Graphical Models: Modeling, Optimization, and Hilbert Space Embedding

Xinhua Zhang

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

March 2010

Declaration

Except where otherwise indicated, this thesis is my own original work.

[illegible]

Xinhua Zhang
March 2010

The following table gives the collaborators of each chapter, and the publications.

Chapter	Collaborators	Publication
1	Douglas Aberdeen and SVN Vishwanthan	ICML 2007, (Zhang et al., 2007)
2	Thore Graepel and Ralf Herbrich	AISTATS 2010, (Zhang et al., 2010a)
3	Alex Smola, Le Song, and Arthur Gretton	NIPS 2008, (Zhang et al., 2009)
4	SVN Vishwanthan and Ankan Saha	NIPS 2010, (Zhang et al., 2010b)

Two other papers published but not included in this thesis are (Song, Zhang, Smola, Gretton, & Schölkopf, 2008b) and (Cheng, Vishwanathan, & Zhang, 2008).

To my parents for their love and support.

Acknowledgements

It is finally the time to summarize my four years' PhD research in a single coherent document, and I would like to take this opportunity to thank all the people who helped me along the way.

First of all, I wish to express my utmost gratitude to S. V. N. Vishwanthan, known to all as Vishy, for his long time guidance in my research and life. Being energetic and enthusiastic with research, Vishy has given me most valuable hand-on guidance in both theoretical and practical problems. I still remember those long discussions into late night, both in Vishy's office and at his home. After becoming a faculty at Purdue University, Vishy kindly offered me a visiting scholar position and I considerably benefited from this visit at Purdue.

My gratitude goes to Alex Smola, who showed me how to perform creative research and impressed me with his novel ideas. Despite the pain of following and digesting his thoughts, it is almost always fruitful to explore in the directions Alex suggested and this high level supervision greatly helped improve my ability to conduct independent research.

Thank you to Douglas Aberdeen for introducing reinforcement learning to me, and guiding me through the project of conditional random fields for reinforcement learning. It was really a pity that Doug left NICTA after the project grew into a good shape.

Many thanks to Microsoft Research Cambridge (MSRC), who offered me a 12-week internship from September 2008. My mentors Thore Graepel and Ralf Herbrich are especially creative, knowledgeable and friendly to help me in both theory and practice.

Thank you to Wray Buntine for teaching me nonparametric Bayesian, and chairing my supervisory panel after Alex and Vishy left NICTA.

My gratitude goes to the CSL/RSISE/CECS administration staff, especially Michelle Moravec, Di Kossatz, and Deb Pioch for creating a friendly and efficient environment for learning and research, keeping me on track, and helping me focus on research. Also thank you to Norma Lucas at Purdue University for all the administrative help.

I have also greatly benefited from discussions with other researchers, students and visitors of NICTA, of the Computer Sciences Lab of ANU, and of the Department of Statistics at Purdue University. The following list is absolutely not complete: Marconi Barbosa, Justin Bedo, Olivier Buffet, Tiberio Caetano, Dmitry Kamenetsky, Li Cheng, Nan Ding, Markus Hegland, Jiayuan Huang, Knut Hüper, Marcus Hutter, Quoc Viet

Le, Novi Quadrianto, Mark Reed, Scott Sanner, Nic Schraudolph, Hao Shen, Javen (Qinfeng) Shi, Le Song, Owen Thomas, Choon Hui Teo, Tao Wang, Chris Webers, and Jin Yu.

I gratefully acknowledge the scholarship from NICTA, fellowship from Microsoft Research Asia, Vice-Chancellor's grant from ANU for attending conferences and visiting Purdue University, and travel grants from Pascal, Pascal 2, ICML, and NIPS.

Xinhua Zhang
March 2010

Abstract

Over the past two decades graphical models have been widely used as powerful tools for compactly representing distributions. On the other hand, kernel methods have been used extensively to come up with rich representations. This thesis aims to combine graphical models with kernels to produce compact models with rich representational abilities.

Graphical models are a powerful underlying formalism in machine learning. Their graph theoretic properties provide both an intuitive modular interface to model the interacting factors, and a data structure facilitating efficient learning and inference. The probabilistic nature ensures the global consistency of the whole framework, and allows convenient interface of models to data.

Kernel methods, on the other hand, provide an effective means of representing rich classes of features for general objects, and at the same time allow efficient search for the optimal model. Recently, kernels have been used to characterize distributions by embedding them into high dimensional feature space. Interestingly, graphical models again decompose this characterization and lead to novel and direct ways of comparing distributions based on samples.

Among the many uses of graphical models and kernels, this thesis is devoted to the following four areas:

Conditional random fields for multi-agent reinforcement learning Conditional random fields (CRFs) are graphical models for modeling the probability of labels given the observations. They have traditionally been trained with using a set of observation and label pairs. Underlying all CRFs is the assumption that, conditioned on the training data, the label sequences of different training examples are independent and identically distributed (*iid*). We extended the use of CRFs to a class of temporal learning algorithms, namely policy gradient reinforcement learning (RL). Now the labels are no longer *iid*. They are actions that update the environment and affect the next observation. From an RL point of view, CRFs provide a natural way to model joint actions in a decentralized Markov decision process. They define how agents can communicate with each other to choose the optimal joint action. We tested our framework on a synthetic network alignment problem, a distributed sensor network, and a road traffic control system. Using tree sampling by Hamze & de Freitas (2004) for inference, the RL methods employing CRFs clearly outperform those which do not

model the proper joint policy.

Bayesian online multi-label classification Gaussian density filtering (GDF) provides fast and effective inference for graphical models (Maybeck, 1982). Based on this natural online learner, we propose a Bayesian online multi-label classification (BOMC) framework which learns a probabilistic model of the linear classifier. The training labels are incorporated to update the posterior of the classifiers via a graphical model similar to TrueSkill (Herbrich et al., 2007), and inference is based on GDF with expectation propagation. Using samples from the posterior, we label the test data by maximizing the expected F-score. Our experiments on Reuters1-v2 dataset show that BOMC delivers significantly higher macro-averaged F-score than the state-of-the-art online maximum margin learners such as LaSVM (Bordes et al., 2005) and passive-aggressive online learning (Crammer et al., 2006). The online nature of BOMC also allows us to efficiently use a large amount of training data.

Hilbert space embedment of distributions Graphical models are also an essential tool in kernel measures of independence for non-*iid* data. Traditional information theory often requires density estimation, which makes it unideal for statistical estimation. Motivated by the fact that distributions often appear in machine learning via expectations, we can characterize the distance between distributions in terms of distances between means, especially means in reproducing kernel Hilbert spaces which are called kernel *embedment*. Under this framework, the undirected graphical models further allow us to factorize the kernel embedment onto cliques, which yields efficient measures of independence for non-*iid* data (Zhang et al., 2009). We show the effectiveness of this framework for ICA and sequence segmentation, and a number of further applications and research questions are identified.

Optimization in maximum margin models for structured data Maximum margin estimation for structured data, *e.g.* (Taskar et al., 2004), is an important task in machine learning where graphical models also play a key role. They are special cases of regularized risk minimization, for which bundle methods (BMRM, Teo et al., 2007) and the closely related SVM$^{\text{Struct}}$ (Tsochantaridis et al., 2005) are state-of-the-art general purpose solvers. Smola et al. (2007b) proved that BMRM requires $O(1/\epsilon)$ iterations to converge to an ϵ accurate solution, and we further show that this rate hits the lower bound. By utilizing the structure of the objective function, we devised an algorithm for the structured loss which converges to an ϵ accurate solution in $O(1/\sqrt{\epsilon})$ iterations. This algorithm originates from Nesterov's optimal first order methods (Nesterov, 2003, 2005b).

Contents

List of Figures

List of Tables

List of Symbols

$[n]$	Index set $\{1, 2, \ldots, n\}$	3
$\mathbb{R}_+$	Set of positive real numbers $(0, \infty)$	3
$\overline{\mathbb{R}}_+$	Set of non-negative real numbers $[0, \infty)$	3
$\boldsymbol{\phi}$	Vector of sufficient statistics	3
$\top$	Transpose of matrix or vector	3
$\mathcal{X}$	Sample space, or space of feature vectors	3
$\boldsymbol{\theta}$	Natural parameter	3
d	Number of features	3
ν	Base measure of exponential family of distributions	3
Θ	The set of $\boldsymbol{\theta}$ for which $g(\boldsymbol{\theta}) < +\infty$	4
$\mathcal{P}_{\boldsymbol{\phi}}$	Exponential family generated by $\boldsymbol{\phi}$	4
$g(\boldsymbol{\theta})$	Log partition function	3
C^{∞}	Infinitely differentiable	4
MRF_G	Markov random field corresponding to graph G	6
$X \perp\!\!\!\perp Y \| Z$	Random variable X is independent of Y given Z	6
$\mathcal{C}$	Set of maximal cliques	7
$k(x, y)$	Kernel function	11
$\mathbb{N}$	Set of natural numbers: $\{1, 2, \ldots\}$	12
$\mathbb{N}_0$	$\mathbb{N} \cup \{0\} = \{0, 1, 2, \ldots\}$	12
$\delta(\cdot)$	$\delta(\cdot) = 1$ if $\cdot$ is true, and 0 otherwise	12
$\mathcal{H}(p)$	Reproducing kernel Hilbert space (RKHS)	14
$\overset{\mathcal{H}}{\to}$	Convergence in RKHS norm	14
$\rightrightarrows$	Uniform convergence of functions	14
$C(\mathcal{X})$	Space of continuous bounded functions $\mathcal{X}$	14
$\mathcal{P}_k$	Exponential family generated by kernel k	16
n	Number of data points	17
$\mathcal{Y}^m$	Cartesian product of $\mathcal{Y}$ for m times	18
$\mathrm{KL}(p, q)$	KL divergence from density p to density q: $\int p(x) \frac{p(x)}{q(x)} \mathrm{d}x$	21
$f^{\star}$	Fenchel dual of function f	23
$\mathcal{M}_{\boldsymbol{\phi}}$	Marginal polytope for sufficient statistics $\boldsymbol{\phi}$	24
$\Lambda_{\boldsymbol{\phi}}$	Mapping from $\boldsymbol{\theta} \in \Theta$ to the mean $\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(x)]$	24
ri	Relative interior	24

$H(p)$	Differential entropy of density p	24
$R_{\mathrm{emp}}(\mathbf{w})$	Empirical risk of weight vector $\mathbf{w}$	28
$\Omega(\mathbf{w})$	Regularizer	28
$\lVert\mathbf{w}\rVert_1$	L_1 norm, $\lVert\mathbf{w}\rVert_1 = \sum_i \lvert\mathbf{w}_i\rvert$	28
ϵ_k	Error at step k: $J(\mathbf{w}_k) - J(\mathbf{w}^*)$	32
qp-bmrm	BMRM with full inner solver	38
ls-bmrm	BMRM with line search inner solver	38
$R(\boldsymbol{\theta})$	Expected long term reward under policy $\boldsymbol{\theta}$	46
$\mathbf{e}_t$	Eligibility trace	47
$\mathbf{y}^i$	Label (sequence) for example i	60
$\delta(x)$	Dirac function: $\delta(x) = 0$ for all $x \neq 0$ and $\int_{\mathbb{R}} \delta(x)\mathrm{d}x = 1$	61
$\mathrm{ExpFs}(\mathbf{l})$	Expected F-score of outputting label $\mathbf{l}$	70
$\widetilde{\mathrm{ExpFs}}(\mathbf{l})$	Empirical estimate of $\mathrm{ExpFs}(\mathbf{l})$ based on samples	74
$\mu[\mathbf{P}]$	Mean element of distribution $\mathbf{P}$	94
$\mathrm{MMD}(\mathbf{P}, \mathbf{Q})$	Maximum mean discrepancy between $\mathbf{P}$ and $\mathbf{Q}$	96
$R_n(\mathcal{H}, \mathbf{P})$	Rademacher average of $\mathcal{H}$ wrt $\mathbf{P}$	96
$\mathrm{HSIC}(X, Y)$	Hilbert-Schmidt independence criteria	97
$\widehat{\mathrm{HSIC}}_b(Z_1^n)$	Biased estimator of HSIC	98
$\widehat{\mathrm{HSIC}}_u(Z_1^n)$	Unbiased estimator of HSIC	99
$k_S(x_S, x'_S)$	Kernel on the subset S	101
$\mathcal{H}_S$	Reproducing kernel Hilbert space on subset S	101
CE	Constant-exclusive kernel	107
CCE	Coordinate Constant-exclusive kernel	107
$\bar{k}_x$	Ground kernel on the cliques of X	109
$[\cdot]_+$	Hinge function $[x]_+ := \max\{0, x\}$	126
$\overline{\mathbb{R}}$	$\mathbb{R} \cup \{+\infty\}$	127
L-$l.c.g$	Lipschitz continuous gradient with modulus L	127
σ-s.c.	Strongly convex with modulus σ	127
$\partial f(\mathbf{w})$	The set of all subgradients of f at $\mathbf{w}$	128
$\Delta(\mathbf{y}, \mathbf{y}^i)$	Discrepancy between $\mathbf{y}$ and $\mathbf{y}^i$	141

Chapter 1

Introduction

Exponential family distributions are one of the most versatile unifying frameworks for statistical modeling and inference. By representing probability densities as generalized linear models, they play a key role in graphical models and conditional random fields, and allow us to conveniently make use of rich sufficient statistics via kernelization. Also contributing to their popularity is the ease in parameter estimation which can be boiled down to convex optimization, and therefore a large body of existing research results can be immediately applied.

At the very high level, this thesis is composed of the following four chapters under the framework of exponential families. Chapter 2 studies how to use conditional exponential families to learn optimal joint policies for multi-agent reinforcement learning. Chapter 3 shows how graphical models can be used for multi-label data, and how to learn Bayesian models from large datasets via online density filtering, based on which multi-variate performance measures can be optimized. Chapter 4 extends the kernel embeddings of distributions to non-*iid* data, and uses graphical models to factorize the kernel measure of independence for efficient estimation. Finally, Chapter 5 explores maximum margin estimation for exponential families, including lower bounds for optimization and a new optimal first-order method based on Nesterov's algorithms (Nesterov, 2003, 2005b).

One of the key advantages of exponential families is their natural connection with graphical models, partly thanks to the Hammersley-Clifford theorem (Hammersley & Clifford, 1971). Over the past two decades graphical models have been widely used as powerful tools for compactly representing distributions, and have become a powerful underlying formalism in machine learning. Their graph theoretic properties provide both an intuitive modular interface to model the interacting factors, and a data structure facilitating efficient learning and inference. The probabilistic nature ensures the global consistency of the whole framework, and allows convenient interface of models to data. Extensions to conditional models are also straightforward, *e.g.* conditional random fields (Lafferty et al., 2001) which, among other applications, can be used to coordinate multiple agents in reinforcement learning, as we will show in Chapter 3.

In addition, exponential families also benefit from describing densities as linear models, which paves way to kernelization and easy utilization of rich feature spaces. In machine learning, kernel methods have been used extensively to represent rich classes of features for general objects, and at the same time allow efficient search for the optimal model. Interestingly, the new family of kernelized distributions can again be factorized using graphical models. In the same spirit, kernels are recently extended to characterizing distributions by embedding them into high dimensional feature spaces, which leads to novel and direct ways of comparing distributions based on samples. Chapter 4 will study its application to independence measures for non-*iid* data.

Besides flexibility in modeling, exponential family distributions also admit convenient parameter estimation, *e.g.* maximum likelihood for regression. In the case of graphical model distributions, this usually requires various operations of inference. As exact inference is proved to be NP-hard in general, efficient approximate inference algorithms have been extensively studied and applied with satisfactory empirical performance. Chapter 3 shows an online inference algorithm for optimizing multi-label and multi-variate performance measures. On the other hand, classification problems usually call for a different estimator because here the probability of the true label is only required to be the highest among all labels. This principle has led to a number of non-logistic losses in the framework of *regularized risk minimization*, for which many convex non-smooth optimizers are available. Of particular interest to the research community is the convergence rate, which will be studied in Chapter 5.

The structure of this chapter is illustrated in Figure 1.1. We first introduce the exponential family of distributions in Section 1.1, which will serve as a unifying framework for the rest of the thesis. Then three extensions are given in the subsequent three sections: Section 1.2 discusses graphical model decomposition using Markov random fields, Section 1.3 describes conditional random fields, and Section 1.4 shows how to use features from reproducing kernel Hilbert spaces for exponential families, and how the kernelized distributions can be again factorized by graphical models. Then we move on to parameter estimation for exponential families, and Section 1.5 shows the maximum likelihood estimation for which various approximate inference algorithms are discussed. Finally, we introduce regularized risk minimization framework in Section 1.6 where a brief overview of optimization techniques is given, especially the bundle method for machine learning.

Convex analysis will be extensively used in all parts of the thesis, and we include a brief summary of the relevant results in Appendix A.

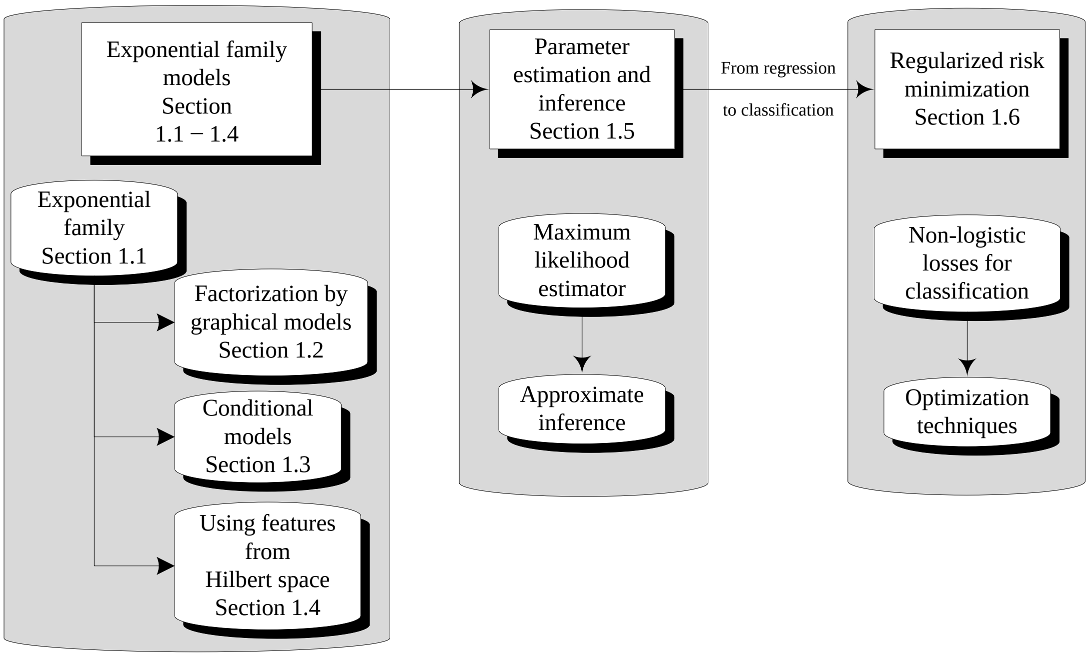

Exponential family models Section 1.1 − 1.4
Exponential family Section 1.1
Factorization by graphical models Section 1.2
Conditional models Section 1.3
Using features from Hilbert space Section 1.4
Parameter estimation and inference Section 1.5
Maximum likelihood estimator
Approximate inference
From regression to classification
Regularized risk minimization Section 1.6
Non-logistic losses for classification
Optimization techniques

Figure 1.1: Structure of introduction.

1.1 Exponential families

Exponential families of distributions (henceforth abbreviated as exponential family) play a key role in machine learning, and have been widely used as a general formalism for graphical models and kernel methods. Suppose we have a random variable X whose sample space is $\mathcal{X}$. Using any arbitrary function $h : \mathcal{X} \mapsto \overline{\mathbb{R}}_+ := [0, +\infty)$, one can endow $\mathcal{X}$ with a measure ν: $\mathrm{d}\nu = h(x)\mathrm{d}x$, where $\mathrm{d}x$ is a counting measure if $\mathcal{X}$ is a discrete space, or the Lebesgue measure if $\mathcal{X}$ is a continuous.

An exponential family is a parametric class of probability densities with respect to measure ν. Let $\phi_i : \mathcal{X} \mapsto \mathbb{R}$ ($i \in [d] := \{1, \ldots, d\}$) be real valued Borel measurable functions, and assemble $\boldsymbol{\phi}(x) := (\phi_1(x), \ldots, \phi_d(x))^\top$. $\boldsymbol{\phi}$ is also known as *sufficient statistics* or *features*. Using any vector $\boldsymbol{\theta} \in \mathbb{R}^d$ which is called *natural parameter*, we can define a probability density function (*pdf* [1])with respect to $\mathrm{d}\nu$:

$$p(x; \boldsymbol{\theta}) := \exp\left(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta})\right), \tag{1.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, and

$$g(\boldsymbol{\theta}) := \log \int \exp\left(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle\right) \nu(\mathrm{d}x)$$

[1]Strictly speaking, when the sample space is discrete, *e.g.* the measure $\mathrm{d}\nu$ is a counting measure, the $p(x)$ is called *probability mass function*. Here and from now on, we will always simply use the term *pdf* with slight sacrifice of mathematical rigor.

ensures $\int p(x; \boldsymbol{\theta}) \nu(\mathrm{d}x) = 1$, and is called *log-partition function.* Of course, this definition is valid only if $g(\boldsymbol{\theta}) < \infty$, and we denote the set of all such $\boldsymbol{\theta}$ as $\Theta := \{\boldsymbol{\theta} : g(\boldsymbol{\theta}) < \infty\}$. Now we define the *exponential family* generated by $\boldsymbol{\phi}$ as the set of *pdf* induced by $\boldsymbol{\theta} \in \Theta$:

$$\mathcal{P}_{\boldsymbol{\phi}} := \{p(x; \boldsymbol{\theta}) := \exp\left(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta})\right) : \boldsymbol{\theta} \in \Theta\}.$$

With a fixed $\boldsymbol{\phi}(x)$, each $\boldsymbol{\theta}$ indexes a particular distribution of the family $\mathcal{P}_{\boldsymbol{\phi}}$. Many classes of distribution are exponential families, *e.g.*, Gaussian, Poisson, multinomial, χ^2, and Dirichlet. The statement "a class of distributions belongs to the exponential family" is implicitly with respect to (wrt) a particular parametrization or choice of natural parameter. For example, the 1-d Laplace distribution $p(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$ $(b > 0)$ is an exponential family with respect to $\frac{1}{b}$ for any fixed μ, and the sufficient statistics are $|x - \mu|$:

$$p(x; \theta) = \exp\left(\langle -|x - \mu|, \theta \rangle - \log \frac{2}{\theta}\right), \qquad \text{where } \theta := \frac{1}{b},$$

and $\Theta = \mathbb{R}_+$. But $p(x)$ is clearly not an exponential family wrt μ or (μ, b) jointly. By the same token, Cauchy, uniform, and t-distribution are not exponential families.

Exponential families have a number of desirable properties. We just list a few and the proof can be found in any related textbook, *e.g.* (Brown, 1986; Dobson & Barnett, 2008). Many properties are related to convex analysis, a brief introduction to which is provided in Appendix A.

First of all, the exponential family is closed under multiplication and division, *i.e.* pq and p/q are in $\mathcal{P}_{\boldsymbol{\phi}}$ up to a normalization constant if $p, q \in \mathcal{P}_{\boldsymbol{\phi}}$, and they correspond to addition and subtraction of natural parameters respectively[2]. This lends a lot of convenience to some inference algorithms such as expectation propagation. However, $\mathcal{P}_{\boldsymbol{\phi}}$ is not closed under addition or subtraction, and is not a convex set in general.

The log partition function $g(\boldsymbol{\theta})$ is very important to exponential families. It is a lower semi-continuous convex function (see Proposition 71 in Appendix A), hence its domain Θ must also be convex. Θ may not necessarily be open, but we only deal with open Θ. Then $g(\boldsymbol{\theta})$ is C^∞ on Θ (see Proposition 74 in Appendix A), and $g(\boldsymbol{\theta})$ is a

[2]As long as the resulting natural parameter is still in Θ.

cumulant generating function, *i.e.* for any $\boldsymbol{\theta} \in \Theta$ we have

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) &= \frac{\nabla_{\boldsymbol{\theta}} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x)}{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x)} \overset{*}{=} \frac{\int \nabla_{\boldsymbol{\theta}} \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x)}{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x)} \\
&= \frac{\int \boldsymbol{\phi}(x) \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x)}{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x)} \qquad (**) \\
&= \mathop{\mathbb{E}}_{x \sim p(x;\boldsymbol{\theta})} [\boldsymbol{\phi}(x)] \\
\frac{\partial^2}{\partial \theta_i \partial \theta_j} g(\boldsymbol{\theta}) &\overset{by\_**}{=} e^{-2g(\boldsymbol{\theta})} \left(e^{g(\boldsymbol{\theta})} \int \phi_i(x)\phi_j(x) \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x) \right. \\
&\qquad \left. - \int \phi_i(x) \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x) \int \phi_j(x) \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)\nu(\mathrm{d}x) \right) \\
&= \mathop{\mathbb{E}}_{x \sim p(x;\boldsymbol{\theta})} [\phi_i(x)\phi_j(x)] - \mathop{\mathbb{E}}_{x \sim p(x;\boldsymbol{\theta})} [\phi_i(x)] \mathop{\mathbb{E}}_{x \sim p(x;\boldsymbol{\theta})} [\phi_j(x)] \\
&= \mathrm{Cov}_{x \sim p(x;\boldsymbol{\theta})} [\phi_i(x)\phi_j(x)].
\end{aligned}$$

The step (*) where differentiation is interchanged with integral can be proven using the dominated convergence theorem, and the detailed proof is available in Appendix A Proposition 73.

As the covariance matrix of the random vector $\boldsymbol{\phi}(x)$ must be positive semi-definite, the Hessian of $g(\boldsymbol{\theta})$ must be positive semi-definite on the open set Θ, and hence $g(\boldsymbol{\theta})$ is convex (Hiriart-Urruty & Lemaréchal, 1993a, Theorem 4.3.1). To further ensure strong convexity, we need the minimality of sufficient statistics.

Definition 1 (Minimality of sufficient statistics) *The sufficient statistics* $\boldsymbol{\phi}(x) \in \mathbb{R}^d$ *is called* minimal *if there does not exist a nonzero vector* $\boldsymbol{\theta} \in \mathbb{R}^d$*, such that* $\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle$ *is a constant* ν*-almost everywhere*[3]*.*

Notice that the $\boldsymbol{\theta}$ in this definition is *not* required to be in Θ.

Proposition 2 *(Wainwright & Jordan, 2008, Proposition 3.1)* $g(\boldsymbol{\theta})$ *is convex in* $\boldsymbol{\theta}$*, and strongly convex if, and only if,* $\boldsymbol{\phi}$ *is minimal.*

Proof is available in Proposition 70 in Appendix A.

In the next three sections, we will introduce three specializations and generalizations of exponential families, namely structural factorization according to graphical models, conditioning, and using sufficient statistics from Hilbert spaces.

[3]This means that the set $\{\boldsymbol{\theta} : \langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle$ is not constant$\}$ has measure 0 under ν.

1.2 Graphical models and factorization

The random variable in the previous section can be multi-variate in general, and in such cases, two new challenges arise:

1. How to characterize the relationship between individual random variables. For example, one key relationship is the conditional independence.

2. In the case of discrete random variables, the joint state space grows exponentially fast with the number of variables. This poses considerable difficulty for the computations such as marginalization and log partition function.

It turns out that conditional independence can be compactly modeled by graphical models (Lauritzen, 1996), which also lend significant savings to the computational tasks via factorizing the joint distribution. In this section, we will introduce one such tool called Markov random fields.

Given a joint *pdf* p of multi-variate random variable (*mrv*) (X, Y, Z), X is said to be independent of Y conditioned on Z (denoted as $X \perp\!\!\!\perp Y|Z$) if $p(x, y|z) = p(x|z)p(y|z)$. Here the lowercase letters stand for the instantiations of the corresponding random variable. For convenience, we will collect all the random variables in question into a *mrv* X, and identify individual random variables by X_i. For a subset of indices A, we denote as X_A the *mrv* consisting of all the corresponding random variables, and its instantiation as x_A (x_A is a vector if A has more than one element, but we prefer not to write $\mathbf{x}_A$).

1.2.1 Markov random fields

This section formally introduces Markov random fields as a tool for modeling conditional independence.

Definition 3 (Graph separation) *Given an undirected graph $G = (V, E)$ where V and E are the set of nodes and edges respectively, let A, B, C be disjoint subsets of nodes. If every path from A to B includes at least one node from C, then C is said to separate A from B in G.*

Definition 4 (Markov random field) *Given an undirected graph G, a Markov random field (MRF) is defined as a set of probability distributions* $\mathrm{MRF}_G := \{p(\mathbf{x}) : p(\mathbf{x}) > 0, \forall p, \mathbf{x}\}$ *such that for all $p \in \mathrm{MRF}_G$ and for any three disjoint subsets A, B, C of G, if C separates A from B then p satisfies $X_A \perp\!\!\!\perp X_B|X_C$. If $p \in \mathrm{MRF}_G$, we often say p* respects *G.*

Like exponential families, MRFs are also a class of distributions. With this definition, naturally we ask two questions: a) given a *pdf*, how to determine whether it is in MRF_G, *i.e.* how to efficiently check all the conditional independence relationships encoded in G; b) for all distributions in MRF_G, how their *pdf* should look like. It turns out that strong results are available, and a formal description calls for the following graph theoretic definition.

Definition 5 (Cliques and maximal cliques) *A* clique *of a graph is a subgraph of it where each pair of nodes is connected by an edge. The* maximal clique *of a graph is a clique which is not a proper subset of another clique.*

We usually denote the set of maximal cliques by $\mathcal{C}$.

Definition 6 (Factorization wrt an undirected graph) *A* pdf $p(\mathbf{x})$ *is said to factorize wrt a given undirected graph G if it can be written as:*

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c), \tag{1.2}$$

where ψ_c is an arbitrary non-negative real valued function called potential functions, and the constant Z ensures $\int p(\mathbf{x}) \mathrm{d}\mathbf{x} = 1$.

This definition gives a clear form of *pdf* based on the maximal cliques, which seems to have nothing to do with the conditional independence relationships in G. However, they are almost equivalent as stated by the next two theorems.

Theorem 7 (Factorization implies conditional independence) *If a* pdf *p factorizes according to an undirected graph G, then $p \in \text{MRF}_G$,* i.e., *if A, B, and C are disjoint subsets of nodes such that C separates A from B in G, then p satisfies $X_A \perp\!\!\!\perp X_B | X_C$.*

This theorem is actually not hard to prove, but the converse result is more involved and is well known as the Hammersley-Clifford theorem.

Theorem 8 (Hammersley-Clifford theorem) *(Hammersley & Clifford, 1971) If a* pdf *$p(\mathbf{x}) \in \text{MRF}_G$, then $p(x)$ must also factorize according to G,* i.e., *there exist functions $\varphi_c(\mathbf{x})$ on $c \in \mathcal{C}$, such that*

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{c \in \mathcal{C}} \varphi_c(x_c)\right). \tag{1.3}$$

In addition to undirected graphs, the directed graphs are also widely used, but in a different class of probabilistic models called Bayesian networks. In this thesis, most of our attention is restricted to MRFs.

Decomposed exponential families

Theorem 7 shows that if the sufficient statistics $\boldsymbol{\phi}$ and natural parameters $\boldsymbol{\theta}$ of an exponential family factorize onto the cliques by $\{\phi_c\}_{c\in\mathcal{C}}$ and $\{\theta_c\}_{c\in\mathcal{C}}$ respectively:

$$p(\mathbf{x};\boldsymbol{\theta}) := \exp\left(\sum_{c\in\mathcal{C}} \langle \phi_c(x_c), \theta_c\rangle - g(\boldsymbol{\theta})\right),$$

then all the distributions in $\mathcal{P}_{\boldsymbol{\phi}}$ must respect G. We will write $\boldsymbol{\theta} = \text{vec}_{c\in\mathcal{C}}\{\theta_c\}$ and $\boldsymbol{\phi}(\mathbf{x}) = \text{vec}_{c\in\mathcal{C}}\{\phi_c(x_c)\}$ where the operator vec concatenates vectors. The converse does not immediately hold because the sufficient statistics $\boldsymbol{\phi}$ are fixed, while Eq. (1.3) allows very general log-potential functions φ_c. However, there is a nontrivial result.

Theorem 9 *(Lauritzen, 1996) If <u>all distributions</u> in $\mathcal{P}_{\boldsymbol{\phi}}$ respect graph G, then $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ must factorize onto the cliques by $\{\phi_c\}_{c\in\mathcal{C}}$ and $\{\theta_c\}_{c\in\mathcal{C}}$ respectively.*

In machine learning, we often need to use the data to find a distribution respecting a known graph G. If we restrict ourselves to the exponential family with some pre-specified sufficient statistics that factorize along the maximal cliques of G, then the distribution is guaranteed to respect G and we only need to estimate the clique-wise natural parameters. This gives a parametric model since ϕ_c are fixed.

1.3 Conditional random fields

Conditional probability plays an important role in probability and statistics. In this section, we will show that it can also be effectively modeled by exponential families, where factorization by graphical models is again applicable. In particular, we introduce one such tool called conditional random fields (CRFs).

CRFs are a probabilistic framework proposed by Lafferty et al. (2001) for labeling and segmenting data. It can be interpreted as a conditional MRF, which consists of two types of nodes: observations X and latent states Y. MRFs model the joint distribution $p(x, y)$. However, in many applications, we only need a conditional model $p(y|x)$, and $p(x)$ is not important. For example, in the sequence tagging problem, we are only interested in the probability of the tags given the tokens, and we do not care about how the tokens are generated. In other words, CRFs are a discriminative model.

By this token, CRFs can be equivalently viewed as an MRF on Y, deleting all the nodes of X and their associated edges. As a result, the parameter estimation and inference algorithms in the normal MRFs directly carry over without change. The clique-wise potential functions $\phi_c(y_c; \mathbf{x})$ are parameterized by x, allowing ϕ_c to be influenced by the whole observations instead of merely local observations.

This interpretation further allows us to study CRFs in the framework of exponential family, and we begin with y being a univariate random variable. Given observation $\mathbf{x} \in \mathcal{X}$, we define a conditional distribution over label $y \in \mathcal{Y}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$ and it can be written in its canonical form as

$$p(y|\mathbf{x};\boldsymbol{\theta}) := \exp(\langle \boldsymbol{\phi}(\mathbf{x}, y), \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta}|\mathbf{x})). \tag{1.4}$$

Here, vector $\boldsymbol{\phi}(\mathbf{x}, y)$ is the sufficient statistics and $\boldsymbol{\theta}$ is the natural parameter. $g(\boldsymbol{\theta}|\mathbf{x})$ is the log-partition function for normalization:

$$g(\boldsymbol{\theta}|\mathbf{x}) := \log \int_y \exp(\langle \boldsymbol{\phi}(\mathbf{x}, y), \boldsymbol{\theta} \rangle) \mathrm{d}y \tag{1.5}$$

To make the definition Eq. (1.4) valid, we require that $g(\boldsymbol{\theta}|\mathbf{x}) < \infty$, and hence define the admissible region $\Theta(\mathbf{x}) := \{\boldsymbol{\theta} : g(\boldsymbol{\theta}|\mathbf{x}) < \infty\}$. So now with a given $\mathbf{x}$, we can define a conditional exponential family (CEF) as

$$\mathcal{P}_{\boldsymbol{\phi}|\mathbf{x}} := \{\exp(\langle \boldsymbol{\phi}(\mathbf{x}, y), \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta}|\mathbf{x})) : \boldsymbol{\theta} \in \Theta(\mathbf{x})\}\,. \tag{1.6}$$

The sufficient statistics $\boldsymbol{\phi}(\mathbf{x}, y)$ are problem specific and represent salient features of the input observations. Of fundamental importance is that $\boldsymbol{\theta}$ does *not* depend on $\mathbf{x}$, which gives a discriminative model (otherwise the model is essentially the same as MRFs).

In $\mathcal{P}_{\boldsymbol{\phi}|x}$, it is known that the log-partition function is also the cumulant generating function of the CEF, *e.g.*:

$$\frac{\partial}{\partial \boldsymbol{\theta}} g(\boldsymbol{\theta}|\mathbf{x}) \;=\; \mathbb{E}_{y \sim p(y|\mathbf{x};\boldsymbol{\theta})}[\boldsymbol{\phi}(\mathbf{x}, y)]. \tag{1.7}$$

1.3.1 Factorization of conditional distributions

More generally, we consider the case of structured output where $\mathbf{y} \in \mathcal{Y}^m$ (m nodes). Then we get a counter-part of Theorem 9 for CEF, which states that for any fixed $\mathbf{x}$ if the conditional density $p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ factorizes according to a graph G on $\mathbf{y}$, then the sufficient statistics $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ decompose into terms over the maximal cliques $\mathcal{C}$ of G (Altun et al., 2004b):

$$\phi(\mathbf{x}, \mathbf{y}) = \mathrm{vec}\{\phi_c(\mathbf{x}, \mathbf{y}_c) | c \in \mathcal{C}\} \tag{1.8}$$

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \exp\left(\sum_{c \in \mathcal{C}} \langle \phi_c(\mathbf{x}, \mathbf{y}_c), \boldsymbol{\theta}_c \rangle - g(\boldsymbol{\theta}|\mathbf{x})\right), \tag{1.9}$$

where y_c is the configuration for nodes in clique c.

CRFs are examples of CEFs with special graphs. For 1-D CRFs, the graph is a chain, so the edge features are $\phi_{i,i+1}(\mathbf{x}, y_i, y_{i+1})$. For 2-D grid CRFs shown in Fig-

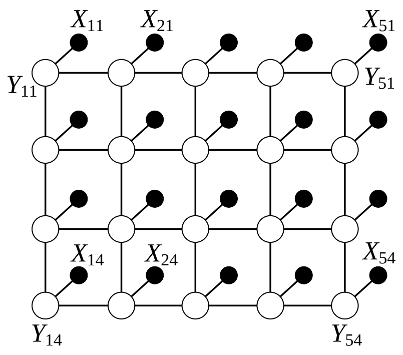


Figure 1.2: 2-D Grid CRF. X_{ij} are observations and Y_{ij} are latent states.

ure 1.2, the edge features are $\phi_{(ij)(i'j')}(\mathbf{x}, y_{ij}, y_{i'j'})$ where nodes are indexed by double coordinates and $|i - i'| + |j - j'| = 1$. In this case, all maximal cliques have size two, *i.e.*, an edge between nodes has a feature associated with it. We can also associate potentials ϕ_{ij} to single nodes Y_{ij}. Node features represent the observation of state available at each node, while the edge features encode the communication or consistency between nodes about their features and states.

Historically, CRFs were motivated from the so-called *label bias problem* that was first observed by Bottou (1991). This problem occurs when the graphical model employs local (or per state) normalization like in hidden Markov models, where at each state the transition only compete among the possibilities from that state, without taking into account the transition probabilities in other parts of the model. Therefore, paths whose states have fewer outgoing transitions receive improper preference. CRFs circumvent this problem by introducing a global normalization to incorporate global interactions, and replacing the local (normalized) transition probability with local (unnormalized) potentials for proper scaling.

1.4 Reproducing kernel Hilbert spaces for exponential families

So far, we have restricted the sufficient statistics $\boldsymbol{\phi}(x)$ and the natural parameters $\boldsymbol{\theta}$ of the exponential families to be from the Euclidean space. As the *pdf* just requires the inner product between $\boldsymbol{\phi}(x)$ and $\boldsymbol{\theta}$, they only need to be from an inner product space. To facilitate convenient learning and inference (see Section 1.5), this space needs to be endowed with richer structure, and in this section we introduce a practical extension: complete inner product spaces where point-wise evaluation is a continuous linear functional, *i.e.* reproducing kernel Hilbert space (RKHSs).

RKHSs have been used extensively in machine learning, with the following key advantages: a) this space of functions can be very rich which provides considerable modeling flexibility, b) only inner product between objects needs to be defined, and this allows us to deal with much more general objects than real vectors, such as strings, graphs, images, and gene sequences, c) searching in this function space to optimize some objective functionals can be conveniently reduced to optimization in Euclidean space.

The objective of this section is to show how RKHSs can be used to define expressive distributions on generic object spaces via exponential families, and how they can be decomposed wrt graphical models. Most results will be given in Section 1.4.3. To this end, we first introduce the necessary building blocks such as positive semi-definite kernels in Section 1.4.1, based on which we rigorously define the RKHS in Section 1.4.2.

1.4.1 Positive semi-definite kernels

Widely used in machine learning, kernel methods first gained their popularity in maximum margin based supervised learning, especially support vector machines. Then they were extended to unsupervised learning such as kernel principal component analysis (Schölkopf et al., 1996), and statistical inferences on probabilities (Smola et al., 2007c).

Intuitively, a kernel is simply a similarity measure between two objects from any arbitrary space. Boser et al. (1992) observed that many supervised learning algorithms depend on training examples only via the inner product of their features, which essentially characterizes their similarity. Based on this insight, they proposed using the kernels directly as a similarity measure, *i.e.* kernels implicitly induce a feature map. The resulting feature space may be extremely rich for some kernels, and surprisingly search in this space is tractable thanks to the representer theorem. This rich feature space has recently been used by Smola et al. (2007c) to embed distributions, with the advantage that statistical properties can be estimated by directly evaluating kernels on samples. We will show in Chapter 4 that the feature space can be factorized wrt graphical models, which leads to novel statistical inference tools for complex domains.

Given a nonempty space of objects $\mathcal{X}$, suppose we have a map $\boldsymbol{\phi}$ from $\mathcal{X}$ to a feature space $\mathcal{H}$ which is Hilbert but not necessarily Euclidean. The inner product $\langle \boldsymbol{\phi}(x_1), \boldsymbol{\phi}(x_2) \rangle$ describes somehow the similarity between x_1 and x_2. Now our motivation is to work the other way round: directly define a similarity measure $k(x_1, x_2)$ which is the inner product of some *unknown and implicit* feature map $\boldsymbol{\phi}$:

$$k(x_1, x_2) = \langle \boldsymbol{\phi}(x_1), \boldsymbol{\phi}(x_2) \rangle_{\mathcal{H}} . \tag{1.10}$$

The benefit of doing so is to directly measure the similarities on complex domains (*e.g.*, genes, images), which is considerably more convenient than indirectly through a feature map $\boldsymbol{\phi}$. The question then is what property is needed from k in order to guarantee the

Table 1.1: Example kernels on $\mathbb{R}^n$

Name	Form of $k(x, x')$
linear kernel	$\langle x, x' \rangle$
polynomial kernel	$(\langle x, x' \rangle + c)^d$, $c > 0, d \in \mathbb{N}$
Gaussian kernel	$\exp\left(-\sigma^2 \left\Vert x - x' \right\Vert^2\right)$
Laplace kernel	$\exp\left(-\sigma^2 \left\Vert x - x' \right\Vert\right)$
Delta kernel	$\delta(x = x')$ ($\delta(\cdot) = 1$ if $\cdot$ is true, and 0 otherwise)

existence of such a $\boldsymbol{\phi}$. Clearly, the following conditions are necessary for k:

1. Finite valued, *i.e.*, $k(x_1, x_2) < \infty$ for all $x_1, x_2 \in \mathcal{X}$.

2. Symmetric. $k(x_1, x_2) = k(x_2, x_1)$.

3. Positive semi-definite. For any $x_1, \ldots, x_n \in \mathcal{X}$, letting $M := (\boldsymbol{\phi}(x_1), \ldots, \boldsymbol{\phi}(x_n))$ and $K := M^\top M = (k(x_i, x_j))_{i,j=1,\ldots,n}$, K must be positive semi-definite (PSD).

This motivates the definition of PSD kernel.

Definition 10 (Positive semi-definite kernel) *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semi-definite kernel if it is finite valued and symmetric, and for any finite set $x_1, \ldots, x_n \in \mathcal{X}$, the Gram matrix $K := (k(x_i, x_j))_{i,j=1,\ldots,n}$ is positive semi-definite.*

Table 1.1 gives some examples of kernel on $\mathbb{R}^n$. More examples on graphs, trees, and strings can be found in (Vishwanathan et al., 2009; Collins & Duffy, 2001; Moschitti, 2006; Haussler, 1999; Teo & Vishwanathan, 2006).

It is surprising that these necessary conditions *are* indeed sufficient as well. Given a PSD kernel k, we can explicitly construct a feature map $\boldsymbol{\phi}$ together with an inner product in the image space, such that Eq. (1.10) is satisfied. Below we give two examples of construction: the first example maps to a possibly infinite dimensional *Euclidean space* with the normal inner product (sum of element-wise product); the second example maps to a *space of functions* where the inner product is more involved. The second example will be detailed in Section 1.4.2, and the first construction is formally stated by Mercer's theorem.

Theorem 11 (Simplified Mercer's theorem) *(Mercer, 1909) Let $\mathcal{X}$ be a compact Hausdorff space, and let k be a continuous symmetric PSD kernel on $\mathcal{X}$. Then there must exist a sequence of functions $\{e_i\}_{i=1}^\infty$ with $e_i : \mathcal{X} \to \mathbb{R}$, and a sequence of nonnegative real numbers $\{\lambda_i\}_{i=1}^\infty$, such that k can be represented as:*

$$k(x_1, x_2) = \sum_{i=1}^\infty \lambda_i e_i(x_1) e_i(x_2), \qquad \textit{for all } x_1, x_2 \in \mathcal{X}, \tag{1.11}$$

where the convergence is absolute and uniform.

Remark 12 1. *Eq. (1.11) implies that to satisfy Eq. (1.10) we can construct the feature map as*

$$\phi(x) = (\sqrt{\lambda_1} e_1(x), \sqrt{\lambda_2} e_2(x), \ldots),$$

with the inner product defined as sum of element-wise product.

2. *In fact, ϕ does map to $\mathcal{H} = \ell_2$ since $k(x,x) = \langle \phi(x), \phi(x) \rangle_{\ell_2} < \infty$.*

3. *The full version of Mercer's theorem also gives the construction of e_i and λ_i.*

4. *The theorem does not cover the uniqueness and surjectivity of these maps.*

In general, we can also drop the PSD condition and have indefinite kernels (Ong et al., 2004). This thesis is restricted to PSD kernels.

1.4.2 Reproducing kernel Hilbert spaces

Associated with a PSD kernel k is a reproducing kernel Hilbert space $\mathcal{H}$. It is a set of functions which is constructed in the following three steps. First include the span of $k(x, \cdot)$[4] for all $x \in \mathcal{X}$:

$$\mathcal{H}_{\frac{1}{2}} = \left\{ \sum_{i=1}^{n} a_i k(x_i, \cdot) : n < \infty, a_i \in \mathbb{R}, x_i \in \mathcal{X} \right\}. \tag{1.12}$$

Second, define an inner product between $f = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$ and $g = \sum_{j=1}^{m} \beta_j k(x'_j, \cdot)$:

$$\langle f, g \rangle := \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, x'_j) = \sum_{j=1}^{m} \beta_j f(x'_j) = \sum_{i=1}^{n} \alpha_i g(x_i). \tag{1.13}$$

Note that although the definition depends on the specific expansion of f and g which may not be unique, it is still well defined because the last two equalities show that the value is independent of the coefficients α_i, x_i, β_j, x'_j given f and g. The other properties required by inner product are clearly satisfied, including symmetry, bilinearity, and positive-definiteness ($\langle f, f \rangle \geq 0$). Since $\langle f, k(x, \cdot) \rangle = f(x)$ for all f, k is called reproducing kernel by Aronszajn (1950).

This inner product and its induced metric further allow us to complete the space $\mathcal{H}_{\frac{1}{2}}$. We define the completed space as the RKHS induced by k:

$$\mathcal{H} = \overline{\mathcal{H}_{\frac{1}{2}}} = \overline{\operatorname{span}\{k(x_i, \cdot) : x_i \in \mathcal{X}\}}. \tag{1.14}$$

[4]This is a function in $\cdot$ parameterized by x.

and the inner product defined on $\mathcal{H}_{\frac{1}{2}}$ is also extended to $\mathcal{H}$. So $\mathcal{H}$ is a Hilbert space.

In the sequel, we will write $f_n \xrightarrow{\mathcal{H}} f$ if f_n converges to f in the RKHS norm. One key consequence of converging in RKHS norm is the point-wise convergence of function sequence, and even uniform convergence "$\rightrightarrows$" if the kernel is bounded.

Proposition 13 *If $f_n \xrightarrow{\mathcal{H}} f$, then f_n converges to f point-wise. If in addition $\sup_{x \in \mathcal{X}} k(x,x) < +\infty$, then f_n converges to f uniformly: $f_n \rightrightarrows f$.*

Proof The point-wise convergence is straightforward due to the continuity of inner product in Hilbert space:

$$\lim_{n\to\infty} f_n(x) = \lim_{n\to\infty} \langle f_n, k(x,\cdot)\rangle = \left\langle \lim_{n\to\infty} f_n, k(x,\cdot)\right\rangle = \langle f, k(x,\cdot)\rangle = f(x).$$

If we further have $B := \sup_{x\in\mathcal{X}} k(x,x) < +\infty$, then

$$\begin{aligned}\sup_x |f_n(x) - f(x)| &= \sup_x |\langle f_n - f, k(x,\cdot)\rangle| \leq \sup_x \|f_n - f\| \, \|k(x,\cdot)\| \\ &= \|f_n - f\| \sup_x \sqrt{k(x,x)} \leq \sqrt{B} \, \|f_n - f\| .\end{aligned}$$

So $\lim_{n\to\infty} \|f_n - f\| = 0$ implies $\sup_x |f_n(x) - f(x)| \to 0$ as $n \to \infty$, *i.e.*, $f_n \rightrightarrows f$. ■

Proposition 13 provides a useful way to evaluation the limit function.

There are other ways to define the RKHS associated with kernel k, see *e.g.*, (Aronszajn, 1950) which motivates from operators and invoke Rietz representor theorem[5] (Moore-Aronszajn theorem). Also note that $\mathcal{H}$ can be an infinite dimensional Hilbert space where some "obvious" operations in finite dimensional spaces may not carry over.

Henceforth, we will use $\phi(x)$ and $k(x,\cdot)$ interchangeably with the inner product defined by Eq. 1.13 (and its extension to the closure in Eq. (1.14)). When it is clear from context, we will also abbreviate $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ and $\|\cdot\|_{\mathcal{H}}$ as $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ respectively.

Properties of RKHS

Theorem 14 (Continuous kernels) *(Steinwart, 2001) Let k be a kernel on a metric space $\mathcal{X}$, and $\phi : X \to \mathcal{H}$ be the feature map to the RKHS of k,* i.e., *$\phi(x) = k(x,\cdot)$. Then k is called a continuous kernel if ϕ is continuous.*

Definition 15 (Universal kernels) *(Steinwart, 2001) Let $C(\mathcal{X})$ be a space of continuous bounded functions on a compact domain $\mathcal{X}$. A continuous kernel k on $\mathcal{X}$ is called* universal *if the RKHS $\mathcal{H}$ induced by k is dense in $C(\mathcal{X})$ in L_∞ sense,* i.e., *for*

[5] In fact, this style of construction will be used in Section 4.1 later.

every function $f \in C(\mathcal{X})$ *and every* $\epsilon > 0$, *there exists a function* $g \in \mathcal{H}$ *such that* $\|f - g\|_\infty < \epsilon$.

Gaussian and Laplace kernels are universal, while linear and polynomial kernels are not. Delta kernel is not continuous, so not universal. Some characterizations of universality are available in (Steinwart, 2001) which are useful for checking whether a kernel is universal. Many kernel based algorithms demonstrate desirable properties when using a universal kernel.

As RKHS is an infinite dimensional space of functions, it appears hard to optimize a functional over an RKHS. Fortunately, the representer theorem converts this search to Euclidean space, which offers substantial convenience for machine learning.

Theorem 16 (Representer theorem) *(Kimeldorf & Wahba, 1971), (Schölkopf & Smola, 2002, Section 4.2) Denote by* $\Omega : [0, \infty) \to \mathbb{R}$ *a strictly monotonic increasing function, by* $\mathcal{X}$ *a set, and by* $c : \mathcal{X}^n \to \mathbb{R} \cup \{+\infty\}$ *an arbitrary function. Then each minimizer of* $f \in \mathcal{H}$ *of the functional:*

$$c(f(x_1), \ldots, f(x_n)) + \Omega(\|f\|^2_{\mathcal{H}})$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x).$$

1.4.3 Kernel exponential families and decomposition

Combining kernels and graphical models in the framework of exponential families results in very powerful probabilistic models. On the one hand, kernels induce a very rich feature space, which can be used to generate expressive exponential families. On the other, the factorization of distributions by graphical models has strong connections with the factorization of kernels (Altun et al., 2004b). This section will review their results.

Section 1.2.1 demonstrated the connection between conditional independence and the factorization of the density formula. Now we can extend the Hammersley-Clifford theorem to the exponential families generated by kernels.

Definition 17 (Exponential family generated by a kernel) *Let* k *be a kernel on a domain* $\mathcal{X}$ *which is measurable with respect to the Lebesgue measure. Let its associated RKHS be* $\mathcal{H}$. *The kernelized exponential family generated by* k *is then defined as the* *following set of distributions:*

$$\mathcal{P}_k := \left\{ p(x) = \exp(f(x) - g(f)) : f \in \mathcal{H}, g(f) := \log \int \exp(f(x)) \mathrm{d}x < \infty \right\}. \quad (1.15)$$

Since $f(x) = \langle \phi(x), f \rangle$, we can view $\phi(x)$ as the sufficient statistics, and f as the natural parameter. k is called the generator of $\mathcal{P}_k$.

The benefit of kernel exponential family lies in its rich feature space. Indeed, when the kernel k is universal, $\mathcal{P}_k$ must be dense in the space of "smooth bounded densities":

Theorem 18 (Dense distribution) *(Altun et al., 2004b, Proposition 3) If the kernel k is universal, then $\mathcal{P}_k$ is dense in the space of distributions on $\mathcal{X}$ whose density is continuous and whose infinity norm is finite,* i.e., $\mathcal{P}(C_0) := \{p \in C(\mathcal{X}) : \max_{x \in \mathcal{X}} p(x) < \infty\}$. *Moreover, for any $p \in \mathcal{P}(C_0)$ with $|\log p| \le C$ and $\epsilon < 1$, we have $\|\log p - f\|_\infty \le$ implies $D(p||p_f) \le 2\epsilon$ and $\|p - p_f\|_\infty \le 4\epsilon e^C$.*

This theorem essentially says that the exponential family with universal kernels is rich and can approximate a large class of distributions arbitrarily well in both L_∞ and KL divergence sense. So restricting our attention to the kernel exponential families will not sacrifice much generality.

Now suppose the multi-variate random variable X is endowed with a graphical model G, then for this family of distributions, we also have a result similar to the Hammersley-Clifford theorem. Before stating the result, we define the notion of factorization for kernels:

Definition 19 (Factorization of kernels) *Suppose a kernel k is defined on a domain $\mathcal{X}$, and an undirected graph G is also associated with it. Let $\mathcal{C}$ be the maximal clique set of G. Then k is said to factorize wrt G if for any clique $c \in \mathcal{C}$ there is a kernel k_c on $\mathcal{X}_c$, such that*

$$k(\mathbf{x}, \mathbf{x}') = \sum_{c \in \mathcal{C}} k_c(x_c, x'_c). \quad (1.16)$$

It is important to contrast the meaning of "factorization" for densities and for kernels. The former means expressing the *pdf* as the product of potential functions on the cliques (see Eq. (1.2) in Definition 6). In contrast, the notion of "factorization" for kernels is defined by the additive decomposition of kernels onto cliques. Now we can state the key result on the equivalence between kernel factorization and conditional independence.

Theorem 20 (Hammersley-Clifford Theorem for kernelized exponential families) *(Altun et al., 2004b, Lemma 5) Suppose a graph G has a maximum clique set $\mathcal{C}$. If the kernel k defined on $\mathcal{X}$ factorizes according to G in the sense of Definition 19, then any density $p \in \mathcal{P}_k$ must respect G,* i.e., *for any three disjoint subsets of node A, B, C such that C separates A from B in G, p must satisfy $X_A \perp\!\!\!\perp X_B | X_C$.*

Conversely, if all distributions in $\mathcal{P}_k$ respect G, then the kernel k must factorize.

Both the Hammersley-Clifford Theorem 7, 8 and the kernelized version Theorem 20 discuss the equivalence between factorization of kernels/densities and conditional independence on a graph. However, they differ in two important ways:

1. The Hammersley-Clifford Theorem is concerned with the conditional independence of a *given* distribution with respect to graph G, while Theorem 20 discusses the conditional independence for *a family of* distributions $\mathcal{P}_k$. Both the assumption and the conclusion are stronger in Theorem 20 than in Theorem 7, 8. This stronger property is useful, because we usually need to search for a function in $\mathcal{H}$ (*i.e.*, the natural parameter) in order to optimize some functional. This is also the case for vanilla exponential families in Euclidean spaces without using kernels.

2. It is clear that kernel factorization Eq. (1.16) implies density factorization Eq. (1.3). However, the opposite is not trivial: although each all densities in $\mathcal{P}_k$ must assume the form of Eq. (1.3), the potentials φ_c *can* depend on the particular density. Hence, it is not obvious that there must exist a *common* set of clique-wise sufficient statistics and natural parameters shared by all densities in $\mathcal{P}_k$.

Altun et al. (2004b) proved Theorem 20 based on the key observation that $\mathcal{H}$, as a Hilbert space, must have a set of basis (Banach spaces may not have a basis in general (Enflo, 1973)). Note the second part of Theorem 20 only gives the existence of $\{k_c\}_{c \in \mathcal{C}}$, while the uniqueness may not hold.

1.5 Learning and inference

The previous three sections focused on modeling in the framework of exponential families, and treated the natural parameters $\boldsymbol{\theta}$ as given. However, in practice, $\boldsymbol{\theta}$ is unknown and we are only given some observations $\mathbf{x}^1, \ldots, \mathbf{x}^n$ which are "related" to the underlying distribution. For the time being, we assume these observations are independent and identically-distributed (*iid*) according to the underlying distribution, and extension to non-*iid* observations will be discussed in Chapter 4.

Our objective in this section is to estimate/infer from the observations a single $\boldsymbol{\theta}$ or a distribution of $\boldsymbol{\theta}$ which best "explains" the data. The former is point estimation and the latter is Bayesian estimation. Similarly for CEFs, we are given a set of observations/label pairs $\left\{(\mathbf{x}^i, \mathbf{y}^i)\right\}_i$ and the task of point estimation is to find a single conditional distribution from $\mathcal{P}_{\boldsymbol{\phi}|\mathbf{x}}$ that best explains how $\mathbf{y}$ depends on $\mathbf{x}$. The most straightforward schemes of point estimation are maximum likelihood or maximal aposterior if some prior of $\boldsymbol{\theta}$ is considered. Below, we use CRF learning as an example.

A commonly used point estimator is maximum likelihood, which finds the maximizer of the likelihood. For exponential families with *iid* observations, it enjoys asymptotic unbiasedness and normality, and is asymptotically efficient in the sense of Cramér-Rao bound (Rao, 1973). Technically, it requires evaluating the log-likelihood:

$$\log p\left(\left\{\mathbf{y}^i\right\}_{i=1}^n \middle| \left\{\mathbf{x}^i\right\}_{i=1}^n ; \boldsymbol{\theta}\right) = \sum_{i=1}^n \log p(\mathbf{y}^i|\mathbf{x}^i;\boldsymbol{\theta}) = \sum_{i=1}^n \left\{ \sum_{c\in\mathcal{C}} \left\langle \phi_c(x_c^i, y_c^i), \theta_c \right\rangle - g(\boldsymbol{\theta}|\mathbf{x}^i) \right\} \tag{1.17}$$

$$= \sum_{c\in\mathcal{C}} \left\langle \sum_{i=1}^n \phi_c(x_c^i, y_c^i), \theta_c \right\rangle - \sum_{i=1}^n g(\boldsymbol{\theta}|\mathbf{x}^i).$$

This formula necessitates the computation of the log-partition function:

$$g(\boldsymbol{\theta}|\mathbf{x}) = \log \int_{\mathcal{Y}^m} \prod_{c\in\mathcal{C}} \exp(\langle \phi_c(\mathbf{x}, y_c), \theta_c \rangle) \mathrm{d}\mathbf{y}. \tag{1.18}$$

Second, to maximize the log-likelihood, many optimization algorithms require the gradient of log-likelihood, which in our case is

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log p\left(\left\{\mathbf{y}^i\right\}_{i=1}^n \middle| \left\{\mathbf{x}^i\right\}_{i=1}^n ; \boldsymbol{\theta}\right) = \sum_{i=1}^n \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \sum_{i=1}^n \mathop{\mathbb{E}}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^i;\boldsymbol{\theta})} [\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y})]$$

$$= \sum_{i=1}^n \operatorname*{vec}_{c\in\mathcal{C}} \left\{ \phi_c(\mathbf{x}^i, y_c^i) - \mathop{\mathbb{E}}_{y_c\sim p(y_c|\mathbf{x}^i;\boldsymbol{\theta})} [\phi_c(\mathbf{x}^i, y_c)] \right\} \tag{1.19}$$

where the first step utilized Eq. (1.7) and the second step utilized Eq. (1.8). Therefore we need to compute the mean of the sufficient statistics.

Finally, once a point estimate $\boldsymbol{\theta}^*$ is obtained and a new instance $\mathbf{x}$ is given, a natural way of labeling is via $\operatorname{argmax}_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}^*)$. This is called decoding, or maximum aposterior inference (MAP).

Now we summarize what operations are need from the graphical models in order to perform parameter estimation and decoding. Given the clique-wise natural parameters

θ_c and potential functions $\phi_c(x_c)$ for all $c \in \mathcal{C}$, which implicitly determine the joint joint *pdf* $p(\mathbf{y}|\mathbf{x})$, inference in CRF refers to the following four tasks:

1. **Marginalization and expectations**. Given a particular $\mathbf{x}$, query the marginal distribution of $p(y_c|\mathbf{x};\boldsymbol{\theta})$ for all $c \in \mathcal{C}$. Or more generally, query the marginal distribution of $p(y_S|\mathbf{x};\boldsymbol{\theta})$ for any index subset S. Closely related is the moments on the cliques, *i.e.* the expectation of features ϕ_c: $\mathbb{E}_{y_c \sim p(y_c|\mathbf{x};\boldsymbol{\theta})}[\phi_c(\mathbf{x}, y_c)]$.
2. **Maximum aposterior (MAP)**. Query the mode $\operatorname{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$.
3. **Partition function**. Compute $Z := \int_{\mathcal{Y}^m} \exp\left(\sum_c \langle \phi_c(\mathbf{x}, y_c), \theta_c\rangle\right) \nu(\mathrm{d}\mathbf{y})$.
4. **Sampling**. Draw samples from $p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$.

Similar operations for unconditioned MRFs can be defined correspondingly.

Cooper (1990) proved that the problem of exact inference is NP-hard in general, unless some assumptions on the topology are made. Roth (1996) further showed that it is NP-hard even to approximate it in the sense that for any algorithm, there exists an example structure for which approximate inference has to take super-polynomial time in the scale of the topology. The complexity of exact inference often grows exponentially with how much the graph is more densely connected than a tree (more details later), therefore approximate inference algorithms are essential for many practical problems. Roughly speaking, approximate inference fall into three categories: message passing, sampling and variational inference. This section will briefly overview these methods.

1.5.1 Exact methods

Exact methods are usually based on dynamic programming and the distributive law: $\sum_{i=1}^n \sum_{j=1}^m a_i b_j = \left(\sum_{i=1}^n a_i\right)\left(\sum_{j=1}^m b_j\right)$, where the left hand side incurs mn multiplications and $mn-1$ additions, while the right hand side takes only one multiplication and $n+m-2$ additions. Instead of showing the exact formula, we draw some intuitions from computing the feature expectation in Eq. 1.19.

Letting Z be the constant normalization term $\exp(g(\boldsymbol{\theta}|\mathbf{x}))$, the expected sufficient statistics for a fixed clique c is[6]

$$\begin{aligned}
\mathbb{E}_{p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})}\left[\phi_c(\mathbf{x}, y_c)\right] &= \int_{\mathcal{Y}^m} \phi_c(\mathbf{x}, y_c) p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \mathrm{d}\mathbf{y} \\
&= Z^{-1} \int_{\mathcal{Y}^m} \phi_c(\mathbf{x}, y_{c'}) \exp \sum_{\bar{c}\in\mathcal{C}} \langle \phi_{\bar{c}}(\mathbf{x}, y_{\bar{c}}), \boldsymbol{\theta}_{\bar{c}}\rangle \,\mathrm{d}\mathbf{y} \\
&= Z^{-1} \int_{\mathcal{Y}^m} \prod_{\bar{c}\in\mathcal{C}} \tilde{\phi}^c_{\bar{c}}(\mathbf{x}, y_{\bar{c}}) \exp \langle \phi_{\bar{c}}(\mathbf{x}, y_{\bar{c}}), \boldsymbol{\theta}_{\bar{c}}\rangle \,\mathrm{d}\mathbf{y}, \qquad (1.20)
\end{aligned}$$

[6]For notational convenience we assume componentwise multiplication of vectors in the last step of Eq. (1.20).

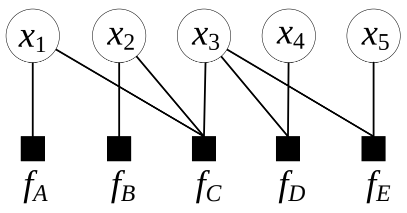


Figure 1.3: Example factor graph

where $\tilde{\phi}^c_{\bar{c}}(\mathbf{x}, y_{\bar{c}}) := \phi_c(\mathbf{x}, y_c)$ if $c = \bar{c}$, and a vector of ones otherwise. Note that both (1.18) and (1.20) are in the sum-product form, which can be computed exactly by belief propagation (BP) (*e.g.*, MacKay, 2003, Chapter 26). Important data structures such as junction trees (Lauritzen, 1996) and factor graphs (Kschischang et al., 2001) have been proposed to formalize the dynamic programming based on the sum-product form, and to apply the generalized distributive law (Kschischang et al., 2001). These algorithms usually have time complexity $O(m\,|\mathcal{Y}|^{w+1})$, where m is the number of nodes and w is the tree width of the graph, *i.e.*, the size of its largest clique minus 1 after the graph is optimally triangulated. For trees and 1-D CRFs (chains), $w = 1$, so that calculating (1.20) directly is feasible. However, for more general cases like 2-D grid CRFs, the tree width w is prohibitively high, and one has to resort to approximate approaches.

1.5.2 Message passing

Message passing schemes essentially propagate local factor information to the other factors and try to achieve global consistency via enforcing local consistency. These algorithms can be most conveniently described by using factor graphs.

Definition 21 (Factor graph) *(Kschischang et al., 2001) Given a* pdf *which factorizes onto groups of nodes: $p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \varphi_c(x_c)$, its factor graph is defined as a bipartite graph, where one side consists the original nodes and the other side consists of the factors given by the prescribed factorization of p. A node i is linked with a factor c if, and only if, i is involved in the factor c ($i \in c$).*

For example, given a joint distribution:

$$p(x_1, \ldots, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5),$$

the corresponding factor graph is Figure 1.3. It is clear that all graphical models can be represented by a factor graph, whose factors are subsets of the maximal cliques.

Interpreting c as the set of nodes associated with factor c, one can define the scheme called belief propagation (BP) which consists of two types of message passing on the factor graph:

$$\begin{aligned} \text{variable } i \in c \text{ to factor } c\text{:} \quad & m_{i \to c}(x_i) = \prod_{c' : i \in c', c' \neq c} m_{c' \to i}(x_i), \\ \text{factor } c \text{ to variable } i \in c\text{:} \quad & m_{c \to i}(x_i) = \sum_{x_{c \setminus \{i\}}} \left(f(x_i, x_{c \setminus \{i\}}) \prod_{j \in c \setminus \{i\}} m_{j \to c}(x_j) \right), \end{aligned}$$

and the final marginal distribution can be obtained by $p(x_c) := \prod_{i \in c} m_{i \to c}(x_i)$ up to a normalization constant. By replacing the above sum-product with max-product, the same scheme can be used for MAP inference. This is the idea of generalized distributive law with different semi-rings (Kschischang et al., 2001).

BP is guaranteed to converge on graphs with at most one loop (Weiss, 2000) or when the joint distribution is Gaussian with arbitrary topology (Weiss, 2001). Unfortunately, when there is more than one loop, no guarantee can be made on convergence, or convergence to the true marginal. Ihler et al. (2005) provided some convergence analysis and conditions using contraction of dynamic range. In general, it is still an open issue although loopy BP often performs well in practice.

A major progress in message passing inference was made by Minka (2001), called expectation propagation (EP). In a nutshell, it approximates all the factors $f_c(x_c)$ with some restricted (simple) forms $\tilde{f}_c(x_c)$ such as product of independent Gaussians, so that the inference on the joint approximation $\{\tilde{f}_c(x_c)\}_c$ is tractable. The approximation criteria is to optimize the KL divergence between the given *pdf* $q \propto \prod_c f_c(x_c)$ and the approximant $p \propto \prod_c \tilde{f}_c(x_c)$. If the approximant is restricted to exponential families, this is equivalent to moment matching. For computational tractability, a cavity update scheme is employed, *i.e.*, cycle through all the factors and each time optimize the factor's approximation in the context of other factors' current approximation. Here we sketch some technical details because EP will be used extensively in Chapter 3, and the full details can be found in (Minka, 2001).

Suppose we have a pre-specified exponential family $\mathcal{P}_{\boldsymbol{\phi}}$ for which efficient inference is available. Now we are given an arbitrary *pdf* q and inference is intractable on it. A natural idea is to approximate q by some distribution $p(\mathbf{x}; \boldsymbol{\theta}) \in \mathcal{P}_{\boldsymbol{\phi}}$, and then simply use the marginals and partition functions etc of $p(\mathbf{x}; \boldsymbol{\theta})$ as the surrogate of those of q. The above approximation can be in the sense of projecting q to $\mathcal{P}_{\boldsymbol{\phi}}$ in KL divergence:

$$\min_{\boldsymbol{\theta} \in \Theta} \mathrm{KL}(q || p(\mathbf{x}; \boldsymbol{\theta})) \;\Leftrightarrow\; \min_{\boldsymbol{\theta} \in \Theta} \mathrm{KL}(q || \exp(\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta}))).$$

Taking gradient wrt $\boldsymbol{\theta}$ and equating to 0 yield the optimality condition:

$$\mathbb{E}_{\mathbf{x} \sim q}[\boldsymbol{\phi}(\mathbf{x})] = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}; \boldsymbol{\theta})}[\boldsymbol{\phi}(\mathbf{x})],$$

which means matching the expectation of features.

Now suppose the distributions have graphical model structures, and q factorizes as

$$q(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} f_c(x_c),$$

and now we naturally wish to project into $\mathcal{P}_{\boldsymbol{\phi}}$ where $\boldsymbol{\phi}$ factorizes into $\mathrm{vec}_{c \in \mathcal{C}} \{\phi_c(x_c)\}$:

$$\min_{\boldsymbol{\theta} \in \Theta} \mathrm{KL}(q||p(\mathbf{x}; \boldsymbol{\theta})) \;\Leftrightarrow\; \min_{\boldsymbol{\theta} \in \Theta} \mathrm{KL}\left(\frac{1}{Z} \prod_{c \in \mathcal{C}} f_c(x_c) \middle|\middle| \exp\left(\sum_{c \in \mathcal{C}} \langle \boldsymbol{\phi}_c(x_c), \theta_c \rangle - g(\boldsymbol{\theta}) \right) \right). \tag{1.21}$$

Although the result of moment matching still holds, it is now intractable to compute the moment in general and in fact this is the problem we want to tackle in the first place. This obstacle also precludes clique-wise block coordinate descent. Despite the computational feasibility of matching the moment clique by clique independently, it does not give good approximations unless all the cliques are disjoint.

Let us first ignore the normalizer and write $\tilde{f}_c(x_c; \theta_c) := \exp(\langle \phi_c(x_c), \theta_c \rangle$. EP takes a cavity approach (Opper & Winther, 2000): cycle through all the cliques, and for each clique c, find the best approximant $\tilde{f}_c$ of f_c keeping the other current approximants $\tilde{f}_{c'}$ $(c' \neq c)$ fixed, *i.e.*

$$\min_{\theta_c} \;\mathrm{KL}\left(f_c(x_c) \prod_{c' \neq c} \tilde{f}_{c'}(x_{c'}; \theta_{c'}) \middle|\middle| \tilde{f}_c(x_c; \theta_c) \prod_{c' \neq c} \tilde{f}_{c'}(x_{c'}; \theta_{c'}) \right).$$

Since only one factor from q, f_c, is involved, this optimization over θ_c is feasible via moment matching. Different algorithms can be derived by further assuming different forms of the exponential family. For example, loopy belief propagation can be recovered when each ϕ_c completely decomposes onto individual nodes: $\phi_c(x_c) = \mathrm{vec}_{i \in c}\, \phi_{c,i}(x_i)$. The normalization factor can be obtained by matching the zero-th order moment, and is usually done after the above cyclic procedure terminates.

Unfortunately, EP still has no convergence guarantee and even when it converges, there is no guarantee that it will give the correct inference results. Again it works pretty well in practice, and a theoretical analysis is available in (Minka, 2005) which also provides unified comparisons with some other inference algorithms.

A simplified version of EP takes only one pass through the factors in q. This is known as assumed density filtering (ADF) (Maybeck, 1982; Opper, 1998), and is useful for online learning where factors are revealed in a sequence and must be discarded before the next factor arrives (for privacy or storage constraint). In general, the accuracy of ADF is inferior to EP and is susceptible to the order in which the factors are revealed.

1.5.3 Sampling

In many applications, the probability p is used to compute the expectation of some function f. Sampling tries to obtain a set of samples $\{\mathbf{x}^i\}$ (typically *iid* from p) to approximate $\mathbb{E}_p[f] := \int f(\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}$ by $\frac{1}{N}\sum_{i=1}^N f(\mathbf{x}^i)$. In many cases, *iid* samples are hard or expensive to draw, hence Markov chain Monte Carlo (MCMC) methods were introduced which asymptotically approach *iid* samples, *e.g.*, Metropolis-Hastings and Gibbs sampling. There is a large volume of literature on sampling methods for machine learning such as (Doucet et al., 2001) and (Andrieu et al., 2003) and the references therein.

Sampling from an undirected graphical model is not easy except for tree structured graphs, and one general purpose inference engine is Gibbs sampling. After randomly initializing the state of all the nodes to $X_1^{(0)}, \ldots, X_n^{(0)}$, one randomly picks a node X_i and sample its next state conditioned on the current state of all the other nodes $X_i^{(1)} \sim p(X_i|\{X_j^{(0)}; j \neq i\})$. Keep the state $X_j^{(1)} = X_j^{(0)}$ for all $j \neq i$. Next we randomly pick a node again and sample its next state conditioned on the rest nodes' current state. This procedure can be run for ever and will give asymptotically independent samples of the joint distribution. Gibbs sampling has been implemented in the the BUGS (Bayesian inference Using Gibbs Sampling) package, which provides MCMC inference engines for complex statistical models with significant flexibility.

1.5.4 Variational inference

Variational methods refer to the technique of posing some quantities hard to compute as the minimal value of some functions, and then apply optimization algorithms to it. For example, the solution of the linear system $A\mathbf{x} = \mathbf{b}$ is exactly the minimizer of $\frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \langle \mathbf{b}, \mathbf{x}\rangle$ if A is positive definite. For unconstrained quadratics, algorithms such as conjugate gradient (Hestenes & Stiefel, 1952) can optimize it very efficiently.

In the same spirit, Wainwright (2002) formulated the log partition function as the minimum of a certain function with some constraints, and its minimizer is exactly the feature mean. This new framework allows direct application of a large body of optimization techniques, which can be further accelerated by utilizing the structure of the graph (*e.g.*, Wainwright et al., 2003, 2005; Sontag & Jaakkola, 2007). Intuitively, the key idea is the Fenchel-Young equality

$$g(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}} \langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle - g^\star(\boldsymbol{\mu})$$

where $g^\star$ is the Fenchel dual of g. Now three questions arise: a) what is the domain of $g^\star$, b) how to compute $g^\star$, c) how to carry out the optimization. We will answer the first two questions in the next part, and then survey some approximate algorithms for optimization.

Marginal polytope and $g^\star$

Wainwright & Jordan (2008, Theorem 3.4) showed that the domain of $g^\star$ is related to the range of the expectation of $\boldsymbol{\phi}(x)$ wrt all distributions that are absolutely continuous wrt ν.

Definition 22 (Marginal polytope) *Define the marginal polytope of $\boldsymbol{\phi}$ as*

$$\mathcal{M}_{\boldsymbol{\phi}} := \left\{ \boldsymbol{\mu} \in \mathbb{R}^d : \exists\, p(\cdot), \, s.t. \int \boldsymbol{\phi}(x) p(x) \nu(\mathrm{d}x) = \boldsymbol{\mu} \right\}.$$

Note that the *pdf* p in the definition is not required to be in the exponential family $\mathcal{P}_{\boldsymbol{\phi}}$, however, adding this restriction is straightforward. Given $\boldsymbol{\theta} \in \Theta$, we are interested in the expectation of $\boldsymbol{\phi}(x)$ under $p(x; \boldsymbol{\theta})$, and formally we define a mapping $\Lambda_{\boldsymbol{\phi}} : \Theta \mapsto \mathcal{M}$ as

$$\Lambda_{\boldsymbol{\phi}}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(x)] = \int \boldsymbol{\phi}(x) p(x; \boldsymbol{\theta}) \nu(\mathrm{d}x).$$

And then we have the range of $\Lambda_{\boldsymbol{\phi}}$ mapping from Θ, *i.e.* the space of mean parameters wrt $\mathcal{P}_{\phi}$:

$$\Lambda_{\boldsymbol{\phi}}(\Theta) := \left\{ \int \boldsymbol{\phi}(x) p(x) \nu(\mathrm{d}x) : p \in \mathcal{P}_{\phi} \right\}.$$

$\mathcal{M}_{\boldsymbol{\phi}}$ is obviously convex, while $\Lambda_{\boldsymbol{\phi}}(\Theta)$ is not necessarily convex. Hence we call $\mathcal{M}_{\boldsymbol{\phi}}$ marginal polytope. Also, neither $\mathcal{M}$ nor $\Lambda_{\boldsymbol{\phi}}(\Theta)$ is guaranteed to be closed. When $\boldsymbol{\phi}$ is clear from context, we omit the subscript $\boldsymbol{\phi}$ in $\mathcal{M}_{\boldsymbol{\phi}}$, $\Lambda_{\boldsymbol{\phi}}(\boldsymbol{\theta})$, and $\Lambda_{\boldsymbol{\phi}}(\Theta)$. $\Lambda(\Theta)$ and $\mathcal{M}$ are related as follows.

Proposition 23 *(Wainwright & Jordan, 2003, Theorem 1) The mean parameter mapping Λ is onto the relative interior of $\mathcal{M}$,* i.e., *$\Lambda(\Theta) = \mathrm{ri}\mathcal{M}$.*

The mean parameter $\boldsymbol{\mu} = \mathbb{E}_{x \sim p}[\boldsymbol{\phi}(x)]$ can be roughly considered as a signature of the density p. The following is an important theorem which provides an explicit form of the Fenchel dual of log partition function, in terms of the entropy of the distribution.

Theorem 24 (Fenchel dual of $g(\boldsymbol{\theta})$ and entropy) *(Wainwright & Jordan, 2003, Theorem 2) For any $\boldsymbol{\mu} \in \mathrm{ri}\mathcal{M}$, let $\boldsymbol{\theta}(\boldsymbol{\mu})$ denote an element in $\Lambda^{-1}(\boldsymbol{\mu})$. Denote as $H(p)$ the entropy of* pdf *p. The Fenchel-Legendre dual of $g(\boldsymbol{\theta})$ has the form*

$$g^\star(\boldsymbol{\mu}) = \begin{cases} -H(p(x; \boldsymbol{\theta}(\boldsymbol{\mu}))) & \text{if } \boldsymbol{\mu} \in \mathrm{ri}\mathcal{M} \\ +\infty & \text{if } \boldsymbol{\mu} \notin \mathrm{cl}\mathcal{M} \end{cases}.$$

$$\begin{tikzcd}
\boldsymbol{\theta}_1 \arrow[dr] & & \\
& \mathbf{P} \arrow[r] & \mu[\mathbf{P}] \\
\boldsymbol{\theta}_2 \arrow[ur] & &
\end{tikzcd}$$

Figure 1.4: Simple break of injectivity

$$\begin{tikzcd}
\boldsymbol{\theta}_1 \arrow[r] & \mathbf{P} \arrow[dr] & \\
& & \mu[\mathbf{P}] = \mu[\mathbf{Q}] \\
\boldsymbol{\theta}_2 \arrow[r] & \mathbf{Q} \arrow[ur] &
\end{tikzcd}$$

Figure 1.5: Imaginary break of injectivity

For any boundary point $\boldsymbol{\mu} \in \mathrm{bd}\mathcal{M} := \mathrm{cl}\mathcal{M} \backslash \mathrm{ri}\mathcal{M}$, *we have* $g^\star(\boldsymbol{\mu}) = \lim_{n\to\infty} -H(p(x;\theta(\boldsymbol{\mu}^n))$ *taken over a sequence* $\{\boldsymbol{\mu}^n\} \subseteq \mathrm{ri}\mathcal{M}$ *converging to* $\boldsymbol{\mu}$.

This theorem also implies that given the mean parameter $\boldsymbol{\mu}$, the entropy of the distribution is independent of which natural parameter is used from $\Lambda^{-1}(\boldsymbol{\mu})$. Considering that both $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ can serve as a signature of the distribution, it is natural to investigate their relationship which turns out to hinge on the minimality of sufficient statistics.

Theorem 25 (Injectivity of mean mapping) *(Wainwright & Jordan, 2008, Proposition 3.2) The mean map* Λ *is one-to-one if, and only if,* $\boldsymbol{\phi}(x)$ *is minimal.*

Since $\boldsymbol{\theta}$ is mapped to the marginal polytope $\mathcal{M}$ via the *pdf* $p(x;\boldsymbol{\theta})$, injectivity can break in two different ways: a) two different natural parameters giving the same distribution, see Figure 1.4; and b) different distributions in the exponential family giving the same mean, see Figure 1.5. The minimality assumption seems to preclude only the first case. Fortunately, it turns out that this second map $\mathcal{P}_{\boldsymbol{\phi}} \mapsto \mathcal{M}$ is injective irrespective of whether the sufficient statistics are minimal.

Theorem 26 *Using the same notation as in Theorem 25, the mapping from distribution* $p \in \mathcal{P}_{\boldsymbol{\phi}}$ *to the mean* $\mathbb{E}_{x\sim p}[\boldsymbol{\phi}(x)]$ *is injective regardless of whether* $\boldsymbol{\phi}(x)$ *is minimal.*

Proof The proof is based on the maximum entropy interpretation of exponential families. Suppose two *pdf*s $p, q \in \mathcal{P}_{\boldsymbol{\phi}}$ have the same mean $\boldsymbol{\mu}$. Let the *pdf* p^* (not necessarily in $\mathcal{P}_\phi$) be the optimal solution of the following optimization problem:

$$\underset{p}{\text{maximize}} \;\; H(p), \qquad \text{s.t. } \mathbb{E}_{x\sim p}[\boldsymbol{\phi}(x)] = \boldsymbol{\mu}. \tag{1.22}$$

Note the optimization is *not* restricted to $\mathcal{P}_{\boldsymbol{\phi}}$, and the feasible region must be nonempty since p and q satisfy the constraint. Since entropy is a strictly convex functional and the linear constraints form a convex set, the optimal solution p^* is unique and is well known to be in $\mathcal{P}_{\boldsymbol{\phi}}$. As the entropy of exponential family distributions can be fully determined by its mean (Theorem 24), p and q must have the same entropy as p^*. Hence they are also the optimal solutions to the problem 1.22. Then the uniqueness of solution implies $p = q = p^*$. ■

The significance of this theorem is that the mean of sufficient statistics uniquely identifies the distribution in the exponential family, and furthermore if the sufficient statistics are minimal, the natural parameter can also be uniquely identified.

Optimization techniques for variational inference

By Theorem 24, $g^\star(\boldsymbol{\mu})$ is just the negative entropy of the distribution corresponding to $\boldsymbol{\mu}$. However, the hardness of the optimization problem $g(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu}\in\mathcal{M}} \langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle - g^\star(\boldsymbol{\mu})$ is exhibited in two folds: a) the constraint set $\mathcal{M}$ is extremely difficult to characterize explicitly; b) the negative entropy $g^\star$ is defined indirectly, hence it lacks explicit form in $\boldsymbol{\mu}$. Therefore, one resorts to outer or inner bounds of $\mathcal{M}$ and upper or lower bounds of $g^\star$. This leads to various algorithms (Wainwright & Jordan, 2008), such as

- Naive mean field. It only considers a subset (inner approximation) of $\mathcal{M}$ where the mean parameter of the edges is fixed to be the product of the mean of the two end points. This essentially assumes that all the nodes are independent, and yields a lower bound on $g(\boldsymbol{\theta})$. In this case, the entropy factorizes and becomes easy to compute.

- Tree-reweighted sum-product. By noticing that the entropy of trees can be computed efficiently, Wainwright et al. (2005) studied the restriction of any mean parameter $\boldsymbol{\mu}$ to a spanning tree T: $\boldsymbol{\mu}(T)$. Since this restriction removes those constrains corresponding to ignored edges, the entropy of $\boldsymbol{\mu}(T)$ is higher than that of $\boldsymbol{\mu}$, hence the restriction leads to a concave upper bound of $\langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle - g^\star(\boldsymbol{\mu})$. Moreover, as convex combination of upper bounds is still an upper bound, it can be tightened by further optimizing over the convex combination, *e.g.*, the distribution over spanning trees called spanning tree polytope.

- Log-determinant relaxation. Observing that $\mathcal{M}$ can be characterized by constraining the moments to be positive semi-definite to any order, Wainwright & Jordan (2006) proposed a relaxation based on Gaussian approximation.

- Cutting plane. Sontag & Jaakkola (2007) proposed a new class of outer bounds on the marginal polytope, by drawing its equivalence with the cut polytope (Barahona & Mahjoub, 1986). Different from most previous methods which fix the outer bound a priori, Sontag & Jaakkola (2007) progressively tightens the outer bound according to the current infeasible solution. This is done by efficiently finding *a* violated constraint via a series of projections onto the cut polytope.

1.6 Regularized risk estimation and optimizations

Parameter estimation is a key task in machine learning. Eq. (1.17) shows the maximum likelihood estimation for CRFs. Suppose we have a set of feature/label pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ drawn *iid* from some underlying joint distribution. If we endow a Gaussian prior on the parameter $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\Sigma)$, then we get a maximum aposterior estimation:

$$\begin{aligned} & \operatorname*{argmax}_{\boldsymbol{\theta}} \log p\left(\{y_i\}_{i=1}^n \,|\, \{\mathbf{x}_i\}_{i=1}^n ; \boldsymbol{\theta}\right) p(\boldsymbol{\theta}) \\ \Longleftrightarrow \quad & \operatorname*{argmin}_{\boldsymbol{\theta}} - \log p\left(\{y_i\}_{i=1}^n \,|\, \{\mathbf{x}_i\}_{i=1}^n ; \boldsymbol{\theta}\right) + \frac{\lambda}{2}\boldsymbol{\theta}^\top \Sigma^{-1} \boldsymbol{\theta} \\ \overset{iid}{\Longleftrightarrow} \quad & \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^n - \log p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) + \frac{\lambda}{2}\boldsymbol{\theta}^\top \Sigma^{-1} \boldsymbol{\theta} \qquad (1.23) \\ \text{where} \quad & - \log p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) = - \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i), \boldsymbol{\theta} \rangle + \log \sum_{\bar{y}_i} \exp\left(\langle \boldsymbol{\phi}(\mathbf{x}_i, \bar{y}_i), \boldsymbol{\theta}\rangle\right). \qquad (1.24) \end{aligned}$$

Although $p(y_i|\mathbf{x}_i; \boldsymbol{\theta})$ is a reasonable likelihood for regression problems, reconsideration is needed for classification tasks. The objective in Eq. (1.23) is essentially a trade-off between the prior and the likelihood. However for classification, the label is determined by $\operatorname{argmax} p(y|\mathbf{x}_i; \boldsymbol{\theta})$. Therefore, it is no longer the case that the higher the likelihood $p(y_i|\mathbf{x}_i; \boldsymbol{\theta})$ the better: we only need $p(y_i|\mathbf{x}_i; \boldsymbol{\theta})$ to exceed the probability of all other labelings, and the effort saved from increasing $p(y_i|\mathbf{x}_i; \boldsymbol{\theta})$ can be used for optimizing the prior. Technically, we only need to redefine the negative log likelihood into:

$$-\delta(y_i = \operatorname*{argmax}_y p(y|\mathbf{x}_i; \boldsymbol{\theta})) = -\delta(p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) > p(y|\mathbf{x}_i; \boldsymbol{\theta}) \ \forall y \neq y_i) \qquad (1.25)$$

and the maximum a posterior estimator becomes:

$$\operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^n -\delta\left(p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) > p(y|\mathbf{x}_i; \boldsymbol{\theta}) \ \forall y \neq y_i\right) + \frac{\lambda}{2}\boldsymbol{\theta}^\top \Sigma^{-1} \boldsymbol{\theta}. \qquad (1.26)$$

Unfortunately, this objective function is not continuous which makes the optimization hard. One common bypass is to replace the negative log likelihood by a convex upper bound, *e.g.*

$$\max\left\{0, 1 - \min_{y \neq y_i} \frac{p(y_i|\mathbf{x}_i; \boldsymbol{\theta})}{p(y|\mathbf{x}_i; \boldsymbol{\theta})}\right\} = \max\left\{0, 1 - \min_{y \neq y_i} \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \boldsymbol{\theta}\rangle\right\} \qquad (1.27)$$

which, intuitively speaking, encourages that the odd ratio between the true label and all other labels be greater than 1. Otherwise a penalty is incurred. Now the estimator

$$\operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \max \left\{ 0, 1 - \min_{y \neq y_i} \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \boldsymbol{\theta} \rangle \right\} + \frac{\lambda}{2} \boldsymbol{\theta}^\top \Sigma^{-1} \boldsymbol{\theta} \tag{1.28}$$

becomes a continuous and convex optimization problem in $\boldsymbol{\theta}$.

In summary, different from regression, classification problems need redefinitions of likelihood. From Eq. (1.23), (1.26) and (1.28), we can see that the estimation becomes an optimization problem whose objective takes the form of:

$$R_{\text{emp}}(\boldsymbol{\theta}) + \lambda \Omega(\boldsymbol{\theta}) = \sum_{i=1}^{n} l(\mathbf{x}_i, y_i; \boldsymbol{\theta}) + \lambda \Omega(\boldsymbol{\theta}),$$

where $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \Sigma^{-1} \boldsymbol{\theta}$, and $l(\mathbf{x}_i, y_i; \boldsymbol{\theta})$ can have different forms such as Eq. (1.24), (1.25), and (1.27). Theoretically, this formulation can be as well interpreted from the statistical learning perspective (Vapnik, 1995), where $\Omega(\boldsymbol{\theta})$ is the *regularizer* and $R_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} l(\mathbf{x}_i, y_i; \boldsymbol{\theta})$ is the *empirical risk*. Intuitively, the empirical risk quantifies the discrepancy between the true label y_i and the prediction for example $\mathbf{x}_i$ using the model parameter $\boldsymbol{\theta}$. The regularizer, on the other hand, measures how *complex* the model is, and *simple* models are preferred. This intuition was known as Occam's razor (among other names), and has been solidly justified in theory, *e.g.* Tikhonov regularization (Tikhonov, 1943, 1963), Vapnik-Chervonenkis dimension and structural/regularized risk minimization (Vapnik, 1995), entropy or covering numbers (Guo et al., 1999), and minimum description length (Grünwald, 2007).

This decomposition of empirical risk and regularization will motivate the general framework of regularized risk minimization in Section 1.6.1, and we will demonstrate how it encompasses many important machine learning algorithms. Section 1.6.2 will survey various algorithms which optimize this functional. A specific general purpose solver, cutting plane method, will be introduced in Section 1.6.3, and some major improvements will be detailed in Section 1.6.4, especially the bundle method for machine learning (BMRM). To comply with the common notations in statistical learning theory, we will change $\boldsymbol{\theta}$ to $\mathbf{w}$, meaning weight vector which also makes sense for exponential families because the natural parameter does specify a weight on the sufficient statistics.

1.6.1 Regularized risk minimization

Besides the quadratic regularizer in Eq. (1.23), many other measures of model complexity exist. For example the L_1 norm $\|\mathbf{w}\|_1 := \sum_i |w_i|$ encourages sparse solution where many w_i are zero meaning the corresponding features are unimportant (Tibshirani, 1996; Candes & Tao, 2005). Entropy or relative entropy is also commonly used when

Table 1.2: Loss for multi-class classification

Name	Definition
"0-1" loss	$\delta(\text{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{\phi}(\mathbf{x}_i, y), \mathbf{w} \rangle) \neq y_i)$
hinge loss	$\max \{0, 1 - \min_{y \neq y_i} \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \mathbf{w} \rangle \}$ $= \max_{y \in \mathcal{Y}} \{ \langle \boldsymbol{\phi}(\mathbf{x}_i, y) - \boldsymbol{\phi}(\mathbf{x}_i, y_i), \mathbf{w} \rangle + \delta(y = y_i) \}$
logistic loss	$- \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i), \mathbf{w} \rangle + \log \sum_{\bar{y}_i} \exp (\langle \boldsymbol{\phi}(\mathbf{x}_i, \bar{y}_i), \mathbf{w} \rangle)$

Table 1.3: Loss for binary classification

Name	Definition
"0-1" loss	$\delta(\text{sign}(\langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w} \rangle) \neq y_i)$
hinge loss	$\max \{0, 1 - y_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w} \rangle \}$
logistic loss	$\log(1 + \exp(-y_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w} \rangle))$
exponential loss	$\exp(-y_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w} \rangle)$

$\mathbf{w}$ corresponds to a distribution on the features, and this prior encourages a uniform distribution. It is noteworthy that the L_2 norm and entropy are strongly convex and smooth while L_1 norm is just convex but not strongly convex or differentiable.

On the other hand, empirical risk also admits a wide range of choice. In the simplest case of the statistical query model (Kearns, 1998), it can be decomposed additively to the loss on individual training examples $l(\mathbf{x}_i, y_i; \mathbf{w})$. Examples include

- Logistic loss as in Eq. (1.24) named in analogy to logistic regression,
- 0-1 loss as in Eq. (1.25) which simply checks whether the output label is correct,
- Hinge loss as in Eq. (1.27) which looks at all the incorrect labels and encourages their discriminant values to be less than the correct label's value by at least 1 (margin).

We summarize these losses in Table 1.2.

When specialized to binary classification with $y \in \{-1, 1\}$, the above definitions can be simplified by letting $\boldsymbol{\phi}(\mathbf{x}_i, y) := y\boldsymbol{\phi}(\mathbf{x}_i)/2$, and are summarized in Table 1.3.

All the four losses in Table 1.3 for binary classification are plotted in Figure 1.6. Exponential loss is used in boosting (Hastie et al., 2009, Section 10.4). Hinge loss leads to maximum margin models, and the commonly used support vector machine (SVM) for binary classification is simply a combination of hinge loss and L_2 regularization. Notice that "0-1" loss is neither convex nor continuous. Hinge loss is convex and continuous but not differentiable at one point. Logistic loss and exponential loss are both smooth,

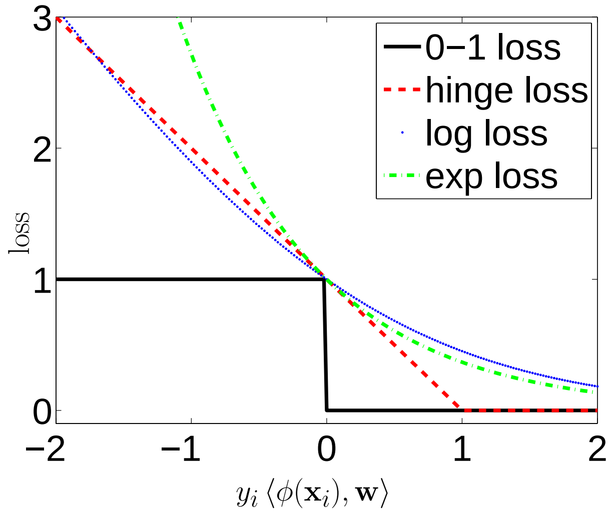


Figure 1.6: 0-1 loss, hinge loss, logistic loss and exponential loss for binary classification.

strongly convex, and have Lipschitz continuous gradient on any compact subset of $\mathbb{R}$. Historically, although 0-1 loss was the real objective that one wants to minimize, its discontinuity and nonconvexity prompted people to use other convex upper bounds as surrogates for easier optimization. The statistical consequences of these surrogates are under research, *e.g.* (Bartlett et al., 2006).

More general loss functions can be defined for regression, ranking, novelty detection, *etc.*. In the case of multi-class classification, the hinge loss defined above can be generalized in two ways which can be summarized by

$$\max_{y \in \mathcal{Y}} \left\{ \rho(y, y_i) \left[\langle \boldsymbol{\phi}(\mathbf{x}_i, y) - \boldsymbol{\phi}(\mathbf{x}_i, y_i), \mathbf{w} \rangle + \Delta(y, y_i) \right] \right\}.$$

Here $\Delta(y, y_i)$ gives a more refined comparison between the proposed label y and the correct label y_i, characterizing *to what extent* the proposed label is wrong. This is much more informative than $\delta(y = y_i)$ which merely checks whether the labeling is correct. For instance, when the output space is a sequence, $\Delta(y, y_i)$ can be the Hamming distance. Path distances (Dekel et al., 2004) or H-loss (Cesa-Bianchi et al., 2006) can also be used when the output space has hierarchies or ontology. $\rho(y, y_i)$ yields a similar effect of penalizing different mistakes differently, but in a different way from $\Delta(y, y_i)$. This can be best illustrated by using two concrete examples: a) margin rescaling where $\rho(y, y_i) = 1$ and $\Delta(y, y_i) = 2$, and b) slack rescaling where $\rho(y, y_i) = 2$ and $\Delta(y, y_i) = 1$:

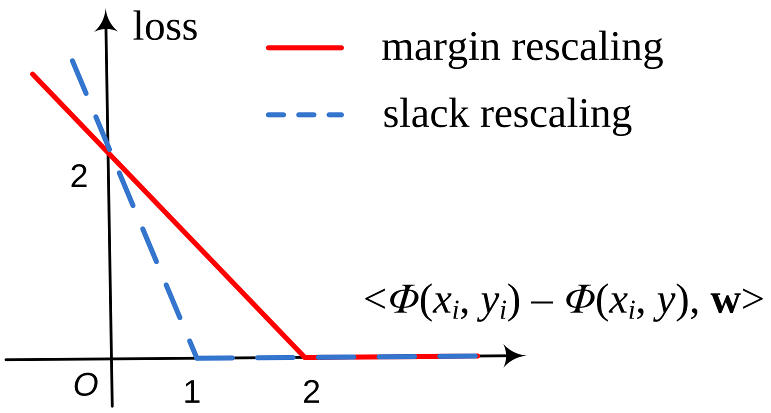


Figure 1.7: Slack rescaling and margin rescaling.

Name	Example	Proposed by
margin rescaling	$\max\{0, 2 - \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \mathbf{w}\rangle\}$	(Taskar et al., 2004)
slack rescaling	$2\max\{0, 1 - \langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \mathbf{w}\rangle\}$	(Tsochantaridis et al., 2005)

Plotting these two rescalings in Figure 1.7, we can see that the margin rescaling starts to penalize early but mildly: once $\langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \mathbf{w}\rangle$ falls below 2, it starts to incur a unit loss for each unit gap. In contrast, slack rescaling starts to penalize only after $\langle \boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y), \mathbf{w}\rangle$ falls below 1, but once it kicks in, the penalty is severe: two units for each unit gap.

When the output space $\mathcal{Y}$ is equipped with a graphical model, the sufficient statistics $\boldsymbol{\phi}$ decomposes, and furthermore Taskar et al. (2004) assumed the same factorization of $\Delta(y, y')$:

$$\Delta(\mathbf{y}, \mathbf{y}') = \sum_{c \in \mathcal{C}} \Delta_c(y_c, y'_c).$$

This factorization is crucial for efficient maximum margin estimation for structured data with margin rescaling. We will revisit it in Section 5.4.1.

Finally, non-decomposable loss functions are also common, especially in applications like information retrieval. For example, the F-score and area under ROC curve (Joachims, 2005). Optimization for these multivariate performance measures is nonconvex and harder, and we will introduce an approximate method for optimizing F-score in Chapter 3.

To summarize, from the examples above we can abstract out the *regularized risk* *estimation* framework (RRM):

$$\min_{\mathbf{w}} J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R_{\text{emp}}(\mathbf{w}), \quad \text{where } R_{\text{emp}}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{x}_i, y_i; \mathbf{w}).$$

Here $\Omega(\mathbf{w})$ is the regularizer and $R_{\text{emp}}(\mathbf{w})$ is the empirical risk. l is a loss function measuring the discrepancy between the true label y_i and the output of model $\mathbf{w}$. We will consider the optimization for this framework of problems, with special focus on strongly convex $\Omega(\mathbf{w})$ and convex (nonsmooth) R_{emp}. This makes optimization relatively simple (Boyd & Vandenberghe, 2004), and allows one to focus on modeling without being entangled with numerical stability or suboptimal solutions due to local minima. In addition, this assumption is not too restrictive as we have shown above that a large number of machine learning models do fit in this framework.

1.6.2 Survey of existing optimization algorithms

With the RRM model well established, the next challenge is to find the optimizer $\mathbf{w}^* := \operatorname{argmin}_{\mathbf{w}} J(\mathbf{w})$. Most existing solvers are iterative: generate a trace of weights $\mathbf{w}_1, \mathbf{w}_2, \ldots$ which approaches the solution $\mathbf{w}^*$.

In general, a solver is evaluated against the following criteria:

1. **Rate of convergence.** Each $\mathbf{w}_k$ incurs a gap in function value $\epsilon_k := J(\mathbf{w}_k) - J(\mathbf{w}^*)$. For any given precision/tolerance $\epsilon > 0$, we are interested in how many steps/iterations are needed before ϵ_k can be reduced to less than ϵ:

$$\epsilon_k := J(\mathbf{w}_k) - J(\mathbf{w}^*) < \epsilon.$$

   Typical rates include $k = O\left(\frac{1}{\epsilon^p}\right)$ $(p > 0)$ and $k = O\left(\log \frac{1}{\epsilon}\right)$ (called linear convergence[7]), and $k = O\left(\log\log \frac{1}{\epsilon}\right)$ (called quadratic convergence). Different variants of linear convergence also exist such as Q-linear and R-linear (Nocedal & Wright, 2006, pp. 619–620).

2. **Cost per iteration.** This cost includes all types of computing and storage resources, such as CPU time, memory, bus or hard drive IO, *etc.*. Comparison in this aspect is very case specific because different computing environments may have different resource bottlenecks, which may also vary with time.

3. **Generality.** Ideally a general solver is useful which can be applied to a wide range of problems, without being restricted to the form of the objective function or constraints. Granted that special purpose solvers can often perform better

[7]Not to be confused with $O\left(\frac{1}{\epsilon}\right)$ rate.

by exploiting the structure of the problem, generic solvers provide a reasonable off-the-shelf baseline which allows fast prototyping.

4. **Parallelization.** This could be subsumed in the above point 2. However we highlight it because the recent revolutionary development in parallel computing has ushered in a new era of multi-core. As the scale of machine learning applications is also growing rapidly, it will be crucial and interesting to develop learning algorithms which make full use of parallel facilities.

Using traditional optimizers

In theory, any general solver can be used for RRM. For example, linear programming (Vanderbei, 2008) can be used to solve L_1 regularized hinge loss. Interior point (IP) has been used by Koh et al. (2006) to solve L_1-Regularized logistic regression and by Ferris & Munson (2000) to train large scale SVMs. Andrew & Gao (2007) applied quasi-Newton methods for L_1 regularized log-linear models. Coordinate descent can also be used to train SVMs with linear convergence (Tseng & Yun, 2008). The main challenge in these methods is scalability: high dimension, highly nonsmooth objective, and a large number of constraints, resulting from the large number of data points and features. Therefore, they must be customized somehow to utilize the structures in the problem.

Using mild composite structure of machine learning objectives

Instead of directly applying general optimization methods which treat the objective function as a black box, one can slightly assume some general structure such as RRM. The bundle method for machine learning (BMRM) by Teo et al. (2007) is one effective algorithm that progressively builds a piecewise linear lower bound of the empirical risk, and solve the regularized model at each iteration. `SVMPerf` by Joachims (2005); Joachims et al. (2009) employs a similar cutting plane scheme, and can optimize multivariate performance measures which may not be decomposable. Tsochantaridis et al. (2005) finds the most violating constraints in each iteration for structured output data, and this greedy update also guarantees convergence at reasonable rate.

Solvers tailored for decomposable risk

If we further specialize to decomposable risk, then a lot of decomposition methods have been proposed in the past decade, most of which work for a specific loss/regularizer. Sequential minimal optimization (SMO) for binary nonlinear SVM optimizes two dual variables analytically in each iteration, and common ways of choosing the two variables implicitly require visiting the whole dataset (Platt, 1998; Keerthi & Gilbert, 2002). When training structured output data with decomposable loss, exponentiated gradient (Kivinen & Warmuth, 1995) is efficient and allows implicit clique-wise updates (Collins et al., 2008). Primal methods are also popular, often in the form of projected subgradient descent (Shor, 1985; Nedic, 2002; Bertsekas, 1976; Duchi & Singer, 2009).

(Stochastic) online learning. A direct consequence of decomposed risk is the possibility of learning sample by sample, called online learning as opposed to the traditional batch learning which visits the whole dataset in each iteration. When the dataset is large and the computing facility is relatively limited, a simple idea is to sample a subset of the dataset to train, and as a result all theoretical guarantees must be probabilistic. One extreme is that each update of the model uses only one training example. This scheme is also the only choice when the data points come in stream, and must be discarded before the next data point becomes available (due to privacy or storage constraints). Not surprisingly, as proved by Shalev-Shwartz et al. (2007) and Bottou & Bousquet (2007), stochastic online learning can reduce the regularized risk of binary SVM to any precision with reasonable confidence at a cost independent of the training set size. Shalev-Schwartz & Srebro (2008) further proved that in order to achieve any fixed generalization error, the runtime can be inversely proportional to the number of data points.

Online learning for binary SVM is a particularly fruitful research area. Online dual optimizers rely on the natural duality relationship between data points and dual variables, hence the well known convergence analyses of coordinate descent can be immediately applied (Luo & Tseng, 1992; Tseng & Yun, 2008, 2009). Hsieh et al. (2008a) proposed liblinear which performs dual coordinate Newton descent and enjoys linear convergence (Luo & Tseng, 1992). In fact, this method is closely related to Hildreth's QP algorithm (Hildreth, 1957; Iusem & Pierro, 1990), passive-aggressive method (Crammer et al., 2003), and implicit updates (Cheng et al., 2006). Primal methods such as SGD (Bottou & LeCun, 2004; Bordes et al., 2009) typically perform projected subgradient descent using approximate (*stochastic*) subgradients calculated from a random subset of the dataset. Its convergence (in probability) usually originates from the standard stochastic approximation theory (Tsypkin, 1971; Robbins & Monro, 1951). Another efficient primal stochastic gradient solver is pegasos proposed by Shalev-Shwartz et al. (2007), which guarantees that by drawing a constant number (can be 1) of random samples at each iteration, with probability $1-\delta$, $f(\mathbf{w}_k) - \min_{\mathbf{w}} f(\mathbf{w}) \leq O\left(\frac{\ln k}{k\delta}\right)$, *i.e.* $O(1/\epsilon)$ rate.

Online learning is often susceptible to the order of the samples, and rely heavily on randomization. Despite its popularity and effectiveness when data overwhelms computing power, the latest development of parallel computing is making it possible to efficiently visit the whole dataset. With the change of bottleneck, batch methods are projected to regain popularity, and now the key challenge in algorithm design becomes minimizing the sequential part of computation according to the Amdahl's law (Amdahl, 1967). Most existing parallel machine learning algorithms require additive decomposition of empirical risk, *i.e.* data parallelization. Examples include (Teo et al., 2007; Chu et al., 2007; Catanzaro et al., 2008; Graf et al., 2004).

Outlook

In Chapter 5, we will consider a new middle ground lying between RRM and decomposable loss:

$$J(\mathbf{w}) = \lambda\Omega(\mathbf{w}) + g^\star(A\mathbf{w}), \quad \text{where } A := (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top.$$

We assume Ω is strongly convex and g is convex with Lipschitz continuous gradient. Intuitively, we are assuming a linear predictor, *i.e.* the prediction for each example $\mathbf{x}_i$ is $\langle \mathbf{x}_i, \mathbf{w}\rangle$ and in such a case data parallelization is again straightforward. However, the overall empirical risk is now allowed to depend on the prediction of the examples via a general convex function $g^\star$. In terms of optimization, this class of objectives admit direct application of Nesterov's first-order methods which yield optimal rate of convergence (Nesterov, 1983, 2003, 2005a,b, 2007). Most closely related optimizers are the cutting plane methods and bundle methods, which we will detail in the next two sections. Lower bounds for these methods will be a central topic of Chapter 5.

1.6.3 Cutting plane

Cutting plane algorithms are based on the key property of convex functions: any closed convex function f can be written as the upper envelope of infinitely many minorizing affine functions:

$$f(\mathbf{w}) = \sup_{\mathbf{a},b} \left\{ \left\langle \mathbf{a}, \mathbf{w}' \right\rangle + b : \left\langle \mathbf{a}, \mathbf{w}' \right\rangle + b \le f(\mathbf{w}') \text{ for all } \mathbf{w}' \in \mathrm{dom} f \right\}.$$

Suppose somehow we have t points in dom f: $\{\mathbf{w}_i\}_{i=0}^{t-1}$, and subgradients $\mathbf{a}_{i+1} \in \partial f(\mathbf{w}_i)$ $(t \ge 0)$. Let $b_{i+1} = f(\mathbf{w}_i) - \langle \mathbf{a}_{i+1}, \mathbf{w}_i \rangle$ such that the hyperplane $(\mathbf{w}, \langle \mathbf{a}_{i+1}, \mathbf{w}\rangle + b_{i+1})$ is tangent to f at $\mathbf{w}_i$. Then we obtain a piecewise linear lower bound approximation of f which is exact at $\{\mathbf{w}_i\}_{i=0}^{t-1}$:

$$f_t^{\mathrm{cp}}(\mathbf{w}) := \max_{i\in[t]} f(\mathbf{w}_{i-1}) + \langle \mathbf{a}_i, \mathbf{w} - \mathbf{w}_{i-1} \rangle = \max_{i\in[t]} \langle \mathbf{a}_i, \mathbf{w} \rangle + b_i.$$

Then we optimize this piecewise linear approximant f_t^{cp} as a surrogate of f. Intuitively when one gets more and more $\mathbf{w}_i$ as t increases, f_t^{cp} will approximate f better and better, and $\min_{\mathbf{w}} f_t^{\mathrm{cp}}(\mathbf{w})$ will also approach $\min_{\mathbf{w}} f(\mathbf{w})$. Obviously the key challenge is how to pick $\mathbf{w}_i$ so that with as few $\{\mathbf{w}_i\}_{i=0}^t$ as possible, f_t^{cp} captures the nadir of f as well as possible.

Kelley (1960) and Cheney & Goldstein (1959) proposed a greedy scheme to progressively pick the landmark points $\mathbf{w}_t$:

$$\mathbf{w}_t := \operatorname*{argmin}_{\mathbf{w}\in \mathrm{dom} f} f_t^{\mathrm{cp}}(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \max_{i\in[t]} \langle \mathbf{a}_i, \mathbf{w}\rangle + b_i. \tag{1.29}$$

And f_t^{cp} is then updated by:

$$f_{t+1}^{\mathrm{cp}}(\mathbf{w}) := \max\left\{f_t^{\mathrm{cp}}(\mathbf{w}), \langle \mathbf{a}_{t+1}, \mathbf{w}\rangle + b_{t+1}\right\}.$$

The advantage of this algorithm is two folds. First, the optimization problem in Eq. (1.29) is simply a linear programming (as long as $\mathrm{dom} f$ is affine):

$$\begin{aligned} \mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w}\in \mathrm{dom} f} \min_{\xi\in\mathbb{R}} \xi \\ \text{s.t. } \langle \mathbf{a}_i, \mathbf{w}\rangle + b_i \le \xi \quad \forall i \in [t]. \end{aligned} \tag{1.30}$$

Second, the whole scheme is guaranteed to converge in finite time. In particular, define the gap

$$\bar{\epsilon}_t := \min_{i\in[t]} f(\mathbf{w}_i) - f_t^{\mathrm{cp}}(\mathbf{w}_t),$$

then for any pre-specified tolerance $\epsilon > 0$, there must be a finite T such that $\bar{\epsilon}_t < \epsilon$ for all $t > T$. $\bar{\epsilon}_t$ is observable (not requiring the knowledge of f^* or $\mathbf{w}^*$), and it is easy to see that $\bar{\epsilon}_t$ upper bounds the real gap because $f_t^{\mathrm{cp}}(\mathbf{w}) \le f(\mathbf{w})$ for all $\mathbf{w}$:

$$\bar{\epsilon}_t = \min_{i\in[t]} f(\mathbf{w}_i) - \min_{\mathbf{w}} f_t^{\mathrm{cp}}(\mathbf{w}) \ge \min_{i\in[t]} f(\mathbf{w}_i) - \min_{\mathbf{w}} f(\mathbf{w}) =: \epsilon_t.$$

However, cutting plane is also plagued with two major disadvantages. First, the complexity of the inner problem Eq. (1.30) grows with iterations as the linear programming gets more constraints. Second, and even worse, albeit the finite time convergence, the rate of convergence can be extremely slow. Given any arbitrary tolerance ϵ, (Hiriart-Urruty & Lemaréchal, 1993a, Example 1.1.2 of Chapter XV) shows an example, due to Nemirovski, where the cutting plane algorithm takes $t = O\left(\epsilon^{-n/2}\right)$ steps to reduce ϵ_t to less than ϵ. The cause of this phenomenon is the instable zigzagging trace of $\mathbf{w}_t$: the solution of the linear programming Eq. (1.30) is not unique, and $\mathbf{w}_t$ can drift far away from the previous $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$. Therefore stabilization techniques are desired,

Algorithm 1: BMRM

```
   Input: Tolerance $\epsilon > 0$, initial guess $\mathbf{w}_0$.
1  Initialize: $t \leftarrow 0$.
2  repeat
3      $t \leftarrow t + 1$
4      Compute $\mathbf{a}_t \in \partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_{t-1})$, $b_t \leftarrow R_{\text{emp}}(\mathbf{w}_{t-1}) - \langle \mathbf{w}_{t-1}, \mathbf{a}_t \rangle$.
5      Update model $R_t^{\text{cp}}(\mathbf{w}) := \max_{i \in [t]} \langle \mathbf{a}_i, \mathbf{w} \rangle + b_i$.
6      Define regularized model $J_t(\mathbf{w}) := \lambda \Omega(\mathbf{w}) + R_t^{\text{cp}}(\mathbf{w})$.
7      Update $\mathbf{w}_t \leftarrow \text{argmin}_{\mathbf{w}} J_t(\mathbf{w})$ using some inner solver like Algorithm 2 and 3.
8      Check $\epsilon_t \leftarrow \min_{0 \le i \le t} J(\mathbf{w}_i) - J_t(\mathbf{w}_t)$.
9  until $\epsilon_t \le \epsilon$
10 return $\mathbf{w}_t$
```

and the next section will introduce two different remedies.

1.6.4 Bundle methods for regularized risk minimization

A natural heuristic for stabilization is to penalize the displacement of $\mathbf{w}_t$ from $\mathbf{w}_{t-1}$:

$$\mathbf{w}_t := \underset{\mathbf{w}}{\text{argmin}}\, \lambda \left\| \mathbf{w} - \mathbf{w}_{t-1} \right\|^2 + f_t^{\text{cp}}(\mathbf{w}).$$

This idea is called proximal bundle method (Kiwiel, 1990) as the cutting planes $\{\mathbf{a}_i, b_i\}$ are deemed as bundles, and $\mathbf{w}_t$ is attracted to the proximity of $\mathbf{w}_{t-1}$. A large volume of work has been done in this area for decades, *e.g.*, (Kiwiel, 1985) and (Hiriart-Urruty & Lemaréchal, 1993a, Chapter XIII to XV). The underlying idea is Moreau-Yosida regularization (Moreau, 1965; Yosida, 1964), and it guarantees to find a ϵ approximate solution in $O(1/\epsilon^3)$ steps (Kiwiel, 2000). When the objective function is strongly convex, the convergence rate can be linear under some assumptions (Robinson, 1999). Variants of this idea are also widely used, *e.g.*, trust region bundle method (Schramm & Zowe, 1992) which upper bounds the displacement instead of penalizing it; and level set bundle method (Lemaréchal et al., 1995) which minimizes the displacement subject to a level of $f_t^{\text{cp}}(\mathbf{w})$.

It is noteworthy that the above methods treat the objective function as a black box which provides function and gradient evaluation at any given location. However, RRM problems are not black boxes, but explicitly composed of two parts: empirical risk R_{emp} and regularizer Ω. The free availability of the regularizer motivated Teo et al. (2007); Smola et al. (2007b) to perform cutting plane on R_{emp} only, and use Ω as the stabilizer. This is called bundle method for machine learning (BMRM). Different from Moreau-Yosida regularization where $\mathbf{w}_t$ is stabilized about $\mathbf{w}_{t-1}$, $\Omega(\mathbf{w})$ usually attracts $\mathbf{w}$ towards its *fixed* center, *e.g.* origin for L_p regularizer and uniform distribution for

Algorithm 2: Exact inner solver for BMRM (qp-bmrm)

```
Input: Previous subgradients $\{\mathbf{a}_i\}_{i=1}^t$ and intercepts $\{b_i\}_{i=1}^t$.
1 Assemble $A_t := (\mathbf{a}_1, \ldots, \mathbf{a}_t)$ and $\mathbf{b}_t := (b_1, \ldots, b_t)^\top$.
2 Solve $\boldsymbol{\alpha}_t := \text{argmax}_{\boldsymbol{\alpha} \in \Delta_t} -\lambda\Omega^*(-\lambda^{-1} A_t \boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \mathbf{b}_t \rangle$.
3 return $\mathbf{w}_t := \partial\Omega^*(-\lambda^{-1} A_t \boldsymbol{\alpha}_t)$
```

Algorithm 3: Inexact line search inner solver for BMRM (ls-bmrm)

```
Input: Previous subgradients $\{\mathbf{a}_i\}_{i=1}^t$ and intercepts $\{b_i\}_{i=1}^t$.
1 Assemble $A_t := (\mathbf{a}_1, \ldots, \mathbf{a}_t)$ and $\mathbf{b}_t := (b_1, \ldots, b_t)^\top$.
2 Solve $\eta_t := \text{argmax}_{\eta \in [0,1]} -\lambda\Omega^*(-\lambda^{-1} A_t \boldsymbol{\alpha}_t(\eta)) + \langle \boldsymbol{\alpha}_t(\eta), \mathbf{b}_t \rangle$ , where $\boldsymbol{\alpha}_t(\eta) :=$
  $((1-\eta)\boldsymbol{\alpha}_{t-1}^\top, \eta)^\top$.
3 $\boldsymbol{\alpha}_t \leftarrow ((1-\eta_t)\boldsymbol{\alpha}_{t-1}^\top, \eta_t)^\top$.
4 return $\mathbf{w}_t := \partial\Omega^*(-\lambda^{-1} A_t \boldsymbol{\alpha}_t)$
```

entropy regularizer. Technically, BMRM modifies the cutting plane algorithm just by replacing Eq. (1.29) with:

$$\mathbf{w}_t := \underset{\mathbf{w} \in \text{dom} f}{\text{argmin}}\, \lambda\Omega(\mathbf{w}) + R_{\text{emp},t}^{\text{cp}}(\mathbf{w}) = \underset{\mathbf{w}}{\text{argmin}}\, \underbrace{\lambda\Omega(\mathbf{w}) + \max_{i \in [t]} \{\langle \mathbf{a}_i, \mathbf{w} \rangle + b_i\}}_{:= J_t(\mathbf{w})}. \tag{1.31}$$

We summarize the BMRM algorithm in Algorithm 1.

The most expensive steps in BMRM are step 4 and 7 in Algorithm 1. In step 4, the computation of subgradient needs to go through the whole dataset, and this admits straightforward data parallelization. In particular, if R_{emp} sums the loss from individual data points like in the statistical query model (Kearns, 1998), then one can divide the whole dataset into subsets residing on distributed computing devices, compute their contribution to the gradient in parallel, and finally sum them up. This makes BMRM very promising for the coming era when parallel computing is the mainstream.

The other expensive step is to solve the optimization problem Eq. (1.31), *i.e.* step 7 of Algorithm 1. Teo et al. (2007) resorted to the dual problem:

$$\boldsymbol{\alpha}_t := \underset{\boldsymbol{\alpha} \in \Delta_t}{\text{argmax}} -\lambda\Omega^*(-\lambda^{-1} A_t \boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \mathbf{b}_t \rangle, \tag{1.32}$$

where Δ_t is the t-dimensional simplex $\left\{(\alpha_1, \ldots, \alpha_t)^\top \in \mathbb{R}^t : \alpha_i \geq 0, \sum_i \alpha_i = 1\right\}$, $A_t := (\mathbf{a}_1, \ldots, \mathbf{a}_t)$ and $\mathbf{b}_t := (b_1, \ldots, b_t)^\top$. The dual connection is $\mathbf{w}_t = \partial\Omega^*(-\lambda^{-1} A_t \boldsymbol{\alpha}_t)$. See Algorithm 2. Since the Ω^* in this dual problem is assumed to be twice differentiable and the constraint is a simple simplex, one can solve Eq. (1.32) with relatively more ease, *e.g.*, (Dai & Fletcher, 2006) which is specialized to L_2 regularizer $\frac{1}{2}\|\mathbf{w}\|^2$, and penalty/barrier methods (Nocedal & Wright, 1999) in general.

To circumvent the growing cost of solving Eq. (1.31) or Eq. (1.32), Teo et al. (2007) proposed the following approximation. Instead of searching for $\boldsymbol{\alpha}_t$ in Δ_t, we restrict the search domain to a line segment $\left\{((1-\eta)\boldsymbol{\alpha}_{t-1}^\top, \eta)^\top : \eta \in [0,1]\right\}$. See Algorithm 3. If $\Omega(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{w}\right\|^2$, then we are essentially restricting the search for $\mathbf{w}_t$ to the line segment between $\mathbf{w}_{t-1}$ and $-\lambda^{-1}\mathbf{a}_t$. In this case, we call Algorithm 3 ls-bmrm, and Algorithm 2 qp-bmrm as it solves a full quadratic program. As the feasible region of ls-bmrm in Eq. (1.32) is a proper subset of that of qp-bmrm, ls-bmrm makes less progress than qp-bmrm in each iteration, and hence converges more slowly.

The key result on the convergence rate of BMRM is (Teo et al., 2010, Theorem 5):

Theorem 27 (Convergence rate for BMRM) *Assume that $J(\mathbf{w}) > 0$ for all $\mathbf{w}$. Assume $\left\|\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w})\right\| \le G$ for all $\mathbf{w} \in \text{dom} J$. Also assume that Ω^* has bounded curvature,* i.e. *$\left\|\partial^2_{\boldsymbol{\mu}}\Omega^*(\boldsymbol{\mu})\right\| \le H^*$ for all $\boldsymbol{\mu} \in \left\{-\lambda^{-1}\sum_{i=1}^{t+1}\alpha_i \mathbf{a}_i : \boldsymbol{\alpha} \in \Delta_{t+1}\right\}$. For any $\epsilon < 4G^2H^*/\lambda$, the algorithm BMRM converges to the desired precision ϵ after*

$$k \le \log_2 \frac{\lambda J(0)}{G^2 H^*} + \frac{8G^2H^*}{\lambda\epsilon} - 1$$

steps. Furthermore, if the Hessian of $J(\mathbf{w})$ is bounded as $\left\|\partial^2_{\mathbf{w}} J(\mathbf{w})\right\| \le H$, convergence to any $\epsilon \le H/2$ takes at most the following number of steps:

$$k \le \log_2 \frac{\lambda J(\mathbf{0})}{4G^2H^*} + \frac{4H^*}{\lambda}\max\left\{0, H - \frac{8G^2H^*}{\lambda}\right\} + \frac{4HH^*}{\lambda}\log_2\frac{H}{2\epsilon}.$$

Teo et al. (2010, Theorem 5) proved this rate for ls-bmrm where each iteration only solves a simple one-dimensional optimization. In contrast, qp-bmrm performs a much more expensive optimization at every iteration, therefore it was conjectured that the rates of convergence of qp-bmrm could be improved. This was also supported by the empirical convergence behavior of qp-bmrm, which is much better than the theoretically predicted rates on a number of real life problems (Teo et al., 2010, Section 5). In Section 5.2, we answer this question in the negative by explicitly constructing a regularized risk minimization problem for which qp-bmrm takes at least $O(1/\epsilon)$ iterations.

1.7 Outline

The rest of the thesis is organized as follows:

Chapter 2: Conditional random fields for multi-agent reinforcement learning. We first applied graphical models to learn with distributed intelligent agents such as traffic light control, for which conditional random fields (CRFs, Lafferty et al., 2001) emerge as a natural way to model joint actions, and to efficiently search for an optimal joint policy through local communications. Policy gradient RL algorithms (Williams, 1992; Baxter & Bartlett, 2001) require inference in CRFs, and many existing algorithms can be utilized straightfowardly. Viewing our model from a CRF perspective, it extends the usual *iid* training scenario to temporal learning where the labels (actions) are no longer *iid*, but update the environment and affect the next observation. We tested our framework on a synthetic network alignment problem, a distributed sensor network, and a road traffic control system. The RL methods employing CRFs clearly outperform those which do not model the proper joint policy.

Chapter 3: Bayesian online multi-label classification. Data in many real world problems are available only as streams and cannot be stored. Many maximum margin online learners tend to overfit the noise, while Bayesian methods appear more promising because they maintain a distribution over the classifiers and are less susceptible to outliers. We applied a form of Bayesian online learning, Gaussian density filtering (GDF, Maybeck, 1982), to multi-label classification. The training labels are incorporated to update the posterior of the linear classifier via a graphical model similar to TrueSkill$^{\text{TM}}$ (Herbrich et al., 2007) tailored for the multi-label scenario, and inference is based on GDF with expectation propagation. Using samples from the posterior, we optimize the expected F-score on the test data. Our experiments on Reuters dataset show that our Bayesian approach delivers significantly higher macro-averaged F-score than the state-of-the-art online maximum margin learners.

Chapter 4: Kernel measure of independence for non-*iid* data. Although the rich feature space induced by kernels have been extensively utilized in supervised learning, it was observed recently that mapping probability distributions to their mean in an RKHS constitutes a natural characterization or embedding of distributions. This embedding induces new distances between distributions and in addition new measures of independence, which circumvent density estimation required by most information theoretic approaches. Interestingly, the undirected graphical models further allow us to factorize this kernel embedding onto cliques, which yields efficient measures of independence for non-*iid* or structured data. In Chapter 4, we applied our framework to ICA, independence test, and sequence segmentation. Methods taking into account the inter-dependence of observations significantly outperform those treating them as *iid*.

Chapter 5: Lower bounds for BMRM and faster rates for training SVMs. The optimization problems arising from maximum margin estimation are often nonsmooth, and can be effectively solved by BMRM and SVM$^{\texttt{Struct}}$ (Tsochantaridis et al., 2005). Smola et al. (2007b) proved that BMRM requires $O(1/\epsilon)$ iterations to converge to an ϵ accurate solution, and we further show in Chapter 5 that this rate is tight, *i.e.* there exists a function for which BMRM costs $O(1/\epsilon)$ steps. Motivated by Nesterov's optimal first-order methods (Nesterov, 2003, 2005b), we further devised an algorithm for the structured loss which finds an ϵ accurate solution in $O(\sqrt{1/\epsilon})$ iterations.

Extensions and proposed future work are detailed in the individual chapters.

Chapter 2

Conditional Random Fields for Multi-agent Reinforcement Learning

Conditional random fields (CRFs) have been studied in batch settings, where parameters are optimized over a training set; and online settings, where parameters are updated after each *iid* sample is observed. However, there is little work on CRFs for modeling temporal problems such as control or time-series prediction. The reinforcement learning (RL) community, on the other hand, has done work on decentralized (multi-agent) control. RL algorithms optimize a long-term measure of temporally delayed rewards in controlled systems. This chapter seeks to improve decentralized RL methods by using CRF models to exploit the structure between agents exhibited in many decentralized RL domains. Examples include sensor networks, traffic routing for roads or networks, pursuer-evader problems, and job-shop scheduling.

Bernstein et al. (2000) proved that the complexity of learning optimal coordination in decentralized RL is generally NEXP-hard in the number of agents. The simplest algorithms assume all agents are independent, learning to cooperate implicitly via an appropriate reward function (Bagnell & Ng, 2006). More advanced algorithms explicitly share information about states, values, or *proposed* actions (Boutilier, 1999), but still avoid modeling the optimal joint policy. Our work is similar to Guestrin et al. (2002), which does model the optimal joint policy, using the underlying structure to factorize Q-values and choose joint actions. In contrast, our approach focuses on directly optimizing a joint probability distribution over preferred actions. Furthermore, we draw on the wealth of approximate inference methods for graphical models, and CRFs in particular, to evaluate and optimize policies that would otherwise be intractable despite the structured representation.

Traditionally, CRFs use batch training algorithms to learn model $p(y|\mathbf{x};\boldsymbol{\theta})$, the probability of a label y, conditioned on observable variables $\mathbf{x}$ with the CRF parameters

$\boldsymbol{\theta}$ (Lafferty et al., 2001). During training we iterate through a set of training instances $\{\mathbf{x}_i\}_{i=1}^n$ with labels $\{y_i\}_{i=1}^n$, finding $\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} | \{(\mathbf{x}_i, y_i)\}_{i=1}^n)$. To predict the label for a novel observation $\mathbf{x}'$ we select $y' := \arg\max_y p(y|\mathbf{x}'; \boldsymbol{\theta}^*)$. In this work, we show that the same inference methods used for CRFs can be used to sample node actions from a *joint* stochastic RL policy. We also show how to optimize this joint policy by estimating the gradients of the long-term reward with respect to the policy parameters. Similar methods could be used for RL policies based on arbitrary graphical models. From the CRF point of view, we propose a method of using CRFs for modeling temporal processes.

Despite the commonality of using graphical models, our approach is different from the "control as inference" model by Toussaint (2009) and the graphical game by Kearns et al. (2001). Toussaint (2009) used graphical models to reformulate the temporal evolution in the control problem, which serves as a unifying framework for many control algorithms. However, the multiple agents are still treated as a black box, without being factorized by graphical models. Kearns et al. (2001), on the contrary, does factorize the players by a graphical model, but it is only used for computing the Nash equilibria which is a completely different setting from reinforcement learning.

Section 2.1 and Section 2.2 are devoted to describing graphical models and reinforcement learning respectively, with particular emphasis on CRFs and policy-gradient methods for RL. We then elaborate on the combination of CRF and RL in Section 2.3. Section 2.4 describes our experiments before concluding.

2.1 Conditional random fields and inference

CRFs are a probabilistic framework for labeling and segmenting data. Unlike hidden Markov models (HMMs) and Markov random fields (MRFs), which model the joint density $p(\mathbf{x}, y)$ over inputs $\mathbf{x}$ and labels y, CRFs directly model $p(y|\mathbf{x})$ for a given input observation $\mathbf{x}$. Furthermore, instead of maintaining a per-state normalization, which leads to the so-called label bias problem, CRFs use a global normalization that allows them to take global interactions into account (Lafferty et al., 2001).

In Section 1.3, we rigorously formulated CRFs in the framework of conditional exponential family. Learning algorithms leveraging efficient inference were also surveyed in Section 1.5. Now we just briefly recapitulate the concepts and furthermore introduce an efficient inference algorithm called tree sampling, which fits into our RL framework and delivers satisfactory empirical performance.

2.1.1 Conditional exponential families

Given an observation $\mathbf{x} \in \mathcal{X}$ and a finite discrete label set $\mathcal{Y}$, a conditional distribution over labels $y \in \mathcal{Y}$ parameterized by the natural parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ can be defined as

$$p(y|\mathbf{x};\boldsymbol{\theta}) = \exp(\langle \boldsymbol{\phi}(\mathbf{x}, y), \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta}|\mathbf{x})). \tag{2.1}$$

Here, $g(\cdot)$ is the log-partition function for normalization and the vector $\boldsymbol{\phi}(\mathbf{x}, y)$ is the *sufficient statistics* (also called features) which represent salient features of the input observations, and typically depend on the applications and CRF design. Let $\Theta_{\mathbf{x}}$ be the set of all valid $\boldsymbol{\theta}$: $\Theta_{\mathbf{x}} := \{\boldsymbol{\theta} : g(\boldsymbol{\theta}|\mathbf{x}) < \infty\}$. Finally we obtain the conditional exponential family (CEF): $\mathcal{P}_{\boldsymbol{\phi}|\mathbf{x}} := \{p(y|\mathbf{x};\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta_{\mathbf{x}}\}$.

Consider the more general case of structured output $\mathbf{y} \in \mathcal{Y}^m$ (m nodes). The clique decomposition theorem essentially states that if all the conditional densities in $\mathcal{P}_{\boldsymbol{\phi}|\mathbf{x}}$ satisfy the conditional independence relations represented by a graph $G = (V, E)$ on $\mathbf{y}$, then the sufficient statistics $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ decompose along the maximal cliques $\mathcal{C} = \{c_1, \ldots, c_n\}$ of G (Altun et al., 2004b):

$$\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = \operatorname*{vec}_{c \in \mathcal{C}} \{\phi_c(\mathbf{x}, y_c)\}, \tag{2.2}$$

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \exp\left(\sum_{c \in \mathcal{C}} \langle \phi_c(\mathbf{x}, y_c), \boldsymbol{\theta}_c \rangle - g(\boldsymbol{\theta}|\mathbf{x})\right), \tag{2.3}$$

where the vec operator concatenates vectors, c indexes the set of maximal cliques $\mathcal{C}$, and y_c is the label configuration for nodes in clique c. For convenience, we will assume that all maximal cliques have size two, *i.e.* an edge between nodes i and j has a feature ϕ_{ij} associated with it. We will also associate potentials ϕ_i to single nodes i. Node features represent the observation of state available at each node. The edge features encode the communication between nodes about their features and potential actions.

CRFs are examples of conditional exponential families with special graphs. For 1-D CRFs, the graph is a chain, so the edge features are $\phi_{i,i+1}(\mathbf{x}, y_i, y_{i+1})$. For 2-D grid CRFs, the edge features are $\phi_{(ij)(i'j')}(\mathbf{x}, y_{ij}, y_{i'j'})$ where nodes are indexed by double coordinates and $|i - i'| + |j - j'| = 1$.

2.1.2 Inference and gradient computations

CRF training procedures usually minimize the negative log-posterior of the parameters given the observation/label training set. As we will see in Section 2.2.1, policy-gradient algorithms instead draw samples from (2.3) given the parameters $\boldsymbol{\theta}$ and the most recent observation $\mathbf{x}$. CRF training procedures usually minimize the negative log-posterior of the parameters given the observation/label training set. This involves computing the

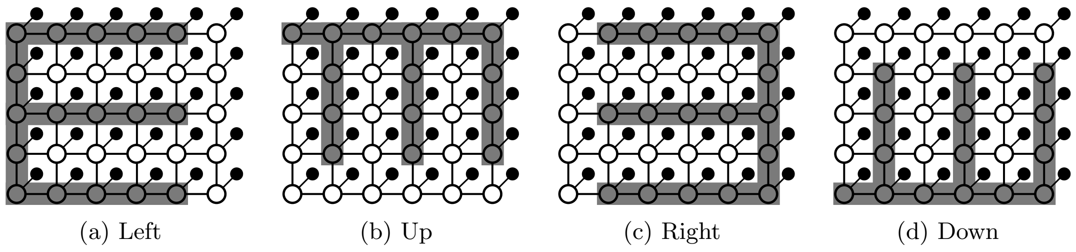


Figure 2.1: Four different partitions of a 5-by-6 CRF. Nodes in shaded and white regions are the two trees and the small black circles represent observations.

log-partition function $g(\boldsymbol{\theta}|\mathbf{x})$. Policy-gradient algorithms also require the gradient of the log probability of sampled labels/actions $\tilde{\mathbf{y}}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ln\ p(\tilde{\mathbf{y}}|\mathbf{x};\boldsymbol{\theta}) = \operatorname*{vec}_{c\in\mathcal{C}} \left\{ \phi_c(\mathbf{x},\tilde{y}_c) - \mathbb{E}_{p(y_c|\mathbf{x};\boldsymbol{\theta})}\left[\phi_c(\mathbf{x},y_c)\right]\right\}, \tag{2.4}$$

which exploits (2.2) and (2.3). Efficient (approximate) algorithms for sampling and computing the feature expectations have been surveyed in Section 1.5. Below we describe in detail one such method used in our experiments.

2.1.3 Tree MCMC sampler for CRFs

The tree Markov chain Monte Carlo (MCMC) sampler of Hamze & de Freitas (2004) is a state-of-the-art algorithm for sampling from posterior distributions and computing expectations of sufficient statistics in undirected graphical models with regular structure and high tree width. Its basic form works on pairwise MRFs or CRFs whose cliques are either nodes or edges.

The algorithm exploits the property that MRFs can be split into several disjoint trees (see Figure 2.1 for four different choices of partitions). Although belief propagation (BP) on the whole MRF is prohibitively expensive, it is cheap to run BP on each of the two trees (their tree width $w = 1$). So a natural idea is to combine analytical and sampling steps: conditioned on a sample of one of the trees, use BP to compute the exact joint conditional distribution of the other tree and draw a sample from it; then alternate between the two trees. Moreover, knowing the exact conditional distribution over the trees makes it possible to Rao-Blackwellize the sampler to reduce the variance (Casella & Robert, 1996). Each partition of the tree has to exclude some edges. In order to reduce the variance in the expectation estimates of these edges, and to cover all edges in the graph, we need to partition the graph in several different ways. This leads to the four partitions in Figure 2.1.

We provide the details of the tree sampler in Algorithm 4, which is specialized to

Algorithm 4: Tree MCMC Sampling, tailored from (Hamze & de Freitas, 2004)

```
   Input:
     • A graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of labels with node set $\mathcal{V}$ and edge set $\mathcal{E}$.
     • A set of features defined on nodes $\phi_i(\cdot)$, $i \in \mathcal{V}$, and on edges $\phi_{ii'}(\cdot,\cdot)$,
       $(i,i') \in \mathcal{E}' \subseteq \mathcal{E}$.
 1 Initialize:
     • Set scores $S_i^a = 0$ for all nodes $i \in \mathcal{V}$ and its possible assignments $a$;
     • Set scores $S_{ii'}^{aa'} = 0$ for all edges $(i,i') \in \mathcal{E}'$ and all its possible assignments $a$, $a$'.
     • Find a set of partitions of node indices $I$ so that $\mathcal{E}'$ is covered by edges of these
       trees:
       $\left\{\Delta_r^p \middle| p = 1 \ldots P, r = 1 \ldots R_p, \Delta_r^p \cap \Delta_{r'}^p = \emptyset \; for \; r \neq r', \cup_{r=1}^{R_p} \Delta_r^p = I, \cup_{r,p} \mathcal{E}_{\Delta_r^p} \supseteq \mathcal{E}' \right\}$,
       where $\mathcal{E}_{\Delta_r^p} \triangleq \{(i,i') \in \mathcal{E} : i, i' \in \Delta_r^p\}$, and similar notations will be used below.
 2
 3 for $p$ = 1 to $P$ /*for all partitions*/ do
 4     Randomly initialize label nodes on all trees in partition $p$ conditioned on
       observations $\mathbf{x}$.
 5     for $t$ = 1 to $T$ /*loop till convergence or exit criteria are met*/ do
 6         for $r$ = 1 to $R_p$ /*all trees in partition $p$*/ do
 7             Apply BP to compute the following smoothing densities:
                 • $p\left(y_i \middle| y_{\Delta_1^p}^t, \ldots, y_{\Delta_{r-1}^p}^t, y_{\Delta_{r+1}^p}^{t-1}, \ldots, y_{\Delta_{R_p}^p}^{t-1}, \mathbf{x}\right)$, for all $i \in \Delta_r^p$;
                 • $p\left(y_{ii'} \middle| y_{\Delta_1^p}^t, \ldots, y_{\Delta_{r-1}^p}^t, y_{\Delta_{r+1}^p}^{t-1}, \ldots, y_{\Delta_{R_p}^p}^{t-1}, \mathbf{x}\right)$, for all $(i,i') \in \mathcal{E}'_{\Delta_r^p}$.
 8
 9         Increment score
             • $S_i^a \leftarrow S_i^a + p\left(y_i = a \middle| y_{\Delta_1^p}^t, \ldots, y_{\Delta_{r-1}^p}^t, y_{\Delta_{r+1}^p}^{t-1}, \ldots, y_{\Delta_{R_p}^p}^{t-1}, \mathbf{x}\right)$, $\forall i \in \Delta_r^p, a$;
             • $S_{ii'}^{aa'} \leftarrow S_{ii'}^{aa'} + p\left(y_{ii'} = aa' \middle| y_{\Delta_1^p}^t, \ldots, y_{\Delta_{r-1}^p}^t, y_{\Delta_{r+1}^p}^{t-1}, \ldots, y_{\Delta_{R_p}^p}^{t-1}, \mathbf{x}\right)$,
                                                                     $\forall (i,i') \in \mathcal{E}'_{\Delta_r^p}, a, a'$.
10
11         Sample $y_{\Delta_r^p}^t \sim p\left(y_{\Delta_r^p} \middle| y_{\Delta_1^p}^t, \ldots, y_{\Delta_{r-1}^p}^t, y_{\Delta_{r+1}^p}^{t-1}, \ldots, y_{\Delta_{R_p}^p}^{t-1}, \mathbf{x}\right)$ using forward
           filtering / backward sampling.
12 Normalize $S_i^a \leftarrow S_i^a / \sum_a S_i^a$, $i \in \mathcal{V}$, and $S_{ii'}^{aa'} \leftarrow S_{ii'}^{aa'} \Big/ \sum_{a,a'} S_{ii'}^{aa'}$, $(i,i') \in \mathcal{E}'$.
13 return Rao-Blackwellised estimators
       $\mathbb{E}(\phi_i) = \sum_a \phi_i(a) S_i^a, \quad \mathbb{E}(\phi_{ii'}) = \sum_{a,a'} \phi_{ii'}(a,a') S_{ii'}^{aa'}$.
```

the pairwise maximal cliques. Empirically tree sampling is considerably more efficient than other partition based sampling schemes and the naïve Gibbs sampler, and with provable faster geometric convergence rate and lower variance (Hamze & de Freitas, 2004).

2.2 Reinforcement learning

A Markov decision process (MDP) consists of a finite set of states $s \in \mathcal{S}$ of the world, actions $y \in \mathcal{Y}$ available to the agent in each state, and a reward function $r(s)$ for each state s. In a partially observable MDP (POMDP), the controller sees only an observation $\mathbf{x} \in \mathcal{X}$ of the current state, sampled stochastically from an unknown distribution $p(\mathbf{x}|s)$. Each action y determines a stochastic matrix $\mathbf{P}(y) = [p(s'|s, y)]$ of transition probabilities from state s to state s' given action y. The methods discussed in this paper do not assume explicit knowledge of $\mathbf{P}(y)$ or of the observation process. All policies are stochastic, with a probability of choosing action y given state s, and parameters $\boldsymbol{\theta} \in \mathbb{R}^n$ of $p(y|\mathbf{x}; \boldsymbol{\theta})$. The evolution of the state s is Markovian, governed by an $|\mathcal{S}| \times |\mathcal{S}|$ transition probability matrix $\mathbf{P}(\boldsymbol{\theta}) = [p(s'|s; \boldsymbol{\theta})]$ with entries

$$p\left(s'|s;\boldsymbol{\theta}\right) = \sum\nolimits_{y \in \mathcal{Y}} p\left(y|s;\boldsymbol{\theta}\right) p\left(s'|s,y\right). \tag{2.5}$$

We assume an average reward setting where the task is to find a policy, or equivalently the parameter $\boldsymbol{\theta}$, which maximizes

$$R(\boldsymbol{\theta}) := \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\boldsymbol{\theta}}\left[\sum_{t=0}^{T-1} r(s_t)\right], \tag{2.6}$$

The expectation $\mathbb{E}_{\boldsymbol{\theta}}$ is over the distribution of state trajectories $\{s_0, s_1, \dots\}$ induced by $\mathbf{P}(\boldsymbol{\theta})$.

The core idea of this paper is to treat CRF distributions over labels, $p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$, exactly as joint distributions over multi-agent RL actions, *i.e.* a stochastic policy. Each node in the CRF will represent a single RL agent. The joint stochastic policy will give the probability of a vector of actions $p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$. The observations available to agent/node i are represented by the sufficient statistics $\phi_i(\mathbf{x}, \mathbf{y}_i)$. However, we also need the edge "observations" $\phi_{ij}(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j)$ to represent the information that can be communicated between neighboring agents i and j. Thus all we need for a CRF-RL model is a family of RL algorithms that directly optimizes stochastic policies.

2.2.1 Policy-gradient algorithms

Policy-gradient (PG) algorithms optimize polices by performing gradient ascent on a parameterized policy (Williams, 1992; Sutton et al., 2000; Baxter & Bartlett, 2001). These algorithms require only a parameterized and differentiable policy model $p(y|\mathbf{x};\boldsymbol{\theta})$, and a way to compute the gradient of the long-term reward $R(\boldsymbol{\theta})$.

A number of algorithms (Williams, 1992; Baxter & Bartlett, 2001; Peters et al., 2005) compute a Monte Carlo approximation of the reward gradient: the agent interacts with the environment, producing an observation, action, reward sequence

$$\{\mathbf{x}_1, y_1, r_1, \mathbf{x}_2, \ldots, \mathbf{x}_T, y_T, r_T\}.^1$$

For example, under mild technical assumptions, including ergodicity and bounding all the terms involved, Baxter & Bartlett (2001) obtain

$$\widehat{\frac{\partial R}{\partial \boldsymbol{\theta}}} = \frac{1}{T}\sum_{t=0}^{T-1} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p(y_t|\mathbf{x}_t;\boldsymbol{\theta}) \sum_{\tau=t+1}^{T} \beta^{\tau-t-1} r_\tau\,, \tag{2.7}$$

where an eligibility discount $\beta \in [0, 1)$ implicitly assumes that rewards are exponentially more likely to be due to recent actions. Without it, rewards would be assigned over a potentially infinite horizon, resulting in gradient estimates with infinite variance. As β decreases, so does the variance, but the bias of the gradient estimate increases (Baxter & Bartlett, 2001). In practice, (2.7) and all other policy-gradient algorithms share the same core estimator that make use of an *eligibility trace*

$$\mathbf{e}_t = \beta \mathbf{e}_{t-1} + \left.\frac{\partial}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \ln p\left(y_t|\mathbf{x}_t;\boldsymbol{\theta}\right) \tag{2.8}$$

Now $\boldsymbol{\delta}_t = r_t \mathbf{e}_t$ is the gradient of $R(\boldsymbol{\theta})$ arising from assigning the instantaneous reward to *all* log probability gradients, where $\beta \in [0, 1)$ gives exponentially more credit to recent actions. Additionally, β may be 1.0 for finite-horizon problems (Williams, 1992). The different policy-gradient algorithms vary in how they use instant gradient estimates $\boldsymbol{\delta}_t$.

For the experiments in this paper we adopt an online variation of the *natural actor-critic* (NAC) algorithm (Peters et al., 2005; Richter et al., 2007). While the NAC algorithm uses the estimator (2.8), it improves performance over (2.7) by: a) using a critic that approximates a projection of value function, with discount factor $\gamma \in [0, 1)$, to reduce variance of the gradient estimates; b) using a clever choice of critic parametrization to naturalize gradients (Amari, 1998); and c) using a least squares approach to solve

[1] We use r_t as shorthand for $r(s_t)$, making it clear that only the reward value is known, not the underlying state.

Algorithm 5: Online Natural Actor-Critic.

```
1  $t = 1,\ A_1^{-1} = I,\ \boldsymbol{\theta}_1 = [0],\ \mathbf{e}_1 = [0]$.
2  $\alpha$=step size, $\gamma$=critic discount, $\beta$=actor discount.
3  Get observation $\mathbf{x}_1$.
4  while not converged do
5      Sample action $\tilde{\mathbf{y}}_t \sim p(\cdot|\mathbf{x}_t, \boldsymbol{\theta}_t)$.
6      $\mathbf{e}_t = \beta \mathbf{e}_{t-1} + [\frac{\partial}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \ln p(\tilde{\mathbf{y}}_t|\mathbf{x}_t;\boldsymbol{\theta})^\intercal, \mathbf{x}_t^\intercal]^\intercal$.
7      Do actions $\tilde{\mathbf{y}}_t$.
8      Get reward $r_t$.
9      $\boldsymbol{\delta}_t = r_t \mathbf{e}_t$.
10     Get observation $\mathbf{x}_{t+1}$.
11     $\mathbf{w}_t = [\frac{\partial}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \ln p(\tilde{\mathbf{y}}_t|\mathbf{x}_t,\boldsymbol{\theta})^\intercal, \mathbf{x}_t^\intercal]^\intercal - \gamma[\mathbf{0}^\intercal, \mathbf{x}_{t+1}^\intercal]^\intercal$.
12     $\epsilon_t = 1 - t^{-1}$.
13     $\mathbf{u}_t = (\epsilon_t^{-1} - 1) A_{t-1}^{-1} \mathbf{e}_t$.
14     $\mathbf{q}_t^\intercal = \epsilon_t^{-1} \mathbf{w}_t^\intercal A_{t-1}^{-1}$.
15     $A_t^{-1} = \epsilon_t^{-1} A_{t-1}^{-1} - \frac{\mathbf{u}_t \mathbf{q}_t^\intercal}{1 + \mathbf{q}_t^\intercal \mathbf{e}_t}$.
16     $[\mathbf{d}_t^\intercal, \mathbf{v}_t^\intercal]^\intercal = A_t^{-1} \boldsymbol{\delta}_t$ (just to extract $\mathbf{d}_t$, $\mathbf{v}_t$ is never used).
17     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \mathbf{d}_t$.
18     $t \leftarrow t + 1$.
```

for the naturalized gradients, making full use of simulated trajectories. Algorithm 5 is used in our experiments (Richter et al., 2007).

2.2.2 Decentralized multi-agent RL

Decentralized (PO)MDPs assume a number of agents, each with a local observation of the state space. Here the action $\mathbf{y}$ becomes a vector giving the action for each agent. In the general case, optimal decision making in Decentralized MDPs is NEXP-hard in the number of agents (Bernstein et al., 2000), due to the combinatorial degree of communication required between the agents to coordinate actions. Many approximate approaches exist including no communication (Peshkin et al., 2000); explicit actions to communicate state information (Boutilier, 1999); local sharing of value functions (Schneider et al., 1999); and others with varying degrees of formalism. Under a common global reward, and some forms of local reward (Bagnell & Ng, 2006), agents that do not communicate can learn to cooperate implicitly to maximize the global reward (Boutilier, 1999). However, unless each agent has access to the full state description, they will generally not be able to act optimally. Our contribution is to introduce a mechanism for agents to efficiently — due to the graph structure of the CRF — communicate in order to converge to a joint policy. Our choice of policy-gradient algorithms is motivated by their ease of integration with CRFs, but they have the additional benefit of being guaranteed to converge (possibly to a poor local maximum) despite the use of function approximation and partial observability. Our model of multi-agent learning is similar to Guestrin et al. (2002), which uses an exact form of BP for factorizing Q-values and choosing jointly optimal actions, and hence may still be intractable for high tree width graphs.

2.3 Conditional random fields for RL

Applying CRFs for distributed RL is relatively straightforward: simply assume that the policy, $p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$, of the POMDP factorizes according to a CRF. The agents correspond to the label nodes, and the edges encode the spatial or temporal collaboration between agents. In order to apply PG methods one needs: a) the ability to draw an action from the policy model (step 5 of Algorithm 5); and b) computation of the gradient of the log-probability of the sampled action (step 6,11 and Eq. (2.4)). Efficient implementations rely on approximate sampling algorithms like the tree MCMC sampler described in Section 2.1.3. One can also easily verify that exponential families in general, and CRFs in particular, satisfy the mild technical conditions required for PG methods to converge, as long as all features are bounded.

Interestingly, the CEF policy representation (2.1) implements exactly the *soft-max* stochastic policy with linear feature combinations commonly encountered in RL applications, *e.g.* (Richter et al., 2007). Only the edge features prevent the trivial factorization of the distribution into independent agents that was demonstrated by Peshkin et al. (2000).

From the perspective of multi-agent RL, CRFs make efficient decentralized RL possible. By using conditional independence assumptions, the search space of the policy-gradient methods factorizes, leading to faster learning. Also, even though a CRF requires only local connections between agents, global interactions are still incorporated by belief propagation.

From a graphical models point of view, our technique is different from the usual online or offline training methods in two important ways. The training data is no longer *iid*. The action at time step t stochastically determines the input at time step $t+1$. Furthermore, the evaluation metric is no longer a loss function but a reward function that depends on both the current state and future states.

Superficially, our setup looks similar to dynamic Bayesian networks (DBNs). DBNs are directed graphical models (in contrast to CRFs which are undirected graphical models) used to represent models that evolve with time. Typically DBNs are used for a) filtering: monitor the hidden system state s over time by computing $p(s_t|\mathbf{x}_1 \ldots \mathbf{x}_t)$; b) prediction: computing $p(s_{t+1}|\mathbf{x}_1 \ldots \mathbf{x}_t)$; or c) smoothing: computing $p(s_{t-1}|\mathbf{x}_1 \ldots \mathbf{x}_t)$.

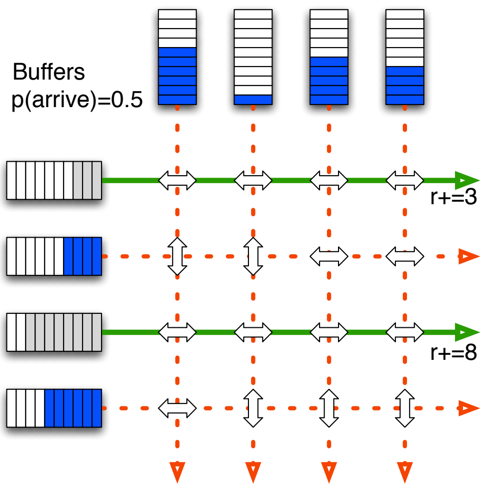


Figure 2.2: Abstract grid alignment domain.

In an RL context DBNs have been used to estimate the state transition matrix $\mathbf{P}(\mathbf{y})$, as well as the distribution $p(\mathbf{x}|s)$, in order to resolve partial observability in a POMDP (Theocharous et al., 2004). In contrast, we use CRFs as a policy, rather than as a state transition model.

We have shown how to learn *reactive* policies that ignore the fact that the true MDP state is unknown. Fortunately, PG methods still converge in this case. To take partial observability into account we could encode (long term) observation history, or a belief state (if $\mathbf{P}(\mathbf{y})$ and $p(\mathbf{x}|s)$ are known), into the sufficient statistics.

2.4 Experimental results

We performed experiments on one toy domain to demonstrate why a joint policy is important, and two benchmark decentralized RL domains.

2.4.1 Grid alignment

We constructed an abstract traffic domain, where it was known that agents would have to coordinate their actions in order to perform well, even in the case of a common global reward. Traffic flows along the edges of an $n \times n$ grid, always traversing to the opposite edge of the grid without turning (see Figure 2.2). Each intersection grid lines is an agent that controls a gate. The actions of a gate allow traffic to flow vertically or horizontally at each time step. Traffic units arrive with probability 0.5 per time step per boundary node (but only the top and left boundaries). Importantly, traffic cannot flow until *all* the gates on the traffic's path line up. When this happens, all waiting traffic for that

Table 2.1: Learning algorithm parameters. Lower β and γ give lower variance but higher bias in the gradient estimation. Too small α leads to slow learning, but too large α causes zigzag which also slows down the learning. α_{node} is N/A for Traffic because it does not use node features.

Domain	α_{ind}	α_{node}	α_{edge}	β	γ	runs
Grid	.002	.00001	.002	0.6	0.5	100
DSN	.0005	.00025	.0005	0.6	0.5	100
Traffic	.001	N/A	.01	0.9	0.95	50

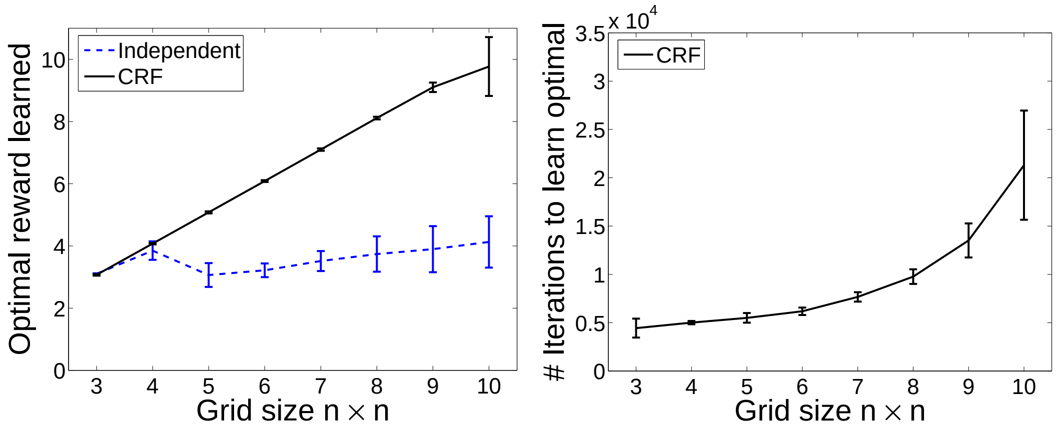


Figure 2.3: Average reward over the last 1000 steps, and iterations to optimality.

line propagates through instantly and each unit of traffic contributes +1 to a global reward. One or more misaligned gates blocks traffic, causing the length 10 buffer to fill up as traffic arrives. Full buffers drop traffic. Two observations per node indicate the normalised number of traffic units waiting for the node to align vertically, and horizontally. Edge features are 1 if the two nodes agree on an alignment, 0 otherwise. The optimal policy is for the *all* the n^2 gates to align in the orientation of the most waiting traffic, but since each node only knows how many traffic units are waiting for it, it must "negotiate" with neighbors on which way to align.

Learning parameters: The CRF model is a 2-D grid, with nodes for each agent, and edges for all nodes connected by a grid line. Due to the 2-D nature of the CRF, MCMC estimation of the log partition function, and its gradient were required. MCMC estimation needed 10 tree samples. To initialize the tree sampler, we randomly picked a spanning tree of the whole graph and sampled from the tree. Empirically, this allows us to obtain a good estimation of the distributions much faster than starting with independent node randomization. Other parameters, including the number of independent learning runs, are summarized in Table 2.1. To prevent poor local maxima when $n > 5$ we needed step sizes for edge feature parameters α_{edge} to be larger than

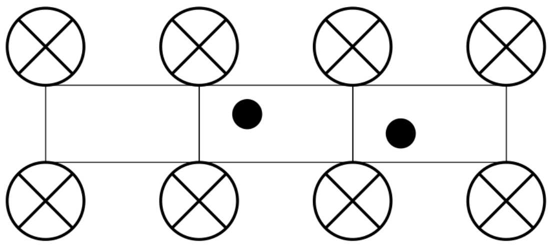


Figure 2.4: Sensor network domain with 8 sensors, 2 targets, and 3 cells.

for node feature parameters α_{node}, boosting the effect of edge features on the policy.

Results: The optimal reward is the grid size n. Figure 2.3 shows the CRF RL approach compared to a naïve implementation with independent agents. The CRF approach obtains the optimal reward all the way to grid size 10 (100 nodes), at which point some runs fail to reach the optimal policy. The number of iterations required to reach the optimal reward for the first time is shown on the right panel.

2.4.2 Sensor networks

The distributed sensor network (DSN) problem is a sequential decision making variant of the distributed constraint optimization problem described in (Dutech et al., 2005). The network consists of two parallel chains of an arbitrary, but equal, number of sensors. The area between the sensors is divided into cells. Each cell is surrounded by four sensors and can be occupied by a target. With equal probability targets can (from left to right) jump to the cell to its left, to its right, or remain where it is. Jumps that would cause a collision are not executed. The goal of the sensors to capture all targets. With initial configuration as in Figure 2.4, there are 37 distinct states. Each sensor can perform three actions resulting in a joint action space of $3^8 = 6561$ actions. The actions are: track a target in the cell to the left, cell to the right, or none. Every track action has a reward of -1. When in one time step at least three of the four surrounding sensors track a target, it is hit and its energy level is decreased by 1. Each target starts with an energy level of 3. When it reaches 0 the target is captured and removed. The three sensors involved in the capture are each provided with a reward of +10, and the goal is to maximize the total reward of all sensors. An epoch finishes when all targets are captured. If the DSN cannot capture all targets within 300 steps, the epoch is terminated, and a new epoch is started. A set of 50 randomly chosen initial states (with replacement) is cycled through for one episode. We run for 200 episodes and study the average reward of each episode. Finally, the whole process is independently repeated for 100 runs, and we report the average optimal reward and number of episodes.

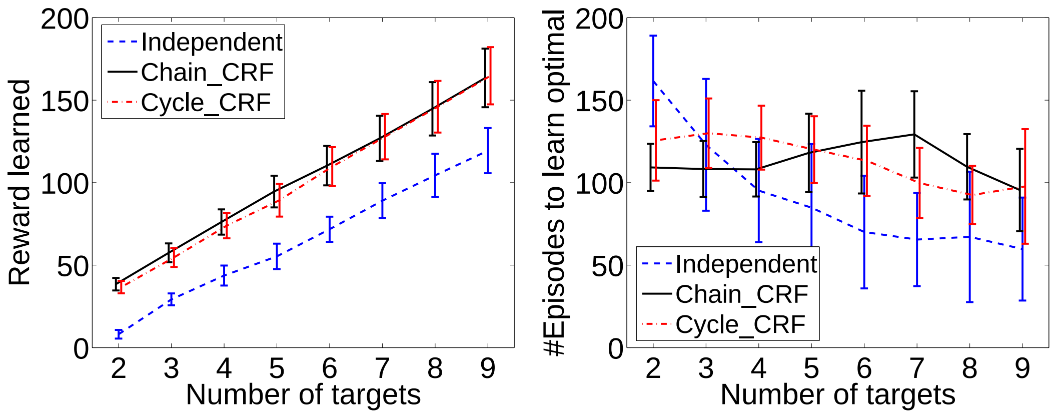


Figure 2.5: Results over 100 runs of the Sensor Network scenario, varying the number of targets.

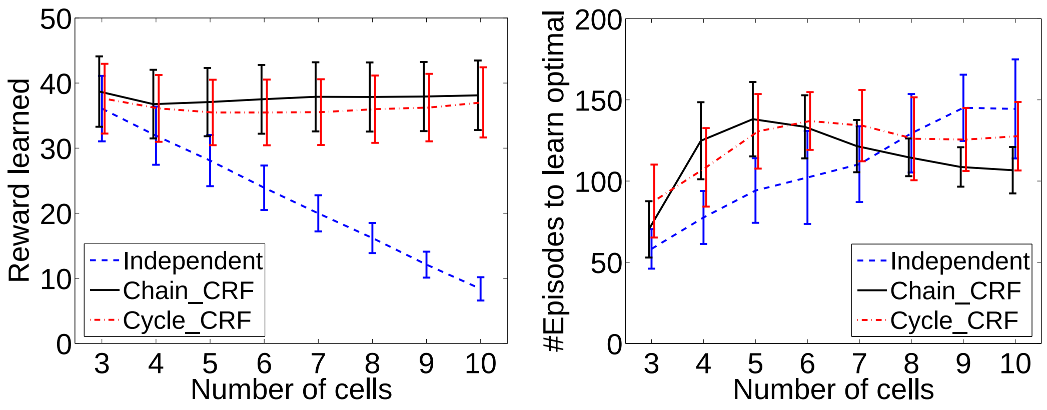


Figure 2.6: Results over 100 runs of the Sensor Network scenario, varying the number of cells.

Learning parameters: We experimented with two alternative CRF models, a *cycle* in which neighboring sensors along the top and bottom are connected as chains, and the cycle is completed with an edge between top and bottom sensors on the left and right ends. The *chain* is a more complex arrangement. Local sensors are bunched into groups of three (one top sensor, two bottom sensors and vice/versa). These meta-sensors form one CRF node. All the meta-sensors are connected in a 1-D chain, so the log-partition function and its gradient can be efficiently estimated.

Each node (or meta-node) has access to whether there is a target in its left and right cells (two binary values with two dummy cells at the two ends always observing no target). For the chain topology, the single edge feature is whether there are at least three out of four sensors focusing on their common cell. For the cycle topology, a single edge feature encodes whether connected sensors are focused on the same cell.

Results: The problem in Figure 2.4 has an optimal long-term average reward of 42. The best results from Dutech et al. (2005) use a distributed Q-learning approach where neighbors' Q-values are averaged (which implicitly assumes communication), achieving an average reward of less than 30. Figure 2.5 shows that CRF modeling with NAC achieves the optimal reward for this problem, and problems where the number of targets is increased up to 10. The ease with which we outperform the distributed Q-learning approach is not surprising, since the CRF allows sensors to agree on which target to focus on. We obtain similarly good results when the number of targets is fixed and more cells are added (Figure 2.6). The curious peak in the required iterations for 5 cells corresponds to the difficult situation where there are not enough sensors, *and* targets are able to jump around with few collisions. Adding more cells also adds more sensors, so that an individual sensor is rarely required to focus left and right at the same time. In both experiments the chain CRF does marginally better, probably due to the tighter coupling of the sensors and exact evaluation of the log-partition function.

2.4.3 Traffic light control

Many drivers have been frustrated by driving along a main street, to be constantly interrupted by red lights. This domain demonstrates learning an offset between neighboring intersections. The domain and simulator code are from Richter et al. (2007), which contains the full implementation details. We model one main road with n controlled intersections and $n + 1$ road links to traverse. It takes cars 2 time units to travel down a road to the next intersection. There are 4 actions, corresponding to the 4 traffic signal patterns that allow traffic to move through the intersection safely in any direction. For this experiment only the action that lets traffic drive straight ahead along the main road is useful. At each time step (about 5 seconds of real-time) the controller decides on the action for the next step. We do not restrict the order of actions, but importantly, we enforce the constraint that all actions must be activated at least once within 8 time steps so that vehicles on side streets would not wait forever. One car enters the leftmost end of the road every 4 time steps. We use realistic local rewards for each intersection: each intersection has an inductive loop sensor that can sense a car waiting, producing a -1 reward. For edge (i, j) where j is downstream of i, the edge parameters receive j's reward. This is different from the grid and sensor network case where a single scalar global reward is used. Our current work focuses on the factorization of the actions of the multiple agents. The influence of the reward factorization is left for future research.

Learning parameters: Each intersection is a CRF node that chooses from one of the four actions. For independent learners the only feature is a constant bias bit that

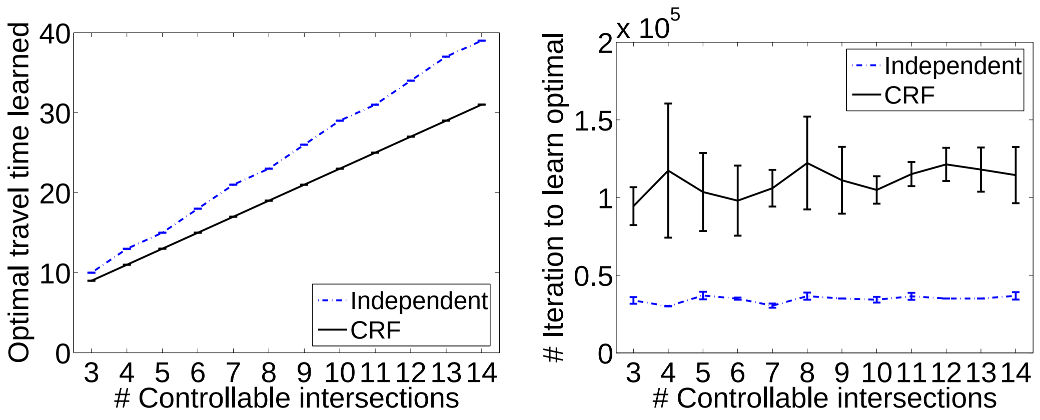


Figure 2.7: Results over 50 runs of the road traffic offset scenario. The X-axis is intersections. The left Y-axis is the travel time.

allows learning of which phase is the most commonly used. No node features were given to the CRF. The edge features for the CRF version are an $a \times a$ binary matrix, where a is the number of actions. Bit i, j is on if the action of the upstream neighbor i from 2 time steps ago (the time required to drive down the road edge) matches the chosen action of the current intersection j. The edge feature matrix has exactly 1 bit set to true in the matrix in any step. Typically a road traffic network would be represented as an undirected graph. But this simple scenario is one long road, thus can be represented as a chain CRF.

Results: Figure 2.7 shows the results on the road traffic domain in the same style as previous results. Again, the CRF model clearly outperforms the independent agents approach. We observed, however, that early in learning both the independent agents and CRF learn that all traffic moves left to right. Giving the maximum possible time to this direction is a strong local minimum. After that, the independent agents fail to improve but the CRF model asymptotically approaches the optimal 0 waiting time.

Figure 2.8 shows convergence plots for the CRF approach versus the independent agents.

2.5 Conclusions

We have shown how to use CRFs to model control processes, or equivalently, how decentralized RL can be performed with CRF optimisation methods. Although all our examples have been related to controlling a process, a special case is where rewards are given simply for predicting the next input, i.e., time series prediction. From a reinforcement learning point of view we have presented an efficient policy-gradient

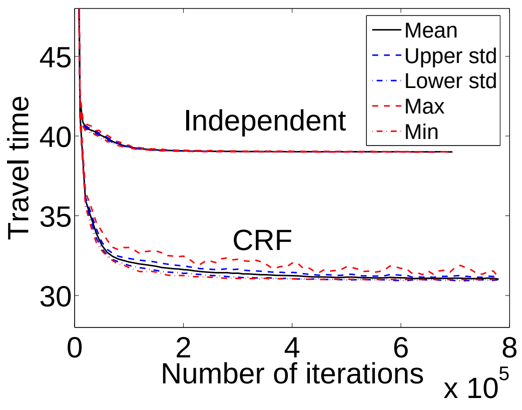


Figure 2.8: Convergence of NAC on the independent agents (upper curve) and on CRF model (lower curve). The number of controllable intersections is 14.

solution to the difficult problem of optimizing joint actions in a decentralised (PO)MDP. Future work could explore RL in general graphical models, and how local rewards may be propagated through a graphical model.

Chapter 3

Bayesian Online Learning for Multi-label and Multi-variate Performance Measures

Many real world applications involve a large number of classes and each example can be associated with multiple classes. For example, a lot of web based objects such as ads, blogs, web pages, RSS feeds are attached with tags which are essentially forms of categorization. A news article on "Obama supported the AIG bailout of $170 billion after some debate in Congress" can be associated with `insurance`, `economics`, and `politics`. This setting is referred to as multi-label classification in machine learning.

Cases in point can be found in search engine industry. Most search engines are free to use, and their revenue comes from users clicking on the ads embedded in the search result. To select and place ads, tags play an important role. Advertisers provide ads and their associated keywords, together with a bid once this ad is clicked. Upon a search request, the ad ranker and filter are invoked to select the ads for display, with an eye to maximizing the expected revenue which depends on a) how likely will the user click it, and b) the price calculated from a Vickrey auction of advertisers' bids. It will be very helpful for ad selection if ads can be automatically attached with *multiple* tags, or categorized into a hierarchy or ontology associated with odds of memberships.

Learning for multi-label data is usually faced with the following practical challenges:

1. The problem scale is huge in a) number of data points n, b) number of feature D, and c) number of class C. Usually, $O(nDC)$ computational complexity is the limit we can afford. Hence efficiency becomes necessary and expensive operations such as pairwise comparison must be avoided.

2. The performance measure is usually more complicated than accuracy, *e.g.* micro-average F-score, area under ROC curve. These measures are usually called multi-variate measure because they couple the labels of all the classes in the dataset in

a nondecomposable way. This incurs two complexities: a) most existing training algorithms simply optimize error rates per class which may lead to poor models even on the training set under this new measure, and b) the application of the trained model need to be calibrated over the whole testing set.

3. The labels can be highly correlated, *e.g.* co-occurrence. More importantly, the labels in many applications employ a tree structured ontology hierarchy, *i.e.* all ancestor classes of a relevant class must also be relevant. Examples include the patent classification hierarchy according to the World International Patent Organization, and the enzyme classification scheme for classifying amino acid sequences of enzymatic proteins. Another example is the Pascal challenge on large scale hierarchical text classification[1] which is based on the ODP web directory data (www.dmoz.org).

Several algorithms have been proposed for multi-label learning, and they can be categorized in three dimensions: a) online v.s. batch, b) frequentist v.s. Bayesian, and c) using structures in the label space v.s. treating the labels as independent. This categorization helps us to analyze how much these algorithms fit for the above three challenges, and also motivates our new algorithm.

The most primitive algorithms are in the batch fashion. A typical frequentist method is based on SVMs (Elisseeff & Weston, 2001), which generalizes the binary hinge loss to the maximum inconsistency on data point $\mathbf{x}$:

$$\max_{c \in \mathcal{R}} \max_{c' \in \mathcal{I}} \langle \mathbf{w}_c, \mathbf{x} \rangle - \langle \mathbf{w}_{c'}, \mathbf{x} \rangle ,$$

where $\mathcal{R}$ and $\mathcal{I}$ are the set of relevant and irrelevant labels of $\mathbf{x}$, and $\mathbf{w}_c$ is the weight vector for class c. This approach has complexity $O(|\mathcal{R}| \, |\mathcal{I}|)$ hence not suitable for large number of class. Among the Bayesian methods, mixture models are the most straightforward. They assume that each document has an unknown "topic", and each word is generated by the topic through a multinomial distribution. To cater for the multi-label scenario, McCallum (1999) proposed blowing up the latent topic space to the power set of all the topics, and EM was used to estimate the parameters with special care paid to overfitting which results from the exponentially large latent space.

Unfortunately, all these batch methods are very expensive in parameter estimation, hence not suitable for the first challenge. For large datasets, one effective and efficient scheme is to visit the data points one by one, and at each step update the model by using just a single data point. This online learning scheme keeps each iteration cheap and incrementally learns a good model. It also allows the training examples to

[1] `http://lshtc.iit.demokritos.gr`

be provided as a stream with no need of storing the data. The any-time property is valuable as well: one can pause the process at any time and use the current model; once new data arrives, learning can be resumed with no need of re-training from scratch.

Although online learning has been widely used for binary classification, it is less studied for the multi-label scenario. Crammer & Singer (2003) proposed an additive online algorithm for category ranking. Given a data point, it compares the output of the model for each pair of relevant and irrelevant classes, and the difference is used to update the weight vector of these two classes. To avoid the quadratic complexity as in (Elisseeff & Weston, 2001), it devised a pre-computation such that the time complexity is reduced to linear in the number of classes, and the space cost to sub-quadratic. Bayesian online learning is also popular, *e.g.* (Opper, 1998). They essentially perform assumed density filtering, where at each step the posterior of the model is updated based on a single data point. We are unaware of any published Bayesian online learner for multi-class or multi-label classification.

In a nutshell, Bayesian methods learn a *distribution* over a family of models, while frequentist methods find the most probable model. They are both useful and commonly used. Intuitively speaking, although learning and using a distribution of the models are generally more expensive in computation, they provide more flexibility in decision making and allow the model to be used in a completely different way from how it was learned (decoupling). A case in point is the multi-variate performance measure. In (Joachims, 2005), the training of SVM *is* customized for the multi-variate measure, however the testing data are still labeled by applying the learned model independently. In contrast, with a distribution of models available, the Bayesian method can a) provide a principled framework of labeling the testing data to optimize multi-variate measures in a batch fashion, and b) allow the distribution of the model to be estimated with a different and decomposable measure (such as square loss), which is especially useful when the large dataset size necessitates online learning.

Finally, to make use of the structure in the label space as desired from the last challenge, frequentist methods such as (Rousu et al., 2006) use the framework of maximum margin Markov network, where the class hierarchy is represented by a Markov tree. This tree plays a key role in the definition of the discrepancy between labels, and of the joint kernels (kernels on the pair of feature and label). On the Bayesian side, the most straightforward way to incorporate label interdependency is the conditional random fields (CRFs), based on which Ghamrawi & McCallum (2005) directly incorporated label co-occurrences into the features. Interestingly, this CRF model can also induce the structure of labels from the data, instead of relying on a *given* structure that is assumed by Rousu et al. (2006). However, they trained the CRF in a batch fashion and it is not clear whether the model can also be learned efficiently in the stochastic online setting.

Summarizing the above discussion, we propose in this chapter a Bayesian online multi-label classification (BOMC) framework which learns a probabilistic model of the linear classifier (Section 3.1). As an initial work, the labels are only loosely coupled in our model for the multi-label scenario, while extension to more general structures is straightforward for future work. The training labels are incorporated to update the posterior of the classifiers via a graphical model similar to TrueSkill$^{\text{TM}}$ (Herbrich et al., 2007), and inference is based on assumed density filtering between training examples and expectation propagation (Minka, 2001) within each training example (Section 3.2). This allows us to efficiently use a large amount of training data. Using samples from the posterior of the model, we label the test data by maximizing the expected F-score (Section 3.3). Experimental results are presented in Section 3.4, including the comparison of macro-average F-score and training time. The whole chapter is concluded in Section 3.5 with several proposal for future research.

3.1 A Bayesian model for multi-label classification

Suppose we have n training examples whose feature vectors are $\left\{\mathbf{x}^i \in \mathbb{R}^D\right\}_{i=1}^n$. Assume there are C classes[2], and the label vector $\mathbf{y}^i$ of each example uses the canonical representation for multi-label: $\mathbf{y}^i \in \{0,1\}^C$ and $y_c^i = 1$ if, and only if, example $\mathbf{x}_i$ is relevant to class c, and 0 otherwise.

Our basic assumptions on the model include the following:

1. Each class is associated with a linear discriminant, *i.e.* weight vector $\mathbf{w}$ and bias b: $\langle \mathbf{w}, \mathbf{x} \rangle - b$.

2. The weight and bias are independent Gaussians. Their mean and variance are estimated from the training data.

In the following sections, we start from a special case of multi-label classification: multi-class classification. This allows us to introduce the most important concepts of the framework without being complicated by the multi-label ingredients. Afterwards, we will propose and compare three models for multi-label classification.

3.1.1 Multi-class classification

In multi-class classification, there are multiple classes and each example belongs to *exactly one* class. For each training example $\mathbf{x}$, a factor graph to model the likelihood

[2]We are very reluctant to use capital letters to denote numbers. However, since the later description of algorithms will require explicit indexing of class and feature dimensions, we find it much more intuitive to use D as the total number of features and d as the index for it. Similarly, C classes indexed by c.

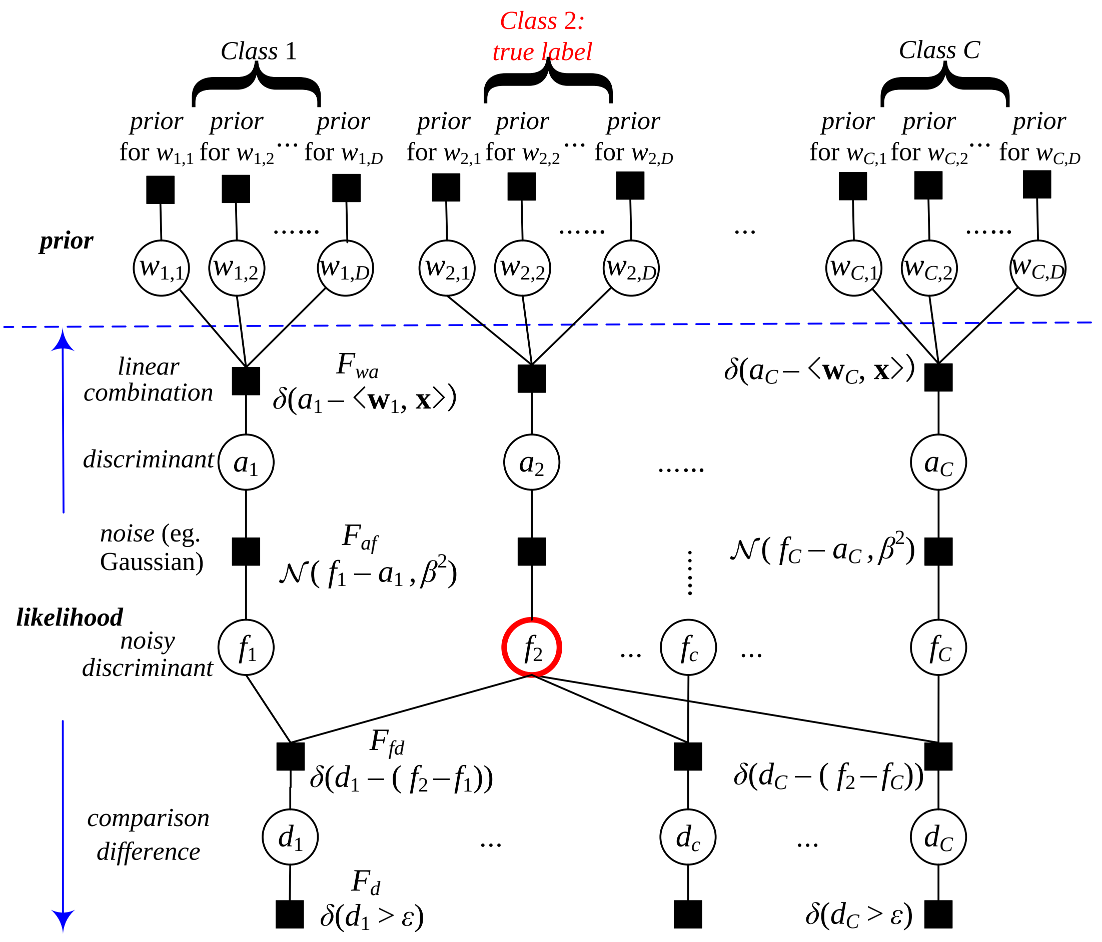


Figure 3.1: A factor graph for multi-class classification. Class 2 is the true label.

is built depending on its label. Figure 3.1 shows the case where example $\mathbf{x}$ has label 2. Round circles represent random variables and solid rectangles represent factors which are detailed below.

Priors. Each class $c \in [C]$ has a corresponding weight vector $\mathbf{w}_c \in \mathbb{R}^D$ endowed with a prior. The simplest form of prior is a diagonal Gaussian, *i.e.* all the elements of $\mathbf{w}_c$ are independent Gaussians as represented in the top row of factors in Figure 3.1.

Linear combination. We use a simple linear discrimination model for all classes: $a_c := \langle \mathbf{w}_c, \mathbf{x}_i \rangle$, and this is encoded by the linear combination factor:

$$F_{wa}(\mathbf{w}_c, a_c) := \delta(a_c - \langle \mathbf{w}_c, \mathbf{x} \rangle),$$

where $\delta(\cdot)$ is the Dirac function: $\delta(x) = 0$ for all $x \neq 0$ and $\int_{\mathbb{R}} \delta(x) \mathrm{d}x = 1$.

Feature noise. To model the feature noise, we define the noisy discriminant by passing a_c through a Gaussian noise factor with a pre-specified variance β^2, *i.e.*

$$F_{af}(f_c, a_c) := \frac{1}{\sqrt{2\pi\beta^2}} \exp\left(-\frac{(f_c - a_c)^2}{2\beta^2}\right). \tag{3.1}$$

Intuitively, this noise serves as a square loss when the dataset is not linearly separable, and β^{-2} corresponds to the regularization parameter.

Label comparison. The label of $\mathbf{x}$ is assumed to be $\operatorname{argmax}_{c\in[C]} f_c$. This assumption is encoded by comparing the f_c of the correct class (2 in this example) with that of the rest classes, and enforcing the difference to be positive by using a $\delta(\cdot > \varepsilon)$ factor, where ε is a small positive *margin*.

$$F_{fd}(f_l, f_c, d_c) = \delta(d_c - (f_l - f_c)), \tag{3.2}$$

$$F_d(d_c) = \delta(d_c > \epsilon), \tag{3.3}$$

where $l = 2$ is the correct label, and c ranges over all the other classes. Note that the δ in Eq. 3.2 is again the Dirac function, while the δ in Eq. (3.3) is 0-1: $\delta(p) = 1$ if p is true, and 0 otherwise.[3]

Clearly, $\delta(d_c > \epsilon)$ is not the only way to enforce the correct label. In the similar spirit of probit regression, one can use $\Phi(d_c)$ as $F_d(d_c)$ where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. Although Φ is smooth, exact inference is still intractable.

A few notes on the connotation of the model in Figure 3.1 is in order. The product of the factors below the dashed line is defined as the likelihood $p(\mathbf{y}^i, \mathbf{a}, \mathbf{f}, \mathbf{d}|\mathbf{w})$ (omitting the conditioning on $\mathbf{x}^i$), while product of the factors above the dashed line equals the prior $p(\mathbf{w})$. So the product of all factors in the graph equals $p(\mathbf{y}^i, \mathbf{a}, \mathbf{f}, \mathbf{d}, \mathbf{w})$. So summing out all the $\mathbf{a}, \mathbf{f}, \mathbf{d}, \mathbf{w}$, we obtain $p(\mathbf{y}^i)$ which is called evidence in the Bayesian theory. It is crucial to understand that this is *not* a function of $\mathbf{y}^i$, but just for the particular given $\mathbf{y}^i$. Our model does *not* specify the probability for any other possible label of $\mathbf{x}^i$, and there is no node that corresponds to the label. Fortunately, this is sufficient for our need because we will be only interested in $p(\mathbf{w}|\mathbf{y}^i)$, which, as a function of $\mathbf{w}$, is equal to $p(\mathbf{w}, \mathbf{y}^i)$ up to a normalizing constant $p(\mathbf{y}^i)$. Now that the product of all the factors in Figure 3.1 is exactly $p(\mathbf{y}^i, \mathbf{a}, \mathbf{f}, \mathbf{d}, \mathbf{w})$, $p(\mathbf{w}, \mathbf{y}^i)$ (hence $p(\mathbf{w}|\mathbf{y}^i)$) as a function of $\mathbf{w}$ can be obtained by summing out $\mathbf{a}, \mathbf{f}, \mathbf{d}$. This can be equivalently achieved by treating Figure 3.1 as a factor graph which defines a joint

[3]The notation of δ is standard in both cases. And this overloading does not cause any ambiguity: when applied on a real number, δ means Dirac function; and when applied on a predicate, it means 0-1.

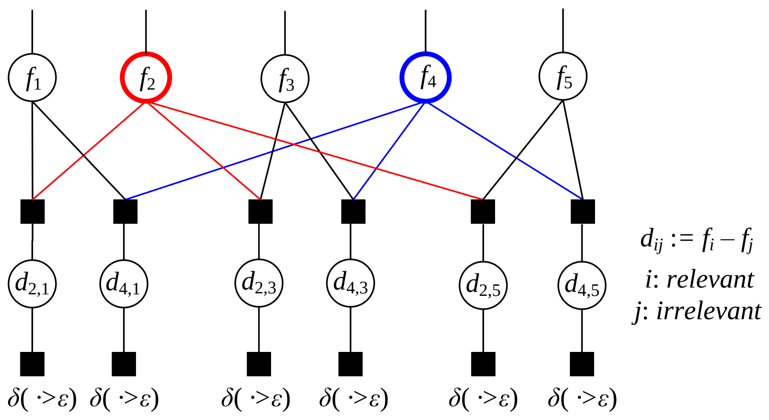


Figure 3.2: A factor graph for multi-label classification via pairwise comparison. Here class 2 and 4 are the relevant labels.

distribution over $\mathbf{a}, \mathbf{f}, \mathbf{d}, \mathbf{w}$ (Definition 21), and then query the marginal distribution of $\mathbf{w}$ by summing out $\mathbf{a}, \mathbf{f}, \mathbf{d}$.

The fact that the label only affects the comparisons between f_c is slightly suggestive of the "gate notation" by Minka & Winn (2009) which conveniently represents mixture models and context-sensitive independence in factor graphs. But in our case there is no point introducing $\mathbf{y}^i$ as a gate variable because $\mathbf{y}^i$ is deterministically given. However, when the given labels are noisy, then the true label becomes a random variable and the gate notation becomes useful.

It is noteworthy that this diagram is very similar to the TrueSkill$^{\text{TM}}$ algorithm (Herbrich et al., 2007), but they are different in the following ways: a) the factor graph in our case corresponds to a fixed example $\mathbf{x}$ instead of multiple examples (players/teams), b) each class is associated with a different weight vector while TrueSkill$^{\text{TM}}$ uses a common weight vector, and c) there is only one winner in our diagram, while TrueSkill$^{\text{TM}}$ has several winners which entails pairwise comparison. In Section 3.2, we will discuss how to learn the model parameters, *i.e.* the mean and variance of the prior.

3.1.2 Multi-label classification

In multi-label classification, each example can be associated with more than one label or no label at all. Accordingly, we only need to modify the comparison part of the model in Figure 3.1, keeping the linear combination and noise part intact. The new model is shown in Figure 3.2.

The fundamental assumption in the new model is that the noisy discriminant value f_c of relevant[4] classes should be higher than that of the irrelevant classes. No comparison is made within relevant classes or irrelevant classes. This idea is the same

[4]We changed "correct" to "relevant" to reflect the multi-label scenario.

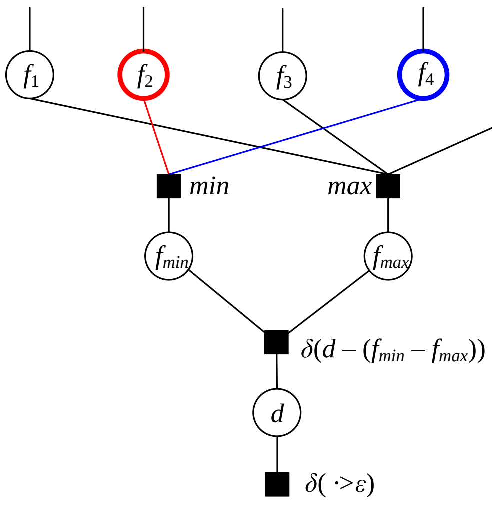


Figure 3.3: A factor graph for multi-label classification using max and min factors.

as in (Elisseeff & Weston, 2001). The biggest problem of this model is the computational cost. An example with R relevant labels and $C-R$ irrelevant labels will cost $O(R(C-R))$ complexity both in time and space.

A simple and equivalent reformulation of pairwise comparison is by introducing the max and min factors:

$$\min_{c:\ c \text{ is relevant}} f_c - \max_{c:\ c \text{ is irrelevant}} f_c > \varepsilon,$$

and the corresponding factor graph is shown in Figure 3.3. This reduces the number of factors back to linear. However, a new problem arises from the inference with max and min factors, for which only approximate message formulae are available and the error analysis is hard. We will present the details on message passing over max and min factors in Appendix B.3.2, which is based on (Afonja, 1972).

A further simplification is by assuming that the relevance of the labels conforms with an underlying total order. This translates to associating some score f_c with each class c, and $f_c > f_{c'}$ implies that c must be relevant once c' is. Equivalently, we can determine the relevance of all classes by thresholding all f_c by a global threshold b which needs to be estimated from the training data as well. We also call b a global bias due to its equivalence to thresholding $f_c - b$ at 0. Figure 3.4 illustrates this idea graphically. One can further incorporate a “local” bias for each individual class by, for example, adding an artificial constant feature. This could also eliminate the need of the global bias. We will compare these two models in the experiment.

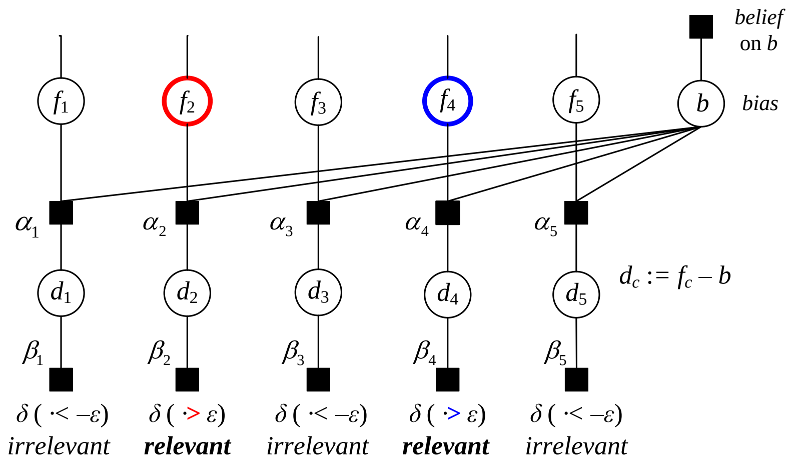


Figure 3.4: A factor graph for multi-label classification via total ordering and a global bias. See text for explanation.

3.2 Online learning and inference

We discuss in this section how to use the training data to learn the model, *i.e.* the distribution of weights and bias. Bear in mind that the graphical models in Figure 3.1, 3.2, 3.3 and 3.4 correspond to one particular training example. So we need to make two decisions:

1. Given a training example and its corresponding graph, how to infer the posterior of the model?

2. How is the set of training data used as a whole, *i.e.* how are the graphs of different training examples connected?

Our answer is: expectation propagation (EP, Minka, 2001) for the first question and Gaussian density filtering (Maybeck, 1982) for the second. Below are the details.

3.2.1 A Bayesian view of learning

Assume we have n feature/label pairs $\left\{(\mathbf{x}^i, \mathbf{y}^i)\right\}_{i=1}^n$ drawn *iid* from some underlying distribution. Suppose we have a prior distribution $p_0(\mathbf{w})$ on the weight vector $\mathbf{w}$, as well as a likelihood model $p(\mathbf{x}^i, \mathbf{y}^i|\mathbf{w})$. In Bayesian learning, we are interested in the posterior distribution of $\mathbf{w}$.

$$p(\mathbf{w}|\left\{(\mathbf{x}^i, \mathbf{y}^i)\right\}) = \frac{p_0(\mathbf{w}) \prod_i p(\mathbf{x}^i, \mathbf{y}^i|\mathbf{w})}{\int p_0(\mathbf{w}) \prod_i p(\mathbf{x}^i, \mathbf{y}^i|\mathbf{w}) \mathrm{d}\mathbf{w}}.$$

The integral in the denominator can be computationally intractable, hence various approximation algorithms have been developed (see Section 1.5). Due to the large

Algorithm 6: Gaussian density filtering.

```
  Input: A set of feature/label pairs for training $\{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^n$.
  Output: Approximate posterior of the model.
1 Initialize: Specify a prior of the model $p_0(\mathbf{w})$.
2 for $i = 1$ to $n$ do
3     Construct the likelihood $p(\mathbf{x}^i, \mathbf{y}^i|\mathbf{w})$ using the training example $(\mathbf{x}^i, \mathbf{y}^i)$.
4     Set the prior of the model to $p_{i-1}(\mathbf{w})$.
5     Find a Gaussian distribution $p_i(\mathbf{w})$ which approximates the posterior
      distribution $p(\mathbf{w}|\mathbf{x}^i, \mathbf{y}^i) \propto p_{i-1}(\mathbf{w})p(\mathbf{x}^i, \mathbf{y}^i|\mathbf{w})$. Different inference algorithms
      differ in the sense of approximation.
6 return $p_n(\mathbf{w})$
```

amount of data in many real life applications, we resort to one of the cheapest approximations: assumed density filtering (ADF). The idea is simple: in each iteration, visit only one data point $(\mathbf{x}^i, \mathbf{y}^i)$, use its likelihood to compute the posterior of the weight, and then use this posterior as the prior for the next iteration. Since each step only deals with one likelihood factor, the posterior inference can be performed efficiently. Algorithm 6 sketches ADF.

In our case the prior of all weights are set to zero mean Gaussians, and the variance will be discussed in the experiment section. The likelihood is modeled by the factor graph in Figure 3.1. If we keep the posterior approximated by Gaussians, then ADF can also be called Gaussian density filtering (GDF). Now the only problem is how to compute the posterior in the step 5 of Algorithm 6.

3.2.2 Inference on the graph of a given training example with EP

Given a training example $(\mathbf{x}, \mathbf{y})$, the discussion in Section 3.1.1 has shown that the posterior $p(\mathbf{w}|\mathbf{x}, \mathbf{y})$ can be derived by querying the marginal distribution of $\mathbf{w}$ in Figure 3.1. This marginal can be computed by EP, which was introduced in Section 1.5. In a nutshell, EP is similar to loopy belief propagation, but further approximates the messages as well as possible. To this end, it approximates the marginals of the factors via Gaussians which match the first and second order moments. Strictly speaking, EP finds a Gaussian approximation of the true posterior. Since our model uses the same set of factors as in TrueSkill$^{\text{TM}}$, we refer the interested readers to the Table 1 in (Herbrich et al., 2007) for a summary of the message formulae, and we provide a detailed derivation in Appendix B.

One important implementation consideration of EP is the message passing schedule (Herbrich et al., 2007). There is no loop in all the graphical models from Figure 3.1 to 3.4. However, they all have a non-Gaussian factor: $\delta(\cdot > \varepsilon)$, which necessitates passing EP messages repeatedly on the graph. Realizing that the shortest paths between these

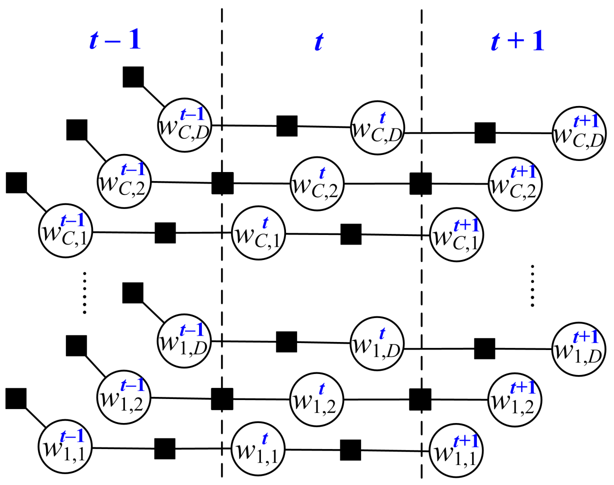


Figure 3.5: A dynamic graphical model with factors between the model of two adjacent examples.

non-Gaussian factors only involve factors $\{\alpha_c, \beta_c\}$ and variables $\{d_c\}$ and b (see Figure 3.4), we only need to run EP iteratively over b and $\{\alpha_c, d_c, \beta_c\}_c$. This significantly reduces the cost of each EP iteration from $O(DC)$ (for all weights) to $O(C)$[5]. In practice, since we only send messages from factors to variables, we just repeatedly do:

$$\alpha_1 \to d_1, \ldots, \alpha_5 \to d_5; \quad \beta_1 \to d_1, \ldots, \ \beta_5 \to d_5; \quad \alpha_1 \to b, \ldots, \alpha_5 \to b.$$

The termination criterion is that the relative difference of messages between two iterations fall below a given tolerance value for all messages. All messages are initialized to zero precision and zero precision-mean.

3.2.3 Dynamic models

So far we have not taken into account the need of different models for different *parts* of the dataset, *i.e.* temporal/spatial variations. This simplification may be unrealistic in many applications. For example, the categorization rule of Reuters news wire may change over the year, so our model needs to evolve through time accordingly. GDF also depends on the random order of training examples and the model information propagates only in the forward direction of the data stream. If the data can be stored, then we may add dynamic factors between the models of adjacent news article to allow smooth temporal variations. See Figure 3.5 for the dynamic graphical model and see (Dangauthier et al., 2008) for how TrueSkill$^{\text{TM}}$ can be extended to a dynamic scenarios. In this case, EP needs to be performed back and forth over the whole dataset. Theoretically appealing, it is very expensive in both time and space, and hence we stick

[5]After EP converges, it still takes $O(DC)$ complexity to record the final posterior.

to GDF in this work.

Our model also admits straightforward active learning, where in each iteration one picks a most informative training example, label it, and train on it. This can be useful when labeling is expensive. In this chapter, we focus on applications where a large number of labeled data is available, and then the bottleneck of computation shifts to finding the most informative training example. This usually requires applying the current model to the whole training set which is expensive, hence we would rather spend that time taking more updates considering its low cost in our model.

From now on, we will refer to our algorithm as Bayesian online multi-label classification (BOMC).

3.3 Generalization for multi-variate performance measure

After obtaining the posterior of weights $w_{c,d} \sim \mathcal{N}\left(\mu_{c,d}, \sigma_{c,d}^2\right)$ for class $c \in [C]$ and feature $d \in [D]$, together with a global threshold $b \sim \mathcal{N}(\mu_0, \sigma_0^2)$, the next task is to find a label in $2^{[C]}$ for test example $\mathbf{x}$. For simplification, unless explicitly highlighted we make predictions class by class and omit the class index c. Denote $\mathbf{w} := (w_1, ..., w_D)^\top$ (for class c whose index we omit) and similarly $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Suppose we are given a set of test data $X_{\text{test}} := \left\{\mathbf{x}^i \in \mathbb{R}^D : i \in [n]\right\}$. Let y^i be a Bernoulli random variable, and $y^i = 1$ means $\mathbf{x}^i$ belongs to class c and 0 otherwise[6]. Given an instantiation of $\mathbf{w}$ and b, we define the label y of a test example $\mathbf{x}$ depending on the sign of $\langle \mathbf{w}, \mathbf{x} \rangle - b$:

$$y := \delta(\langle \mathbf{w}, \mathbf{x} \rangle - b > 0), \quad \textit{i.e.} \quad p(y = 1|\mathbf{x}, \mathbf{w}, b) := \delta(\langle \mathbf{w}, \mathbf{x} \rangle - b > 0). \tag{3.4}$$

Therefore using the posterior of $\mathbf{w}$ and b we have

$$p(y = 1|\mathbf{x}) = \mathop{\mathbb{E}}_{(\mathbf{w},b)\sim p(\mathbf{w},b|X_{\text{train}},Y_{\text{train}})} [\delta(\langle \mathbf{w}, \mathbf{x} \rangle - b > 0)] = \Phi\left(\frac{\langle \boldsymbol{\mu}, \mathbf{x} \rangle - \mu_0}{\sqrt{\sigma_0^2 + \sum_d x_d^2 \sigma_d^2}}\right), \tag{3.5}$$

where Φ is the cumulative distribution of a standard normal distribution. A naïve decision criterion will then attach label c to $\mathbf{x}$ if $p(y = 1|\mathbf{x}) > 0.5$, or equivalently $\langle \boldsymbol{\mu}, \mathbf{x} \rangle - \mu_0 > 0$. However, there is no justification that 0.5 is the best threshold. In this work we will label the test data in a much more principled Bayesian fashion.

To this end, we study the joint distribution of all labels $\mathbf{y} := (y^1, \ldots, y^n)^\top$ and assume the testing data are labeled independently given the model, *i.e.*

$$y^i \perp\!\!\!\perp y^j | \mathbf{w}, b, \mathbf{x}^i, \mathbf{x}^j, \quad \text{and} \quad p(\mathbf{y}|\mathbf{w}, b, X_{\text{test}}) = \prod_{i=1}^n p\left(y^i|\mathbf{x}^i, \mathbf{w}, b\right).$$

[6]Not to be confused with the ground truth. y^i just represents the belief of our model and predictor.

However, the independence is lost after $\mathbf{w}$ and b are integrated out:

$$y^i \not\perp y^j \mid \mathbf{x}^i, \mathbf{x}^j, \quad \text{under} \quad p(\mathbf{y}|X_{\text{test}}) := \mathop{\mathbb{E}}_{(\mathbf{w},b)\sim p(\mathbf{w},b|X_{\text{train}},Y_{\text{train}})} [p(\mathbf{y}|\mathbf{w}, b, X_{\text{test}})] \tag{3.6}$$

The following sections will study how to label the testing data based on $p(\mathbf{y}|X_{\text{test}})$.

Incidentally, we could modify the definition Eq. (3.4) into a soft version:

$$p(y = 1|\mathbf{x}, \mathbf{w}, b) := \Phi(\langle \mathbf{w}, \mathbf{x} \rangle - b), \tag{3.7}$$

and hence

$$p(y = 1|\mathbf{x}) = \mathop{\mathbb{E}}_{(\mathbf{w},b)\sim p(\mathbf{w},b|X_{\text{train}},Y_{\text{train}})} [\Phi(\langle \mathbf{w}, \mathbf{x} \rangle - b)] = \mathop{\mathbb{E}}_{z\sim\mathcal{N}(\langle \boldsymbol{\mu},\mathbf{x}\rangle-\mu_0,\sigma_0^2+\sum_d x_d^2\sigma_d^2)} [\Phi(z)].$$

In this chapter, we will stick to definition Eq. (3.4) for simplicity.

3.3.1 Formulation of expected F-score

Labeling criteria must be designed to optimize some underlying performance measure. Let $\mathbf{l} \in \{0, 1\}^n$ be a reference labeling for n data points, and $\mathbf{y} \in \{0, 1\}^n$ be a predicted labeling. Some performance measures are additively decomposable such as accuracy:

$$\text{Accuracy}(\mathbf{y}, \mathbf{l}) := \frac{1}{n} \sum_{i=1}^{n} \delta(l^i = y^i),$$

while there also exist many undecomposable multi-variate performance measures, *e.g.* precision, recall, F-score, area under ROC curve, *etc.*. In such cases, the predicted labels must be optimized as a whole. For example, the F-score[7] is defined through the following sequence of performance measures:

$$\begin{aligned}
&\text{tp} := \text{true positive} := \sum_{i=1}^{n} l^i y^i & &\text{fn} := \text{false negative} := \sum_{i=1}^{n} (1 - y^i) l^i \\
&\text{fp} := \text{false positive} := \sum_{i=1}^{n} y^i (1 - l^i) & & \\
&\text{Pre} := \text{Precision} := \frac{\text{tp}}{\text{tp} + \text{fp}} & &\text{Rec} := \text{Recall} := \frac{\text{tp}}{\text{tp} + \text{fn}}
\end{aligned}$$

$$\text{F-score}(\mathbf{y}, \mathbf{l}) := 2 \frac{\text{Pre} \times \text{Rec}}{\text{Pre} + \text{Rec}} = \frac{2\ \text{tp}}{\text{tp} + \text{fp} + \text{fn}} = \frac{2 \sum_{i=1}^{n} y^i \cdot l^i}{\sum_{i=1}^{n} y^i + \sum_{i=1}^{n} l^i}. \tag{3.8}$$

[7] Strictly speaking, we use the F_1-score.

This measure is useful in many applications like information retrieval, where the class distribution is skewed and high accuracy can be achieved by blindly classifying all examples to the most common class. F-score, which is the harmonic mean of precision and recall, essentially reweights the positive and negative classes to the same importance, and encourages both of them to be accurately labeled.

Now that the true reference label is unknown, we simply check for each possible labeling $\mathbf{l} \in \{0,1\}^n$ how much the expected F-score is:

$$\mathrm{ExpFs}(\mathbf{l}) := \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y})} \left[\mathrm{F-score}(\mathbf{y}, \mathbf{l}) \right], \tag{3.9}$$

where $p(\mathbf{y})$ is the abbreviation of $p(\mathbf{y}|X_{\mathrm{test}})$ in Eq. (3.6). Finally, we output the maximizer of the expected F-score as the deterministic labeling, *i.e.*

$$\begin{aligned} \mathbf{l}^* &:= \operatorname*{argmax}_{\mathbf{l}\in\{0,1\}^n} \mathrm{ExpFs}(\mathbf{l}) = \operatorname*{argmax}_{\mathbf{l}\in\{0,1\}^n} \mathop{\mathbb{E}}_{\mathbf{y}\sim p(\mathbf{y})} \left[\mathrm{F-score}(\mathbf{y}, \mathbf{l}) \right] \\ &= \operatorname*{argmax}_{\mathbf{l}\in\{0,1\}^n} \mathop{\mathbb{E}}_{\mathbf{y}\sim p(\mathbf{y})} \left[\frac{\sum_{i=1}^n y^i l^i}{\sum_{i=1}^n y^i + \sum_{i=1}^n l^i} \right]. \end{aligned} \tag{3.10}$$

This principle of Bayesian labeling can be easily applied to other multi-variate performance measures. In the multi-label settings, each class has an F-score and an "average F-score" can be defined in two different ways. Suppose we have a set of reference labels $\{\mathbf{l}_c\}_c$ for all classes, and a set of proposed labels $\{\mathbf{y}_c\}_c$. Then we can average over the F-scores for all the classes, which is called "macro-average F-score":

$$\text{Macro-average F-score} := \frac{1}{C} \sum_{c=1}^{C} \{\text{F-score of class } c\} = \frac{2}{C} \sum_{c=1}^{C} \frac{\sum_{i=1}^n y_c^i \cdot l_c^i}{\sum_{i=1}^n y_c^i + \sum_{i=1}^n l_c^i}.$$

Another commonly used average F-score is the micro-average F-score. It first calculates the average of true positive, true negative and false positive, and then use these averages to compute the F-score:

$$\text{Micro-average F-score} := \frac{2\overline{\mathrm{tp}}_c}{\overline{\mathrm{tp}}_c + \overline{\mathrm{fp}}_c + \overline{\mathrm{fn}}_c} = \frac{2\sum_{i=1}^n \sum_{c=1}^C y_c^i \cdot l_c^i}{\sum_{i=1}^n \sum_{c=1}^C y_c^i + \sum_{i=1}^n \sum_{c=1}^C l_c^i}.$$

where

$$\overline{\mathrm{tp}} := \text{average true positive} := \frac{1}{C} \sum_{c\in[C]} \mathrm{tp}_c; \quad \overline{\mathrm{fn}} := \text{average false negative} := \frac{1}{C} \sum_{c\in[C]} \mathrm{fn}_c;$$

$$\overline{\mathrm{fp}} := \text{average false positive} := \frac{1}{C} \sum_{c\in[C]} \mathrm{fp}_c.$$

In general, micro-average F-score is more stable than macro-average F-score when some classes have very few positive examples. In that case, small changes in the predicted labels can make the F-score of those classes jump between 0 and 1.

The principle of Bayesian labeling can be applied here without change. Suppose we have a joint distribution $p\left(\{\mathbf{y}_c\}_c\right)$. Then the optimal labeling should be:

$$\operatorname*{argmax}_{l_c^i \in \{0,1\}} \mathop{\mathbb{E}}_{\{\mathbf{y}_c\}_c \sim p\left(\{\mathbf{y}_c\}_c\right)} \left[\text{Micro-average F-score}(\{\mathbf{y}_c\}_c, \{\mathbf{l}_c\}_c)\right].$$

This discrete optimization problem is very hard in general because the labels of all the classes are coupled in the numerator and denominator of micro-average F-score. In contrast, the macro-average F-score is additively decomposed into the classes, which allows us to find the optimal labels of all the classes independently using the marginal distributions:

$$\begin{aligned}
&\operatorname*{argmax}_{l_c^i \in \{0,1\}} \mathop{\mathbb{E}}_{\{\mathbf{y}_c\}_c \sim p\left(\{\mathbf{y}_c\}_c\right)} \left[\sum_{c \in [C]} \{\text{F-score of class } c \text{ with } (\mathbf{y}_c, \mathbf{l}_c)\}\right] \\
\Leftrightarrow\ &\operatorname*{argmax}_{l_c^i \in \{0,1\}} \sum_{c \in [C]} \mathop{\mathbb{E}}_{\mathbf{y}_c \sim p(\mathbf{y}_c)} [\text{F-score of class } c \text{ with } (\mathbf{y}_c, \mathbf{l}_c)] \\
\Leftrightarrow\ &\operatorname*{argmax}_{l_c^i \in \{0,1\}} \mathop{\mathbb{E}}_{\mathbf{y}_c \sim p(\mathbf{y}_c)} [\text{F-score of class } c \text{ with } (\mathbf{y}_c, \mathbf{l}_c)] \quad \forall\, c \in [C].
\end{aligned}$$

For a fixed class, F-score is a multi-variate measure which cannot be additively decomposed onto the data points. However, some other measures do admit such a decomposition: $\sum_{i=1}^n \text{loss}(y^i, l^i)$, and then the optimal labeling $\operatorname{argmax}_{\mathbf{l}} \mathbb{E}_{\mathbf{y}}[\sum_{i=1}^n \text{loss}(y^i, l^i)]$ can be found by optimizing on each data point separately based on the marginal distributions $p(y^i)$:

$$\max_{\mathbf{l}} \mathop{\mathbb{E}}_{\mathbf{y} \sim p(\mathbf{y})} \left[\sum_{i=1}^n \text{loss}(y^i, l^i)\right] \Leftrightarrow \max_{\mathbf{l}} \sum_{i=1}^n \mathop{\mathbb{E}}_{y^i \sim p(y^i)} \left[\text{loss}(y^i, l^i)\right] \Leftrightarrow \max_{l^i} \mathop{\mathbb{E}}_{y^i \sim p(y^i)} \left[\text{loss}(y^i, l^i)\right] \ \forall i.$$

In addition, when the loss is accuracy for binary classification, the rule of labeling becomes exactly thresholding $p(y^i = 1)$ at 0.5:

$$\begin{aligned}
\operatorname*{argmin}_{l^i \in \{0,1\}} \mathop{\mathbb{E}}_{y^i} \left[\text{loss}(y^i, l^i)\right] &= \operatorname*{argmin}_{l^i \in \{0,1\}} p(y^i = 1)\delta(l^i = 1) + p(y^i = 0)\delta(l^i = 0) \\
&= \delta(p(y^i = 1) > 0.5).
\end{aligned}$$

In general, closed form solutions rarely exist for optimizing multi-variate performance measures, and the expectation in Eq. (3.10) is intractable in the first place, bearing in mind that the space of $\mathbf{l}$ and $\mathbf{y}$ are exponentially large. The rest of Section 3.3 provides some practical approximate solutions.

Algorithm 7: A simple algorithm to maximize ExpFs($\mathbf{l}$).

```
1 for $r = 0$ to $n$ do
2     Find $\mathbf{l}_r := \text{argmax}_{\mathbf{l}:\|\mathbf{l}\|_1 = r} \text{ExpFs}(\mathbf{l})$
3 return $r^* := \text{argmax}_{r \in [n] \cup \{0\}} \text{ExpFs}(\mathbf{l}_r)$ and $\mathbf{l}_{r^*}$
```

3.3.2 Algorithms for maximizing expected F-score

To solve the discrete optimization problem in Eq. (3.10), it is helpful to study the most closely related algorithm: (Jansche, 2007). It relies on the assumption that $y^1, \ldots, y^n$ are independent. This assumption definitely does not hold in our case, but examining this algorithm gives useful insights.

Based on the expression of F-score in Eq. (3.10), the intuition of (Jansche, 2007) is to fix $\|\mathbf{l}\|_1 = \sum_i l^i$ to some value, and then $\mathbf{l}$ appears only in the numerator which makes optimization easier. We outline the idea in Algorithm 7.

The key step is step 2: find $\mathbf{l}_r := \text{argmax}_{\mathbf{l}:\|\mathbf{l}\|_1 = r} \text{ExpFs}(\mathbf{l})$. We now take a closer look. Given that $\|\mathbf{l}\|_1 = r$, we have

$$\text{ExpFs}(\mathbf{l}) = \mathop{\mathbb{E}}_{\mathbf{y} \sim p(\mathbf{y})} \left[\frac{\sum_{i=1}^n y^i l^i}{\sum_{i=1}^n y^i + r} \right] = \sum_{i=1}^n l^i \cdot \underbrace{\mathop{\mathbb{E}}_{\mathbf{y} \sim p(\mathbf{y})} \left[\frac{y^i}{\sum_{t=1}^n y^t + r} \right]}_{:= z^i} .$$

So we only need to sort z^i in decreasing order, and assign the l^i of the top r indices to 1. Although z^i is hard to compute as well, it clearly shows that the correlation among y^i plays an important role, because

$$\begin{aligned}
z^i &= \mathop{\mathbb{E}}_{\mathbf{y} \sim p(\mathbf{y})} \left[\frac{y^i}{\sum_{t=1}^n y^t + r} \right] = \sum_{\mathbf{y}: y^i = 1} \frac{y^i}{\sum_{t=1}^n y^t + r} p(\mathbf{y}) + \sum_{\mathbf{y}: y^i = 0} \frac{y^i}{\sum_{t=1}^n y^t + r} p(\mathbf{y}) \\
&= \sum_{\mathbf{y}: y^i = 1} \frac{y^i}{\sum_{t=1}^n y^t + r} p(\mathbf{y}) \\
&= \sum_{\mathbf{y}^{\setminus i}} \frac{1}{\sum_{t \neq i}^n y^t + 1 + r} p(y^i = 1) p(\mathbf{y}^{\setminus i} | y^i = 1) \quad (\mathbf{y}^{\setminus i} := (y^1, \ldots, y^{i-1}, y^{i+1}, \ldots, y^n)^\top) \\
&= p(y^i = 1) \sum_{s=0}^{n-1} \frac{1}{s + 1 + r} p \left(\left\| \mathbf{y}^{\setminus i} \right\|_1 = s | y^i = 1 \right) . \qquad (3.11)
\end{aligned}$$

So the conditional distribution $p \left(\left\| \mathbf{y}^{\setminus n} \right\|_1 | y^i = 1 \right)$ plays a very important part in the value of z^i, and the marginal probability $p(y^i = 1)$ is only one factor of z^i. Jansche (2007) only sorted $p(y^i = 1)$ because he assumed the independence of $\{y^i\}$ (Theorem 1 therein). Incidentally, we can also derive from Eq. (3.11) that if $\{y^i\}$ were independent,

Algorithm 8: One implementation of the step 2 of Algorithm 7

```
1 Sample $\mathbf{w}$ and $b$.
2 Obtain $\mathbf{y} = (y^1, \ldots, y^n)^\top$ using Eq. (3.4).
3 Repeat step 1 and 2 for many times to obtain many samples of $\mathbf{y}$. Compute
  $p\left(\left\|\mathbf{y}^{\backslash i}\right\|_1 | y^i = 1\right)$ by counting, for all $i \in [n]$ and $s = [n-1] \cup \{0\}$.
4 Compute $z^i$ by applying Eq. (3.11) for all $i \in [n]$.
5 $\max_{\mathbf{l}:\|\mathbf{l}\|_1 = r} \mathrm{ExpFs}(\mathbf{l})$ is exactly the sum of the $r$ greatest values of $z^n$ (in the
  derivation of $z^n$ in Eq. (3.11), we omitted the index $r$).
```

then the order of z^i would be exactly the order of $p(y^i = 1)$:

Proposition 28 *If $\{y^i\}$ are independent, then $p(y^i = 1) \geq p(y^j = 1)$ implies $z^i \geq z^j$.*

Proof First notice that independence implies $p\left(\left\|\mathbf{y}^{\backslash i}\right\|_1 = s | y^i = 1\right) = p\left(\left\|\mathbf{y}^{\backslash i}\right\|_1 = s\right)$. If $p(y^i = 1) \geq p(y^j = 1)$, then $z^i - z^j$ equals

$$
\begin{aligned}
& p(y^i = 1) \sum_{s=0}^{n-1} \frac{1}{s+1+r} \left(p(y^j = 1) p\left(\sum_{t \neq i,j} y^t = s-1 \right) + p(y^j = 0) p\left(\sum_{t \neq i,j} y^t = s \right) \right) \\
& - p(y^j = 1) \sum_{s=0}^{n-1} \frac{1}{s+1+r} \left(p(y^i = 1) p\left(\sum_{t \neq i,j} y^t = s-1 \right) + p(y^i = 0) p\left(\sum_{t \neq i,j} y^t = s \right) \right) \\
= & \sum_{s=0}^{n-1} \frac{1}{s+1+r} \left(p(y^i = 1) p(y^j = 0) - p(y^i = 0) p(y^j = 1) \right) p\left(\sum_{t \neq i,j} y^t = s \right) \\
= & \sum_{s=0}^{n-1} \frac{1}{s+1+r} (p(y^i = 1) - p(y^j = 1)) p\left(\sum_{t \neq i,j} y^t = s \right) \geq 0.
\end{aligned}
$$

■

Eq. (3.11) also provides a way to compute z^i, based on which we can implement the step 2 of Algorithm 7 to find the maximizer of ExpFs($\mathbf{l}$) subject to $\|\mathbf{l}\|_1 = r$. The conditional distribution $p\left(\left\|\mathbf{y}^{\backslash i}\right\|_1 | y^i = 1\right)$ can be estimated by sampling $\mathbf{w}$ and b. We formalize the whole idea in Algorithm 8.

Unfortunately, this approach is usually too expensive. The table of conditional probability $p\left(\left\|\mathbf{y}^{\backslash i}\right\|_1 | y^i = 1\right)$ costs $O(n^2)$ complexity in time and space. Sampling one set of $\mathbf{w}$ costs $O(DC)$ time which can be expensive in practice as well. Statistically, some concentration argument is also needed to bound the variance of the sampling.

Stochastic gradient descent

The objective function ExpFs($\mathbf{l}$) is in the form of expectation, which is amenable for stochastic gradient descent methods. We draw a sample of $\mathbf{w}$ and b which gives a sample $\mathbf{y}$, and then take a step of gradient descent based on this single sample. Asymptotically, convergence to the optimal may be provable, but more theoretical analysis is needed for this discrete problem. Fortunately, ExpFs($\mathbf{l}$) *is* convex in $\mathbf{l}$ after relaxing its domain from $\{0,1\}^n$ to $[0,1]^n$.

A heuristic approach to maximizing expected F-score

We have emphasized that the marginal probability $p(y^i = 1)$ is insufficient to optimize ExpFs($\mathbf{l}$), and the correlations between $\{y^i\}$ are important. However, correlations are expensive for both storage and computation. We finally resort to a simplified heuristic which respects the order of $p(y^i = 1)$ but tunes the threshold: for any arbitrary value of $\theta \in [0,1]$, we consider the class to be relevant if, and only if, $p(y^i = 1) > \theta$, where $p(y^i = 1)$ is given in Eq. (3.5). So the label $\mathbf{l}(\theta)$ is defined as a function of θ

$$\mathbf{l}(\theta) := (\delta(p(y^1 = 1) > \theta), \ldots, \delta(p(y^n = 1) > \theta))^\top. \tag{3.12}$$

Instead of maximizing ExpFs($\mathbf{l}$) over all $\mathbf{l} \in \{0,1\}^n$, we now find the final deterministic labeling by maximizing the expected F-score of $\mathbf{l}(\theta)$ wrt $\theta \in [0,1]$:

$$\theta^* := \operatorname*{argmax}_{\theta\in[0,1]} \ \text{ExpFs}(\mathbf{l}(\theta)). \tag{3.13}$$

In fact, ExpFs($\mathbf{l}(\theta)$) can assume at most n different values (jumping at $\theta = p(y^i = 1)$). This reduction of search space is significant, which makes optimization much easier. Another benefit is that we are no longer in the transductive setting, and can easily handle out-of-sample predictions. However, there is no guarantee that $\mathbf{l}(\theta^*)$ will recover the true $\mathbf{l}^* = \operatorname{argmax}_{\mathbf{l}} \text{ExpFs}(\mathbf{l})$, which may be not contained in the range of $\{\mathbf{l}(\theta) : \theta \in [0,1]\}$ in the first place.

Given θ, $\mathbf{l}(\theta)$ can be computed efficiently from Eq. (3.12) and (3.5). But there is still no closed form for ExpFs($\mathbf{l}$) defined by Eq. (3.9) due to the expectation operation. Therefore, we resort to approximate evaluation of ExpFs($\mathbf{l}$) based on samples $\{\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_S\}$ drawn *iid* from $p(\mathbf{y})$:

$$\widetilde{\text{ExpFs}}(\mathbf{l}) := \frac{1}{S}\sum_{s=1}^{S} \frac{\sum_{i=1}^n \tilde{y}_s^i l^i}{\sum_{i=1}^n \tilde{y}_s^i + \sum_{i=1}^n l^i}. \tag{3.14}$$

To draw sample $\tilde{\mathbf{y}}_s$ from $p(\mathbf{y})$, one only needs to draw sample $\tilde{\mathbf{w}}_s$ from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and

$\tilde{b}_s$ from $\mathcal{N}(\mu_0, \sigma_0^2)$, and set $\tilde{y}_s^i := \delta(\langle \mathbf{x}_i, \tilde{\mathbf{w}}_s \rangle - \tilde{b}_s > 0)$. However, a naïve application of Eq. (3.14) costs $O(nSCD)$ time, which is impractical for large datasets. We will design a more efficient algorithm in Section 3.3.4 using careful precomputation and buffering. Before that, we first justify the labeling criteria Eq. (3.13).

3.3.3 Soundness of approximate Bayesian labeling criteria

We call our labeling criteria $\mathbf{l}^* := \operatorname{argmax}_{\mathbf{l} \in \{0,1\}^n} \operatorname{ExpFs}(\mathbf{l})$ *sound* if $\mathbf{l}^*$ is "close" to the ground truth $\mathbf{y}^*$, as long as $p(\mathbf{w}, b | X_{\text{train}}, Y_{\text{train}})$ is well estimated. The meaning of "close" can be quantified in three possible forms:

1. $\mathrm{F-score}(\mathbf{l}^*, \mathbf{y}^*)$ is high.
2. $\operatorname{ExpFs}(\mathbf{y}^*)$ is close to the maximum of $\operatorname{ExpFs}(\mathbf{l})$.
3. $\|\mathbf{y}^* - \mathbf{l}^*\|_2$ is small.

However, since it is intractable to find $\mathbf{l}^*$, none of these criteria can be computed in practice, which leaves us unable to check the soundness of our exact labeling criteria.

Fortunately, it is possible to indirectly check the soundness of maximizing $\operatorname{ExpFs}(\mathbf{l}(\theta))$ over $\theta \in [0, 1]$. To this end, we simply enumerate θ in [0,1] with a small step size and plot two curves:

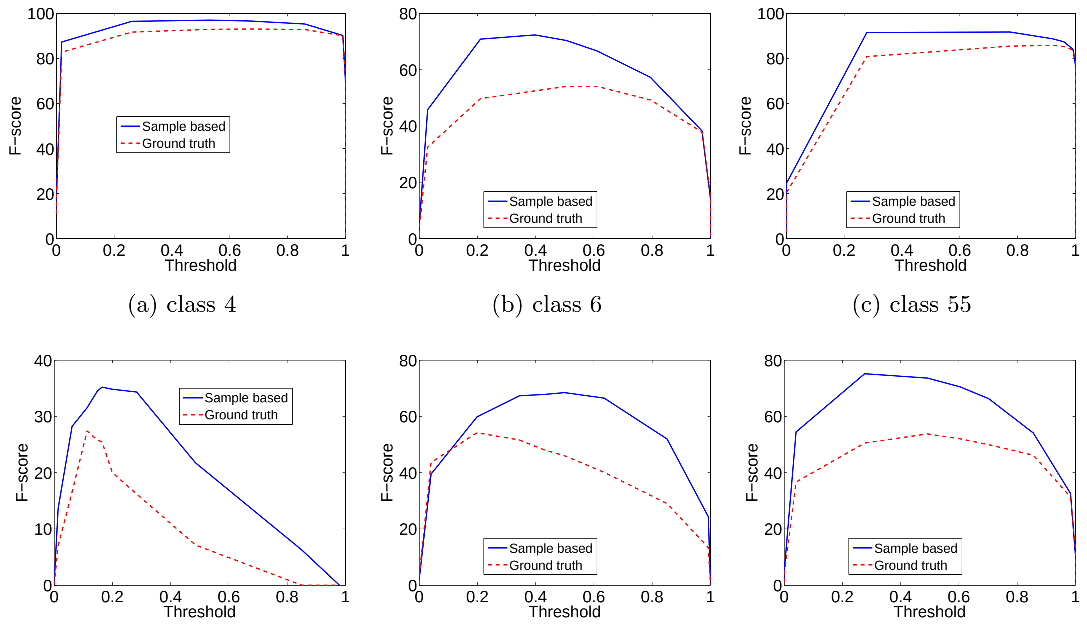
1. Expected F-score: $\operatorname{ExpFs}(\mathbf{l}(\theta))$ versus θ.
2. True F-score: $\text{F-score}(\mathbf{l}(\theta), \mathbf{y}^*)$ versus θ.

If these two figures are "similar", then it suggests that optimizing $\operatorname{ExpFs}(\mathbf{l}(\theta))$ over θ is a good proxy to maximizing the real testing F-score against the ground truth. In practice, $\operatorname{ExpFs}(\mathbf{l}(\theta))$ can only be evaluated approximately via samples of $\mathbf{w}$ and b, *i.e.* $\widetilde{\operatorname{ExpFs}}(\mathbf{l}(\theta))$ in Eq. (3.14), and accordingly we denote the sample based (empirical) optimal threshold by:

$$\tilde{\theta}^* := \operatorname*{argmax}_{\theta} \widetilde{\operatorname{ExpFs}}(\mathbf{l}(\theta))$$

A case study of the soundness of θ^* and $\tilde{\theta}^*$

We present an experimental result to compare $\widetilde{\operatorname{ExpFs}}(\mathbf{l}(\theta))$ and $\text{F-score}(\mathbf{l}(\theta), \mathbf{y}^*)$ as functions of θ. The example uses group = `topics` of Reuters dataset, 10,000 training examples, 100,000 testing examples, and 5 random samples. Noise $\beta^2 = 0.01^2$ in Eq. (3.1), and the prior of global bias has variance 100^2. Since there are 101 classes in all, we can just show some representative plots in Figure C.5 and the full set of 101 figures can be found in Appendix C. In all these figures, the red dotted curve represents



(a) class 4 (b) class 6 (c) class 55

(d) class 74 (e) class 84 (f) class 93

Figure 3.6: Example curves of $\widetilde{\mathrm{ExpFs}}(\mathbf{l}(\theta))$ (blue) and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (red) v.s. θ.

F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$, while the blue solid curve represents $\widetilde{\mathrm{ExpFs}}(\mathbf{l}(\theta))$. The horizontal axis is θ.

In these figures, both curves follow roughly similar trend. In fact, we do not need the maximizer of the blue and red curves to be similar, nor do we require the max of them to be similar. We only hope that the maximizer of the blue solid curve gives approximately the max of the red curve, *i.e.*

$$\text{F-score}(\mathbf{l}(\tilde{\theta}^*), \mathbf{y}^*) \qquad \text{close to} \qquad \max_\theta \text{F-score}(\mathbf{l}(\theta), \mathbf{y}^*).$$

And this is actually pretty much the case in this example. Numerically, the left hand term is 60.97 after summing over all the 101 classes, while the right hand term is 63.26.

3.3.4 Efficient calculation of empirical expected F-score

In the previous diagnosis, the curve F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ versus θ on a set of $\theta \in \{\theta_1, \ldots, \theta_G\}$ can be efficiently computed. However, $\widetilde{\mathrm{ExpFs}}(\mathbf{l}(\theta))$ is expensive in practice. We make a concrete analysis by using the Reuters dataset as an example, where number of class $C = 300$, number of feature $D = 5 \times 10^4$, number of test examples $n = 10^5$, average number of non-zero features per example $\bar{D} = 70$, and number of θ candidate $G = 20$.

1. Due to memory constraints, the testing data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ can only be read in an online fashion, and cannot be stored. In some cases, privacy or other accessibility constraints disallow us to revisit the testing examples. In some other cases, though revisiting the data is allowed, we can only afford at most a dozen of passes due to the computational cost of reading and parsing the data.

2. Sampling is also expensive in time and space. The $\mathbf{w}$ vector costs $O(CD)$ memory. For the Reuters dataset, it costs $8CD$ bytes $= 120$ MB. With regard to computational complexity, one sample takes $O(nC\bar{D})$ time to be applied to all the testing data, so the total cost is about 2×10^9. Therefore we can neither compute nor store more than a dozen samples of $\mathbf{w}$. So we let $S = 10$.

Taking into account the above constraints, we propose two efficient exact algorithms: one takes a single pass over the testing data and the other uses multiple passes. Both algorithms rely on careful buffering which can be best illustrated by writing out the empirical expected F-score in ground terms. For class c, combining the definitions in Eq. (3.12) and (3.14), we have

$$\widetilde{\text{ExpFs}}_c(\mathbf{l}(\theta_g)) = \frac{1}{S}\sum_{s=1}^{S} \frac{\overbrace{\sum_{i=1}^{n} \delta\left(\langle \mathbf{x}^i, \tilde{\mathbf{w}}_{s,c}\rangle - \tilde{b}_s > 0\right) \cdot \delta(p(y_c^i = 1) > \theta_g)}^{:=\alpha_{c,s,g}}}{\underbrace{\sum_{i=1}^{n} \delta\left(\langle \mathbf{x}^i, \tilde{\mathbf{w}}_{s,c}\rangle - \tilde{b}_s > 0\right)}_{:=\beta_{c,s}} + \underbrace{\sum_{i=1}^{n} \delta(p(y_c^i) = 1) > \theta_g)}_{:=\gamma_{c,g}}}.$$

Technically, we maintain three counters: $\alpha_{c,s,g}$, $\beta_{c,s}$ and $\gamma_{c,g}$. They are all cheap in space, costing $O(CSG)$ for $\boldsymbol{\alpha}$, $O(CS)$ for $\boldsymbol{\beta}$, and $O(CG)$ for $\boldsymbol{\gamma}$. $\boldsymbol{\gamma}$ does not depend on the samples, and can be computed efficiently. So the only problem left is $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

Single pass If we are only allowed to visit the test dataset for a single pass, then for each testing example, we must apply all the samples of $\mathbf{w}$. Since there is not enough memory to store all the weight samples, we have to regenerate these samples for every testing example. To ensure good statistical performance, all the testing examples need to "see" the same samples of $\mathbf{w}$, and therefore we store the seed of the random number generator for all the weight components. Algorithm 9 shows the whole algorithm, and it is essentially trading computations (of resampling weights) for IO pass.

Labeling could be done class by class which allows us to store all the weight samples of that class in memory. However, it will require reading through the testing data for C passes, which is forbidden or too expensive.

Algorithm 9: IO bounded computation of $\widetilde{\mathrm{ExpFs}}(\mathbf{l}(\theta))$ for $\theta \in \{\theta_g : g \in [G]\}$.

```
Input: A set of candidate thresholds $\theta \in \{\theta_g : g \in [G]\}$, bias $b \sim \mathcal{N}(\mu_0, \sigma_0^2)$, and
       posterior model $w_{c,d} \sim \mathcal{N}(\mu_{c,d}, \sigma_{c,d}^2)$ for class $c \in [C]$ and feature $d \in [D]$.
Output: $\widetilde{\mathrm{ExpFs}}_c(\mathbf{l}(\theta_g))$ for all $c \in [C]$ and $g \in [G]$.
1  Randomly generate seed $s_{c,d}$ for $c \in [C]$ and $d \in [D]$.
2  Draw iid random samples $\tilde{b}_1, \ldots, \tilde{b}_S$ from $\mathcal{N}(\mu_0, \sigma_0^2)$.
3  Clear buffer $\alpha_{c,s,g} = \beta_{c,s} = \gamma_{c,g} = 0$ for $c \in [C]$, $s \in [S]$, $g \in [G]$.
4  while there is still testing data do
5      Load the next test example $\mathbf{x}$, which has non-zero features $d_1, \ldots, d_F$.
6      for $c \in [C]$ (class index) do
7          $p(y_c = 1) \leftarrow \Phi\left(\frac{\langle \boldsymbol{\mu}_c, \mathbf{x}\rangle - \mu_0}{\sqrt{\sigma_0^2 + \sum x_d^2 \sigma_{c,d}^2}}\right)$ utilizing feature sparsity.
8          for $g \in [G]$ (threshold candidate index) do
9              Increment $\gamma_{c,g}$ by 1 if $p(y_c = 1) > \theta_g$.
10         Create random number generators $r_{d_1}, \ldots, r_{d_F}$ seeded by $s_{c,d_1}, \ldots, s_{c,d_F}$
           resp.
11         for $s \in [S]$ (sample index) do
12             for $d = d_1, d_2, \ldots, d_F$ (index of non-zero features) do
13                 Sample $\tilde{w}_{s,d} \sim \mathcal{N}(\mu_{c,d}, \sigma_{c,d}^2)$ using generator $r_{s,d}$.
14             if $\sum_d x_d \tilde{w}_{s,d} - \tilde{b}_s > 0$ (i.e., $y_c = 1$) then
15                 Increment $\beta_{c,s}$ by 1.
16                 for $g \in [G]$ (threshold candidate index) do
17                     Increment $\alpha_{c,s,g}$ by 1 if $p(y_c = 1) > \theta_g$.
18 for $c \in [C]$ (class index) and $g \in [G]$ (threshold candidate index) do
       Output: $\widetilde{\mathrm{ExpFs}}_c(\mathbf{l}(\theta_g)) = \frac{1}{S}\sum_{s=1}^{S} \frac{\alpha_{c,s,g}}{\beta_{c,s} + \gamma_{c,g}}$.
```

Multiple passes If the testing data can be visited for multiple passes, then we no longer need to regenerate weight samples. For each weight sample, we go through the whole testing data and update the counters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. Since only a dozen of samples are drawn, visiting the testing data for a dozen of passes is affordable. This algorithm is simpler than the single pass version, and we omit the details. Essentially, it trades multiple IO passes for the computational cost of regenerating samples.

Finally although 10 samples seem to be a very low number, the experimental results in Section 3.3.3 and 3.4.3 show that 5 samples already provide a pretty good, though approximate, characterization of how $\mathrm{ExpFs}(\mathbf{l}(\theta))$ and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ depend on θ, which allows us to find the optimal θ approximately. Remember for each weight sample, the whole testing dataset is used to compute the approximation. And we only need the mode of $\mathrm{ExpFs}(\mathbf{l}(\theta))$, which could probably be roughly captured with a small set of weight samples.

3.4 Empirical evaluation

In this section, we compare the empirical performance of several variants of BOMC with batch SVM and two state-of-the-art online learning classifiers. Our focus is on macro-average F-score and training time, and the dataset used is Reuters1-v2.

3.4.1 Dataset

The Reuters1-v2 dataset was studied by Lewis et al. (2004) and we downloaded the raw tokens from
`http://jmlr.csail.mit.edu/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm.`

Labels. The dataset consists of three groups of categories: `topics`, `industries`, and `regions`, which contain 103, 354, and 366 categories (or called classes/labels in machine learning terminology) respectively.

Examples. There are 804,414 documents (news wire) in the whole dataset. Every document is associated with zero or more labels from each of the three groups. In fact, all documents have at least one label from the `topics` group while many documents are not relevant to any label in the `industries` and `regions` group.

In the experiment, the training and testing sets were both sampled uniformly random from the whole dataset. We varied the number of training examples in $\{10^4, 2\times 10^4, 4\times 10^4, 8\times 10^4\}$, and this allowed us to plot curves. We also used different sizes of testing set in $\{10^5, 2\times 10^5, 4\times 10^5, 7\times 10^5\}$ which will correspond to different subfigures. Note Lewis et al. (2004) used 23,149 documents for training, and the rest 781,255 documents were used for testing.

Features. The feature representation of documents is basically `tf-idf`. Suppose we are given a training set $\mathcal{D}$, then the weight (feature value) of a token t in a document d is defined as follows:

$$w_d(t) = \underbrace{(1 + \log n(t,d))}_{tf} \times \underbrace{\log \frac{|\mathcal{D}|}{n(t)}}_{idf}.$$

where $n(t,d)$ is the number of occurrences of token t in document d, $n(t)$ is the number of documents in $\mathcal{D}$ that contain token t, and $|\mathcal{D}|$ is the number of the training documents. Unfortunately, this definition caters for the batch learning scenario, and is not immediately suitable for our online learning scenario. In particular, the `idf` vector

Table 3.1: Number of features under different sizes of `idf` set.

`ip (%)`	3	10	30	50
Number of features	58909	105269	172423	214887

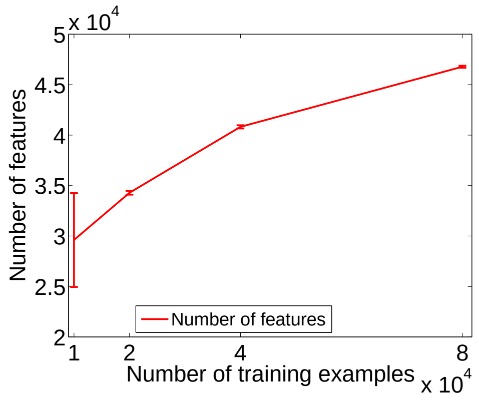


Figure 3.7: Total number of features in the training set versus the number of training examples.

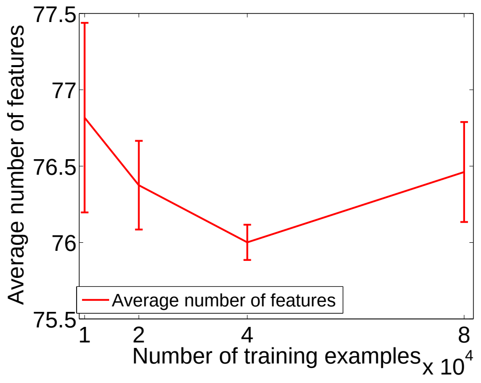


Figure 3.8: Average number of non-zero features per example versus the number of training examples.

$n(t)$ cannot be calculated offline because the training data (including both the feature and label) comes online. Inspired by (Lewis et al., 2004, Section 7)[8], we assume that some statistics about the features can be computed offline, including the `idf` vector. Importantly, their labels are still *unknown* and are revealed online. This `idf` vector can be computed from the whole dataset, a subset of it, or even borrowed from other sources of documents. We call that source as `idf` set $\mathcal{I}$. Our underlying assumption is that $\frac{|\mathcal{I}|}{n(t)}$ in $\mathcal{I}$ is similar to that in $\mathcal{D}$ for all tokens t, and it is more realistic for common tokens than for rare ones.

In our experiment, we computed the `idf` vector offline by using `ip` = 3%, 10%, 30%, 50% samples drawn uniformly random from the whole dataset. Note we did *not* require that $\mathcal{I}$ contain all possible tokens, i.e., $n(t) \geq 1$ for all t. So if a token did not appear in $\mathcal{I}$, then it was ignored even if it did appear later in the training set. Table 3.1 gives the number of features with respect to various values of `ip` under a draw of `idf` set. In practice, using `ip` = 50% does not noticeably improve the testing F-score compared with `ip` = 3%, therefore we will only present the results for `ip` = 3%.

For a specific random draw of training set, only the features and labels that appeared in the training set were considered. Figure 3.7 shows the number of features in the whole training set as a function of the number of training examples. The error

[8]Quote: "This (is) a legitimate use of additional unlabeled data. Only the document text from the additional documents was used, not their codes."

bars are based on 5 random draws of the training set.

Although there is a large number of features in the whole dataset, each data point (document) has only a small number of non-zero features, *i.e.*, tokens that appears in the document after pre-processing like punctuation removal, stemming, and stop word removal. On average there are 77 non-zero features in each data point (77 tokens in each document), and Figure 3.8 shows the average number of features per training example.

3.4.2 Algorithms

We compared different variants of BOMC with three state-of-the-art online learners.

BOMC was trained with the following settings.

1. The order of the training examples was randomized.

2. The prior of the feature weights were Gaussians $\mathcal{N}(0, 1)$. No class-wise bias was used. The single global bias had prior $\mathcal{N}(0, 10^4)$. The noise level in Eq. (3.1) is set to $\beta = 0.01$, reflecting the fact that text data lies in a high dimensional space which is pretty likely to be linearly separable.

3. EP was used for inference. The convergence criterion of EP was that the relative change of all messages fell below 10^{-5}. On average, it just took about 3 iterations for EP to converge.

In practice, many classes only have very few positive examples and over 90% examples are negative. This skewed ratio is commonly dealt with by two heuristics. The first approach tunes the threshold by using cross validation (CV) (Yang, 2001; Lewis et al., 2004; Fan & Lin, 2007). Intuitively it translates the original separating hyperplane towards the negative example region. However, CV is very expensive and relies heavily on the batch setting. The second approach requires more prior knowledge but is much cheaper. It uses different costs for misclassifying positive and negative examples, *e.g.* the "-j" parameter in SVM$^{\text{light}}$. Intuitively it increases the influence of less common classes. Lewis (2001) won the TREC-2001 Batch Filtering Evaluation by using this heuristic with SVM$^{\text{light}}$. Theoretically, Musicant et al. (2003) proved that such cost models approximately optimize F-score.

All the competing algorithm in our experiment perform very poorly when neither heuristic is used. Therefore we assume some prior knowledge such as the relative frequency of positive and negative examples (denoted by r). BOMC can encode this prior in the delta factor $\delta(\cdot > \epsilon)$. For negative examples, the loss factor is set to $\delta(d < -1)$, while for positive examples the loss factor is set to $\delta(d > \ln(e + 1/r))$.

BMOC with sampling (`BOMC_Sample`) To label the test data, we sampled from the posterior of the learned model as shown in Algorithm 9. 5 samples were drawn since the experiment showed that drawing 10 or 20 samples did not improve the F-score significantly. We call this algorithm `BOMC_Sample`.

Special care was required when a class was never "activated" by samples, *i.e.* for all test examples the inner product of feature and sampled weight being less than the sampled bias. Not being activated by 5 samples probably should not rule out the possibility of activating the class in the test set. Suppose the learned model is $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and $b \sim \mathcal{N}(\mu_0, \sigma_0)$, we set the threshold of that class to the maximum of membership probability (given by Eq. (3.5))

$$p(y = 1|\mathbf{x}) = \Phi\left(\frac{\langle \boldsymbol{\mu}, \mathbf{x}\rangle - \mu_0}{\sqrt{\sigma_0^2 + \sum_d x_d^2 \sigma_d^2}}\right),$$

over all testing examples $\mathbf{x}$.

BOMC with Class Mass Normalization (`BOMC_CMN`) A much simpler but non-Bayesian heuristic for picking the threshold is by matching the zero-th order moment: making the class ratio in the testing set identical to that in the training set. This heuristic was proposed by Zhu et al. (2003) to solve a similar threshold tuning problem in semi-supervised learning. Technically, we sorted this membership probability (Eq. (3.5)) of all testing examples in decreasing order, and labeled the top p percent to be positive, where p is the fraction of positive examples in the *training* set. This approach is called class mass normalization (CMN) by Zhu et al. (2003), so we call this variant of BOMC as `BOMC_CMN`.

BMOC: Training all classes independently (`BOMC_IND_CMN` and `BOMC_IND_Sample`) We also tried training each class independently, *i.e.* each class c has its own bias b_c and the shared global bias is no longer used. Now the posterior of each class can be computed in closed form for each training example. During testing, both CMN and sampling are again applicable, and hence called `BOMC_IND_CMN` and `BOMC_IND_Sample`, respectively.

All the variants of BOMC were implemented in F#[9], and can downloaded from `http://www.stat.purdue.edu/~zhang305/code/bomc.tar.bz2`.

Batch SVM (`SVM_Batch`) As a baseline for benchmark, we compared with SVM whose batch nature is an unfair advantage over BOMC as an online learner. We

[9]`http://research.microsoft.com/en-us/um/cambridge/projects/fsharp`

trained one SVM for each class independently. Since many classes in Reuters have very few positive examples, we applied the heuristic of using different cost for mislabeling positive and negative examples. The cost was chosen by 5 fold CV, and the final result was rather poor. So we tried the algorithm in (Yang, 2001) which tunes the threshold, and it yielded very competitive performance. The final algorithm relies highly on CV: besides using CV for picking the loss-regularization tradeoff parameter C, it also employs a nontrivial 2-level CV strategy to tune the bias of SVM (Fan & Lin, 2007). So in total, CV with k folds costs k^3 rounds. We call this method `SVM_Batch`.

As the 3-level CV is very expensive, our experiment used 3 folds for each level of CV, and so the underlying trainer was called for $3^3 + 1 = 28$ times. We tried 5 folds on some random samples of training and testing data and it gave almost the same result.

We used the `C` implementation of `liblinear` as the batch SVM solver[10], and wrote a Matlab script to deal with the multi-label data.

LaSVM (`LaSVM`) `LaSVM` is an online optimizer for SVM objective proposed by Bordes et al. (2005), who showed that by going through the dataset for a single pass, `LaSVM` achieves almost as good generalization performance as the batch SVM. Operating in the dual which allows nonlinear kernels, `LaSVM` maintains the active/support vector set, and employs a removal heuristic to avoid overfitting especially when the data is noisy. Strictly speaking, it is not a stream learner because it memorizes some data points (support vectors).

For our multi-label problem, we again trained all classes separately. The experiment showed that using different cost for positive and negative examples did not improve the testing F-score of `LaSVM` on imbalanced data, hence we resorted to the CV based strategy to tune the bias as in `SVM_Batch`. Due to the high computational cost of `LaSVM`, we only managed to use 2 folds for each level/parameter under CV. This means calling `LaSVM` for $2^3 + 1 = 9$ times.

We used the `C` implementation of `LaSVM`[11], and wrote a Matlab script for the multi-label scenario. Although only linear kernels are used here, this `LaSVM` implementation was not optimized for this specialization, hence inefficient.

Passive-Aggressive (`PA`) This online algorithm has been repeatedly proposed (under different names) for training SVMs, *e.g.* (Cheng et al., 2006; Crammer et al., 2006; Hsieh et al., 2008b). The idea is simple: given a current model $\mathbf{w}_t$ and a new training

[10] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

[11] `http://leon.bottou.org/projects/lasvm`

example $(\mathbf{x}_t, y_t)$, find a new $\mathbf{w}$ which minimizes

$$\mathbf{w}_{t+1} := \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2} \left\| \mathbf{w} - \mathbf{w}_t \right\|_{\mathcal{H}}^2 + \text{loss}(\mathbf{x}_t, y_t, \mathbf{w}). \tag{3.15}$$

`PA` does not naturally accommodate the bias in SVM. Hence we applied the same CV strategy used in `SVM_Batch` to find the optimal bias. Here, CV may either use `PA` or batch SVM, which we call `PA_OnlineCV` and `PA_BatchCV` respectively.

Due to the equivalence of `PA` and running one pass of liblinear, we simply used liblinear with the iteration number set to 1.

3.4.3 Performance measure

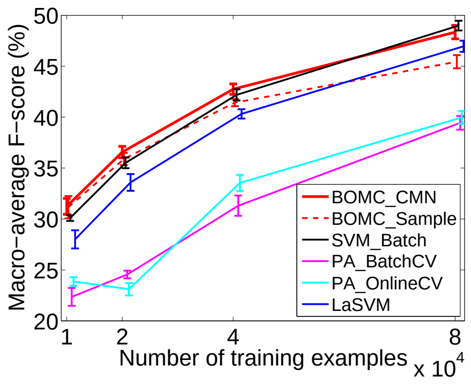
In this experiment, we compared the generalization performance in terms of macro-average F-score. It is useful for decision making and is easier for optimization due to the decoupling of classes.

If a class had no positive example in the training set, then it was ignored in testing. However if positive examples did appear in the training set but not in the testing set, special care was needed. With ground truth $\mathbf{y}$, the F-score of predicting $\mathbf{l}$ is defined as $\frac{\sum_i l^i y^i}{\sum_i l^i + \sum_i y^i}$. Now that all y^i are 0, the F-score becomes `NaN` if all l^i are 0. In information retrieval community, there has been some discussion on what the proper F-score should be in such a case. As we focus on machine learning, we simply ignored this class when computing the macro-average F-score. However, if some testing examples are erroneously predicted to be positive, *i.e.* $l_i = 1$, then the definition gives 0 F-score and we did take this 0 into the macro-average.

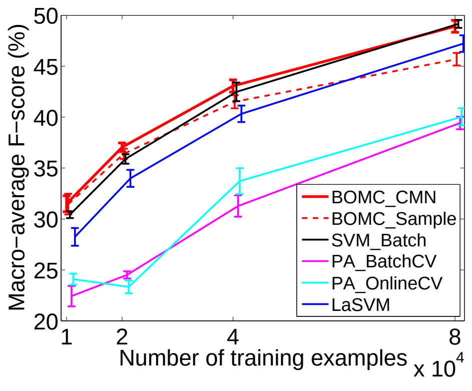
In addition, we compared the CPU time cost for training the algorithms. All competing algorithms of `BOMC` used CV for model selection and/or threshold tuning, so we only measured the time cost for training the final model after CV. In Matlab, CPU time can be simply obtained by the command `cputime`. In `C/C++`, `getrusage` was used to query the system and user time process of the current process. In F#, we called `Process.GetCurrentProcess().TotalProcessorTime` or `Sys.time`.

3.4.4 Results

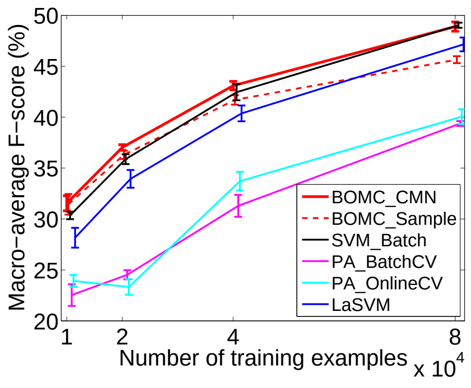
We compared the macro-average F-score and training time for the above algorithms. The random sampling of training and testing data was repeated for 5 times which allowed us to plot error bars.



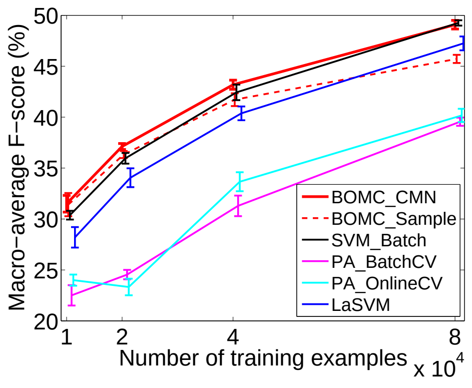
(a) #test = 100,000



(b) #test = 200,000



(c) #test = 400,000



(d) #test = 700,000

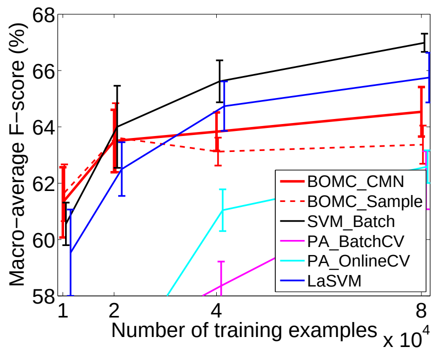
Figure 3.9: F-score for the category group `industries`.

Macro-average F-score

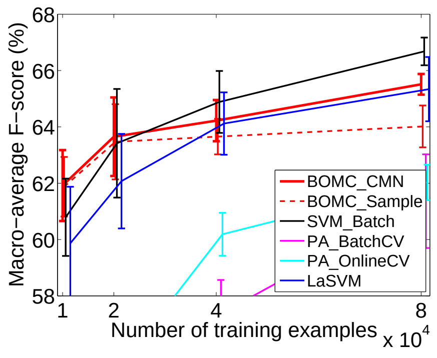
Figure 3.9 to Figure 3.11 show the macro-average F-score as a function of the number of training examples. The following observations can be drawn:

1. Among all online learners, `BOMC_CMN` achieves the highest macro-average F-score most of the time. Comparing Figure 3.10a and Figure 3.10d for the group `regions`, we observe that `BOMC_CMN` significantly benefits from a large size of testing set. This is not surprising because the class mass normalization rule assumes that the testing data has the same class ratio as in the training data, and this assumption is more likely to hold when the test set is large. In contrast, none of the other classifiers here label uses this assumption.

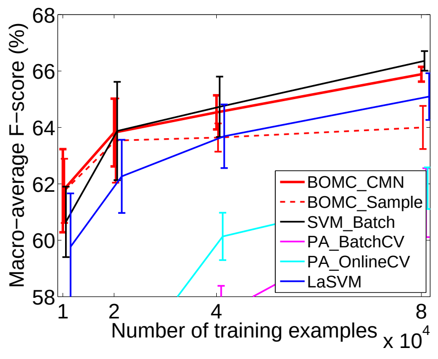
2. The macro-average F-score of `BOMC_Sample` is inferior to `BOMC_CMN`, but still competitive. Notice that CMN is also a method to choose the threshold, so it suggests that the training of the model is fine, and our sample based approach to threshold finding can be improved.



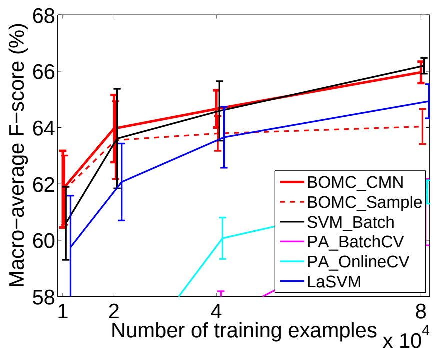
(a) #test = 100,000



(b) #test = 200,000

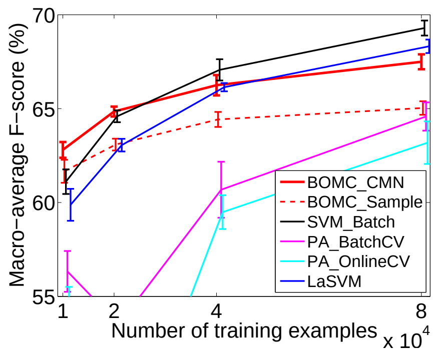


(c) #test = 400,000

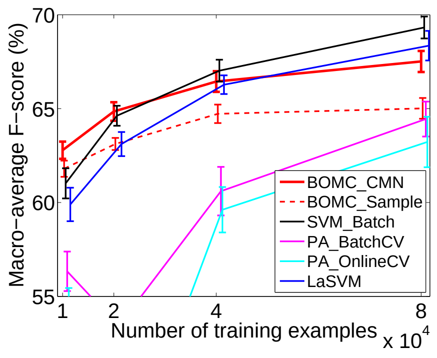


(d) #test = 700,000

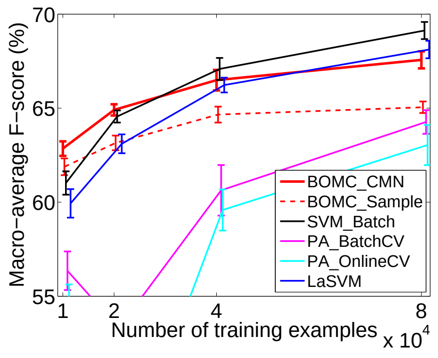
Figure 3.10: F-score for the category group `regions`.



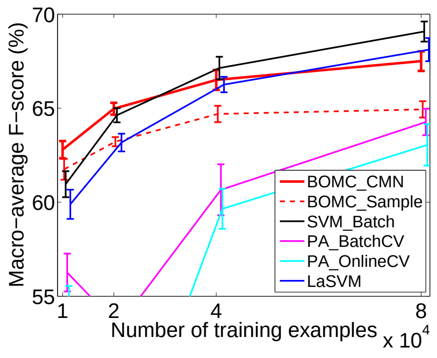
(a) #test = 100,000



(b) #test = 200,000



(c) #test = 400,000



(d) #test = 700,000

Figure 3.11: F-score for the category group `topics`.

3. Not surprisingly, `SVM_Batch` usually yields the highest F-score. However, `BOMC_CMN` often performs as well as or even better than `SVM_Batch` by a single pass, especially on the dataset `industries`, and on other datasets when the training set size is medium.

4. `PA_OnlineCV` and `PA_BatchCV` perform worse than other algorithms. One reason is probably that the noise in the data leads to a high number of support vectors in online learning, hence overfitting.

5. In contrast, `LaSVM` employs a removal step to handle noise, and provably converges to the true SVM solution if multiple passes are run. The macro-average F-score of `LaSVM` is slightly worse than `BOMC_CMN`, but competitive.

Comparing coupled and decoupled training for BOMC

It is easy to observe from Figure 3.4 that our multi-label model only loosely couples all the classes via the global bias. A natural question is why not introduce a "local" bias to all the classes and learn the model of all the classes independently. In this experiment we address this concern by comparing the macro-average F-score of BOMC trained in two different ways:

1. `BOMC_IND_CMN`: All classes have their own local bias and no global bias is used. Closed form inference is available for this case. We set the prior of all b_c to $\mathcal{N}(0, 10^4)$ which gave the highest macro-average F-score.

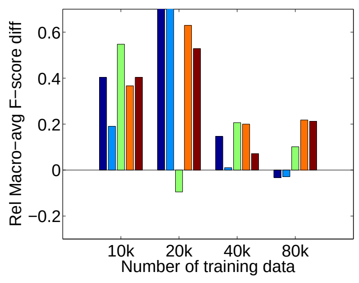
2. `BOMC_CMN`: All classes share a global bias and no local bias is used. EP was used for approximate inference;

Figure 3.12 to 3.14 demonstrate how much the macro-average F-score of `BOMC_CMN` (F_{coupled}) is relatively higher than that of `BOMC_IND_CMN` (F_{ind}):

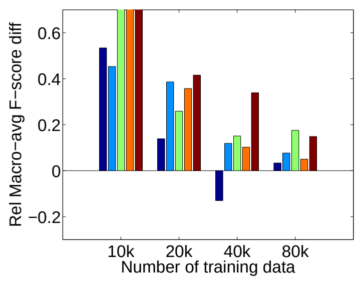
$$200 \times \frac{F_{\text{coupled}} - F_{\text{ind}}}{F_{\text{coupled}} + F_{\text{ind}}}.$$

It can be seen that on `industries` and `regions`, `BOMC_CMN` delivers significantly higher macro-average F-score than `BOMC_IND_CMN`. As we observed in the final learned model, the precision of the global bias in `BOMC_CMN` is much higher than that of the feature weights, and also higher than that of the local bias in `BOMC_IND_CMN`. This is no surprise because for each training example, the global bias serves as a hub for all the classes and is often updated. In contrast, due to feature sparsity, many feature weights are updated only on a few training examples, resulting in less confidence in the final posterior. On `topics` group, `BOMC_IND_CMN` slightly outperforms.

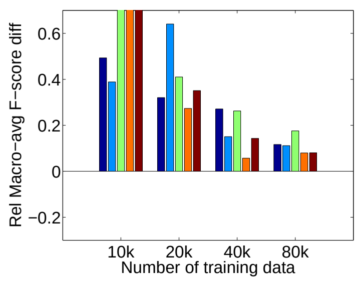
As we will discuss in Section 3.5, graphical models provide considerable flexibility in modeling important factors such as label noise and co-occurrence of labels or hierarchies



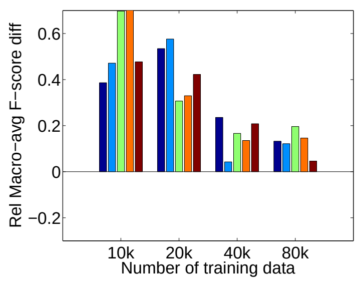
(a) #test = 100,000



(b) #test = 200,000

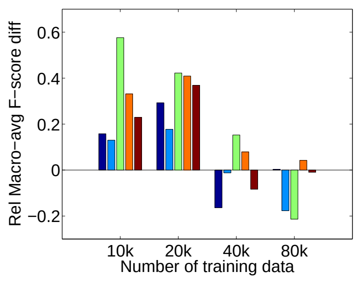


(c) #test = 400,000

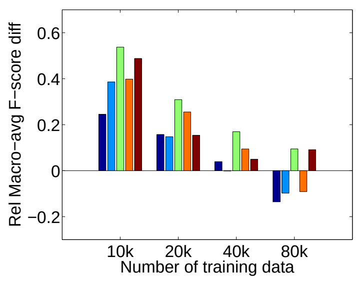


(d) #test = 700,000

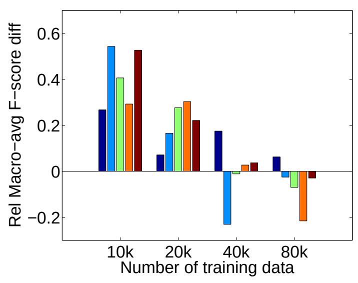
Figure 3.12: `industries`



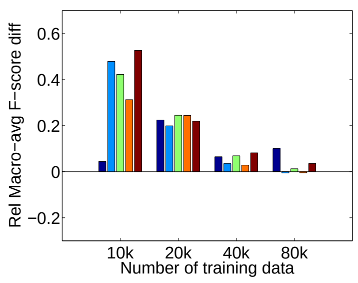
(a) #test = 100,000



(b) #test = 200,000

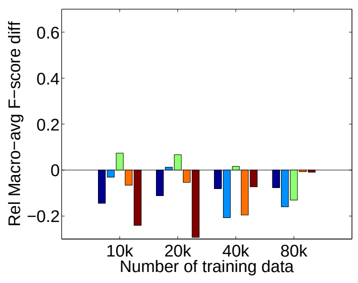


(c) #test = 400,000

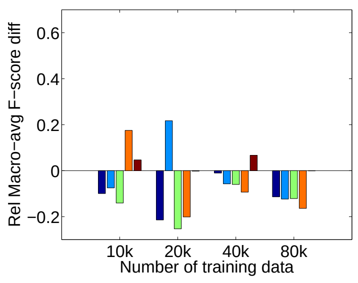


(d) #test = 700,000

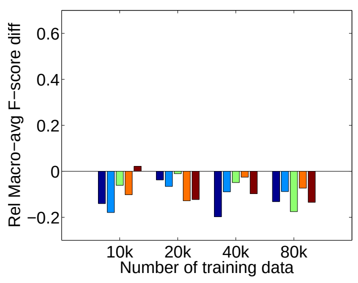
Figure 3.13: `regions`



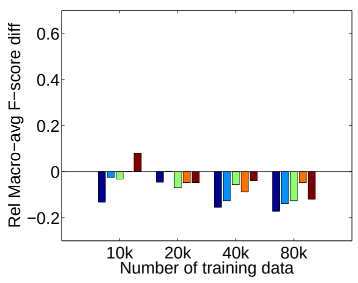
(a) #test = 100,000



(b) #test = 200,000



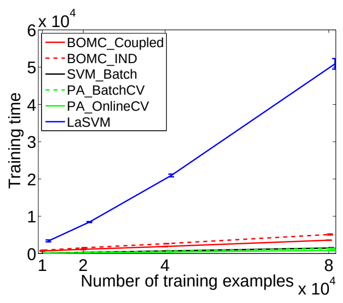
(c) #test = 400,000



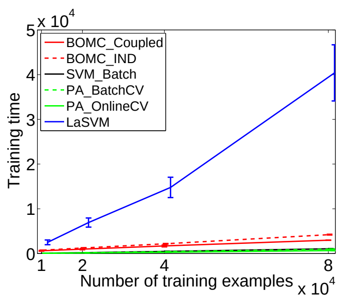
(d) #test = 700,000

Figure 3.14: `topics`

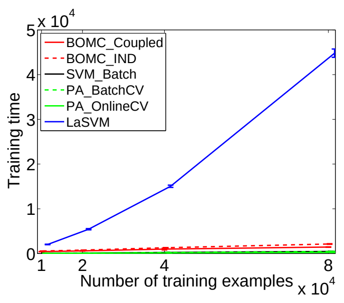
in the labels. The benefit of modeling them has been confirmed by existing algorithms such as (Rousu et al., 2006; Ghamrawi & McCallum, 2005).



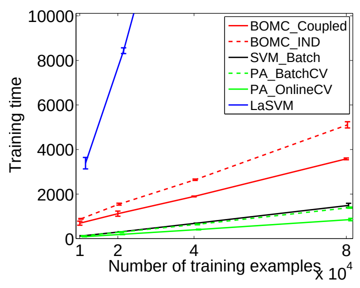
(a) `industries`



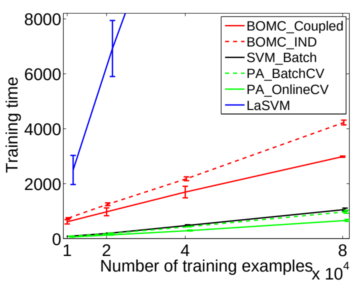
(b) `regions`



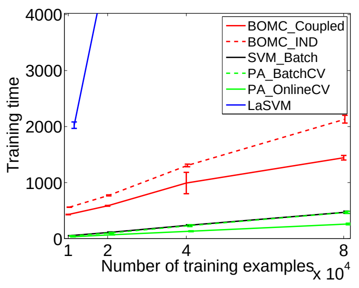
(c) `topics`



(d) `industries` (zoom in)



(e) `regions` (zoom in)



(f) `topics` (zoom in)

Figure 3.15: CPU time for training.

Training Time

Figure 3.15 presents the CPU time cost for training these algorithms, under various number of training examples.

The most important observation is that the training time of all algorithms except `LaSVM` is linear in the number of training examples. This matches the theoretical property. Training `BOMC_IND_CMN` takes slightly more time than `BOMC_CMN`. On the one hand, the inference step in `BOMC_IND_CMN` can be conducted in closed form without running EP repeatedly. On the other, `BOMC_IND_CMN` uses local bias for each class while `BOMC_CMN` uses only one global bias, and EP converges in just 3 iterations on average. Empirically the latter factor seems to be more influential. Passive-Aggressive and batch SVM can be trained faster than `BOMC` by a factor of 2–3. This is probably because they are implemented in pure C while `BOMC` was written in F#.

Although `LaSVM` is the online learner which achieves closest testing F-score to `BOMC`, it takes a lot of time for training and is slightly super-linear in the training set size. This is because it operates in the dual and has not been customized/optimized for the linear kernels.

3.5 Conclusion and future directions

In this chapter, we proposed a Bayesian learning algorithm for multi-label classification. The model is assumed to be a probabilistic linear classifier, and training is based on Gaussian density filtering. It is online, efficient, and allows the model to be trained incrementally. In contrast to ad hoc thresholding schemes used in frequentist approaches like SVM, it labels unseen data in a much more principled manner, namely maximizing the expected F-score by using samples from the posterior of the model. Empirically, our method delivers state-of-the-art macro-average F-score compared with batch SVM, `LaSVM`, and passive-aggressive updates. The method is also efficient in time and space.

In the future, the following three extensions are straightforward.

1. Dynamic training with EP through the dataset. It is expected to learn a better model at a higher cost.

2. Model the label noise. We used to assume that the label is just thresholding at 0, *i.e.* $y = \delta(f \geq 0)$ in Eq. (3.4) where f is the linear discriminant. However, labels can also be noisy, and to take this into account, the models proposed by Kim & Ghahramani (2006) can be easily incorporated into our framework by simply changing the factor $\delta(f \geq \epsilon)$ into

$$p(y|f) := \begin{cases} \frac{1}{1+\exp(-(2y-1)f)} & \text{sigmoid} \\ \Phi((2y-1)f) & \text{cumulative normal} \\ \rho + (1-2\rho)\delta((2y-1)f \geq 0) & \text{noisy threshold} \end{cases}$$

   where the noise $\rho \in [0, 0.5)$. For example, a common type of label noise is the flipping noise where the observed label simply flips the correct label. This can be due to typos. To model it, we simple replace the factor $\delta(d > \epsilon)$ by $\rho\delta(d > \epsilon) + (1-\rho)\delta(d < -\epsilon)$.

3. Modeling label hierarchies. We propose a model in Appendix D.

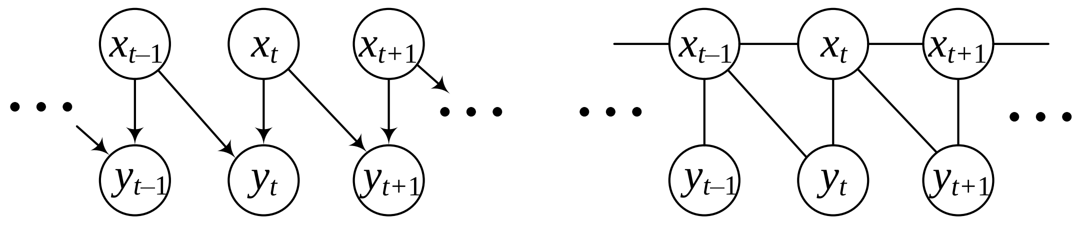
Chapter 4

Kernel Measures of Independence for non-*iid* Data

Statistical dependence measures have been proposed as a unifying framework to address many machine learning problems. For instance, clustering can be viewed as a problem where one strives to maximize the dependence between the observations and a discrete set of labels (Song et al., 2007b). Conversely, if labels are given, feature selection can be achieved by finding a subset of features in the observations which maximize the dependence between labels and features (Song et al., 2007c). Similarly in supervised dimensionality reduction (Song et al., 2008a), one looks for a low dimensional embedding which retains additional side information such as class labels. Likewise, blind source separation (BSS) tries to unmix independent sources, which requires a contrast function quantifying the dependence of the unmixed signals.

The use of mutual information is well established in this context, as it is theoretically well justified. Unfortunately, it typically involves nontrivial intermediate steps such as density estimation, space partitioning (Learned-Miller, 2004), bias correction (Stögbauer et al., 2004; Nemenman et al., 2002), etc. These operations often require sophisticated optimization procedures (Nguyen et al., 2008) or are not well underpinned in theory for high dimensional data (Learned-Miller, 2004). In addition, most methods work only for distributions in Euclidean spaces. Borgwardt & Ghahramani (2009) used flexible models like Dirichlet process mixtures to test independence, which encodes the inductive bias in terms of probability distributions (Dirichlet distribution). However, it is essentially still based on density estimation and requires sophisticated graphical model inference algorithms.

These problems can be averted by using the Hilbert Schmidt Independence Criterion (HSIC). It can be computed directly from the dataset without needing any intermediate step. It also enjoys concentration of measure properties and can be computed efficiently on any domain where a reproducing kernel Hilbert space (RKHS) can be defined. The RKHS essentially encodes the inductive bias, and allows the algorithm designers to



(a) Directed model for XOR with causality (b) Moralized undirected graphical model

Figure 4.1: Graphical model for the XOR problem $y_t = x_t \otimes x_{t-1}$.

focus on the properties of a distribution that are most important to their problems.

However, the application of HSIC is limited to independent and identically distributed (*iid*) data, a property that many problems do not share (*e.g.*, BSS on audio data). For instance many random variables have a pronounced temporal or spatial structure. A simple motivating example is given in Figure 4.1. Assume that the observations x_t are drawn *iid* from a uniform distribution on $\{0, 1\}$ and y_t is determined by an XOR operation via $y_t = x_t \otimes x_{t-1}$. Algorithms which treat the observation pairs $\{(x_t, y_t)\}_{t=1}^{\infty}$ as *iid* will consider the random variables x, y as independent. However, it is trivial to detect the XOR dependence by using the information that x and y are, in fact, sequences.

In view of its importance, temporal correlation has been exploited in the independence test for blind source separation. For instance, Hosseni & Jutten (2003) used this insight to reject nontrivial nonseparability of nonlinear mixtures, and Ziehe & Müller (1998) exploited multiple time-lagged second-order correlations to decorrelate over time.

These methods work well in practice. But they are rather *ad hoc* and appear very different from standard criteria. In this paper, we propose a framework which extends HSIC to structured non-*iid* data. Our new approach is built upon the connection between exponential family models and the marginal polytope in an RKHS. This is doubly attractive since distributions can be uniquely identified by the expectation operator in the RKHS and moreover, for distributions with conditional independence properties the expectation operator decomposes according to the clique structure of the underlying undirected graphical model (Altun et al., 2004b).

In this chapter, we will first survey the existing works on Hilbert space embeddings of distributions and kernel measures of independence for *iid* data (Section 4.1). Of central importance is the empirical estimators, especially their computational efficiency and statistical properties such as concentration of measure and sample efficiency. This framework allows us to further decompose the probability embeddings and the kernel independence criterion when the distribution and kernel factorize with respect to a graphical model. Accordingly, the empirical estimators can also be decomposed onto the cliques which we will demonstrate in Section 4.2. Example estimators for typical graphical models and various choices of kernels will be given in Section 4.3, where we also show how homogeneity and stationarity can improve the efficiency of estimation. Interestingly, for first order Markov chains, the asymptotic bounds on the concentration of measure can be obtained under mild regularity conditions. Finally, Section 4.4 will provide the experimental results on independence test for non-*iid* data, ICA, and sequence segmentation. In all these problems, methods taking into account the inter-dependence of observations significantly outperform those treating them as *iid*.

4.1 Preliminaries of RKHS embeddings of probabilities

Many theoretical and practical problems involve comparison of distributions, and a lot of measures have been proposed over the years, such as KL divergence, total variance, dynamic range, L_p norm, and earth mover's distance. However, we usually only have samples of the distributions and therefore density estimation is often required before these measures can be applied. This adds more complexity, and it is desirable for a measure to be directly estimated from samples.

One solution is the discrepancy in moments, which can be directly estimated from samples. More generally, we can compare the mean of distributions $\mathbf{P}$ and $\mathbf{Q}$ on a class of *touchstone* functions $\mathcal{F}$. This is especially suitable for many applications where distributions matter only via their expectations, and the same idea motivated the definition of weak convergence for random variables, where bounded continuous functions are used. With a kernel k and its induced RKHS $\mathcal{H}$, Shawe-Taylor & Dolia (2007) proposed using samples from the unit ball of $\mathcal{H}$ as the touchstone function class, and Song et al. (2008b) used the sup of the mean discrepancy over this ball. The advantage of the latter is two folds: a) it can be easily estimated from samples of $\mathbf{P}$, $\mathbf{Q}$ with tight concentration, and b) the space of RKHS can be rich enough to capture all the high order moments of the distributions. Intuitively, we measure:

$$\begin{aligned}
&\sup_{f\in\mathcal{H}:\|f\|\le 1}\left|\mathop{\mathbb{E}}_{x\sim\mathbf{P}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbf{Q}}[f(x)]\right| = \sup_{f\in\mathcal{H}:\|f\|\le 1}\left|\int f(x)\mathrm{d}\mathbf{P} - \int f(x)\mathrm{d}\mathbf{Q}\right| \\
= &\sup_{f\in\mathcal{H}:\|f\|\le 1}\left|\int \langle f, k(x,\cdot)\rangle\,\mathrm{d}\mathbf{P} - \int \langle f, k(x,\cdot)\rangle\,\mathrm{d}\mathbf{Q}\right| \quad \text{(Using the reproducing property)} \\
= &\sup_{f\in\mathcal{H}:\|f\|\le 1}\left|\left\langle f, \int k(x,\cdot)\mathrm{d}\mathbf{P} - \int k(x,\cdot)\mathrm{d}\mathbf{Q}\right\rangle\right| \\
= &\sup_{f\in\mathcal{H}:\|f\|\le 1}\left|\left\langle f, \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[k(x,\cdot)] - \mathop{\mathbb{E}}_{x\sim\mathbf{Q}}[k(x,\cdot)]\right\rangle\right|.
\end{aligned}$$

The last form indicates a key object that captures the property of a distribution $\mathbf{P}$: $\mathbb{E}_{x\sim\mathbf{P}}[k(x,\cdot)]$. It gives the expectation of any arbitrary function by a simple inner product. This motivates the abstraction of mean mapping which we formulate in this section.

Given a distribution $\mathbf{P}$ on domain $\mathcal{X}$, define a mean operator $T_{\mathbf{P}} : \mathcal{H} \to \mathbb{R}$ as

$$T_{\mathbf{P}}(f) = \mathbb{E}_{x\sim\mathbf{P}}[f(x)].$$

$T_{\mathbf{P}}$ is obviously linear. To check whether it is bounded, notice

$$\sup_{f:\|f\|_{\mathcal{H}}=1} \left| \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[f(x)] \right| = \sup_{f:\|f\|_{\mathcal{H}}=1} \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[|\langle f, k(x,\cdot)\rangle_{\mathcal{H}}|] \leq \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[\|k(x,\cdot)\|_{\mathcal{H}}] = \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[\sqrt{k(x,x)}].$$

Hence, $T_{\mathbf{P}}$ is bounded if $\mathbb{E}_{x\sim\mathbf{P}}[\sqrt{k(x,x)}] < \infty$, and by Rietz representer theorem, there exists an element $\mu[\mathbf{P}] \in \mathcal{H}$ such that

$$\langle \mu[\mathbf{P}], f\rangle = T_{\mathbf{P}}(f) = \mathbb{E}_{x\sim\mathbf{P}}[f(x)] \quad \text{for all } f \in \mathcal{H}.$$

Formally, we define:

Definition 29 (Mean operators) *Let $\mathbf{P}$ be a distribution on domain $\mathcal{X}$ and k be a kernel on $\mathcal{X}$ with RKHS $\mathcal{H}$. If $\mathbb{E}_{x\sim\mathbf{P}}[\sqrt{k(x,x)}] < \infty$ then there exists an element $\mu[\mathbf{P}] \in \mathcal{H}$ such that for all $f \in \mathcal{H}$ we have $\langle \mu[\mathbf{P}], f\rangle = \mathbb{E}_{x\sim\mathbf{P}}[f(x)]$, and we call $\mu[\mathbf{P}]$ the mean element of $\mathbf{P}$.*

If we have a finite set of samples $x_1, \ldots, x_n$ from $\mathcal{X}$, then the empirical mean element is defined by $\frac{1}{n}\sum_{i=1}^{n} k(x_i,\cdot)$, which is obviously in $\mathcal{H}$.

Note $\mu[\mathbf{P}]$ maps a function $f \in \mathcal{H}$ to its mean under $\mathbf{P}$ (a scalar), while μ maps a distribution to $\mathcal{H}$. We will call $\mu[\mathbf{P}]$ the *mean element* of $\mathbf{P}$, and call μ the *mean operator*. From definition, the following useful property is immediate:

Property 1 *If $\mathbb{E}_{x\sim\mathbf{P}}[\sqrt{k(x,x)}] < \infty$ then*

$$\langle \mu[\mathbf{P}], \mu[\mathbf{Q}]\rangle = \mathop{\mathbb{E}}_{x\sim\mathbf{P}} \mathop{\mathbb{E}}_{x'\sim\mathbf{Q}}[k(x,x')], \quad \textit{in particular } \|\mu[\mathbf{P}]\|^2 = \mathop{\mathbb{E}}_{x\sim\mathbf{P}} \mathop{\mathbb{E}}_{x'\sim\mathbf{P}}[k(x,x')].$$

Proof

$$\langle \mu[\mathbf{P}], \mu[\mathbf{Q}]\rangle = \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[\mu[\mathbf{Q}](x)] = \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[\langle \mu[\mathbf{Q}], k(x,\cdot)\rangle] = \mathop{\mathbb{E}}_{x\sim\mathbf{P}} \mathop{\mathbb{E}}_{x'\sim\mathbf{Q}}[k(x,x')].$$

Incidentally, notice that $\|\mu[\mathbf{P}]\|^2 \neq \mathbb{E}_{x\sim\mathbf{P}}[k(x,x)]$ in general. ■

Remark 30 *It is possible to define $\mu[\mathbf{P}]$ in the following more intuitive form:*

$$\mu[\mathbf{P}] = \int k(x, \cdot) \, \mathrm{d}\mathbf{P},$$

but then the proof of well-definedness for integral in the RKHS is more involved than our definition. To this end, a sufficient condition required by Smola et al. (2007a) is that $\mathbb{E}_x[k(x,x)] < \infty$. However, this condition is strictly stronger than our $\mathbb{E}_x[\sqrt{k(x,x)}] < \infty$. For example if $\mathbf{P}$ has density $p(x) \propto \frac{1}{|x|^3+1}$ on $\mathbb{R}$ and a linear kernel is used, then $\mathbb{E}_x[\sqrt{k(x,x)}] < \infty$ but $\mathbb{E}_x[k(x,x)] = \infty$.

We naturally want $\mu[\mathbf{P}]$ to be a signature of $\mathbf{P}$, *i.e.*, different distributions have different mean elements in $\mathcal{H}$. This is obviously not true for linear kernels because different distributions can easily have the same mean. So more assumptions are needed such as in the following theorem.

Theorem 31 (Injectivity of mean operator) *If the kernel k is universal (see Definition 15), then the mean operator $\mathbf{P} \mapsto \mu[\mathbf{P}]$ is injective.*

A simple example is Gaussian RBF kernels. Let $\alpha_i, \alpha'_i \in \mathbb{R} \backslash \{0\}$ and $\mathbf{x}_i, \mathbf{x}'_i \in \mathbb{R}^d$ where $\{\mathbf{x}_i\}_{i=1}^n$ are distinct and $\{\mathbf{x}'_i\}_{i=1}^m$ are distinct, then $\sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \cdot) = \sum_{i=1}^m \alpha'_i k(\mathbf{x}'_i, \cdot)$ iff $m = n$ and $\{(\alpha_i, \mathbf{x}_i)\}_{i=1}^n$ is a permutation of $\{(\alpha'_i, \mathbf{x}'_i)\}_{i=1}^n$.

In analogy to the characteristic functions of distributions, a kernel is called *characteristic* if its induced mean operator is injective from the whole set of distributions on $\mathcal{X}$. Universality on compact domains is a sufficient but not necessary condition of characteristicity. More properties on characteristic kernels can be found in (Sriperumbudur et al., 2008; Fukumizu et al., 2009).

Empirical estimation

If we have n *iid* samples $X_1^n := \{X_i\}_{i=1}^n$ of $\mathbf{P}$, a natural estimator of $\mu[\mathbf{P}]$ is

$$\mu[X_1^n] := \frac{1}{n} \sum_{i=1}^n k(X_i, \cdot).$$

To quantify the concentration of measure for $\mu[X_1^n]$, we first note the following relationship:

$$\sup_{f:\|f\|\leq 1} \left| \mathbb{E}_X[f(X)] - \frac{1}{n}\sum_{i=1}^n f(X_i) \right| = \sup_{f:\|f\|\leq 1} |\langle \mu[\mathbf{P}] - \mu[X_1^n], f \rangle| = \| \mu[\mathbf{P}] - \mu[X_1^n] \| .$$

The LHS is exactly the uniform deviation in statistical estimation (see Appendix E.5). Therefore, the results like Eq. (E.14) translate directly to the following concentration:

Theorem 32 (Concentration of measure) *Assume $f \in [0, 1]$ for all $f \in \mathcal{H}_1 := \{g \in \mathcal{H} : \|g\| \leq 1\}$. Given n* iid *samples $X_1^n = \{X_1, \ldots, X_n\}$ of $\mathbf{P}$, with probability $1 - \delta$ we have*

$$\|\mu[\mathbf{P}] - \mu[X_1^n]\| \leq R_n(\mathcal{H}) + \sqrt{\frac{\log(2/\delta)}{2n}} \leq \frac{2}{n}\mathbb{E}\left[\sqrt{\sum_{i=1}^{n} k(X_i, X_i)}\right] + \sqrt{\frac{\log(2/\delta)}{2n}},$$

where $R_n(\mathcal{H})$ is the Rademacher average of function space $\mathcal{H}$ (see definition in Eq. (E.13)). This procedure is in complete analogy to the Glivenko-Cantelli lemma, which is used to bound the deviations between empirical and expected means of functions, and at the same time gives rise to the Kolmogorov-Smirnov statistics for comparing distributions.

4.1.1 Distance between distributions

The embedding $\mu[\mathbf{P}]$ in the Hilbert space $\mathcal{H}$ immediately induces a distance between two distributions $\mathbf{P}$ and $\mathbf{Q}$:

$$\begin{aligned} D(\mathbf{P}, \mathbf{Q})^2 &:= \|\mu[\mathbf{P}] - \mu[\mathbf{Q}]\|^2 = \|\mu[\mathbf{P}]\|^2 + \|\mu[\mathbf{Q}]\|^2 - 2\langle \mu[\mathbf{P}], \mu[\mathbf{Q}]\rangle \\ &= \mathop{\mathbb{E}}_{x\sim\mathbf{P}} \mathop{\mathbb{E}}_{x'\sim\mathbf{P}} \left[k(x, x')\right] + \mathop{\mathbb{E}}_{y\sim\mathbf{Q}} \mathop{\mathbb{E}}_{y'\sim\mathbf{Q}} \left[k(y, y')\right] - 2 \mathop{\mathbb{E}}_{x\sim\mathbf{P}} \mathop{\mathbb{E}}_{y\sim\mathbf{Q}} \left[k(x, y)\right]. \end{aligned} \tag{4.1}$$

In view of

$$D(\mathbf{P}, \mathbf{Q}) = \|\mu[\mathbf{P}] - \mu[\mathbf{Q}]\| = \sup_{f:\|f\|\leq 1} |\langle \mu[\mathbf{P}] - \mu[\mathbf{Q}], f\rangle| = \sup_{f:\|f\|\leq 1} \left| \mathop{\mathbb{E}}_{x\sim\mathbf{P}}[f(x)] - \mathop{\mathbb{E}}_{y\sim\mathbf{Q}}[f(y)] \right|,$$

we will call $D(\mathbf{P}, \mathbf{Q})$ the maximum mean discrepancy $\mathrm{MMD}(\mathbf{P}, \mathbf{Q})$.

Empirical estimation

Suppose we have n *iid* samples $\{X_i\}_{i=1}^n$ of $\mathbf{P}$ and m *iid* samples $\{Y_i\}_{i=1}^m$ of $\mathbf{Q}$, then a natural estimator for $\mathrm{MMD}(\mathbf{P}, \mathbf{Q})$ is $\|\mu[X_1^n] - \mu[Y_1^m]\|$, and the concentration bound can be easily derived by using the diagram:

$$\begin{tikzcd} \mu[\mathbf{P}] \arrow[d, dash] & \mu[\mathbf{Q}] \arrow[d, dash] \\ \mu[X_1^n] \arrow[r, dash] & \mu[Y_1^m] \end{tikzcd}$$

Theorem 33 *Suppose $\{X_i\}_{i=1}^n$ and $\{Y_i\}_{i=1}^m$ are* iid *samples from $\mathbf{P}$ and $\mathbf{Q}$ respectively. Let $R_n(\mathcal{H}, \mathbf{P})$ be the Rademacher average of $\mathcal{H}$ wrt $\mathbf{P}$. Then with probability $1 - \delta$, we* *have*

$$\mathrm{MMD}(\mathbf{P},\mathbf{Q}) \le \|\mu[X_1^n] - \mu[Y_1^m]\| + R_n(\mathcal{H},\mathbf{P}) + R_m(\mathcal{H},\mathbf{Q}) + \sqrt{\frac{\log(2/\delta)}{2m}} + \sqrt{\frac{\log(2/\delta)}{2n}}.$$

Another estimator can be obtained by noticing that $\mathrm{MMD}(\mathbf{P},\mathbf{Q})$ is a U-parameter according to definition Eq. (4.1), and its corresponding U-statistic has the minimal variance among all unbiased estimators (see Appendix E.2). But this estimator imposes the constraint that $\mathbf{P}$ and $\mathbf{Q}$ have the same number of samples ($n = m$).

One advantage of using $\mathrm{MMD}(\mathbf{P},\mathbf{Q})$ as a distance measure between $\mathbf{P}$ and $\mathbf{Q}$ is that it can be directly estimated from the samples with no need of density estimation. If a universal kernel is used, then it captures the difference of all high order moments, while a polynomial kernel of order d allows one to focus on the first d order moments. And the use of kernel allows the distribution to be from any domain where kernels can be defined, such as strings and graphs.

4.1.2 Hilbert-Schmidt independence criteria

Using the MMD distance, we can further quantify the (in)dependence of two random variables X, Y on domains $\mathcal{X}$ and $\mathcal{Y}$ respectively. Define the product space $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$. Let the kernels on $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be k_x, k_y, and k_z respectively, and the corresponding RKHS be $\mathcal{H}_x$, $\mathcal{H}_y$, $\mathcal{H}_z$ respectively. Let the joint distribution on $\mathcal{Z}$ be $\mathbf{P}_{XY}$ and the marginal distributions be $\mathbf{P}_X$ and $\mathbf{P}_Y$. Since X and Y are independent iff $\mathbf{P}_{XY} = \mathbf{P}_X\mathbf{P}_Y$, an independence measure can be naturally defined via the distance between the RKHS embeddings of $\mathbf{P}_{XY}$ and $\mathbf{P}_X\mathbf{P}_Y$:

$$\mathrm{HSIC}(X,Y) := \mathrm{MMD}(\mathbf{P}_{XY},\mathbf{P}_X\mathbf{P}_Y) = \left\| \mathop{\mathbb{E}}_{(x,y)\sim\mathbf{P}_{XY}} k_z((x,y),\cdot) - \mathop{\mathbb{E}}_{x\sim\mathbf{P}_X} \mathop{\mathbb{E}}_{y\sim\mathbf{P}_Y} k_z((x,y),\cdot) \right\|. \tag{4.2}$$

The name HSIC stands for Hilbert-Schmidt independence criteria, because this measure was first proposed as a Hilbert-Schmidt norm of the cross-covariance operator between $\mathcal{X}$ and $\mathcal{Y}$ (Gretton et al., 2005a), and then it was observed that this can be equally motivated as above.

The definition in Eq. (4.2) requires a joint kernel k_z on $\mathcal{Z}$. A factorized kernel k_z allows one to substantially concretize the expression of HSIC. Let

$$k_z((x,y),(x',y')) := k_x(x,x')k_y(y,y'), \quad \textit{i.e.}, \; k_z((x,y),(\cdot,:)) := k_x(x,\cdot) \otimes k_y(y,:),$$

which means $\mathcal{H}_z$ is the tensor product of $\mathcal{H}_x$ and $\mathcal{H}_y$. Now Eq. (4.2) becomes

$$\begin{aligned}
\mathrm{HSIC}(X,Y)^2 &= \left\| \mathop{\mathbb{E}}_{(x,y)\sim \mathbf{P}_{XY}} [k_x(x,\cdot) \otimes k_y(y,:)] - \mathop{\mathbb{E}}_{x\sim \mathbf{P}_X} [k_x(x,\cdot)] \otimes \mathop{\mathbb{E}}_{y\sim \mathbf{P}_Y} [k_y(y,:)] \right\|^2 \\
&= \mathop{\mathbb{E}}_{(x,y)\sim \mathbf{P}_{XY}} \mathop{\mathbb{E}}_{(x',y')\sim \mathbf{P}_{XY}} \left[k_x(x,x')k_y(y,y')\right] && (4.3) \\
&\quad - 2 \mathop{\mathbb{E}}_{(x,y)\sim \mathbf{P}_{XY}} \left[\mathop{\mathbb{E}}_{x'\sim \mathbf{P}_X} \left[k_x(x,x')\right] \mathop{\mathbb{E}}_{y'\sim \mathbf{P}_Y} \left[k_y(y,y')\right] \right] && (4.4) \\
&\quad + \mathop{\mathbb{E}}_{x\sim \mathbf{P}_X} \mathop{\mathbb{E}}_{x'\sim \mathbf{P}_X} \left[k_x(x,x')\right] \mathop{\mathbb{E}}_{y\sim \mathbf{P}_Y} \mathop{\mathbb{E}}_{y'\sim \mathbf{P}_Y} \left[k_y(y,y')\right]. && (4.5)
\end{aligned}$$

A key property of HSIC is that it correctly detects independence:

Theorem 34 *(Gretton et al., 2005a, Appendix B) If k_x and k_y are both universal kernels on compact domains $\mathcal{X}$ and $\mathcal{Y}$ respectively, then* HSIC $= 0$ *if, and only if, X and Y are independent.*

Empirical estimation

Let Z_1^n be n pairs of observations $\{Z_i := (X_i, Y_i)\}_{i=1}^n$ drawn *iid* from the joint distribution $\mathbf{P}_{XY}$. A natural estimator is to replace all the expectations in Eq. (4.3), (4.4), and (4.5), with the respective empirical means, namely $n^{-2} \sum_{ij} k_x(X_i, X_j) k_y(Y_i, Y_j)$, $n^{-3} \sum_{ij} \sum_s k_x(X_i, X_s) k_y(Y_s, Y_j)$, and $n^{-4} \sum_{ij} k_x(X_i, X_j) \sum_{st} k_y(Y_s, Y_t)$ respectively. So we finally obtain an estimator:

$$\widehat{\mathrm{HSIC}}_b(Z_1^n) := \frac{1}{n^2} \operatorname{tr}(K_x H K_y H), \tag{4.6}$$

where $K_x := (k_x(X_i, X_j))_{ij}$, $K_y := (k_y(Y_i, Y_j))_{ij}$, and H is the n-by-n centering matrix $H := \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^\top$.

This estimator is clearly *biased* because of the self-interacting terms. For example, check the first half of Eq. (4.5):

$$\begin{aligned}
\mathbb{E}_{X_1^n} \left[\frac{1}{n^2} \sum_{ij} k_x(X_i, X_j) \right] &= \frac{n-1}{n} \mathbb{E}_x \mathbb{E}_{x'} [k_x(x,x')] + \frac{1}{n} \mathbb{E}_x [k(x,x)] && (4.7) \\
&\neq \mathbb{E}_x \mathbb{E}_{x'} [k_x(x,x')].
\end{aligned}$$

An obvious remedy is to remove these self-interacting terms as

$$\mathbb{E}_{X_1^n} \left[\frac{1}{n(n-1)} \sum_{i \neq j} k_x(X_i, X_j) \right] = \mathbb{E}_x \mathbb{E}_{x'} [k_x(x,x')].$$

This modification leads to an unbiased estimator for HSIC:

$$\widehat{\mathrm{HSIC}}_u := \frac{1}{n(n-3)}\left(\mathrm{tr}(\bar{K}_x\bar{K}_y) - \frac{2}{n-2}\mathbf{1}^\top\bar{K}_x\bar{K}_y\mathbf{1} + \frac{\mathbf{1}^\top\bar{K}_x\mathbf{1}\cdot\mathbf{1}^\top\bar{K}_y\mathbf{1}}{(n-1)(n-2)}\right), \tag{4.8}$$

where $\bar{K}_x$ is equal to K_x except that the diagonal terms are all set to 0.

Despite the biasedness of $\widehat{\mathrm{HSIC}}_b$, its computational form is much simpler and in fact the bias diminishes with more samples, as can be intuitively seen from the last term in Eq. (4.7).

Theorem 35 *The estimator* $\widehat{\mathrm{HSIC}}_b$ *has bias* $\left|\mathrm{HSIC} - \mathbb{E}_{Z_1^n}[\widehat{\mathrm{HSIC}}_b(Z_1^n)]\right| = O(n^{-1})$.

Finally we study the concentration properties of $\widehat{\mathrm{HSIC}}_u$. To start with, it is important to recognize that HSIC is a U-parameter and $\widehat{\mathrm{HSIC}}_u$ happens to be its corresponding U-statistic. The background of U-statistics can be found in Appendix E.2.

Theorem 36 HSIC *can be written as a U-parameter with the kernel*

$$h(i,j,q,r) = \frac{1}{24}\sum_{(s,t,u,v)}^{(i,j,q,r)} K_x(s,t)(K_y(s,t) + K_y(u,v) + 2K_y(s,u)) \tag{4.9}$$

where the summation means (s,t,u,v) *enumerates all the permutations of* (i,j,q,r). $\widehat{\mathrm{HSIC}}_u$ *is exactly the corresponding U-statistic*

$$\widehat{\mathrm{HSIC}}_u = \begin{pmatrix} n \\ 4 \end{pmatrix}^{-1} \sum_{1\le i,j,q,r\le n \text{ and are distinct}} h(i,j,q,r).$$

Rewriting $\widehat{\mathrm{HSIC}}_u$ as a U-statistics allows one to make use of a large body of literature on U-statistics. The following uniform bounds and asymptotic bounds are straightforward from Theorem 80, 82, and 83 in Appendix E.

Theorem 37 (Uniform bounds) *Assume* k_x *and* k_y *are nonnegative and bounded almost everywhere by 1. Then with probability* $1-\delta$ *we have for all* $\mathbf{P}_{XY}$*:*

$$\left|\widehat{\mathrm{HSIC}}_b - \mathrm{HSIC}\right| \le 4\sqrt{\frac{2\log(2/\delta)}{n}}.$$

Theorem 38 (Asymptotic normality if not independent) *Suppose* $\mathbb{E}[h^2] < \infty$ *where* h *is defined in Eq. (4.9). If* X *and* Y *are not independent, then*

$$\sqrt{n}\left(\widehat{\mathrm{HSIC}}_u(Z_1^n) - \mathrm{HSIC}\right) \xrightarrow{\mathcal{D}} \mathcal{N}(0,\sigma^2) \quad \text{as } n\to\infty.$$

Here $\sigma^2 = 16\left(\mathbb{E}_{Z_i}\left[(\mathbb{E}_{Z_j,Z_q,Z_r}[h(i,j,q,r)])^2\right] - \mathrm{HSIC}^2\right)$.

When X and Y are independent, the variance of $\widehat{\mathrm{HSIC}}_u$ becomes degenerate, and a different asymptotic bound can be derived similar to Theorem 83.

4.1.3 Applications of HSIC

Independence measures are very useful in machine learning. A large variety of learning algorithms can be posed in this framework, and here we just give some intuitions and links.

In supervised learning, the labels are given. So in general, we seek for a systematic way to transform the features such that:

1. The class of transform is restricted, *e.g.*, linear transform.

2. A kernel can be defined on the range of the transform.

3. We look for the transformation which maximizes the dependence between the transformed image and the labels, measured by HSIC or its empirical estimates.

One simple transformation is just inner product with a weight vector, and this recovers (kernel) fisher discriminant analysis (Mika et al., 1999). Or, one can look for a subset of features with a prescribed cardinality, and maximize the dependence between the selected features and the labels. This leads to feature selection (Song et al., 2007a). Furthermore, the transform can be a permutation of the training examples, which gives kernelized sorting (Quadrianto et al., 2009). If we map to a lower dimensional space keeping the distance between nearest neighbors intact, then we obtain supervised dimensionality reduction (Song et al., 2008a).

In unsupervised learning, there is no label. While we still seek for a systematic transformation of the features in some prescribed form, we now maximize the dependence between the transformed image and the original data. For example, principal component analysis restricts the transform to projection to a direction. If the transform must take value in categories on which delta kernels are applied, we recover clustering such as k-means, spectral clustering, and normalized graph cut (Song et al., 2007b).

4.2 Embedding distributions with graphical models

Bearing in mind that the domains $\mathcal{X}$ and $\mathcal{Y}$ are fully general, we will discuss a number of different structural assumptions on them in Section 4.3 which allow us to recover existing and propose new measures of dependence. For instance X and Y may represent sequences or a mesh for which we wish to establish dependence. To this end, graphical models can be used as a unifying framework to model the relationship of conditional independence and dependence, and we introduced graphical models in Section 1.2.

The starting point of our extension is how the RKHS embeddings of distributions can be factorized onto cliques when the domain is associated with an undirected graph. This involves two aspects: kernels k and distributions $\mathbf{P}$ which can be completely independent in general. Naturally, we will assume factorization of both kernels and distributions in the sense of Eq. (1.16) and (1.2) respectively. Since a lot is known about the decomposition of distributions, we will develop some results on the decomposition of kernels and their RKHS in Section 4.2.2. Based on these insights, we will study the injectivity of the embeddings in Section 4.2.3, with special attention paid to the exponential family generated by k whose RKHS we embed into. These analyses will immediately facilitate the factorization of HSIC, which will be detailed in Section 4.2.4.

4.2.1 Factorization of mean operators

When the kernel and/or distribution factorize by an undirected graph, we naturally conjecture that the RKHS embedding also factorizes onto the cliques:

Conjecture 39 *Let $\mathbf{P}$ be a distribution which satisfies all conditional independence relationships given by a graph G. Suppose a kernel k on $\mathcal{Z}$ decomposes along the cliques as in Eq. (1.16), with joint RKHS $\mathcal{H}$ and clique-wise RKHSs $\mathcal{H}_c$ ($c \in \mathcal{C}$). Then for any (joint) distribution $\mathbf{P}$ on $\mathcal{Z}$ that also factorizes wrt G as Eq. (1.2), its joint mean map $\mu[\mathbf{P}]$ is related to the the mean map of $\mu_c[\mathbf{P}_c]$ where $\mathbf{P}_c$ is the marginal distribution of $\mathbf{P}$ on clique c and $\mu_c[\mathbf{P}_c]$ is the mean map of $\mathbf{P}_c$ in $\mathcal{H}_c$. Furthermore, if k_c are all characteristic, then the joint map μ is injective.*

This intuition turns out only roughly correct. This section aims to rigorously establish the factorization results. We first show that even without making any assumption on the conditional independence of the distribution, the factorization of kernels itself is enough to guarantee the factorization of the square norm of mean elements.

Theorem 40 (Factorization of mean element) *Let $S_1, S_2, \ldots$ be subsets of the node set, and denote $\mathcal{S} := \{S_1, S_2, \ldots\}$. Suppose the kernel k on $\mathcal{Z}$ can be decomposed by:*

$$k(\mathbf{z}, \mathbf{z}') = \sum_{S \in \mathcal{S}} k_S(z_S, z'_S), \tag{4.10}$$

where k_S are kernels on $\mathcal{Z}_S$. Let the RKHSs of k and k_S be $\mathcal{H}$ and $\mathcal{H}_S$ respectively, and let μ_S be the mean operator induced by k_S. Then for any arbitrary distribution $\mathbf{P}$

and $\mathbf{Q}$ *on* $\mathcal{Z}$ *whose marginal distribution on* $\mathcal{Z}_S$ *is* $\mathbf{P}_S$ *and* $\mathbf{Q}_S$ *respectively, we have*

$$\langle \mu[\mathbf{P}], \mu[\mathbf{Q}] \rangle_{\mathcal{H}} = \sum_{S \in \mathcal{S}} \langle \mu_S[\mathbf{P}_S], \mu_S[\mathbf{Q}_S] \rangle_{\mathcal{H}_S} ,$$

and in particular

$$\| \mu[\mathbf{P}] \|^2_{\mathcal{H}} = \sum_{S \in \mathcal{S}} \| \mu_S[\mathbf{P}_S] \|^2_{\mathcal{H}_S} .$$

Proof The proof is straightforward from Property 1.

$$\begin{aligned} \langle \mu[\mathbf{P}], \mu[\mathbf{Q}] \rangle_{\mathcal{H}} &= \mathop{\mathbb{E}}_{\mathbf{z} \sim \mathbf{P}} \mathop{\mathbb{E}}_{\mathbf{z}' \sim \mathbf{Q}} \left[k(\mathbf{z}, \mathbf{z}') \right] = \mathop{\mathbb{E}}_{\mathbf{z} \sim \mathbf{P}} \mathop{\mathbb{E}}_{\mathbf{z}' \sim \mathbf{Q}} \left[\sum_{S \in \mathcal{S}} k_S(z_S, z'_S) \right] \\ &= \sum_{S \in \mathcal{S}} \mathop{\mathbb{E}}_{z_S \sim \mathbf{P}_S} \mathop{\mathbb{E}}_{z'_S \sim \mathbf{Q}_S} \left[k_S(z_S, z'_S) \right] = \sum_{S \in \mathcal{S}} \langle \mu_S[\mathbf{P}_S], \mu_S[\mathbf{Q}_S] \rangle_{\mathcal{H}_S} . \end{aligned}$$

■

Theorem 40 is very suggestive of the decomposition of $\mathcal{H}$: $\mathcal{H}$ factorizes into the direct product (or direct sum[1]) of $\mathcal{H}_S$: $\mathcal{H} = \oplus_S \mathcal{H}_S$. However, this turns out to be true only when all S in $\mathcal{S}$ are disjoint, and the next section will discuss the details.

4.2.2 Factorization of RKHS for factorized kernels

In this section, we temporarily ignore the distributions, and focus on the relationship between $\mathcal{H}$ and $\{\mathcal{H}_S\}_{S \in \mathcal{S}}$ when the kernel k factorizes according to Eq. (4.10). One key encouraging property is the following existential theorem:

Theorem 41 *Suppose kernel* k *factorizes by Eq.* (4.10). *Then for any* $f \in \mathcal{H}$*, there must exist* $f_S \in \mathcal{H}_S$ *such that*

$$f(\mathbf{z}) = \sum_{S \in \mathcal{S}} f_S(z_S), \ \textit{for all } \mathbf{z} \in \mathcal{Z}, \qquad \textit{and } \|f\|^2_{\mathcal{H}} = \sum_{S \in \mathcal{S}} \|f_S\|^2_{\mathcal{H}_S} .$$

Proof First, if $f(\cdot) = k(\hat{\mathbf{z}}, \cdot)$, then $f(\mathbf{z}) = k(\hat{\mathbf{z}}, \mathbf{z}) = \sum_{S \in \mathcal{S}} k_S(\hat{z}_S, z_S)$ for all $\mathbf{z} \in \mathcal{Z}$. Next, if $f(\cdot) = \sum_i \alpha_i k(\hat{\mathbf{z}}^i, \cdot)$, then for all $\mathbf{z} \in \mathcal{Z}$ we have

$$f(\mathbf{z}) = \sum_i k(\hat{\mathbf{z}}^i, \mathbf{z}) = \sum_i \sum_{S \in \mathcal{S}} \alpha_i k_S(\hat{z}^i_S, z_S) = \sum_{S \in \mathcal{S}} \underbrace{\sum_i \alpha_i k_S(\hat{z}^i_S, z_S)}_{:= f_S(z_S) \in \mathcal{H}_S} . \tag{4.11}$$

[1]The direct product is the same as the direct sum when the cardinality of $\mathcal{S}$ is finite.

To check the norm, notice

$$\|f\|^2 = \sum_{ij} \alpha_i \alpha_j k(\hat{\mathbf{z}}^i, \hat{\mathbf{z}}^j) = \sum_{ij} \alpha_i \alpha_j \sum_S k(\hat{z}_S^i, \hat{z}_S^j) = \sum_S \sum_{ij} \alpha_i \alpha_j k(\hat{z}_S^i, \hat{z}_S^j) = \sum_S \|f_S\|^2 .$$

Finally, if f is a limit point of the linear span $\{k(\mathbf{z}, \cdot) : \mathbf{z} \in \mathcal{Z}\}$, then there exists a sequence of $\{f^n\}_{n\in\mathbb{N}}$ in the linear span, and $f^n \overset{\mathcal{H}}{\to} f$. For each n, due to the above result, there exist functions $f_S^n \in \mathcal{H}_S$, such that $f^n(\mathbf{z}) = \sum_S f_S^n(z_S)$ for all $\mathbf{z} \in \mathcal{Z}$ and $\|f^n\|^2 = \sum_S \|f_S^n\|^2$. Since $f^n \overset{\mathcal{H}}{\to} f$, so $\{\|f^n\|\}_{n\in\mathbb{N}}$ is bounded, hence $\{\|f_S^n\|\}_{n\in\mathbb{N}}$ is bounded for all $S \in \mathcal{S}$. Call the sets in $\mathcal{S}$ as $S_1, S_2, \ldots$. For S_1, the sequence $\{f_{S_1}^n\}_{n\in\mathbb{N}}$ must have a cluster point $f_{S_1}^* \in \mathcal{H}_{S_1}$, which is the limit of a subsequence $\left\{f_{S_1}^{n_k}\right\}_{k\in\mathbb{N}}$. Without loss of generality, assume this subsequence is $\{f_{S_1}^n\}_{n\in\mathbb{N}}$ itself. Similar procedure of subsequence selection can be run for $S_2, S_3, \ldots$. Finally, we obtain $f_S^* \in \mathcal{H}_S$ for all $S \in \mathcal{S}$ and $f_S^n \overset{\mathcal{H}_S}{\to} f_S^*$ as $n \to \infty$. So

$$\begin{aligned} f(\mathbf{z}) &= \lim_{n\to\infty} f^n(\mathbf{z}) = \lim_{n\to\infty} \sum_S f_S^n(z_S) = \sum_S \lim_{n\to\infty} f_S^n(z_S) = \sum_S f_S^*(z_S), \\ \|f\|^2 &= \lim_{n\to\infty} \|f^n\|^2 = \lim_{n\to\infty} \sum_S \|f_S^n\|^2 = \sum_S \lim_{n\to\infty} \|f_S^n\|^2 = \sum_S \|f_S^*\|^2 . \end{aligned}$$

Pathological behavior may occur when $|\mathcal{S}| = \infty$[2], which we do not consider here. ■

It is convenient to formalize the above map from $f \in \mathcal{H}$ to $\{f_S : S \in \mathcal{S}\}$.

Definition 42 (Function factorization mapping) *The function factorization mapping $\tau_{\mathcal{S}}$ from $\mathcal{H}$ to the power set of $\oplus_{S\in\mathcal{S}}\mathcal{H}_S$ is defined by mapping any function $f \in \mathcal{H}$ to the set of all possible factorizations of f:*

$$\tau_{\mathcal{S}}(f) := \left\{ \{f_S \in \mathcal{H}_S\}_S : f(z) = \sum_{S\in\mathcal{S}} f_S(z_S),\ \forall z \in \mathcal{Z} \right\}. \tag{4.12}$$

So Theorem 41 says $|\tau_{\mathcal{S}}(f)| \geq 1$ for all $f \in \mathcal{H}$.

For any arbitrary kernel, it is not hard to see that if the index sets in $\mathcal{S}$ constitute a partition of all the nodes[3], the factorization of any $f \in \mathcal{H}$ is unique, *i.e.*, $|\tau_{\mathcal{S}}(f)| = 1$. Conversely, for any set of functions $\{f_S \in \mathcal{H}_S : S \in \mathcal{S}\}$, $f(Z) := \sum_{S\in\mathcal{S}} f_S(Z_S)$ must be in $\mathcal{H}$. However, if the index sets in $\mathcal{S}$ overlap, then in general neither uniqueness nor surjectivity holds even if $\mathcal{S}$ is the set of maximal cliques. We give two examples.

[2] $|A|$ denotes the cardinality of the set A.

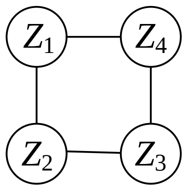
[3] A partition of a set A is any set of subsets $\{A_i \subseteq A : i \in \mathcal{I}\}$ where $\mathcal{I}$ is an index set, such that $\cup_{i\in\mathcal{I}} A_i = A$, and $A_i \cap A_j = \emptyset$ for all $i, j \in \mathcal{I}$ and $i \neq j$.

Proposition 43 (Breakdown of uniqueness) *One can construct an example with the following elements: a) a graph with maximal clique set $\mathcal{C}$, b) positive definite kernels k_c on all cliques $c \in \mathcal{C}$, c) a joint kernel k defined by Eq. (4.10), and d) a function $f \in \mathcal{H}$ such that its factorization is not unique,* i.e., $|\tau_{\mathcal{C}}(f)| \geq 2$, *and different factorizations have different sum of squared norm $\sum_c \|f_c\|^2$.*

Proof Let the graphical model of the random variable $Z \in \mathbb{R}^4$ be a loop with $\mathcal{C} = \{\{Z_1, Z_2\}, \{Z_2, Z_3\}, \{Z_3, Z_4\}, \{Z_4, Z_1\}\}$. Let the kernel on clique $\{Z_i, Z_j\}$ be

$$k_{ij}((z_i, z_j), (z_i', z_j')) := \exp(-(z_i - z_i')^2) + \exp(-(z_j - z_j')^2),$$

and $k(\mathbf{z}, \mathbf{z}') := \sum_{c \in \mathcal{C}} k_S(z_c, z_c')$. Set $f(\mathbf{z}) := \frac{1}{2} k(\mathbf{0}, \mathbf{z}) = \sum_{i=1}^4 \exp(-z_i^2)$ which is obviously in the RKHS of k. It is not hard to check that the following $\{f_c\}_c$ satisfy $f(\mathbf{z}) = \sum_c f_c(z_c)$ for all $\alpha \in \mathbb{R}$:



$$\begin{aligned}
f_{12}(z_1, z_2) &= \alpha \exp(-z_1^2) + \alpha \exp(-z_2^2) \\
f_{23}(z_2, z_3) &= (1-\alpha) \exp(-z_2^2) + (1-\alpha) \exp(-z_3^2) \\
f_{34}(z_3, z_4) &= \alpha \exp(-z_3^2) + \alpha \exp(-z_4^2) \\
f_{41}(z_4, z_1) &= (1-\alpha) \exp(-z_4^2) + (1-\alpha) \exp(-z_1^2).
\end{aligned}$$

Besides, $\sum_S \|f_S\|_{\mathcal{H}_S}^2 = 8\alpha^2 - 8\alpha + 4$ which equals $\|f\|_{\mathcal{H}}^2 = 4$ only when $\alpha = 0$ or 1. ■

Proposition 44 (Breakdown of surjectivity) *With the same conditions a), b), c) as in Proposition 43, there exist $\{f_c \in \mathcal{H}_c\}$ such that $f(\mathbf{z}) := \sum_c f_c(z_c)$ is not in $\mathcal{H}$.*

Proof Consider a simple three node chain on $\mathbb{R}^3$ with cliques $\{\{Z_1, Z_2\}, \{Z_2, Z_3\}\}$,

$$Z_1 \text{ —— } Z_2 \text{ —— } Z_3$$

and let the kernel on the cliques be Gaussian:

$$k_{ij}((z_i, z_j), (z_i', z_j')) := \exp(-(z_i - z_i')^2 - (z_j - z_j')^2) \qquad \text{for } \{i, j\} = \{1, 2\} \text{ or } \{2, 3\},$$

and the joint kernel be $k_{12} + k_{23}$:

$$k(\mathbf{z}, \mathbf{z}') =: k_{12}((z_1, z_2), (z_1', z_2')) + k_{23}((z_2, z_3), (z_2', z_3')).$$

Pick two functions $f_{12} \in \mathcal{H}_{12}$, $f_{23} \in \mathcal{H}_{23}$, and define f by

$$\begin{aligned} f_{12}(x_1, x_2) &:= 0, \\ f_{23}(x_2, x_3) &:= k_{23}(\mathbf{0}, (z_2, z_3)) = \exp\left(-z_2^2 - z_3^2\right), \\ f(\mathbf{z}) = f(z_1, z_2, z_3) &:= f_{12}(x_1, x_2) + f_{23}(x_2, x_3) = \exp\left(-z_2^2 - z_3^2\right). \end{aligned} \tag{4.13}$$

The proof of f not being in $\mathcal{H}$ is lengthy, hence moved to Appendix F. It is based on the orthonormal basis of real Gaussian RKHS. ■

In conclusion, the relationship between $\mathcal{H}$ and $\{\mathcal{H}_c\}_{c\in\mathcal{C}}$ is much more involved than just direct product, which holds only when the maximal cliques in $\mathcal{C}$ are mutually disjoint. It will be interesting for future research to investigate when the f_c are unique ($|\tau_{\mathcal{C}}(f)| = 1$) and when $\|f\|_{\mathcal{H}}^2 = \sum_c \|f_c\|_{\mathcal{H}_c}^2$. Example conditions may be $\mathcal{C}$ being the maximum clique set of a *triangulated* graph, or kernels being universal. Since the example in Proposition 44 satisfies both, it indicates that some new conditions are needed.

Fortunately, Theorem 40 is enough for our subsequent discussions on the independence criteria.

4.2.3 Injectivity of factored mean operators

When the kernel factorizes by Eq. (4.10), it is again important to study the sufficient and necessary conditions for the operator μ to be injective. In particular we study the case where the kernel and the distribution factorize wrt the same graphical model.

In the cases without factorization, the injectivity results such as by Fukumizu et al. (2009); Sriperumbudur et al. (2008) usually seek conditions on the kernels such as universality, while almost no assumption is made on the distribution space. The following theorem shows the feasibility of the other way round: weaker assumption on kernels and stronger assumptions on the distributions.

Theorem 45 *For any kernel k, the mean operator from $\mathcal{P}_k$ to $\mathcal{H}$ is injective.*

Notice that except for tree structured graphs, the marginal distribution on the cliques does not uniquely identify the global distribution in general. However, among them, only one can be in the exponential family.

Proof Immediate from Theorem 26. ■

Restricting the distributions to kernel exponential families does not cost much generality, thanks to Theorem 18 which says that if the kernel k is universal, then $\mathcal{P}_k$ can approximate a very general class of densities arbitrarily well in the sense of L_∞ norm or KL divergence. Now we generalize this result to the case where the distribution and kernel decompose by a graphical model.

Theorem 46 (Dense distribution wrt graphical models) *Given a graphical model G on a multivariate random variable Z with maximal clique set $\mathcal{C}$. Suppose the domain of Z, $\mathcal{Z}$, is measurable with respect to the Lebesgue measure. Let $\mathcal{P}_G$ be the set of all distributions that a) have full support on $\mathcal{Z}$, b) satisfy all the conditional independence relations encoded by G, and c) there exists a constant B such that $\|p\|_\infty < B$ for all $p \in \mathcal{P}_G$. Assume a joint kernel k on $\mathcal{Z}$ decomposes along the cliques by $k(\mathbf{z}, \mathbf{z}') = \sum_c k_c(z_c, z'_c)$, where the kernel k_c on $\mathcal{Z}_c$ are all universal. Suppose*

$$\sigma(\{\mathcal{H}_c\}_c) := \left\{ f(\mathbf{z}) := \sum_{c \in \mathcal{C}} f_c(z_c) : f_c \in \mathcal{H}_c \right\}$$

is dense in $\mathcal{H}$ in L_∞ norm. Then $\mathcal{P}_k$ is dense in $\mathcal{P}_G$ in the sense that for any $\epsilon > 0$ and any distribution in $\mathcal{P}_G$ with density p, there exists a density p_f in $\mathcal{P}_k$ with natural parameter $f \in \mathcal{H}$, such that $\|p - p_f\|_\infty \leq \epsilon$.

Proof For any distribution in $\mathcal{P}_G$ with density p, the Hammersley-Clifford theorem guarantees that there exists a potential function ψ_c on $\mathcal{Z}_c$ for all $c \in \mathcal{C}$, such that

$$p(\mathbf{z}) = \exp\left(\sum_c \psi_c(z_c) - g\right), \quad \text{where } g = \log \int \exp\left(\sum_c \psi_c(z_c)\right) \mathrm{d}\mathbf{z}.$$

Let $m := |\mathcal{C}|$ be the number of maximal cliques. Since k_c is universal, there must exist a function $f_c \in \mathcal{H}_c$ such that $\|f_c - (\psi_c - g/m)\|_\infty < \frac{\epsilon}{2m}$. Denoting $\psi(\mathbf{z}) := \sum_c \psi_c(z_c)$ and $\tilde{f}(\mathbf{z}) := \sum_c f_c(z_c)$, we have $\left\|\log p - \tilde{f}\right\|_\infty = \|\psi - g - f\|_\infty < \epsilon/2$. Since $\sigma(\{\mathcal{H}_c\}_c)$ is dense in $\mathcal{H}$ in L_∞ sense, there must exist a function $f \in \mathcal{H}$ such that $\left\|f - \tilde{f}\right\|_\infty < \epsilon/2$. Hence $\|\log p - f\|_\infty < \epsilon$.

f determines a distribution in $\mathcal{P}_k$ with density $p_f(\mathbf{z}) = \exp(f(\mathbf{z}) - g_f)$, where

$$|g_f| = \left|\log \int \exp(f(\mathbf{z}))\mathrm{d}\mathbf{z}\right| = \left|\log \int \exp(f(\mathbf{z}) - \log p(\mathbf{z}))p(\mathbf{z})\mathrm{d}\mathbf{z}\right| \leq \left|\log \int e^{\epsilon} p(\mathbf{z})\mathrm{d}\mathbf{z}\right| = \epsilon.$$

Therefore,

$$\begin{aligned}
\|\log p_f - \log p\|_\infty &= \|f - g_f - \log p\|_\infty \leq \|f - \log p\|_\infty + |g_f| < \epsilon + \epsilon = 2\epsilon, \\
\|p_f - p\|_\infty &= \|\exp(\log p_f) - \exp(\log p)\|_\infty \\
&\leq \|p\|_\infty \|\exp(\log p_f - \log p) - 1\|_\infty < \|p\|_\infty (e^{2\epsilon} - 1) = \|p\|_\infty o(\epsilon).
\end{aligned}$$

Noting the uniform bound B on $\|p\|_\infty$ for all $p \in \mathcal{P}_G$ completes the proof. ■

Incidentally, the density of $\mathcal{P}_k$ in $\mathcal{P}_G$ also holds in the sense of KL divergence (both $\mathrm{KL}(p||p_f)$ and $\mathrm{KL}(p_f||p)$ for $p \in \mathcal{P}_G$ and $p_f \in \mathcal{P}_k$), which can be derived in exactly the same fashion as Altun & Smola (2006, Proposition 3).

So below we focus on $\mathcal{P}_k$, where another map $\Lambda_k : \mathcal{H} \mapsto \mathcal{H}$ naturally arises in complete analogy with the mean operator for vanilla exponential family (from the natural parameter to the marginal polytope): $\Lambda_k[f] := \mu[\mathbf{P}(x; f)]$, *i.e.*

$$\begin{array}{ccccc} f & \longmapsto & p(x;f) = \exp(f(x) - g) & \longmapsto & \mathbb{E}_{x\sim p(x;f)}[k(x,\cdot)] \\ \in \mathcal{H} & & \in \mathcal{P}_k & & \in \mathcal{H} \end{array}$$

Studying the properties of Λ_k can be interesting for future research on the inference in kernel exponential families with graphical models, hence generalizing the large volume of existing literature (*e.g.* Wainwright & Jordan, 2008).

In $\mathcal{P}_k$, the natural parameters are the function $f \in \mathcal{H}$ and the sufficient statistics are the evaluating elements $k(x, \cdot)$. So according to Theorem 25, the map Λ_k from $\mathcal{H}$ to the mean of the sufficient statistics is injective if, and only if, $k(x, \cdot)$ is minimal, *i.e.*, there does not exist any nonzero function $f \in \mathcal{H}$ such that $\langle f, k(x, \cdot)\rangle$ is constant. Hence we just require that $\mathcal{H}$ not contain constant functions.

Proposition 47 *If the RKHS of kernel k does not contain nonzero constant functions, then the map Λ_k from $\mathcal{H}$ to $\mathcal{H}$ via $\mathcal{P}_k$ is injective.*

In view of the important role played by the existence of nonzero constant functions in RKHS, we formally define:

Definition 48 (Constant-exclusive (CE) kernels) *A kernel is called* constant exclusive *(CE) if its RKHS $\mathcal{H}$ does not contain nonzero constant functions. Furthermore, if the kernel is defined on the space $\mathcal{Z}_1 \times \ldots \times \mathcal{Z}_n$, then the kernel is called* coordinate constant exclusive *(CCE) if $\mathcal{H}$ does not contain any function f for which there is a coordinate $i \in [n]$ and a set of assignment $\{\tilde{x}_j\}_{j\neq i}$ such that $f(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, x_i, \tilde{x}_{i-1}, \ldots)$ is constant in x_i and nonzero.*

The CE property looks simple but has not been well studied. Steinwart & Christmann (2008, Corollary 4.44) gave a nontrivial proof that nonzero constant functions are not in the RKHS of Gaussian kernels on a subset of $\mathbb{R}^n$ which contains open set, hence is CE. Lemma 92 further proves that it is CCE as well. However, in general, there seems to be no implication between a kernel being universal/characteristic and CE/CCE. Even universal kernels are not necessarily CE or CCE, *e.g.*, Gaussian kernel plus 1 is universal but not CE or CCE. Polynomial kernels are also clearly not CE or CCE. The RKHS of linear kernels only contains linear functions, hence CE but not necessarily CCE. The kernel $k((x_1, x_2), (x'_1, x'_2)) = \exp(-(x_1 - x'_1)^2) + 1$ is CE but not CCE. Conversely, CCE clearly guarantees CE.

The following Theorem 49 gives an interesting result which connects three important aspects: a) the graphical model topology, b) CCE of the kernels on the cliques, and c) CE of the joint kernel.

Theorem 49 *Suppose G is a tree and the kernels k_c are all CCE, then k is CE and hence the mean operator Λ_k is injective.*

Proof Suppose k is not CE and its RKHS has a non-zero constant function f. Then there must exist functions $\{f_c : \mathcal{Z}_c \to \mathbb{R}, c \in \mathcal{C}\}$ such that $f(Z) = \sum_{c\in\mathcal{C}} f_c(Z_c)$. Since the maximal cliques of a tree are just edges, take an edge $c = (l, n)$ where l is a leaf. Since l does not appear in other $c \in \mathcal{C}$, so f_c must be constant in l. However, k_c is CCE, so f_c can only be 0. This argument can be run recursively from all leaves, to parent of leaves, and finally to the root, resulting in f_c being 0 for all c, hence $f \equiv 0$. This contradiction means k must be CE. ∎

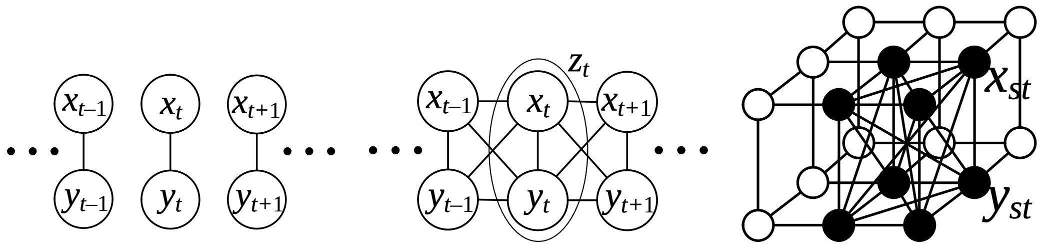
4.2.4 Factorization of independence criteria

Theorem 40 implies that we will be able to perform all subsequent operations on structured domains simply by dealing with mean operators on the corresponding maximal cliques. In addition, it implies that if the kernel decomposes along an undirected graph G we may decompose HSIC(X, Y) further into

$$\begin{aligned} I(X,Y) &= \sum\nolimits_{c\in\mathcal{C}} \left\|\mu_c[\mathbf{P}_{X_cY_c}] - \mu_c[\mathbf{P}_{X_c}\mathbf{P}_{Y_c}]\right\|^2_{\mathcal{H}_c} \\ &= \sum\nolimits_{c\in\mathcal{C}} \left\{\mathbb{E}_{(x_cy_c)(x'_cy'_c)} + \mathbb{E}_{x_cy_cx'_cy'_c} - 2\mathbb{E}_{(x_cy_c)x'_cy'_c}\right\} [k_{z,c}((x_c, y_c), (x'_c, y'_c))] \end{aligned} \tag{4.14}$$

where bracketed random variables in the subscripts are drawn from their joint distributions and un-bracketed ones are from their respective marginals, *e.g.*, $\mathbb{E}_{(x_cy_c)x'_cy'_c} := \mathbb{E}_{(x_cy_c)}\mathbb{E}_{x'_c}\mathbb{E}_{y'_c}$ (refer to Eq. (4.3) to (4.5) for the full expansion). Obviously the challenge is to find good empirical estimates of (4.14). In its simplest form we may replace each of the expectations by sums over samples, that is, by replacing

$$\mathbb{E}_{(x,y)}[f(x,y)] \leftarrow \frac{1}{n}\sum_{i=1}^{n} f(x_i, y_i) \quad \text{and} \quad \mathbb{E}_{(x)(y)}[f(x,y)] \leftarrow \frac{1}{n^2}\sum_{i,j=1}^{n} f(x_i, y_j). \tag{4.15}$$



(a) *iid* (b) First order sequential (c) 2-Dim mesh

Figure 4.2: From left to right: (a) a graphical model representing *iid* observations, (b) a graphical model for first order sequential data, and (c) a graphical model for dependency on a two dimensional mesh.

4.3 Estimates for special structures

To illustrate the versatility of our approach we apply our model to a number of graphical models ranging from independent random variables to meshes proceeding according to the following recipe:

1. Define a conditional independence graph.
2. Identify the maximal cliques.
3. Choose suitable joint kernels on the maximal cliques.
4. Exploit stationarity (if existent) in $I(X, Y)$ in (4.14).
5. Derive the corresponding empirical estimators for each clique, and hence for all of $I(X, Y)$.

4.3.1 Independent and identically distributed data

As the simplest case, we first consider the graphical model in Figure 4.2a, where $\{(x_t, y_t)\}_{t=1}^T$ are *iid* random variables. Correspondingly the maximal cliques are $\{(x_t, y_t)\}_{t=1}^T$. We choose the joint kernel on the cliques to be

$$\bar{k}_{z,t}((x_t, y_t), (x'_t, y'_t)) := \bar{k}_x(x_t, x'_t)\bar{k}_y(y_t, y'_t) \tag{4.16}$$

$$\text{hence} \quad k_z((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \sum_{t=1}^{T} \bar{k}_x(x_t, x'_t)\bar{k}_y(y_t, y'_t), \tag{4.17}$$

where $\bar{k}_x$ and $\bar{k}_y$ are ground kernels on the cliques of $\mathcal{X}$ and $\mathcal{Y}$ respectively. The representation for $k_{z,t}$ implies that we are taking an outer product between the Hilbert spaces on x_t and y_t induced by kernels $\bar{k}_x$ and $\bar{k}_y$ respectively. If the pairs of random variables (x_t, y_t) are *not* identically distributed, all that is left is to use (4.17) to obtain an empirical estimate via (4.15).

We may improve the estimate considerably if we are able to assume that all pairs (x_t, y_t) are drawn from the *same* distribution $p(x_t, y_t)$. Consequently all coordinates of the mean map are identical and we can use all the data to estimate just one of the discrepancies $\|\mu_c[p_c(x_c, y_c)] - \mu_c[p_c(x_c)p_c(y_c)]\|^2$. The latter expression is identical to the standard HSIC criterion and we obtain the biased estimate

$$\hat{I}(X,Y) = \tfrac{1}{T} \operatorname{tr} HK_xHK_y \tag{4.18}$$

$$\text{where} \qquad (K_x)_{st} := \bar{k}_x(x_s, x_t), (K_y)_{st} := \bar{k}_y(y_s, y_t) \text{ and } H_{st} := \delta_{st} - \tfrac{1}{T}.$$

4.3.2 Sequence data

A more interesting application beyond *iid* data is sequences with a Markovian dependence as depicted in Figure 4.2b. Here the maximal cliques are the sets $\{(x_t, x_{t+1}, y_t, y_{t+1})\}_{t=1}^{T-1}$. More generally, for longer range dependency of order $\tau \in \mathbb{N}$, the maximal cliques will involve the random variables $(x_t, \ldots, x_{t+\tau}, y_t, \ldots, y_{t+\tau}) =: (x_{t,\tau}, y_{t,\tau})$.

We assume *homogeneity* and *stationarity* of the random variables: that is, all cliques share the same sufficient statistics (feature map) and their expected value is identical. In this case the kernel

$$\bar{k}_z((x_{t,\tau}, y_{t,\tau}), (x'_{t,\tau}, y'_{t,\tau})) := \bar{k}_x(x_{t,\tau}, x'_{t,\tau})\bar{k}_y(y_{t,\tau}, y'_{t,\tau})$$

can be used to measure discrepancy between the random variables. Stationarity means that $\mu_c[p_c(x_c, y_c)]$ and $\mu_c[p_c(x_c)p_c(y_c)]$ are the same for all cliques c, hence $I(X,Y)$ is a multiple of the difference for a single clique.

Using the same argument as in the *iid* case, we can obtain a biased estimate of the dependence measure by using $(K_x)_{ij} = \bar{k}_x(x_{i,\tau}, x_{j,\tau})$ and $(K_y)_{ij} = \bar{k}_y(y_{i,\tau}, y_{j,\tau})$ instead of the definitions of K_x and K_y in (4.18). This works well in experiments. In order to obtain an unbiased estimate we need some more work. Recall the unbiased estimate of $I(X,Y)$ is a fourth order U-statistic (see Theorem 36 and (Gretton et al., 2008)).

Theorem 50 *An unbiased empirical estimator for* $\|\mu[p(x,y)] - \mu[p(x)p(y)]\|^2$ *is*

$$\hat{I}(X,Y) := \frac{(n-4)!}{n!} \sum_{(i,j,q,r)} h(x_i, y_i, \ldots, x_r, y_r), \tag{4.19}$$

where the sum is over all terms such that i, j, q, r *are mutually different, and*

$$h(x_1, y_1, \ldots, x_4, y_4) := \frac{1}{4!} \sum_{(t,u,v,w)}^{(1,2,3,4)} \bar{k}_x(x_t, x_u)\bar{k}_y(x_t, x_u) + \bar{k}_x(x_t, x_u)\bar{k}_y(x_v, x_w)$$
$$- 2\bar{k}_x(x_t, x_u)\bar{k}_y(x_t, x_v),$$

and the latter sum denotes all ordered quadruples (t, u, v, w) *drawn from* $(1, 2, 3, 4)$.

The theorem implies that in expectation h takes on the value of the dependence measure. To establish that this also holds for dependent random variables we use a result from (Aaronson et al., 1996) which establishes convergence for stationary mixing sequences under mild regularity conditions, namely whenever the kernel of the U-statistic h is bounded and the process generating the observations is absolutely regular. See also Appendix E.3 and (Borovkova et al., 2001, Section 4). We note that Kontorovich (2007) developed a similar result for uniform bound, and his results are highlighted in Appendix E.4.

Theorem 51 *Whenever* $I(X, Y) > 0$*, that is, whenever the random variables* are *dependent, the estimate* $\hat{I}(X, Y)$ *is asymptotically normal with*

$$\sqrt{n}(\hat{I}(X,Y) - I(X,Y)) \xrightarrow{d} \mathcal{N}(0, 4\sigma^2) \tag{4.20}$$

where the variance is given by

$$\sigma^2 = \mathrm{Var}\left[h_3(x_1, y_1)\right]^2 + 2\sum_{t=1}^{\infty} \mathrm{Cov}(h_3(x_1, y_1), h_3(x_t, y_t)) \tag{4.21}$$

$$\textit{and} \quad h_3(x_1, y_1) := \mathbb{E}_{(x_2,y_2,x_3,y_3,x_4,y_4)}[h(x_1, y_1, \ldots, x_4, y_4)] \tag{4.22}$$

This follows from (Borovkova et al., 2001, Theorem 7), again under mild regularity conditions (note that Borovkova et al. (2001) state their results for U-statistics of second order, and claim the results hold for higher orders). The proof is tedious but does not require additional techniques.

4.3.3 TD-SEP as a special case

So far we did not discuss the freedom of choosing different kernels. In general, an RBF kernel will lead to an effective criterion for measuring the dependence between random variables, especially in time-series applications. However, we could also choose linear kernels for $\bar{k}_x$ and $\bar{k}_y$, for instance, to obtain computational savings.

For a specific choice of cliques and kernels, we can recover the work of Ziehe & Müller (1998) as a special case of our framework. In (Ziehe & Müller, 1998), for two centered scalar time series x and y, the contrast function is chosen as the sum of same-time and time-lagged cross-covariance $\mathbb{E}[x_t y_t] + \mathbb{E}[x_t y_{t+\tau}]$. Using our framework, two types of cliques, (x_t, y_t) and $(x_t, y_{t+\tau})$, are considered in the corresponding graphical model. Furthermore, we use a joint kernel of the form

$$\langle x_s, x_t \rangle \langle y_s, y_t \rangle + \langle x_s, x_t \rangle \langle y_{s+\tau}, y_{t+\tau} \rangle\,, \tag{4.23}$$

which leads to the estimator of structured HSIC:

$$\hat{I}(X, Y) = \frac{1}{T} \left(\operatorname{tr} HK_x HK_y + \operatorname{tr} HK_x HK_y^\tau \right).$$

Here K_y^τ denotes the linear covariance matrix for the time lagged y signals. For scalar time series, basic algebra shows that $\operatorname{tr} HK_x HK_y$ and $\operatorname{tr} HK_x HK_y^\tau$ are the estimators of $\mathbb{E}[x_t y_t]$ and $\mathbb{E}[x_t y_{t+\tau}]$ respectively (up to a multiplicative constant).

Further generalization can incorporate several time lagged cross-covariances into the contrast function. For instance, TD-SEP (Ziehe & Müller, 1998) uses a range of time lags from 1 to τ. That said, by using a nonlinear kernel we are able to obtain better contrast functions, as we will show in our experiments.

4.3.4 Grid structured data

Structured HSIC can go beyond sequence data and be applied to more general dependence structures such as 2-D grids for images. Figure 4.2c shows the corresponding graphical model. Here each node of the graphical model is indexed by two subscripts, i for row and j for column. In the simplest case, the maximal cliques are

$$\mathcal{C} = \{(x_{ij}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}, y_{ij}, y_{i+1,j}, y_{i,j+1}, y_{i+1,j+1})\}_{ij}.$$

In other words, we are using a cross-shaped stencil to connect vertices. Provided that the kernel $\bar{k}_z$ can also be decomposed into the product of $\bar{k}_x$ and $\bar{k}_y$, then a biased estimate of the independence measure can be again formulated as $\operatorname{tr} HK_x HK_y$ up to a multiplicative constant. The statistical analysis of U-statistics for stationary Markov random fields is highly nontrivial. We are not aware of results equivalent to those discussed in Section 4.3.2. Kontorovich (2007), when dealing with uniform bounds, also did not give any result on grid structured data.

4.4 Experiments

Having a dependence measure for structured spaces is useful for a range of applications. Analogous to *iid* HSIC, structured HSIC can be applied to non-*iid* data in applications such as independent component analysis (Shen et al., 2009), independence test (Gretton et al., 2008), feature selection (Song et al., 2007c), clustering (Song et al., 2007b), and dimensionality reduction (Song et al., 2008a). The fact that structured HSIC can take into account the interdependency between observations provides us with a principled generalization of these algorithms to, *e.g.*, time series analysis. In this thesis, we will focus on three examples:

1. Independence test where structured HSIC is used as a test statistic;
2. Independent component analysis where we wish to minimize the dependence;
3. Time series segmentation where we wish to maximize the dependence.

4.4.1 Independence test

We first present two experiments that use the structured HSIC as an independence measure for non-*iid* data, namely XOR binary sequence and Gaussian process. With structured HSIC as a test statistic, we still need an approach to building up the distribution of the test statistic under the null hypothesis $\text{H0} : \mathbf{x} \perp\!\!\!\perp \mathbf{y}$. For this purpose, we generalize the random shuffling technique commonly used for *iid* observations (Gretton et al., 2008) into a clique-bundled shuffling. This shuffling technique randomly pairs up the observations in $\mathbf{x}$ and $\mathbf{y}$. Depending on the clique configurations of structured HSIC, one observation in $\mathbf{x}$ may be paired up with several observations in $\mathbf{y}$. The observations corresponding to an instance of a maximal clique need to be bundled together and shuffled in blocks. For instance, if the maximal cliques are $\{(x_t, y_t, y_{t+1})\}$, after shuffling we may have pairs such as (x_3, y_8, y_9) and (x_8, y_3, y_4), but never have pairs such as (x_3, y_4, y_9) or (x_4, y_3, y_8), because y_3 is bundled with y_4, and y_8 is bundled with y_9. If the structured HSIC has a form of (4.18) with Gram matrices K_x and K_y possibly assuming more general forms like $\bar{k}_x(x_{i,\tau}, x_{j,\tau})$, the shuffling can be performed directly on the matrix entries. In this case, K_x and K_y can be computed offline and separately. Given a permutation π, a shuffle will change $(K_y)_{st}$ into $(K_y)_{\pi(s)\pi(t)}$. The random shuffling is usually carried out many times and structured HSIC is computed at each time, which results in the null distribution.

Independence test for XOR binary sequences

In this experiment, we compared *iid* HSIC and structured HSIC for independence test. We generated two binary sequences $\mathbf{x}$ and $\mathbf{y}$ of length $T = 400$. The observations in $\mathbf{x}$ were drawn *iid* from a uniform distribution over $\{0, 1\}$. $\mathbf{y}$ were determined by an XOR operation over observations from $\mathbf{x}$: $y_t = x_t \otimes x_{t-1}$. If we treat the observation pairs as *iid*, then the two sequences must appear independent. The undirected graphical model for this data is shown in Figure 4.1b.

For *iid* HSIC, we used maximal cliques $\{(x_t, y_t)\}$ to reflect its underlying *iid* assumption. The corresponding kernel is $\delta(x_s, x_t)\delta(y_s, y_t)$. The maximal cliques for structured HSIC are $\{(x_{t-1}, x_t, y_t)\}$, which takes into account the interdependent nature of the observations. The corresponding kernel is $\delta(x_{s-1}, x_{t-1})\delta(x_s, x_t)\delta(y_s, y_t)$. We tested the null hypothesis $\text{H0} : \mathbf{x} \perp\!\!\!\perp \mathbf{y}$ with both methods at significance level 0.01.

Table 4.1: The number of times HSIC and structured HSIC rejected the null hypothesis.

data	HSIC	p-value	Structured HSIC	p-value
XOR	1	0.44±0.29	100	0±0
RAND	1	0.49±0.28	0	0.49±0.31

The distributions of the test statistics was built by shuffling the paring of kernel entries for 1000 times.

We randomly instantiated the two sequences for 100 times, then counted the number of times each method rejected the null hypothesis (Table 4.1 XOR row). Structured HSIC did a perfect job in detecting the dependence between the sequences, while normal HSIC almost completely missed that out. For comparison, we also generated a second dataset with two independent and uniformly distributed binary sequences. Now both methods correctly detected the independence (Table 4.1 RAND row). We also report the mean and standard deviation of the p-values over the 100 instantiations of the experiment to give a rough picture of the distribution of the p-values.

Independence test for Gaussian processes

In this experiment, we generated two sequences $\mathbf{x} = \{x_t\}_{t=1}^T$ and $\mathbf{y} = \{y_t\}_{t=1}^T$ using the following formulae:

$$\mathbf{x} = A\mathbf{u} \quad \text{and} \quad \mathbf{y} = A\left(\epsilon\mathbf{u} + \sqrt{1-\epsilon^2}\mathbf{v}\right), \tag{4.24}$$

where $A \in \mathbb{R}^{T\times T}$ is a mixing matrix, and $\mathbf{u} = \{u_t\}_{t=1}^T$ and $\mathbf{v} = \{v_t\}_{t=1}^T$ are sequences of *iid* zero-mean and unit-variance normal observations. $\epsilon \in [0,1]$ and larger values of ϵ lead to higher dependence between sequences $\mathbf{x}$ and $\mathbf{y}$. In this setting, both $\mathbf{x}$ and $\mathbf{y}$ are stationary Gaussian processes. Furthermore, due to the mixing matrix A (especially its non-zero off-diagonal elements), observations within $\mathbf{x}$ and $\mathbf{y}$ are interdependent. We expect that an independence test which takes into account this structure will outperform tests assuming *iid* observations. In our experiment, we used $T = 2000$ and $A_{ab} = \exp(-|a-b|/25)$ with all elements below 0.7 clamped to 0. This banded matrix makes the interdependence in $\mathbf{x}$ and $\mathbf{y}$ localized. For structured HSIC, we used the maximal cliques $\{(x_{t,\tau}, y_{t,\tau})\}$ where $\tau = 10$ and linear kernel $\langle x_{s,10}, x_{t,10}\rangle \langle y_{s,10}, y_{t,10}\rangle$.

We varied $\epsilon \in \{0, 0.05, 0.1, \ldots, 0.7\}$. For each value of ϵ, we randomly instantiated $\mathbf{u}$ and $\mathbf{v}$ for 1000 times. For each instantiation, we followed the strategy in (Karvanen, 2005) which formed a new subsequence of length 200 by resampling every d observations and here we used $d = 5$. We tested the null hypothesis H0 : $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$ with 500 random shuffles, and the nominal risk level was set to $\alpha = 0.01$. When $\epsilon = 0$ we are interested

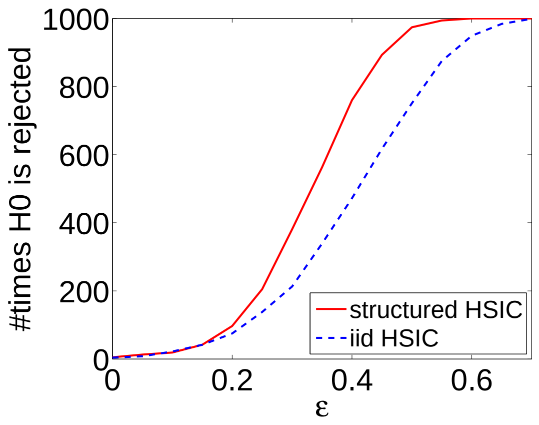


Figure 4.3: Independence test for a Gaussian process.

in the Type I error, *i.e.*, the fraction of times when H0 is rejected which should be no greater than the α. When $\epsilon > 0$ we are concerned about the same fraction, but now called *empirical power* of the test because a higher value is favored. d and τ were chosen to make the comparison fair. Smaller d includes more autocorrelation and increases the empirical power for both *iid* HSIC and structured HSIC, but it causes higher Type I error (see *e.g.*, Table II in Karvanen, 2005). We chose $d = 5$ since it is the smallest d such that Type I error is close to the nominal risk level $\alpha = 0.01$. τ is only for structured HSIC, and in our experiment higher values of τ did not significantly improve the empirical power, but just make the kernels more expensive to compute.

In Figure 4.3, we plot the number of times H0 is rejected. When $\epsilon = 0$, $\mathbf{x}$ and $\mathbf{y}$ are independent and both *iid* HSIC and structured HSIC almost always accept H0. When $\epsilon \in [0.05, 0.2]$, *i.e.*, $\mathbf{x}$ and $\mathbf{y}$ are slightly dependent, both tests have a low empirical power. When $\epsilon > 0.2$, structured HSIC is considerably more sensitive in detecting dependency and consistently rejects H0 more frequently. Note $\mathbf{u}$ and $\mathbf{v}$ have the same weight in Eq. (4.24) when $\epsilon = 2^{-1/2} = 0.71$.

4.4.2 Independent component analysis

In independent component analysis (ICA), we observe a time series of vectors $\mathbf{t}$ that corresponds to a linear mixture $\mathbf{t} = A\mathbf{s}$ of n mutually independent sources $\mathbf{s}$ (each entry in the source vector $\mathbf{s}$ here is a random process, and depends on its past values; examples include music and EEG time series). Based on the series of observations $\mathbf{t}$, we wish to recover the sources using only the independence assumption on $\mathbf{s}$. Note that sources can only be recovered up to scaling and permutation. The core of ICA is a contrast function that measures the independence of the estimated sources. An ICA algorithm searches over the space of mixing matrix A such that this contrast function is minimized. Thus, we propose to use structured HSIC as the contrast function for ICA. By incorporating time lagged variables in the cliques, we expect that structured HSIC can better deal with the non-*iid* nature of time series. In this respect, we generalize the TD-SEP algorithm (Ziehe & Müller, 1998), which implements this idea using a linear kernel on the signal. Thus, we address the question of whether correlations between higher order moments, as encoded using non-linear kernels, can improve the performance of TD-SEP on real data.

Data Following the setting of (Gretton et al., 2005b, Section 5.5), we unmixed various musical sources, combined using a randomly generated orthogonal matrix A (since optimization over the orthogonal part of a general mixing matrix is the more difficult step in ICA). We considered mixtures of two to four sources, drawn at random without replacement from 17 possibilities. We used the sum of pairwise dependencies as the overall contrast function when more than two sources were present.

Methods We compared structured HSIC to TD-SEP and *iid* HSIC. While *iid* HSIC does not take the temporal dependence in the signal into account, it has been shown to perform very well for *iid* data (Shen et al., 2009). Following Gretton et al. (2005b), we employed a Laplace kernel, $\bar{k}_x(x, x') = \exp(-\lambda\|x - x'\|)$ with $\lambda = 3$ for both structured and *iid* HSIC. For both structured and *iid* HSIC, we used gradient descent over the orthogonal group with a Golden search, and low rank Cholesky decompositions of the Gram matrices to reduce computational cost, as in (Bach & Jordan, 2002).

Results We chose the Amari divergence as the index for comparing performance of the various ICA methods. This is a divergence measure between the estimated and true unmixing matrices, which is invariant to the output ordering and scaling ambiguities. A smaller Amari divergence indicates better performance. Results are shown in Table 4.2. Overall, contrast functions that take time delayed information into account perform best, although the best time lag is different when the number of sources varies.

4.4.3 Time series clustering and segmentation

We can also extend clustering to time series and sequences using structured HSIC. This is carried out in a similar way to the *iid* case. One can formulate clustering as generating the labels $\mathbf{y}$ from a finite discrete set, such that their dependence on $\mathbf{x}$ is maximized (Song et al., 2007b):

$$\text{maximize}_{\mathbf{y}} \quad \operatorname{tr} HK_xHK_y \quad \textit{subject to} \text{ constraints on } \mathbf{y}. \tag{4.25}$$

Table 4.2: Median performance of ICA on music using HSIC, TDSEP, and structured HSIC. In the top row, the number m of sources and n of samples are given. In the second row, the number of time lags τ used by TDSEP and structured HSIC are given: thus the observation vectors $x_t, x_{t-1}, \ldots, x_{t-\tau}$ were compared. The remaining rows contain the median Amari divergence (multiplied by 100) for the three methods tested. The original HSIC method does not take into account time dependence ($\tau = 0$), and returns a single performance number. Results are in all cases averaged over 136 repetitions: for two sources, this represents all possible pairings, whereas for larger m the sources are chosen at random without replacement.

Method	$m = 2$, $n = 5000$			$m = 3$, $n = 10000$			$m = 4$, $n = 10000$		
	1	2	3	1	2	3	1	2	3
HSIC	1.51			1.70			2.68		
TDSEP	1.54	1.62	1.74	1.84	1.72	1.54	2.90	2.08	1.91
Structured HSIC	1.48	1.62	1.64	1.65	1.58	1.56	2.65	2.12	1.83

Here K_x and K_y are the kernel matrices for $\mathbf{x}$ and the generated $\mathbf{y}$ respectively. More specifically, assuming $(K_y)_{st} := \delta(y_s, y_t)$ for discrete labels $\mathbf{y}$, we recover clustering. Relaxing discrete labels to $y_t \in \mathbb{R}$ with bounded norm $\|\mathbf{y}\|_2$ and setting $(K_y)_{st} := y_s y_t$, we obtain principal component analysis.

This reasoning for *iid* data carries over to sequences by introducing additional dependence structure through the kernels: $(K_x)_{st} := \bar{k}_x(x_{s,\tau}, x_{t,\tau})$ and $(K_y)_{st} := \bar{k}_y(y_{s,\tau}, y_{t,\tau})$. In general, the interacting label sequences make the optimization in (4.25) intractable. However, for a class of kernels $\bar{k}_y$ an efficient decomposition can be found by applying a reverse convolution on $\bar{k}_x$.

Efficient optimization for convolution kernels

Suppose the kernel $\bar{k}_y$ assumes a special form given by

$$\bar{k}_y(y_{s,\tau}, y_{t,\tau}) = \sum\nolimits_{u,v=0}^{\tau} \bar{k}_y^{\circ}(y_{s+u}, y_{t+v}) M_{uv}, \tag{4.26}$$

where $M \in \mathbb{R}^{(\tau+1)\times(\tau+1)}$ is positive semi-definite, and $\bar{k}_y^{\circ}$ is a base kernel between individual time points. A common choice is $\bar{k}_y^{\circ}(y_s, y_t) = \delta(y_s, y_t)$. In this case we can rewrite $\operatorname{tr} HK_xHK_y$ by applying the summation over M to HK_xH, *i.e.*,

$$\sum_{s,t=1}^{T} [HK_xH]_{ij} \sum_{u,v=0}^{\tau} \bar{k}_y^{\circ}(y_{s+u}, y_{t+v}) M_{uv} = \sum_{s,t=1}^{T+\tau} \underbrace{\sum_{\substack{u,v=0 \\ s-u,t-v\in[1,T]}}^{\tau} M_{uv} [HK_xH]_{s-u,t-v}}_{:=K^{\star}_{st}} \bar{k}_y^{\circ}(y_s, y_t) \tag{4.27}$$

This means that we may apply the matrix M to HK_xH and thereby we are able to decouple the dependency within $\mathbf{y}$. That is, in contrast to $\bar{k}_y$ which couples two subsequences of $\mathbf{y}$, $\bar{k}^\circ_y$ only couples two individual elements of $\mathbf{y}$. As a result, the optimization over $\mathbf{y}$ is made much easier. Denoting the convolution by $K^\star = [HK_xH] \star M$, we can directly apply (4.25) to time series and sequence data in the same way as *iid* data, treating $K^\star$ as the original K_x. In practice, approximate algorithms such as incomplete Cholesky decomposition are needed to efficiently compute and represent $K^\star$ and the details can be found in Appendix G.

Empirical Evaluation

Datasets We studied two datasets in this experiment.

1. **Swimming dataset**. The first dataset was collected by the Australian Institute of Sports (AIS) from a 3-channel orientation sensor attached to a swimmer which monitors: 1. the body orientation by a 3-channel magnetometer; 2. the acceleration by a 3-channel accelerometer. The three time series we used in our experiment have the following configurations: $T = 23000$ time steps with 4 laps; $T = 47000$ time steps with 16 laps; and $T = 67000$ time steps with 20 laps. The task is to automatically find the starting and finishing time of each lap based on the sensor signals. We treated this problem as a segmentation problem, and used orientation data for our experiments because they lead to better results than the acceleration signals. Since the dataset contains four different styles of swimming, we assumed there are six states/clusters for the sequence: four clusters for the four styles of swim, two clusters for approaching and leaving the end of the pool (finishing and starting a lap, respectively).

2. **BCI dataset**. The second dataset is a brain-computer interface data (data IVb of Berlin BCI group[4]). It contains EEG signals collected when a subject was performing three types of cued imagination: `left`, `foot`, and `relax`. Between every two successive imaginations, there is an interim. So an example state sequence is:

```
left, interim, relax, interim, foot, interim, relax, interim,...
```

Therefore, the `left/foot/relax` states correspond to the swimming styles and the interim corresponds to the turning at the end or beginning of the laps. Including the interim period, the dataset consists of $T = 10000$ time points with 16 different segments (32 boundaries). The task is to automatically detect the start and end of an imagination. We used four clusters for this problem.

We preprocessed the raw signal sequences by applying them to a bandpass filter which only keeps the frequency range from 12Hz to 14Hz. Besides, we followed the

[4]http://ida.first.fraunhofer.de/projects/bci/competition-iii/desc-IVb.html

common practice and only used the following electrode channels (basically those in the middle of the test region):

```
33,34,35,36,37,38,39,42,43,44,45,46,47,48,49,51,52,53,54,
55,56,57,59,60,61,62,63,64,65,66,69,70,71,72,73,74,75.
```

Finally, for both swimming and BCI datasets, we smoothed the raw data with moving averages, *i.e.*, $x_t \leftarrow \sum_{\tau=-w}^{w} x_{t+\tau}^{\text{raw}}$ followed by normalization to zero mean and unit variance for each feature dimension. Here w is set to 100 for swimming data and 50 for BCI data due to its higher frequency of state switching. This smoothed and normalized x was used by *all* the three algorithms.

Methods We compared three algorithms: structured HSIC for clustering, spectral clustering (Ng et al., 2002), and HMM.

1. **Structured HSIC**. For the three swimming datasets, we used the maximal cliques of $\{(x_t, y_{t-50,100})\}$ for structured HSIC, where $\mathbf{y}$ is the discrete label sequence to be generated. Time lagged labels in the maximal cliques reflect the fact that clustering labels keep the same for a period of time. The kernel $\bar{k}_y$ took the form of equation (4.26), with $M \in \mathbb{R}^{101\times 101}$ and $M_{ab} := \exp(-(a-b)^2)$. We used the technique described in Section 4.4.3 to shift the dependence within $\mathbf{y}$ into $\mathbf{x}$. The kernel $\bar{k}_x$ was RBF: $\exp(-\|x_s - x_t\|^2)$. We performed kernel k-means clustering based on the convolved kernel matrix $K^\star$. To avoid the local minima of k-means, we randomly initialized it for 20 times and reported the error made by the model which has the lowest sum of point-to-centroid distances. The parameters for BCI dataset are the same, except that $M \in \mathbb{R}^{51\times 51}$ to reflect the fact that state changes more frequently in this dataset.

2. **Spectral clustering.** We first applied the algorithm in (Ng et al., 2002) on $\mathbf{x}$ and it yielded far larger error, and hence is not reported here. Then we applied its kernelized version to the convolved kernel $K^\star$. We used 100 nearest neighbors with distance function $\exp(-\|x_i - x_j\|^2)$. These parameters delivered uniformly best result.

3. **HMM**. We trained a first order homogeneous HMM by the EM algorithm with 6 hidden states for swimming dataset and 4 states for BCI dataset, and its observation model contained diagonal Gaussians. After training, we used Viterbi decoding to determine the cluster labels. We used the implementation from Torch[5]. To regularize, we tried a range of minimum variance $\sigma \in \{0.5, 0.6, ..., 2.0\}$. For each σ, we randomly initialized the training of HMM for 50 times to avoid local maxima of EM, and computed the error incurred by the model which yielded the highest likelihood on the whole sequence. Finally, we reported the minimum error over all σ.

[5]http://www.torch.ch

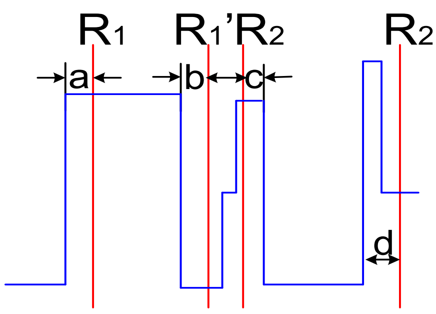


Figure 4.4: Illustration of error calculation. Red lines denote the ground truth and blues line are the segmentation results. The error introduced for segment R_1 to R'_1 is $a + b$, while that for segment R_2 to R'_2 is $c + d$. The overall error in this example is then $(a + b + c + d)/4$.

Table 4.3: Segmentation errors by various methods on the four studied time series.

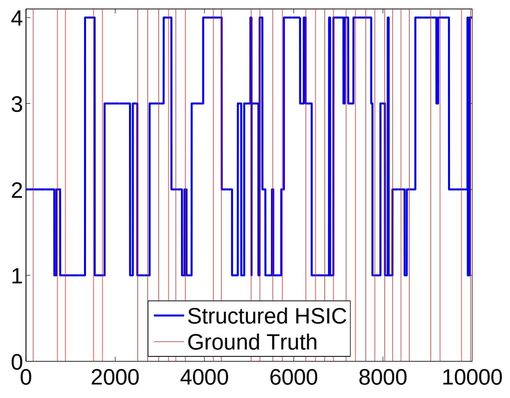
Method	Swimming 1	Swimming 2	Swimming 3	BCI
structured HSIC	**99.0**	**118.5**	**108.6**	**111.5**
spectral clustering	125	212.3	143.9	162
HMM	153.2	120	150	168

Results To evaluate the segmentation quality, the boundaries found by various methods were compared against the ground truth. First, each detected boundary was matched to a true boundary, and then the discrepancy between them was counted into the error. The overall error was this sum divided by the number of boundaries. Figure 4.4 gives an example on how to compute this error.

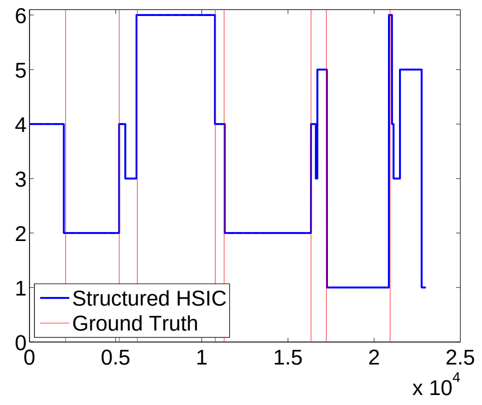
According to Table 4.3, in all of the four time series we studied, segmentation using structured HSIC leads to lower error compared with spectral clustering and HMM. For instance, structured HSIC reduces nearly 1/3 of the segmentation error in the BCI dataset. We also plot the true boundaries together with the segmentation results produced by structured HSIC, spectral clustering, and HMM respectively. Figures 4.6 to 4.8 present the results for the three swimming datasets, and Figure 4.5 for the BCI dataset. Although the results of swimming data in Figure 4.6 to 4.8 are visually similar among all algorithms, the average error produced by structured HSIC is much smaller than that of HMM or spectral clustering. Finally, the segment boundaries of BCI data produced by structured HSIC clearly fit better with the ground truth.

4.5 Conclusion

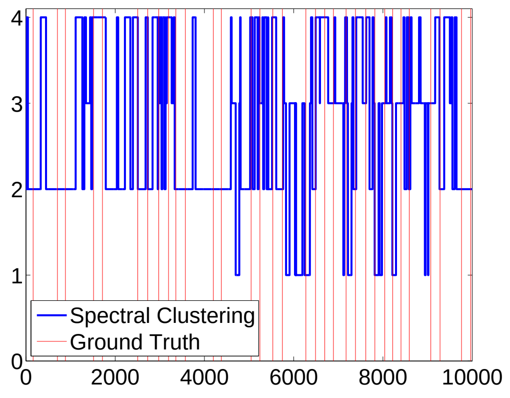
In this paper, we extended the Hilbert Schmidt Independence Criterion from *iid* data to structured and non-*iid* data. Our approach is based on RKHS embeddings of distributions, and utilizes the efficient factorizations provided by the exponential family



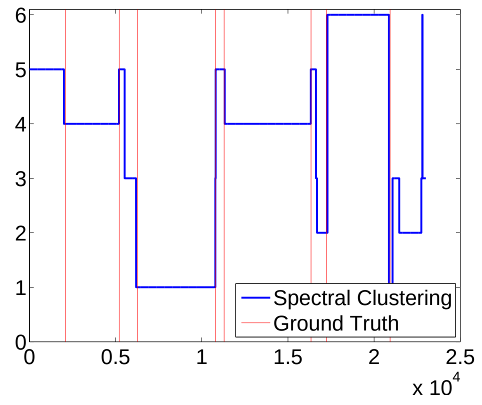
(a) Structured HSIC



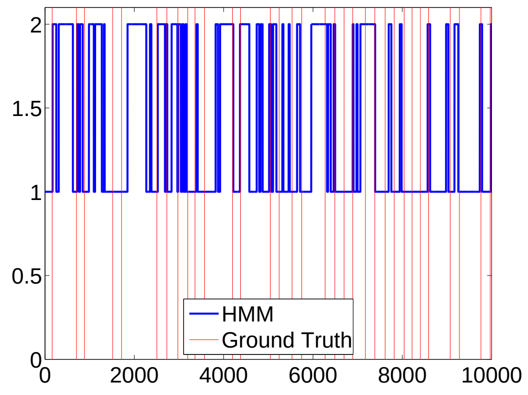
(a) Structured HSIC



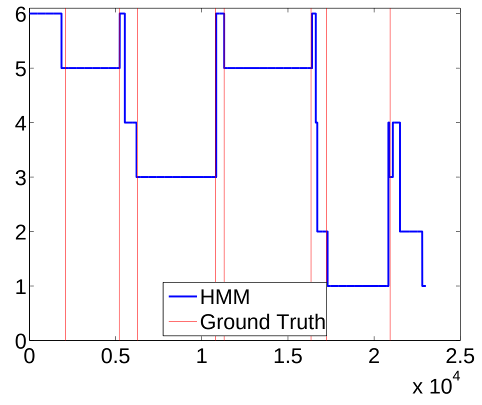
(b) Spectral Clustering



(b) Spectral Clustering



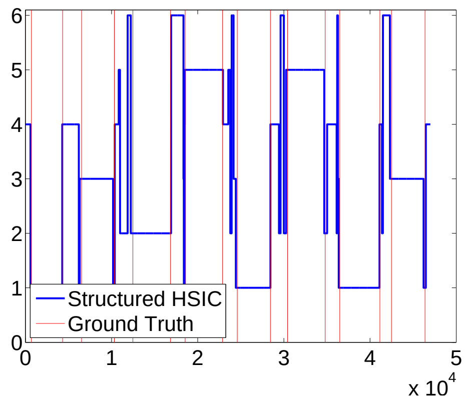
(c) HMM



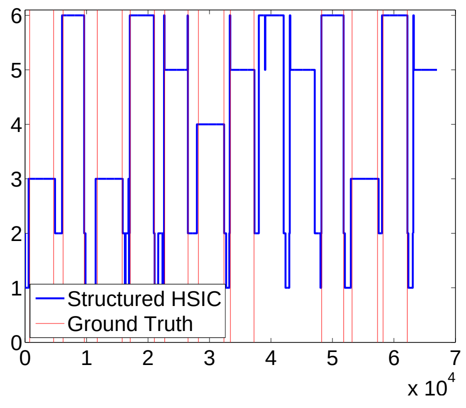
(c) HMM

Figure 4.5: Segmentation results of BCI dataset produced by (a) structured HSIC, (b) spectral clustering and (c) HMM. In (c), we did specify 4 hidden states, but the Viterbi decoding showed only two states were used.

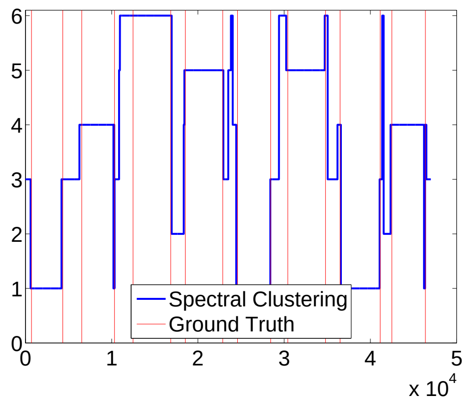
Figure 4.6: Segmentation results of swimming dataset 1 produced by (a) structured HSIC, (b) spectral clustering and (c) HMM.



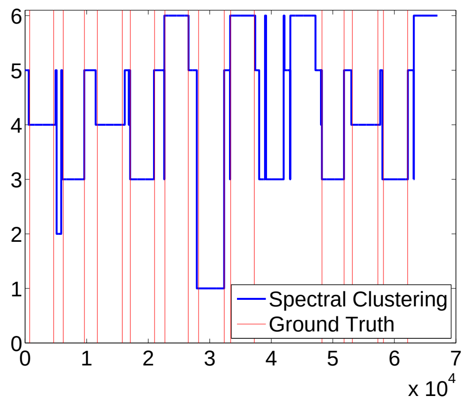
(a) Structured HSIC



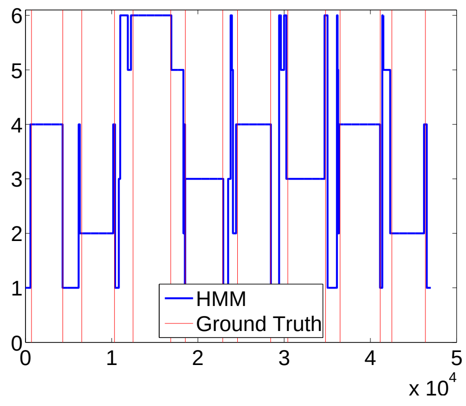
(a) Structured HSIC



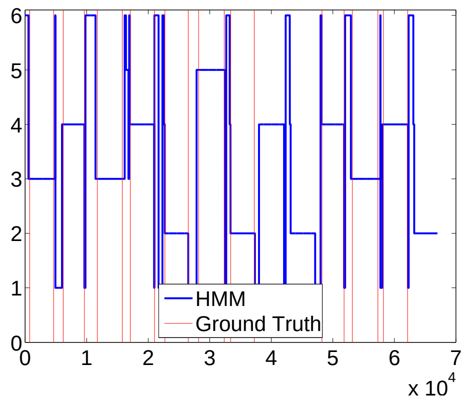
(b) Spectral Clustering



(b) Spectral Clustering



(c) HMM



(c) HMM

Figure 4.7: Segmentation results of swimming dataset 2 produced by (a) structured HSIC, (b) spectral clustering and (c) HMM.

Figure 4.8: Segmentation results of swimming dataset 3 produced by (a) structured HSIC, (b) spectral clustering and (c) HMM.

associated with undirected graphical models. Encouraging experimental results were demonstrated on independence test, ICA, and segmentation for time series. Further work will be done in the direction of applying structured HSIC to PCA and feature selection on structured data. It will be also impacting in theory to study the asymptotic bounds for grid structured non-*iid* data, in the same line as Section 4.3.2 and (Kontorovich, 2007).

Chapter 5

Lower Bounds for BMRM and Faster Rates for Training SVMs

CRFs are log-linear models for learning from structured data. A similar approach to this task is maximum margin models (*e.g.*, Taskar et al., 2004) which a) allows decomposition of loss and variable parameterization along the graphical models, and b) admits straightforward kernelization to implicitly model nonlinear dependencies with a much richer feature space while still retaining linear estimation. However, the nonsmooth objective function poses new challenges in optimization (Collins et al., 2008; Taskar et al., 2006). Fortunately, the problem can be cast as an example of regularized risk minimization, for which bundle methods (BMRM, Teo et al., 2007) and the closely related `SVM`$^{\texttt{Struct}}$ (Tsochantaridis et al., 2005) are state-of-the-art general purpose solvers. Section 1.6 provided a brief introduction to these solvers.

Smola et al. (2007b) proved that BMRM requires $O(1/\epsilon)$ iterations to converge to an ϵ accurate solution, and we will show in this chapter that this rate is tight, *i.e.* there exists a function for which BMRM requires $O(1/\epsilon)$ steps. Motivated by Nesterov's optimal first-order methods (Nesterov, 1983, 2005a, 2007), we further devise an algorithm for the structured loss which finds an ϵ accurate solution in $O(1/\sqrt{\epsilon})$ iterations.

Let $\mathbf{x}^i \in \mathcal{X} \subseteq \mathbb{R}^d$ denote the feature vector of examples and $y_i \in \mathcal{Y}$ be the corresponding labels[1]. Given a training set of n sample label pairs $\left\{(\mathbf{x}^i, y_i)\right\}_{i=1}^n$, drawn i.i.d. from a joint probability distribution on $\mathcal{X} \times \mathcal{Y}$, many machine learning algorithms solve the following regularized risk minimization problem:

$$\min_{\mathbf{w}} J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R_{\text{emp}}(\mathbf{w}), \text{ where } R_{\text{emp}}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{x}^i, y_i; \mathbf{w}). \tag{5.1}$$

Here $l(\mathbf{x}^i, y_i; \mathbf{w})$ denotes the loss on instance $(\mathbf{x}^i, y_i)$ using the current model $\mathbf{w}$ and

[1]We first discuss binary SVMs and therefore use the symbol y_i without boldface. Structured outputs will be discussed in Section 5.4 where we will use $\mathbf{y}$ to denote the structured labels.

$R_{\mathrm{emp}}(\mathbf{w})$, the empirical risk, is the average loss on the training set. The regularizer $\Omega(\mathbf{w})$ acts as a penalty on the complexity of the classifier and prevents overfitting. Usually the loss is convex in $\mathbf{w}$ but can be nonsmooth while the regularizer is usually a smooth strongly convex function. Binary support vector machines (SVMs) are a prototypical example of such regularized risk minimization problems where $\mathcal{Y} = \{1, -1\}$ and the loss considered is the binary hinge loss:

$$l(\mathbf{x}^i, y_i; \mathbf{w}) = \left[1 - y_i \left\langle \mathbf{w}, \mathbf{x}^i \right\rangle\right]_+, \text{ with } [\cdot]_+ := \max(0, \cdot). \tag{5.2}$$

Recently, a number of solvers have been proposed for the regularized risk minimization problem. The first and perhaps the best known solver is SVM$^{\texttt{Struct}}$ by Tsochantaridis et al. (2005), which was shown to converge in $O(1/\epsilon^2)$ iterations to an ϵ accurate solution. The convergence analysis of SVM$^{\texttt{Struct}}$ was improved to $O(1/\epsilon)$ iterations by Smola et al. (2007b). In fact, Smola et al. (2007b) showed that their convergence analysis holds for a more general solver than SVM$^{\texttt{Struct}}$ namely BMRM (Bundle method for regularized risk minimization).

At every iteration BMRM replaces R_{emp} by a piecewise linear lower bound R_t^{cp} and optimizes

$$\min_{\mathbf{w}} J_t(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R_t^{\mathrm{cp}}(\mathbf{w}), \quad \text{where } R_t^{\mathrm{cp}}(\mathbf{w}) := \max_{1 \le i \le t} \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i, \tag{5.3}$$

to obtain the next iterate $\mathbf{w}_t$. Here $\mathbf{a}_i \in \partial R_{\mathrm{emp}}(\mathbf{w}_{i-1})$ denotes an arbitrary subgradient of R_{emp} at $\mathbf{w}_{i-1}$ and $b_i = R_{\mathrm{emp}}(\mathbf{w}_{i-1}) - \langle \mathbf{w}_{i-1}, \mathbf{a}_i \rangle$. The piecewise linear lower bound is successively tightened until the gap

$$\epsilon_t := \min_{0 \le t' \le t} J(\mathbf{w}_{t'}) - J_t(\mathbf{w}_t), \tag{5.4}$$

falls below a predefined tolerance ϵ. The full details of BMRM can be found in Section 1.6.4 and (Teo et al., 2010).

Even though BMRM solves an expensive optimization problem at every iteration, the convergence analysis only uses a simple one-dimensional line search (Algorithm 3 in Chapter 1) to bound the decrease in ϵ_t. Furthermore, the empirical convergence behavior of BMRM is much better than the theoretically predicted rates on a number of real life problems (Teo et al., 2010, Section 5). It was therefore conjectured that the rates of convergence of BMRM could be improved. In Section 5.2, we answer this question in the negative by explicitly constructing a regularized risk minimization problem for which BMRM takes at least $O(1/\epsilon)$ iterations.

One possible way to circumvent the $O(1/\epsilon)$ lower bound is to solve the problem in the dual. Using a very old result of Nesterov (1983) we obtain in Section 5.3 an algorithm for SVMs which only requires $O(1/\sqrt{\epsilon})$ iterations to converge to an ϵ accurate solution; each iteration of the algorithm requiring $O(nd)$ work. Although we primarily focus on the regularized risk minimization with the binary hinge loss, our algorithm can also be used whenever the empirical risk is piecewise linear and contains a small number of pieces. Examples of this include multi-class, multi-label, and ordinal regression hinge loss and other related losses. Extension to more general structured output spaces is also feasible as long as it factorizes according to a graphical model. Section 5.4 shows the details. Finally, experimental results will be presented in Section 5.5 which confirm our bounds.

5.1 Preliminaries

In this section, we quickly recap the necessary convex analysis concepts. A brief introduction to convex analysis is available in Appendix A, and more details can be found in textbooks like (Hiriart-Urruty & Lemaréchal, 1993a; Rockafellar, 1970). Unless specified otherwise, $\|\cdot\|$ refers to the Euclidean norm $\|\mathbf{w}\| := \left(\sum_{i=1}^{n} w_i^2\right)^{\frac{1}{2}}$. $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$, and $[t] := \{1, \ldots, t\}$. The dom of a convex function f is defined by $\operatorname{dom} f := \{\mathbf{w} : f(\mathbf{w}) < \infty\}$. Δ_k refers to the k dimensional simplex. The following three notions will be used extensively:

Definition 52 (Strong convexity) *A convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is strongly convex (s.c.) wrt norm $\|\cdot\|$ if there exists a constant $\sigma > 0$ such that $f - \frac{\sigma}{2}\|\cdot\|^2$ is convex. σ is called the modulus of strong convexity of f, and for brevity we will call f σ-strongly convex or σ-s.c..*

Definition 53 (Lipschitz continuous gradient) *A function f is said to have Lipschitz continuous gradient (l.c.g) if there exists a constant L such that*

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\| \qquad \forall\ \mathbf{w} \text{ and } \mathbf{w}'. \tag{5.5}$$

For brevity, we will call f L-l.c.g.

Definition 54 (Fenchel duality) *The Fenchel dual of a function $f : E_1 \to E_2$, is a function $f^\star : E_2^\star \to E_1^\star$ given by*

$$f^\star(\mathbf{w}^\star) = \sup_{\mathbf{w} \in E_1} \{\langle \mathbf{w}, \mathbf{w}^\star \rangle - F(\mathbf{w})\} \tag{5.6}$$

The following theorem specifies the relationship between strong convexity of a primal function and Lipschitz continuity of the gradient of its Fenchel dual.

Theorem 55 *(Hiriart-Urruty & Lemaréchal, 1993a, Theorem 4.2.1 and 4.2.2)*

1. *If $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is σ-strongly convex, then* $\operatorname{dom} f^\star = \mathbb{R}^n$ *and $\nabla F^\star$ is $\frac{1}{\sigma}$-l.c.g.*
2. *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-l.c.g, then $f^\star$ is $\frac{1}{L}$-strongly convex.*

Subgradients generalize the concept of gradients to nonsmooth functions. For $\mathbf{w} \in \operatorname{dom} f$, $\boldsymbol{\mu}$ is called a subgradient of f at $\mathbf{w}$ if

$$f(\mathbf{w}') \geq f(\mathbf{w}) + \left\langle \mathbf{w}' - \mathbf{w}, \boldsymbol{\mu} \right\rangle \quad \forall \mathbf{w}'. \tag{5.7}$$

The set of all subgradients at $\mathbf{w}$ is called the subdifferential, denoted by $\partial f(\mathbf{w})$. If f is convex, then $\partial f(\mathbf{w}) \neq \emptyset$ for all $\mathbf{w} \in \operatorname{dom} F$, and is a singleton if, and only if, f is differentiable (Hiriart-Urruty & Lemaréchal, 1993a).

Any piecewise linear convex function $f(\mathbf{w})$ with t linear pieces can be written as

$$f(\mathbf{w}) = \max_{i \in [t]} \{\langle \mathbf{a}_i, \mathbf{w} \rangle + b_i\}, \tag{5.8}$$

for some $\mathbf{a}_i$ and b_i. If the empirical risk R_{emp} is a piecewise linear function then the convex optimization problem in (5.1) can be expressed as

$$\min_{\mathbf{w}} J(\mathbf{w}) := \min_{\mathbf{w}} \max_{i \in [t]} \{\langle \mathbf{a}_i, \mathbf{w} \rangle + b_i\} + \lambda \Omega(\mathbf{w}). \tag{5.9}$$

Let $A := (\mathbf{a}_1, \ldots, \mathbf{a}_t)$ and $\mathbf{b} := (b_1, \ldots, b_n)$, then the *adjoint* form of $J(\mathbf{w})$ can be written as

$$D(\boldsymbol{\alpha}) := -\lambda \Omega^\star(-\lambda^{-1} A \boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \mathbf{b} \rangle \text{ with } \boldsymbol{\alpha} \in \Delta_t \tag{5.10}$$

where the primal and the adjoint optimum are related by

$$\mathbf{w}^* = \partial \Omega^\star(-\lambda^{-1} A \boldsymbol{\alpha}^*). \tag{5.11}$$

In fact, using concepts of strong duality (see *e.g.*Theorem 2 of (Teo et al., 2010)), it can be shown that

$$\inf_{\mathbf{w} \in \mathbb{R}^d} \left\{ \max_{i \in [n]} \langle \mathbf{a}_i, \mathbf{w} \rangle + b_i + \lambda \Omega(\mathbf{w}) \right\} = \sup_{\boldsymbol{\alpha} \in \Delta_t} \left\{ -\lambda \Omega^\star(-\lambda^{-1} A \boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \mathbf{b} \rangle \right\} \tag{5.12}$$

5.2 Lower bounds

The following result was shown by Smola et al. (2007b):

Theorem 56 (Theorem 4 of (Smola et al., 2007b)) *Assume that $J(\mathbf{w}) \geq 0$ for all $\mathbf{w}$, and that $\|\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w})\| \leq G$ for all $\mathbf{w} \in W$, where W is some domain of interest containing all $\mathbf{w}_{t'}$ for $t' \leq t$. Also assume that Ω^* has bounded curvature, i.e. let*

$\left\|\partial^2_{\boldsymbol{\mu}}\Omega^*(\boldsymbol{\mu})\right\| \leq H^*$ *for all* $\boldsymbol{\mu} \in \left\{-\lambda^{-1}A\boldsymbol{\alpha} \text{ where } \boldsymbol{\alpha} \in \Delta_t\right\}$*. Then, for any* $\epsilon < 4G^2H^*/\lambda$ *we have* $\epsilon_t < \epsilon$ *after at most*

$$\log_2 \frac{\lambda J(\mathbf{0})}{G^2H^*} + \frac{8G^2H^*}{\lambda\epsilon} - 4 \tag{5.13}$$

steps.

Although the above theorem proves an upper bound of $O(1/\epsilon)$ on the number of iterations, the tightness of this bound has been an open question. We now demonstrate a function which satisfies all the conditions of the above theorem, and yet takes $\Omega(1/\epsilon)$ iterations to converge.

5.2.1 Concepts and notations

Since most rates of convergence discussed in machine learning community are upper bounds, we will first clarify the meaning of lower bound wrt ϵ, with special attention paid to the qualifiers of objective functions and optimization algorithms.

Given a function f and an optimization algorithm, we define $T(\epsilon; f)$ as the number of steps required to reduce the gap defined in Eq. (5.4) to less than ϵ[2]:

$$T(\epsilon; f) = \max\left\{k : f(\mathbf{w}_k) - f^* \geq \epsilon\right\}.$$

Since convergence rates are often expressed in $O(\cdot)$ form, comparisons need to be redefined up to multiplicative constants. It is intuitive to define the following total order on the convergence rate.

Type	Meaning
$g(\epsilon) \prec h(\epsilon)$	$\lim_{\epsilon\to 0} g(\epsilon)/h(\epsilon) = 0$
$g(\epsilon) \succ h(\epsilon)$	$\lim_{\epsilon\to 0} h(\epsilon)/g(\epsilon) = 0$
$g(\epsilon) \sim h(\epsilon)$	$\lim_{\epsilon\to 0} g(\epsilon)/h(\epsilon) = C \in (0, +\infty)$
$g(\epsilon) \preccurlyeq h(\epsilon)$	$g(\epsilon) \prec h(\epsilon)$ or $g(\epsilon) \sim h(\epsilon)$
$g(\epsilon) \succcurlyeq h(\epsilon)$	$g(\epsilon) \succ h(\epsilon)$ or $g(\epsilon) \sim h(\epsilon)$

Special attention should be paid to the qualifications in upper and lower bounds. Upper bounds are usually qualified by "for all functions and for all ϵ", it takes at most

[2]Indeed the initial point also matters: in the best case one just starts from the optimal $x_0 = x^*$. So rigorously, it should be $T(\epsilon; f, x_0)$. Here for simplicity, we omit the x_0 which is always qualified as existential in lower bounds and universal in upper bounds. Furthermore, an algorithm can be random to some extent, *e.g.*, pick a subgradient in the subdifferential. Again, we qualify it as existential in lower bound and universal in upper bound.

$O(g(\epsilon))$ steps to reduce the gap to ϵ. However, for lower bounds, the two "for all" may be turned into "there exist" in two different ways.

Type	Meaning
Upper bound:	For all function f, $T(\epsilon; f) \preccurlyeq g(\epsilon)$
Strong lower bound (SLB):	There exists a function f, such that $T(\epsilon; f) \succcurlyeq g(\epsilon)$.
Weak lower bound (WLB):	For all ϵ, there exists a function f_ϵ which may depend on ϵ, such that $T(\epsilon; f_\epsilon) \geq g(\epsilon)$.

Clearly, if SLB holds, WLB must hold, but not vice versa. A simple example is the well known WLB for cutting plane (Hiriart-Urruty & Lemaréchal, 1993a, Example 1.1.2 of Chapter XV): for all ϵ, there exists a function f_ϵ with n variables such that it takes $k \geq O(1/\epsilon^{n/2})$ steps to ensure $f(\mathbf{w}_k) - f^* < \epsilon$. However, after that many steps, the gap immediately drops to 0, hence it is not admissible as a SLB example. Incidentally, there is no known SLB example for cutting plane algorithm (Nemirovski, 2009).

Both upper bound and SLB are defined in an asymptotic fashion, while WLB is not. Obviously SLB and upper bound are incompatible, *i.e.*, if $g(\epsilon) \prec h(\epsilon)$, then the following cannot be true at the same time: a) for all function f, $T(\epsilon; f) \preccurlyeq g(\epsilon)$; and b) there exists a function f such that $T(\epsilon; f) \succcurlyeq h(\epsilon)$. However, WLB and upper bound are compatible, *i.e.*, given an optimization algorithm there can be two rates $g(\epsilon) \prec h(\epsilon)$ such that the following two hold simultaneously:

1. for all function f there exist a constant C_f such that for all ϵ, it takes *at most* $k = C_f g(\epsilon)$ steps to ensure $f(x_k) - f^* < \epsilon$
2. for all ϵ, there exists a function f_ϵ which may depend on ϵ, such that it takes at least $k = h(\epsilon)$ steps to ensure $f_\epsilon(\mathbf{w}_k) - f_\epsilon^* < \epsilon$.[3]

The objective function domain $\mathcal{F}$ under consideration is also important. When discussing upper bounds where f is universally qualified, broader $\mathcal{F}$ leads to stronger statements. However, when discussing lower bounds where f is existential, a narrower $\mathcal{F}$ means stronger claims.

For both upper and lower bounds, a statement is stronger if the domain of the algorithm is more general. For example, Nesterov (2003) showed a WLB for all algorithms that satisfy the condition that $\mathbf{w}_{k+1}$ lies in the linear span of previous subgradients $\{\nabla f(\mathbf{w}_i)\}_{i=0}^{k}$.

[3]Note we do not further allow a function-dependent constant C_f since f is qualified as existential.

To understand optimization algorithms, both WLB and SLB are important. The algorithm domain of this section is restricted to BMRM, and we construct SLB and WLB examples for binary linear SVM objectives, as well as a WLB example for L_2 regularized piecewise linear objectives.

5.2.2 Strong lower bounds

Strong lower bounds are harder to construct and prove. We now demonstrate one for ls-bmrm (see Section 1.6.4 for an introduction to ls-bmrm and qp-bmrm). In particular, we show that the primal gap of ls-bmrm on an SVM training example is decreased at $O(1/k)$ rate where k is the step index. Similar examples can be constructed to show the $O(1/\epsilon)$ SLB for pegasos, which is given in Appendix H. The SLB for qp-bmrm is an open problem.

Consider the following training instances in 1-d space. Let $x^i \in \mathbb{R}$ be features and $y^i \in \{-1, 1\}$ be labels. Pick $(x^1, y^1) = (1, 1)$, $(x^2, y^2) = (-1, -1)$, $(x^3, y^3) = (\frac{1}{2}, 1)$, and $(x^4, y^4) = (-\frac{1}{2}, -1)$. Set $\lambda = \frac{1}{16}$. Then the objective function of a SVM can be written as:

$$\min_{w \in \mathbb{R}} J(w) = \min_{w \in \mathbb{R}} \frac{1}{32} w^2 + \frac{1}{2} [1 - w]_+ + \frac{1}{2} \left[1 - \frac{w}{2}\right]_+ . \tag{5.14}$$

Our main result is the following Theorem:

Theorem 57 $\lim_{k \to \infty} k(J(w_k) - J(w^*)) = \frac{1}{4}$, *where* $w^* = \operatorname{argmin}_w J(w)$.

The proof is based on the fact that $\{w_k\}$ oscillates about and approaches $w^* = 2$ at the rate of $1/k$:

Lemma 58 $\lim_{k \to \infty} k \, |2 - w_k| = 2$ *with* $w_{2k+1} > 2$ *and* $w_{2k} \in (1, 2)$.

To this end, we establish a recursive relation between w_k and $\alpha_{k,1}$, the first element of the solution $\boldsymbol{\alpha}_k$ of the inner dual problem in BMRM (Eq. (5.10) or step 2 of Algorithm 2 in Chapter 1).

Lemma 59 *For* $k \geq 1$, *we have*

$$w_{2k+1} = 2 \frac{w_{2k-1}^3 + 12\alpha_{2k-1,1} w_{2k-1}^2 + 16 w_{2k-1} \alpha_{2k-1,1}^2 - 64 \alpha_{2k-1,1}^3}{w_{2k-1} \left(w_{2k-1} + 4\alpha_{2k-1,1}\right)^2} > 2, \tag{5.15}$$

$$\alpha_{2k+1,1} = \frac{w_{2k-1}^2 + 16\alpha_{2k-1,1}^2}{\left(w_{2k-1} + 4\alpha_{2k-1,1}\right)^2} \alpha_{2k-1,1}, \tag{5.16}$$

$$w_{2k} = 2 - \frac{8\alpha_{2k-1,1}}{w_{2k-1}} \in (1, 2). \tag{5.17}$$

(5.15) *and* (5.16) *provide recursive formulae to compute* w_{2k+1} *and* $\alpha_{2k+1,1}$ *based on* w_{2k-1} *and* $\alpha_{2k-1,1}$, *and* (5.17) *gives* w_{2k}.

The straightforward but technical proof for Theorem 57, Lemma 58 and 59 can be found in Appendix H. Also notice that these results only show how fast the optimization problem 5.14 can be solved, while the question of how fast the generalization performance gets improved is still open.

5.2.3 Weak lower bounds

Although we are unable to construct a SLB for qp-bmrm on binary SVM, we manage to prove a WLB: $O(1/\epsilon)$. For ease of presentation, we postpone the description and first show a WLB example for L_2 regularized piecewise linear objectives.

Our example of L_2 regularized piecewise linear objective can be written in the framework of RRM as:

$$\Omega(\mathbf{w}) := \frac{1}{2}\left\|\mathbf{w}\right\|^2, \; R_{\text{emp}}(\mathbf{w}) := \max_{i\in[n]} w_i = \max_{i\in[n]} \left\langle \mathbf{e}_i, \mathbf{w} \right\rangle, \; J(\mathbf{w}) := \Omega(\mathbf{w}) + R_{\text{emp}}, \tag{5.18}$$

where $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{e}_i$ is the i-th coordinate vector (straight 0 except the i-th coordinate being 1). Our key result is the following:

Theorem 60 *Let* $\mathbf{w}_0 = \mathbf{0}$ *and* $\mathbf{w}^* := \operatorname{argmin}_{\mathbf{w}} J(\mathbf{w})$. *Suppose running* qp-bmrm *on the objective* $J(\mathbf{w})$ *in Eq.* (5.18) *gives* $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k, \ldots$. *Then for all* $k \in [n]$ *we have*

$$\min_{i\in[k]} J(\mathbf{w}_i) - J(\mathbf{w}^*) = \frac{1}{2k} + \frac{1}{2n}. \tag{5.19}$$

And $J(\mathbf{w}_k) = J(\mathbf{w}^*)$ *for all* $k > n$.

Proof We prove Theorem 60 by simply running qp-bmrm by hand. The R_{emp} in Eq. (5.18) consists of n hyperplanes. Since qp-bmrm cuts a tangent plane of R_{emp} at each iteration, we can assume without loss of generality that after k steps, the first k hyperplanes cut by qp-bmrm are: $\langle \mathbf{e}_1, \mathbf{w}\rangle, \ldots, \langle \mathbf{e}_k, \mathbf{w}\rangle$. Then we obtain a regularized lower approximation $J_k(\mathbf{w})$ to minimize:

$$J_k(\mathbf{w}) := \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \max_{i\in[k]} \left\langle \mathbf{e}_i, \mathbf{w}\right\rangle = \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \max_{i\in[k]} w_i.$$

It is not hard to see that the optimal solution of $J_k(\mathbf{w})$ is

$$\mathbf{w}_k = \operatorname*{argmin}_{\mathbf{w}} J_k(\mathbf{w}) = \Big(\overbrace{\frac{-1}{k}, \ldots, \frac{-1}{k}}^{k \text{ copies}}, 0, \ldots\Big)^\top,$$

because $\partial J_k(\mathbf{w}_k) = \left\{\mathbf{w}_k + \sum_{i\in[k]} \alpha_i \mathbf{e}_i : \boldsymbol{\alpha} \in \Delta_k\right\} \ni \mathbf{0}$. Similarly, we can derive that $\mathbf{w}^* = -\frac{1}{n}\mathbf{1}$. Plugging $\mathbf{w}_k$ and $\mathbf{w}^*$ into the definition of $J(\mathbf{w})$ we immediately derive

Eq. (5.20). ■

To complete our construction, for any arbitrary ϵ, set n to $\lfloor 1/\epsilon \rfloor$. Then for all $k \le n < 1/\epsilon$, $\min_{i\in[k]} J(\mathbf{w}_i) - J(\mathbf{w}^*) = \frac{1}{2k} + \frac{1}{2n} \ge \frac{1}{2n} + \frac{1}{2n} = \frac{1}{n} > \epsilon$. Since ls-bmrm cannot converge faster than qp-bmrm, this is also a WLB of $O(1/\epsilon)$ for ls-bmrm.

In fact, a careful look at the proof of Theorem 60 shows that the rate in Eq. (5.20) holds not only for qp-bmrm, but also for any optimizer which satisfies:

$$\mathbf{w}_{k+1} \in \operatorname{span}\left\{\nabla R_{\text{emp}}(\mathbf{w}_i) : i \in [k]\right\}, \qquad \text{where } \nabla R_{\text{emp}}(\mathbf{w}_i) \in \partial R_{\text{emp}}(\mathbf{w}_i).$$

Now coming back to SVM, the definition of hinge loss $[1 - y\langle \mathbf{x}, \mathbf{w}\rangle]_+$ indicates that we can treat any data point $(\mathbf{x}, y)$ $(y \in \{-1, 1\})$ as a new example $y\mathbf{x}$ with fixed label $+1$. So let the n "new" examples in $\mathbb{R}^{n+1}$ be $\mathbf{x}_1 = (\sqrt{n}, n, 0, 0, \ldots)$, $\mathbf{x}_2 = (\sqrt{n}, 0, n, 0, \ldots)$, $\mathbf{x}_3 = (\sqrt{n}, 0, 0, n, 0, \ldots)$, *i.e.*, $\mathbf{x}^i = n\mathbf{e}_{i+1} + \sqrt{n}\mathbf{e}_1$ (n will be set later). So the objective function is

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}[1 - \langle \mathbf{w}, \mathbf{x}^i\rangle]_+ = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}[1 - \sqrt{n}w_1 - nw_{i+1}]_+.$$

Our key result is the following.

Theorem 61 *Let $\mathbf{w}_0 = (n^{-1/2}, 0, 0, \ldots)^\top$. Suppose running* qp-bmrm *on the objective function Eq. (5.18) gives $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k, \ldots$. Then for all $k \in [n]$ we have*

$$\min_{i\in[k]} J(\mathbf{w}_i) - J(\mathbf{w}^*) = \frac{1}{2k} - \frac{1}{2n}. \tag{5.20}$$

And $J(\mathbf{w}_k) = J(\mathbf{w}^*)$ for all $k > n$.

Proof Again we run qp-bmrm by hand. Since $\partial R_{\text{emp}}(\mathbf{w}_0) = \left\{\frac{-1}{n}\sum_{i=1}^{n}\alpha_i\mathbf{x}^i : \boldsymbol{\alpha} \in \Delta_n\right\}$, we can choose

$$\mathbf{a}_1 = -n^{-1}\mathbf{x}_1 = (-n^{-1/2}, -1, 0, \ldots)^\top$$
$$b_1 = R_{\text{emp}}(\mathbf{w}_0) - \langle \mathbf{a}_1, \mathbf{w}_0\rangle = 0 - n^{-1} = -n^{-1}$$
$$\mathbf{w}_1 = \operatorname*{argmin}_{\mathbf{w}}\left\{\frac{1}{2}\|\mathbf{w}\|^2 - n^{-1/2}w_1 - w_2 - n^{-1}\right\} = (n^{-1/2}, 1, 0, \ldots)^\top.$$

Since $\partial R_{\text{emp}}(\mathbf{w}_1) = \left\{\frac{-1}{n}\sum_{i=2}^{n} \alpha_i \mathbf{x}^i : \boldsymbol{\alpha} \in \Delta_{n-1}\right\}$, we can choose

$$\begin{aligned}
\mathbf{a}_2 &= -n^{-1}\mathbf{x}_2 = (-n^{-1/2}, 0, -1, 0, \ldots)^\top \\
b_2 &= R_{\text{emp}}(\mathbf{w}_1) - \langle \mathbf{a}_2, \mathbf{w}_1 \rangle = 0 - n^{-1} = -n^{-1} \\
\mathbf{w}_2 &= \operatorname*{argmin}_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \max\left\{-n^{-1/2}w_1 - w_2 - n^{-1}, -n^{-1/2}w_1 - w_3 - n^{-1}\right\}\right\} \\
&= \left(\frac{1}{\sqrt{n}}, \frac{1}{2}, \frac{1}{2}, 0, \ldots\right)^\top.
\end{aligned}$$

Proceeding in the same way, we can show that

$$\mathbf{w}_k = \Big(\frac{1}{\sqrt{n}}, \overbrace{\frac{1}{k}, \ldots, \frac{1}{k}}^{k \text{ copies}}, 0, \ldots\Big)^\top,$$

And $\mathbf{w}^* = \mathbf{w}_n = \left(\frac{1}{\sqrt{n}}, \frac{1}{n}, \frac{1}{n}, \ldots\right)^\top$. Hence $J(\mathbf{w}_k) - J(\mathbf{w}^*) = \frac{1}{2k} - \frac{1}{2n}$. ■

To complete the construction of WLB example, set $n = \lceil 2/\epsilon \rceil$, then for all $k < \frac{2}{5\epsilon} < n$, we have

$$J(\mathbf{w}_k) - J(\mathbf{w}^*) > \frac{5\epsilon}{4} - \frac{\epsilon}{4} = \epsilon.$$

That is, qp-bmrm has WLB $O(1/\epsilon)$ on binary linear SVM problem. As ls-bmrm converges no faster than qp-bmrm, this is also a WLB example for ls-bmrm.

5.3 A new algorithm with convergence rates $O(1/\sqrt{\epsilon})$

We now turn our attention to the regularized risk minimization with the binary hinge loss, and propose a new algorithm. Our algorithm is based on (Nesterov, 1983) and (Nesterov, 2005a) which proposed a non-trivial scheme of minimizing an L-$l.c.g$ function to ϵ-precision in $O(1/\sqrt{\epsilon})$ iterations. Our contributions are two fold. First, we show that the dual of the regularized risk minimization problem is indeed a L-$l.c.g$ function. Second, we introduce an $O(n)$ time algorithm for projecting onto an n-dimensional simplex or in general an n-dimensional box with a single linear equality constraint, thus improving upon the $O(n \log n)$ deterministic algorithm of Duchi et al. (2008) (who also gave a randomized algorithm having expected complexity $O(n)$). This projection is repeatedly invoked as a subroutine by Nesterov's algorithm when specialized to our problem.

Consider the problem of minimizing a function $J(\mathbf{w})$ with the following structure over a closed convex set Q_1:

$$J(\mathbf{w}) = f(\mathbf{w}) + g^\star(A\mathbf{w}). \tag{5.21}$$

Here f is strongly convex on Q_1, A is a linear operator which maps Q_1 to another closed convex set Q_2, and g is convex and *l.c.g* on Q_2. Nesterov (2005a) works with the adjoint form of J:

$$D(\boldsymbol{\alpha}) = -g(\boldsymbol{\alpha}) - f^\star(-A^\top \boldsymbol{\alpha}), \tag{5.22}$$

which is *l.c.g* according to Theorem 55. Under some mild constraint qualifications which we omit for the sake of brevity (see *e.g.* Theorem 3.3.5 of (Borwein & Lewis, 2000)) we have

$$J(\mathbf{w}) \geq D(\boldsymbol{\alpha}) \quad \forall\, \mathbf{w}, \boldsymbol{\alpha} \qquad \text{and} \qquad \inf_{\mathbf{w}\in Q_1} J(\mathbf{w}) = \sup_{\boldsymbol{\alpha}\in Q_2} D(\boldsymbol{\alpha}). \tag{5.23}$$

By using the algorithm in (Nesterov, 1983) to maximize $D(\boldsymbol{\alpha})$, one can obtain an algorithm which converges to an ϵ accurate solution of $J(\mathbf{w})$ in $O(1/\sqrt{\epsilon})$ iterations.

The regularized risk minimization with the binary hinge loss can be identified with (5.21) by setting

$$J(\mathbf{w}) = \underbrace{\frac{\lambda}{2}\left\|\mathbf{w}\right\|^2}_{f(\mathbf{w})} + \underbrace{\min_{b\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i(\langle \mathbf{x}^i, \mathbf{w}\rangle + b)\right]_+}_{g^\star(A\mathbf{w})} \tag{5.24}$$

The latter, $g^\star$, is the dual of $g(\boldsymbol{\alpha}) = -\sum_i \alpha_i$ (see Appendix Example 3). Here $Q_1 = \mathbb{R}^d$. Let $A := -YX^\top$ where $Y := \mathrm{diag}(y^1, \ldots, y^n)$, $X := \left(\mathbf{x}^1, \ldots, \mathbf{x}^n\right)$. Then the adjoint can be written as :

$$D(\boldsymbol{\alpha}) := -g(\boldsymbol{\alpha}) - f^\star(-A^\top\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2\lambda}\boldsymbol{\alpha}^\top Y X^\top X Y \boldsymbol{\alpha} \qquad \text{with} \tag{5.25}$$

$$Q_2 = \left\{\boldsymbol{\alpha} \in [0, n^{-1}]^n : \sum_i y_i \alpha_i = 0\right\}. \tag{5.26}$$

In fact, this is the well known SVM dual objective function with the bias incorporated.

Now we present the algorithm of (Nesterov, 2005a) in Algorithm 10. Since it optimizes the primal $J(\mathbf{w})$ and the adjoint $D(\boldsymbol{\alpha})$ simultaneously, we call it pragam (PRimal-Adjoint GAp Minimization). It requires a σ_2-strongly convex prox-function on Q_2: $d_2(\boldsymbol{\alpha}) = \frac{\sigma_2}{2}\left\|\boldsymbol{\alpha}\right\|^2$, and sets $D_2 = \max_{\boldsymbol{\alpha}\in Q_2} d_2(\boldsymbol{\alpha})$. Let the Lipschitz constant of $\nabla D(\boldsymbol{\alpha})$ be L. Algorithm 10 is based on two mappings $\boldsymbol{\alpha}_\mu(\mathbf{w}) : Q_1 \mapsto Q_2$ and $\mathbf{w}(\boldsymbol{\alpha}) : Q_2 \mapsto Q_1$, together with an auxiliary mapping $v : Q_2 \mapsto Q_2$. They are defined by

Algorithm 10: pragam: an $O(1/k^2)$ rate primal-adjoint solver (Nesterov, 2005a).

```
Input: Objective function $f$ which has a composite form as Eq. (5.21). $L$ as a
       conservative estimate (i.e. upper bound) of the Lipschitz constant of
       $\nabla D(\boldsymbol{\alpha})$.
Output: Two sequences $\mathbf{w}_k$ and $\boldsymbol{\alpha}_k$ which reduce the duality gap
        $J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k)$ at $O(1/k^2)$ rate.
1 Initialize: Randomly pick $\boldsymbol{\alpha}_{-1}$ in $Q_2$. Let $\mu_0 = 2L$, $\boldsymbol{\alpha}_0 \leftarrow v(\boldsymbol{\alpha}_{-1})$,
  $\mathbf{w}_0 \leftarrow \mathbf{w}(\boldsymbol{\alpha}_{-1})$.
2 for $k = 0, 1, 2, \ldots$ do
3     $\tau_k = \frac{2}{k+3}$, $\boldsymbol{\beta}_k \leftarrow (1-\tau_k)\boldsymbol{\alpha}_k + \tau_k \boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$.
4     Set $\mathbf{w}_{k+1} \leftarrow (1-\tau_k)\mathbf{w}_k + \tau_k \mathbf{w}(\boldsymbol{\beta}_k)$, $\boldsymbol{\alpha}_{k+1} \leftarrow v(\boldsymbol{\beta}_k)$, $\mu_{k+1} \leftarrow (1-\tau_k)\mu_k$.
```

$$\boldsymbol{\alpha}_\mu(\mathbf{w}) := \operatorname*{argmin}_{\boldsymbol{\alpha}\in Q_2} \mu d_2(\boldsymbol{\alpha}) - \langle A\mathbf{w}, \boldsymbol{\alpha}\rangle + g(\boldsymbol{\alpha}) \tag{5.27}$$

$$= \operatorname*{argmin}_{\boldsymbol{\alpha}\in Q_2} \frac{\mu}{2}\|\boldsymbol{\alpha}\|^2 + \mathbf{w}^\top XY\boldsymbol{\alpha} - \sum_i \alpha_i, \tag{5.28}$$

$$\mathbf{w}(\boldsymbol{\alpha}) := \operatorname*{argmin}_{\mathbf{w}\in Q_1} \langle A\mathbf{w}, \boldsymbol{\alpha}\rangle + f(\mathbf{w}) \tag{5.29}$$

$$= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} -\mathbf{w}^\top XY\boldsymbol{\alpha} + \frac{\lambda}{2}\|\mathbf{w}\|^2 = \frac{1}{\lambda}XY\boldsymbol{\alpha}, \tag{5.30}$$

$$v(\boldsymbol{\alpha}) := \operatorname*{argmin}_{\boldsymbol{\alpha}'\in Q_2} \frac{L}{2}\left\|\boldsymbol{\alpha}' - \boldsymbol{\alpha}\right\|^2 - \left\langle \nabla D(\boldsymbol{\alpha}), \boldsymbol{\alpha}' - \boldsymbol{\alpha}\right\rangle. \tag{5.31}$$

Eq. (5.30) is exactly the dual relationship in binary SVM. Eq. (5.28) and (5.31) are examples of a box constrained QP with a single equality constraint. In Section 5.3.2, we provide a linear time algorithm to find the minimizer of such a QP. The overall complexity of each iteration is thus $O(nd)$ due to the gradient calculation in (5.31) and the matrix multiplication in (5.30).

5.3.1 Convergence rates

According to (Nesterov, 2005a), on running Algorithm pragam for k iterations, the $\boldsymbol{\alpha}_k$ and $\mathbf{w}_k$ satisfy:

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq \frac{4LD_2}{(k+1)(k+2)\sigma_2}. \tag{5.32}$$

For SVMs, $L = \frac{1}{\lambda}\|A\|_{1,2}^2$ where $\|A\|_{1,2} = \max\{\langle A\mathbf{w}, \boldsymbol{\alpha}\rangle : \|\boldsymbol{\alpha}\| = 1, \|\mathbf{w}\| = 1\}$, $\sigma_2 = 1$, $D_2 = \frac{1}{2n}$. Assuming $\left\|\mathbf{x}^i\right\| \leq R$,

$$|\langle A\mathbf{w}, \boldsymbol{\alpha}\rangle|^2 \leq \|\boldsymbol{\alpha}\|^2 \left\|YX^\top\mathbf{w}\right\|^2 = \left\|X^\top\mathbf{w}\right\|^2 = \sum_i \left\langle \mathbf{x}^i, \mathbf{w}\right\rangle^2 \leq \sum_i \|\mathbf{w}\|^2 \left\|\mathbf{x}^i\right\|^2 \leq nR^2.$$

Thus by (5.32), we conclude

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq \frac{4LD_2}{(k+1)(k+2)\sigma_2} \leq \frac{2R^2}{\lambda(k+1)(k+2)} < \epsilon,$$

which gives

$$k > O\left(\frac{R}{\sqrt{\lambda\epsilon}}\right).$$

This $O(\sqrt{1/\epsilon})$ rate improves upon the $O(1/\epsilon)$ rate in state-of-the-art SVM solvers like pegasos (Shalev-Shwartz et al., 2007), `SVMPerf` (Joachims, 2006), `SVM`$^{\texttt{Struct}}$ (Tsochantaridis et al., 2005), and BMRM (Teo et al., 2010). It should also be noted that our algorithm has a better dependence on λ compared to these methods which have a factor of $\frac{1}{\lambda}$ in their convergence rates. Our rate of convergence is also data dependent, showing how the correlation of the dataset $XY = (y^1\mathbf{x}^1, \ldots, y^n\mathbf{x}^n)$ affects the rate via the Lipschitz constant L, which is equal to the square of the maximum singular value of XY (or the maximum eigenvalue of $YX^\top XY$). On one extreme, if $\mathbf{x}^i$ is the i-th dimensional unit vector then $L = 1$, while $L = n$ if all $y^i\mathbf{x}^i$ are identical.

5.3.2 A linear time algorithm for simple QP

It is easy to see that the dual optimization problem $D(\boldsymbol{\alpha})$ from (5.25) is a box constrained QP with a single linear equality constraint.

In this section, we focus on solving the following simple QP:

$$\begin{aligned}
\min \quad & \frac{1}{2}\sum_{i=1}^{n} d_i^2(\alpha_i - m_i)^2 \\
s.t. \quad & l_i \leq \alpha_i \leq u_i \qquad \forall i \in [n]; \\
& \sum_{i=1}^{n} \sigma_i\alpha_i = z.
\end{aligned} \tag{5.33}$$

Without loss of generality, we assume $l_i < u_i$ and $d_i \neq 0$ for all i. Also assume $\sigma_i \neq 0$ because otherwise α_i can be solved independently. To ensure the feasible region is nonempty, we further assume

$$\sum_i \sigma_i(\delta(\sigma_i > 0)l_i + \delta(\sigma_i < 0)u_i) \leq z \leq \sum_i \sigma_i(\delta(\sigma_i > 0)u_i + \delta(\sigma_i < 0)l_i).$$

The algorithm we describe below stems from (Pardalos & Kovoor, 1990) and finds the exact optimal solution in $O(n)$ time, faster than the $O(n\log n)$ complexity in (Duchi et al., 2008).

With a simple change of variable $\beta_i = \sigma_i(\alpha_i - m_i)$, the problem is simplified as

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{n} \bar{d}_i^2 \beta_i^2 \\ s.t. \quad & l_i' \leq \beta_i \leq u_i' \quad \forall i \in [n]; \\ & \sum_{i=1}^{n} \beta_i = z', \end{aligned} \qquad \text{where} \qquad \begin{aligned} l_i' &= \begin{cases} \sigma_i(l_i - m_i) & \text{if } \sigma_i > 0 \\ \sigma_i(u_i - m_i) & \text{if } \sigma_i < 0 \end{cases}, \\ u_i' &= \begin{cases} \sigma_i(u_i - m_i) & \text{if } \sigma_i > 0 \\ \sigma_i(l_i - m_i) & \text{if } \sigma_i < 0 \end{cases}, \\ \bar{d}_i^2 &= \frac{d_i^2}{\sigma_i^2}, \quad z' = z - \sum_i \sigma_i m_i. \end{aligned}$$

We derive its dual via the standard Lagrangian.

$$L = \frac{1}{2}\sum_i \bar{d}_i^2 \beta_i^2 - \sum_i \rho_i^+ (\beta_i - l_i') + \sum_i \rho_i^- (\beta_i - u_i') - \lambda \left(\sum_i \beta_i - z' \right).$$

Taking derivative:

$$\frac{\partial L}{\partial \beta_i} = \bar{d}_i^2 \beta_i - \rho_i^+ + \rho_i^- - \lambda = 0 \quad \Rightarrow \quad \beta_i = \bar{d}_i^{-2}(\rho_i^+ - \rho_i^- + \lambda). \tag{5.34}$$

Substituting into L, we get the dual optimization problem

$$\begin{aligned} \min D(\lambda, \rho_i^+, \rho_i^-) &= \frac{1}{2}\sum_i \bar{d}_i^{-2}(\rho_i^+ - \rho_i^- + \lambda)^2 - \sum_i \rho_i^+ l_i' + \sum_i \rho_i^+ u_i' - \lambda z' \\ s.t. \quad & \rho_i^+ \geq 0, \quad \rho_i^- \geq 0 \quad \forall i \in [n]. \end{aligned}$$

Taking derivative of D with respect to λ, we get:

$$\sum_i \bar{d}_i^{-2}(\rho_i^+ - \rho_i^- + \lambda) - z' = 0. \tag{5.35}$$

The KKT condition gives:

$$\rho_i^+ (\beta_i - l_i') = 0, \tag{5.36a}$$

$$\rho_i^- (\beta_i - u_i') = 0. \tag{5.36b}$$

Now we enumerate four cases.

1. $\rho_i^+ > 0$**,** $\rho_i^- > 0$**.** This implies that $l_i' = \beta_i = u_i'$, which is contradictory to our assumption.

2. $\rho_i^+ = 0$**,** $\rho_i^- = 0$**.** Then by (5.34), $\beta_i = \bar{d}_i^{-2}\lambda \in [l_i', u_i']$, hence $\lambda \in [\bar{d}_i^2 l_i', \bar{d}_i^2 u_i']$.

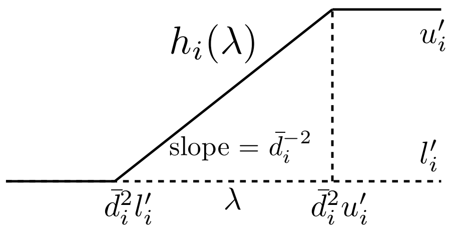


Figure 5.1: $h_i(\lambda)$

Algorithm 11: $O(n)$ algorithm to find the root of $f(\lambda)$. Ignoring boundary condition checks.

```
1 Set kink set $S \leftarrow \{\bar{d}_i^2 l_i' : i \in [n]\} \cup \{\bar{d}_i^2 u_i' : i \in [n]\}$.
2 while $|S| > 2$ do
3     Find median of $S$: $m \leftarrow \mathrm{MED}(S)$.
4     if $f(m) \geq 0$ then
5         $S \leftarrow \{x \in S : x \leq m\}$.
6     else
7         $S \leftarrow \{x \in S : x \geq m\}$.
8 return Root $\frac{lf(u)-uf(l)}{f(u)-f(l)}$ where $S = \{l, u\}$.
```

3. $\rho_i^+ > 0$**,** $\rho_i^- = 0$**.** Now by (5.36) and (5.34), we have $l_i' = \beta_i = \bar{d}_i^{-2}(\rho_i^+ + \lambda) > \bar{d}_i^{-2}\lambda$, hence $\lambda < \bar{d}_i^2 l_i'$ and $\rho_i^+ = \bar{d}_i^2 l_i' - \lambda$.

4. $\rho_i^+ = 0$**,** $\rho_i^- > 0$**.** Now by (5.36) and (5.34), we have $u_i' = \beta_i = \bar{d}_i^{-2}(-\rho_i^- + \lambda) < \bar{d}_i^{-2}\lambda$, hence $\lambda > \bar{d}_i^2 u_i'$ and $\rho_i^- = -\bar{d}_i^2 u_i' + \lambda$.

In sum, we have $\rho_i^+ = [\bar{d}_i^2 l_i' - \lambda]_+$ and $\rho_i^- = [\lambda - \bar{d}_i^2 u_i']_+$. Now (5.35) turns into

$$f(\lambda) := \sum_i \underbrace{\bar{d}_i^{-2}([\bar{d}_i^2 l_i' - \lambda]_+ - [\lambda - \bar{d}_i^2 u_i']_+ + \lambda)}_{=:h_i(\lambda)} - z' = 0. \tag{5.37}$$

In other words, we only need to find the root of $f(\lambda)$ in (5.37). $h_i(\lambda)$ is plotted in Figure 5.1. Note that $h_i(\lambda)$ is a monotonically increasing function of λ, so the whole $f(\lambda)$ is monotonically increasing in λ. Since $f(\infty) \geq 0$ by $z' \leq \sum_i u_i'$ and $f(-\infty) \leq 0$ by $z' \geq \sum_i l_i'$, the root must exist. Considering that f has at most $2n$ kinks (nonsmooth points) and is linear between two adjacent kinks, the simplest idea is to sort $\{\bar{d}_i^2 l_i', \bar{d}_i^2 u_i' : i \in [n]\}$ into $s^{(1)} \leq \ldots \leq s^{(2n)}$. If $f(s^{(i)})$ and $f(s^{(i+1)})$ have different signs, then the root must lie between them and can be easily found because f is linear in $[s^{(i)}, s^{(i+1)}]$. This algorithm takes at least $O(n \log n)$ time because of sorting.

However, this complexity can be reduced to $O(n)$ by making use of the fact that the median of n (unsorted) elements can be found in $O(n)$ time. Notice that due to the

Algorithm 12: $O(1/k^2)$ rate optimization for *l.c.g* functions (Nesterov, 1983).

```
Input: A l.c.g function $f$, a conservative estimate (upper bound) of the
       Lipschitz constant of its gradient, an oracle which gives the gradient of
       $f$ at any query point $\mathbf{x}$, a proxy-function $d(\mathbf{x})$ which is $\sigma$-strongly
       convex on $Q$ wrt a norm $\|\cdot\|$.
Output: A sequence $\{\mathbf{y}^k\}$ which converges to the optimal solution at $O(1/k^2)$
        rate.
1 Initialize: Set $\mathbf{x}^0$ to a random value in $Q$.
2 for $k = 0, 1, 2, \ldots$ do
3     Query the gradient of $f$ at point $\mathbf{x}^k$: $\nabla f(\mathbf{x}^k)$.
4     Find $\mathbf{y}^k \leftarrow \mathrm{argmin}_{\mathbf{x} \in Q} \left\langle \nabla f(\mathbf{x}^k), x - \mathbf{x}^k \right\rangle + \frac{1}{2} L \left\| \mathbf{x} - \mathbf{x}^k \right\|^2$ .
5     Find $\mathbf{z}^k \leftarrow \mathrm{argmin}_{\mathbf{x} \in Q} \frac{L}{\sigma} d(\mathbf{x}) + \sum_{i=0}^{k} \frac{i+1}{2} \left\langle \nabla f(\mathbf{x}^i), \mathbf{x} - \mathbf{x}^i \right\rangle$.
6     Update $\mathbf{x}^{k+1} \leftarrow \frac{2}{k+3} \mathbf{z}^k + \frac{k+1}{k+3} \mathbf{y}^k$.
```

monotonicity of f, the median of a set S gives exactly the median of function values, *i.e.*, $f(\mathrm{MED}(S)) = \mathrm{MED}(\{f(x) : x \in S\})$. Algorithm 11 sketches the idea of binary search. The while loop terminates in $\log_2(2n)$ iterations because the set S is halved in each iteration. And in each iteration, the time complexity is linear to $|S|$, the size of current S. So the total complexity is $O(n)$. Note the evaluation of $f(m)$ potentially involves summing up n terms as in (5.37). However by some clever aggregation of slope and offset, this can be reduced to $O(|S|)$.

5.3.3 Other versions of Neseterov's algorithms

The Algorithm 10 is one of the three major algorithms proposed by Nesterov which offer a $O\left(\sqrt{1/\epsilon}\right)$ rate of convergence. We detail it because a) it captures all the important techniques in this series of work, b) it is primal-dual, which bounds the duality gap instead of merely the gap for primal or dual objective, and c) it is not too complicated to describe. Below we sketch the other two important variants of Nesterov's algorithm:

1. (Nesterov, 1983) outlined in Algorithm 12. This was the first algorithm that gives the $O\left(\sqrt{1/\epsilon}\right)$ rate of convergence in our setting. It works only on *l.c.g* functions constrained to a convex set which allows efficient projection. It works purely in the primal. Similar to (Nesterov, 2005a), it also needs the explicit knowledge of the Lipschitz constant of the gradient, which is often expensive in practice.

2. (Nesterov, 2007) outlined in Algorithm 13. The main contribution of this work is to automatically estimate the Lipschitz constant of the gradient via geometric scaling. Its objective function is assumed to be the same as the primal of (Nesterov, 2005a) in Eq. (5.21), *i.e.* composite. This algorithm works only in the primal, hence not bounding the duality gap.

Algorithm 13: $O(1/k^2)$ solver for composite functions as Eq. (5.21), with built-in Lipschitz constant estimation (Nesterov, 2007).

```
Input: $L_0 \in (0, L]$ as an optimistic estimate (lower bound) of $L$, two scaling
       parameters $\gamma_u > 1$ and $\gamma_d \geq 1$.
Output: A sequence $\mathbf{x}^k$ which converges to the optimal solution at $O(1/k^2)$
        rate.
1  Initialize: Set $A_k = 0$, $\tilde{L} := L_0/\gamma_u$, set $\mathbf{x}^0$ randomly and $\phi_0(\mathbf{x}) := \frac{1}{2}\left\|\mathbf{x} - \mathbf{x}^0\right\|^2$.
2  for $k = 0, 1, 2, \ldots$ do
3      repeat
4          $\tilde{L} \leftarrow \gamma_u \tilde{L}$.
5          Find the positive root of $\frac{a^2}{A_k + a} = 2\frac{1 + \mu A_k}{\tilde{L}}$.
6          Set $\mathbf{y} = \frac{A_k \mathbf{x}^k + a \mathbf{v}_k}{A_k + a}$, where $\mathbf{v}_k := \mathrm{argmin}_{\mathbf{x}}\, \phi_k(\mathbf{x})$.
7          Set $T_{\tilde{L}}(\mathbf{y}) := \mathrm{argmin}_{\mathbf{x}} \frac{\tilde{L}}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle + \Psi(\mathbf{x}) + f(\mathbf{y})$.
8      until $\langle \phi'(T_{\tilde{L}}(\mathbf{y})), \mathbf{y} - T_{\tilde{L}}(\mathbf{y})\rangle \geq \tilde{L}^{-1}\left\|\phi'(T_{\tilde{L}}(\mathbf{y}))\right\|^2$, where
       $\phi'(\mathbf{x}) \in \nabla \hat{f}(\mathbf{x}) + \partial\Psi(\mathbf{x})$.
9      Set $A_{k+1} := A_k + a$, $\mathbf{x}^{k+1} := T_{\tilde{L}}(\mathbf{y})$ and finally $\tilde{L} \leftarrow \tilde{L}/(\gamma_d \gamma_u)$.
10     Set $\phi_{k+1}(\mathbf{x}) := \phi_k(\mathbf{x}) + a\left[\hat{f}(\mathbf{x}^{k+1}) + \left\langle \nabla \hat{f}(\mathbf{x}^{k+1}), \mathbf{x} - \mathbf{x}^{k+1}\right\rangle + \Psi(\mathbf{x})\right]$.
```

5.4 Structured output space

It is noteworthy that applying pragam to structured data is straightforward, and this section sketches the basic ideas. As we will see, a key interesting problem here is how to project onto a probability simplex such that the image decomposes according to a graphical model.

Recall the margin rescaled hinge loss for multi-class classification in Section 1.6.1:

$$l(\mathbf{x}^i, \mathbf{y}^i; \mathbf{w}) = \max_{\mathbf{y} \in \mathcal{Y}} \left\{ \Delta(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) - \left\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) \right\rangle \right\},$$

where we moved the sample index to superscript. For structured output space $\mathcal{Y}$, optimization becomes intractable as there are exponentially many candidates in the max operation. Therefore, the graphical model structure must be exploited to factorize the features $\boldsymbol{\phi}$ and discrepancy $\Delta(\mathbf{y}, \mathbf{y}^i)$, which leads to parameter estimation based on cliques. We illustrate this idea using the maximum margin Markov network (Taskar et al., 2004), and show how pragam can be applied. For ease of exposition, the output space of all training examples is assumed to have the same graphical model structure with maximal clique set $\mathcal{C}$, and this restriction can be easily relaxed.

5.4.1 Margin scaled maximum margin Markov network

The maximum margin Markov network (M^{3}N) by Taskar et al. (2004) uses square norm regularizer and margin rescaled hinge loss:

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_i \max_{\mathbf{y}\in\mathcal{Y}} \left\{ \Delta(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) - \left\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) \right\rangle \right\}, \tag{5.38}$$

where both $\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y})$ and $\Delta(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ are assumed to decompose onto the cliques:

$$\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) = \bigoplus_{c\in\mathcal{C}} \boldsymbol{\phi}_c(x_c^i, y_c), \qquad \Delta(\mathbf{y}^i, \mathbf{y}; \mathbf{x}^i) = \sum_{c\in\mathcal{C}} l_c(y_c^i, y_c; \mathbf{x}^i), \tag{5.39}$$

where $\oplus$ means Cartesian product.

Viewing the primal objective J in Eq. (5.38) as a composite function in the same way as in Eq. (5.24), we can derive the adjoint form:

$$D(\boldsymbol{\alpha}) = \frac{1}{2\lambda} \sum_{i,j} \sum_{\mathbf{y},\mathbf{y}'} A_{(i,\mathbf{y}),(j,\mathbf{y}')} \alpha^i(\mathbf{y}) \alpha^j(\mathbf{y}') - \sum_i \sum_{\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) \alpha^i(\mathbf{y}), \tag{5.40}$$

where $A_{(i,\mathbf{y}),(j,\mathbf{y}')} := \left\langle \boldsymbol{\psi}^i(\mathbf{y}), \boldsymbol{\psi}^j(\mathbf{y}') \right\rangle$ denoting $\boldsymbol{\psi}^i(\mathbf{y}) := \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y})$. The constraints are that $\boldsymbol{\alpha}^i$ be in the simplex:

$$\alpha^i(\mathbf{y}) \geq 0 \quad \forall\, i, \mathbf{y}, \qquad \text{and} \qquad \sum_{\mathbf{y}} \alpha^i(\mathbf{y}) = 1 \quad \forall\, i. \tag{5.41}$$

The dual connection is:

$$\mathbf{w} = \sum_i \sum_{\mathbf{y}} \alpha^i(\mathbf{y}) \boldsymbol{\psi}^i(\mathbf{y}). \tag{5.42}$$

Now incorporating the decomposition in Eq. (5.39), we can derive the factorized adjoint form:

$$D(\boldsymbol{\alpha}) = \frac{1}{2\lambda} \sum_{i,j} \sum_{c,c'} \sum_{y_c, y'_{c'}} A_{(i,c,y_c),(j,c',y'_{c'})} \alpha_c^i(y_c) \alpha_{c'}^j(y'_{c'}) - \sum_{i,c,y_c} l_c(y_c, y_c^i; \mathbf{x}^i) \alpha_c^i(y_c), \tag{5.43}$$

where $\alpha_c^i(y_c) := \sum_{\mathbf{y}\sim y_c} \alpha_c^i(\mathbf{y})$. Here $\mathbf{y} \sim y_c$ means ranging over all possible assignments of $\mathbf{y}$ which match y_c on the clique c. Then the constraints in Eq. (5.41) become

$$\alpha_c^i(y_c) \geq 0 \quad \forall\, i, c, y_c, \qquad \text{and} \qquad \sum_{y_c} \alpha_c^i(y_c) = 1 \quad \forall\, i. \tag{5.44}$$

In addition, a new set of constraints need to be introduced to enforce consistency:

$$\sum_{y_c \sim y_{c\cap c'}} \alpha_{i,c}(y_c) = \sum_{y_{c'} \sim y_{c\cap c'}} \alpha_{i,c'}(y_{c'}) \qquad \forall i, \forall c, c' \in \mathcal{C} : c \cap c' \neq \emptyset, \forall y_{c\cap c'}, \tag{5.45}$$

which ensures that the marginal distribution of $y_{c\cap c'}$ computed from the marginal distribution of clique c is consistent with that computed from clique c'. Notice that the simplex conditions Eq. (5.44) and the local consistency conditions Eq. (5.45) are just necessary but not sufficient conditions of global consistency Eq. (5.41). When the graph is tree structured, they are equivalent.

Finally, the dual connection becomes

$$\mathbf{w} = \bigoplus_{c\in\mathcal{C}} \sum_i \sum_{y_c} \alpha_c^i(y_c) \boldsymbol{\psi}_c^i(y_c). \tag{5.46}$$

Now it turns out straightforward to apply Algorithm 12 (Nesterov, 1983) and Algorithm 13 (Nesterov, 2007) to optimize the dual objective Eq. (5.43) subject to the constraints Eq. (5.44) and (5.45). Both algorithms only require the following form of projection as the inner solver:

$$\begin{aligned}
&\min && \frac{1}{2}\sum_c d_c^2 \left\|\boldsymbol{\alpha}_c - \mathbf{m}_c\right\|_2^2 && \qquad (5.47)\\
&s.t. && \boldsymbol{\alpha}_c \in \Delta_{|V_c|} && \forall\, c \in \mathcal{C}\\
& && \sum_{y_c \sim y_{c\cap c'}} \alpha_c(y_c) = \sum_{y_{c'} \sim y_{c\cap c'}} \alpha_{c'}(y_{c'}) && \forall\, c \cap c' \neq \emptyset, \forall\, y_{c\cap c'}.
\end{aligned}$$

where V_c is the range of assignments that y_c can assume, and $\mathbf{m}_c$ is an arbitrary vector in $\mathbb{R}^{|V_c|}$ that is not necessarily a distribution. This problem bears significant resemblance to the inner solver for binary SVM in Eq. (5.33). The key difference is that we now have to enforce additional local consistency constraints originating from the graphical models. Intuitively speaking, we are again projecting to a probability simplex, but subject to the conditional independence relations encoded in a graphical model. More details on how to solve this constrained projection will be given in Section 5.4.2.

Application of the primal-dual Algorithm 10 (Nesterov, 2005a) is still hard, because the factorized problem Eq. (5.43) is not exactly the adjoint form of the primal problem Eq. (5.38). Technically, the key obstacle is that the projection in Eq. (5.31) measures the L_2 distance of the *joint distribution*:

$$\left\|\boldsymbol{\alpha} - \mathbf{m}\right\|_2^2, \tag{5.48}$$

where $\boldsymbol{\alpha}$ is the joint distribution, and this square distance can *not* be decomposed onto the cliques as in Eq. (5.47). This disallows us to apply the trick in (Collins et al., 2008): conceptually optimize wrt joint distributions $\{\alpha^i(\mathbf{y})\}_{i,\mathbf{y}}$ via practically updating the marginals on the cliques $\{\alpha_c^i(y_c)\}_{i,c,y_c}$.

Most machine learning models for structured output data perform parameter estimation by graphical model decomposition, and it is not hard to apply the same idea here to those models, *e.g.* Gaussian process for sequence labeling (Altun et al., 2004a).

5.4.2 Efficient projection onto factorized simplex

We consider in this section how to solve the constrained projection (5.47), which extends the simple projection in Section 5.3.2. In addition to projecting onto the n dimensional simplex wrt L_2 distance, we also restrict the image to be factorized by a graphical model. Formally, given a set of marginal parameters on the cliques $\{\mathbf{m}_c \in \mathbb{R}^{|V_c|} : c \in \mathcal{C}\}$ where $\mathbf{m}_c$ may not be a distribution, we want to find a set of marginal distributions $\{\boldsymbol{\alpha}_c \in \Delta_{|V_c|} : c \in \mathcal{C}\}$ which minimize:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\sum_c d_c^2 \left\|\boldsymbol{\alpha}_c - \mathbf{m}_c\right\|_2^2 \\
s.t. \quad & \boldsymbol{\alpha}_c \in \Delta_{|V_c|} & \forall\, c \in \mathcal{C} \\
& \sum_{y_c \sim y_{c\cap c'}} \alpha_c(y_c) = \sum_{y_{c'} \sim y_{c\cap c'}} \alpha_{c'}(y_{c'}) & \forall c \cap c' \neq \emptyset, \forall\, y_{c\cap c'}.
\end{aligned}
$$

We proceed by writing out the standard Lagrangian:

$$
\begin{aligned}
\mathcal{L} =& \frac{1}{2}\sum_c d_c^2 \sum_{y_c}(\alpha_c(y_c) - m_c(y_c))^2 - \sum_c \lambda_c\left(\sum_{y_c}\alpha_c(y_c) - 1\right) - \sum_{c,y_c}\xi_c(y_c)\alpha_c(y_c) \\
& - \sum_{c,c' : c\cap c' \neq \emptyset}\sum_{y_{c\cap c'}} \bar{\mu}_{c,c'}(y_{c\cap c'})\left(\sum_{y_c : y_c \sim y_{c\cap c'}} \alpha_c(y_c) - \sum_{y_{c'} \sim y_{c\cap c'}} \alpha_{c'}(y_{c'})\right).
\end{aligned}
$$

Taking derivative over $\alpha_c(y_c)$:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \alpha_c(y_c)} &= d_c^2(\alpha_c(y_c) - m_c(y_c)) - \lambda_c - \xi_c(y_c) - \sum_{c'}\bar{\mu}_{c,c'}(y_{c\cap c'}) + \sum_{c'}\bar{\mu}_{c',c}(y_{c\cap c'}) = 0, \\
&\Rightarrow \alpha_c(y_c) = m_c(y_c) + d_c^{-2}\left(\lambda_c + \xi_c(y_c) + \sum_{c'}\mu_{c,c'}(y_{c\cap c'})\right), \qquad (5.49)
\end{aligned}
$$

where $\mu_{c,c'}(y_{c\cap c'}) := \bar{\mu}_{c,c'}(y_{c\cap c'}) - \bar{\mu}_{c',c}(y_{c\cap c'})$. Plugging it back into $\mathcal{L}$, we derive the dual problem:

$$\begin{aligned}
\min D(\lambda_c, \xi_c(y_c), \mu_{c,c'}(y_{c\cap c'})) =& \frac{1}{2}\sum_c d_c^{-2} \sum_{y_c} \left(\lambda_c + \xi_c(y_c) + \sum_{c'} \mu_{c,c'}(y_{c\cap c'})\right)^2 \\
&+ \sum_c \sum_{y_c} m_c(y_c) \left(\lambda_c + \xi_c(y_c) + \sum_{c'} \mu_{c,c'}(y_{c\cap c'})\right) \\
&- \sum_c \lambda_c \\
s.t. \qquad & \xi_c(y_c) \geq 0.
\end{aligned} \tag{5.50}$$

This problem is essentially a QP over $\lambda_c, \xi_c(y_c), \mu_{c,c'}(y_{c\cap c'})$ with the only constraint that $\xi_c(y_c) \geq 0$. Similar to Section 5.3.2, one can write $\xi_c(y_c)$ as a hinge function of λ_c and $\mu_{c,c'}(y_{c\cap c'})$. However since it is no longer a single variable function, it is very hard to apply the median trick here. So we resort to a simple block coordinate descent as detailed in Algorithm 14 with reference to the following expressions of gradient:

$$\frac{\partial D}{\partial \xi_c(y_c)} = -d_c^{-2}(\lambda_c + \xi_c(y_c) + \sum_{c'} \mu_{c,c'}(y_{c\cap c'})) + m_c(y_c) = 0 \tag{5.51a}$$

$$\frac{\partial D}{\partial \lambda_c} = d_c^{-2} \sum_{y_c} \left(\lambda_c + \xi_c(y_c) + \sum_{c'} \mu_{c,c'}(y_{c\cap c'})\right) + \sum_{y_c} m_c(y_c) - 1 \tag{5.51b}$$

$$\frac{\partial D}{\partial \mu_{c,c'}(y_{c\cap c'})} = d_c^{-2} \sum_{y'_c \sim y_{c\cap c'}} \left(\lambda_c + \xi_c(y'_c) + \sum_{\bar{c}} \mu_{c,\bar{c}}(y'_{c,\bar{c}})\right) + \sum_{y'_c \sim y_{c\cap c'}} m_c(y'_c). \tag{5.51c}$$

From (5.51a) and $\xi_c(y_c) \geq 0$, we can derive

$$\xi_c(y_c) = \left[-d_c^2 m_c(y_c) - \lambda_c - \sum_{c'} \mu_{c,c'}(y_{c\cap c'})\right]_+ . \tag{5.52}$$

Example: sequence

Suppose the graph is simply a sequence: $y_1 - y_2 - \ldots - y_L$ and each node can take value in $[m]$. Then the cliques are $\{(y_t, y_{t+1}) : t \in [L-1]\}$ and the primal is:

$$\begin{aligned}
\min \quad & \frac{1}{2}\sum_{t=1}^{L-1} d_t^2 \sum_{i,j=1}^{m} (\alpha_t(i,j) - m_t(i,j))^2 \\
s.t. \quad & \boldsymbol{\alpha}_t \in \Delta_{m^2} && \forall t \in [L-1] \\
& \sum_i \alpha_t(i,j) = \sum_k \alpha_{t+1}(j,k) && \forall t \in [L-2], j \in [m].
\end{aligned}$$

Algorithm 14: A coordinate descent scheme for minimizing the dual problem (5.50).

```
1 Initialize:  Randomly set $\{\lambda_c : c\}, \{\xi_c(y_c) : c, y_c\}, \{\mu_{c,c'}(y_{c\cap c'}) : c, c', y_{c\cap c'}\}$.
2 while not converged do
3     Fixing $\xi_c(y_c)$, apply conjugate gradient to minimize the unconstrained
      quadratic form in (5.50) with respect to $\{\lambda_c : c\}$ and
      $\{\mu_{c,c'}(y_{c\cap c'}) : c, c', y_{c\cap c'}\}$. The necessary gradients are given in (5.51b) and
      (5.51c).
4     Set $\xi_c(y_c) \leftarrow \left[-d_c^2 m_c(y_c) - \lambda_c - \sum_{c'} \mu_{c,c'}(y_{c\cap c'})\right]_+$ for all $c \in \mathcal{C}$ and $y_c$.
5 Compute $\alpha_c(y_c)$ according to Eq. (5.49).
6 return $\alpha_c(y_c)_{c,y_c}$.
```

Proceeding with the standard Lagrangian:

$$\begin{aligned}\mathcal{L} \;=\; & \sum_{t=1}^{L-1} d_t^2 \sum_{i,j=1}^{m} (\alpha_t(i,j) - m_t(i,j))^2 - \sum_{t=1}^{L-1} \lambda_t \left(\sum_{i,j} \alpha_t(i,j) - 1 \right) \\ & - \sum_{t=1}^{L-1} \sum_{i,j} \xi_t(i,j)\alpha_t(i,j) - \sum_{t=1}^{L-2} \sum_{j} \mu_t(j) \left(\sum_i \alpha_t(i,j) - \sum_k \alpha_{t+1}(j,k) \right).\end{aligned}$$

Taking derivative over $\alpha_t(i,j)$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha_t(i,j)} &= d_t^2(\alpha_t(i,j) - m_t(i,j)) - \lambda_t - \xi_t(i,j) - \mu_t(j) + \mu_{t-1}(i) = 0 \\ &\Rightarrow \alpha_t(i,j) = d_t^{-2}(\lambda_t + \xi_t(i,j) + \mu_t(j) - \mu_{t-1}(i)) + m_t(i,j), \qquad (5.53)\end{aligned}$$

where we define $\mu_0(j) := 0$. Plugging into $\mathcal{L}$, we derive the dual problem:

$$\begin{aligned}\min D(\lambda_t, \xi_t(i,j), \mu_t(i)) =& \frac{1}{2} \sum_{t=1}^{L-1} d_t^2 \sum_{i,j} (\lambda_t + \xi_t(i,j) + \mu_t(j) - \mu_{t-1}(i))^2 \qquad (5.54) \\ & + \sum_{t=1}^{L-1} \sum_{i,j} m_t(i,j)(\lambda_t + \xi_t(i,j) + \mu_t(j) - \mu_{t-1}(i)) - \sum_{t=1}^{L-1} \lambda_t \\ s.t. \qquad & \xi_t(i,j) \geq 0. \quad \forall t \in [L-1],\, i,j \in [m].\end{aligned}$$

Taking derivatives:

$$\begin{aligned}\frac{\partial D}{\partial \xi_t(i,j)} &= d_t^{-2}(\lambda_t + \xi_t(i,j) + \mu_t(j) - \mu_{t-1}(i)) + m_t(i,j) = 0 \qquad \forall t \in [L-1] \\ &\Rightarrow \xi_t(i,j) = [-d_t^2 m_t(i,j) - \lambda_t - \mu_t(j) + \mu_{t-1}(i)]_+\end{aligned}$$

Table 5.1: Dataset statistics. n: #examples, d: #features, s: feature density.

dataset	n	d	$s(\%)$	dataset	n	d	$s(\%)$	dataset	n	d	$s(\%)$
adult9	32,561	123	11.28	covertype	522,911	6,274,932	22.22	reuters-c11	23,149	1,757,801	0.16
astro-ph	62,369	99,757	0.077	news20	15,960	7,264,867	0.033	reuters-ccat	23,149	1,757,801	0.16
aut-avn	56,862	20,707	0.25	real-sim	57,763	2,969,737	0.25	web8	45,546	579,586	4.24

$$\frac{\partial D}{\partial \lambda_t} = d_t^{-2} \sum_{i,j} (\lambda_t + \xi_t(i,j) + \mu_t(j) - \mu_{t-1}(i)) + \sum_{i,j} m_t(i,j) - 1 \qquad \forall t \in [L-1]$$

$$\frac{\partial D}{\partial \mu_t(i)} = d_t^{-2} \sum_{j} (\lambda_t + \xi_t(j,i) + \mu_t(i) - \mu_{t-1}(j)) \qquad \forall t \in [L-2]$$

$$+ d_{t+1}^{-2} \sum_{j} (\lambda_{t+1} + \xi_{t+1}(i,j) + \mu_{t+1}(j) - \mu_t(i)) + \sum_{j} m_t(j,i) - \sum_{j} m_{t+1}(i,j),$$

where we further define $\mu_{L-1}(j) := 0$. Obviously it takes $O(Lm^2)$ time to compute all the gradients, and so is $\{\xi_t(i,j)\}$.

5.5 Experimental results

In this section, we compare the empirical performance of our pragam with state-of-the-art binary linear SVM solvers, including two variants of pegasos[4] (Shalev-Shwartz et al., 2007), and two variants of BMRM[5] (Teo et al., 2010).

Datasets Table 5.1 lists the statistics of the dataset. `adult9`, `astro-ph`, `news20`, `real-sim`, `reuters-c11`, `reuters-ccat` are from the same source as in (Hsieh et al., 2008a). `aut-avn` classifies documents on auto and aviation (http://www.cs.umass.edu/~mccallum/data/sraa.tar.gz). `covertype` is from UCI repository. We did not normalize the feature vectors and no bias was used.

Algorithms Closest to pragam in spirit is the line search BMRM (ls-bmrm) which minimizes the current piecewise lower bound of regularized R_{emp} via a one dimensional line search between the current $\mathbf{w}_t$ and the latest subgradient. This simple update was enough for Smola et al. (2007b) to prove the $1/\epsilon$ rate of convergence. Interpreted in the adjoint form, this update corresponds to coordinate descent with the coordinate being chosen by the Gauss-Southwell rule (Bollen, 1984). In contrast, pragam performs a parallel update of all coordinates in each iteration and achieves faster convergence

[4]http://ttic.uchicago.edu/~shai/code/pegasos.tgz
[5]http://users.rsise.anu.edu.au/~chteo/BMRM.html

Table 5.2: λ for datasets.

dataset	λ	dataset	λ	dataset	λ	dataset	λ
adult	2^{-18}	astro-ph	2^{-17}	aut-avn	2^{-17}	covertype	2^{-17}
news20	2^{-14}	reuters-c11	2^{-19}	reuters-ccat	2^{-19}	real-sim	2^{-16}
web8	2^{-17}						

rate. So in this section, our main focus is to show that pragam converges faster than ls-bmrm.

We also present the results of pegasos, which is a primal estimated subgradient solver for SVM with L_1 hinge loss. We tested two extreme variants of pegasos: pegasos-n where all the training examples are used in each iteration, and pegasos-1 where only one randomly chosen example is used. Finally, we also compare with the qp-bmrm which solves the full QP in (5.10) in each iteration.

It should be noted that SVM$^{\text{Struct}}$ (Tsochantaridis et al., 2005) is also a general purpose regularized risk minimizer, and when specialized to binary SVMs, the SVMPerf (Joachims, 2005, 2006) gave the first linear time algorithm for training linear SVMs. We did not compare with SVMPerf because its cutting plane nature is very similar to BMRM when specialized to binary linear SVMs.

For pragam, since the Lipschitz constant L of the gradient of the SVM dual is unknown in practice, we resort to Algorithm 13 (Nesterov, 2007) which automatically estimates L while the rates presented in Section 5.3.1 are unchanged. We further implemented pragam-b, the pragam algorithm which uses SVM bias. In this case the inner optimization is a QP with box constraints and a single linear equality constraint.

For all datasets, we obtained the best $\lambda \in \left\{2^{-20}, \ldots, 2^{0}\right\}$ using their corresponding validation sets, and the chosen λ's are given in Table 5.2.

Results We first compared how fast $\mathsf{err}_t := \min_{t'<t} J(\mathbf{w}_{t'}) - J(\mathbf{w}^*)$ decreases with respect to the iteration index t. We used err_t instead of $J(\mathbf{w}_t) - J(\mathbf{w}^*)$ because $J(\mathbf{w}_t)$ in pegasos and ls-bmrm fluctuates drastically on some datasets. The results in Figure 5.2 show pragam converges faster than ls-bmrm and pegasos-n which both have $1/\epsilon$ rates. qp-bmrm converges faster than the rest algorithms in general. pegasos-1 is not included because it converges very slowly in terms of iterations.

Next, we compared in Figure 5.3 how fast err_t decreases in wall clock time. pragam is not fast in decreasing err_t to low accuracies like 10^{-3}. But it becomes quite competitive when higher accuracy is desired, whereas ls-bmrm and pegasos-1 often take a long time in this case. Again, qp-bmrm is much faster than the other algorithms.

Another important comparison is on generalization performance: how fast a solver finds a model with reasonable testing accuracy. At iteration t, we examined the test accuracy of $\mathbf{w}_{t'}$ where $t' := \text{argmin}_{t' \leq t} J(\mathbf{w}_{t'})$, and the result is presented in Figures 5.4 and 5.5 with respect to number of iterations and time respectively. It can be seen that although pragam manages to minimize the primal function fast, its generalization power is not improved efficiently. This is probably because this generalization performance hinges on the sparsity of the solution (or number of support vectors, (Graepel et al., 2000)), and compared with all the other algorithms pragam does not achieve any sparsity in the process of optimization. Asymptotically, all the solvers achieve very similar testing accuracy.

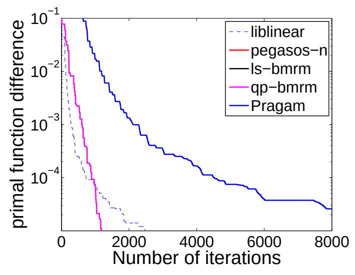
Since the objective function of pragam-b has a different feasible region than other optimizers which do not use bias, we only included it when comparing test accuracy. In Figures 5.4 and 5.5, the test accuracy of the optimal solution found by pragam-b is always higher than or similar to that of the other solvers. In most cases, pragam-b achieves the same test accuracy faster than pragam both in number of iterations and time.

5.6 Discussion and conclusions

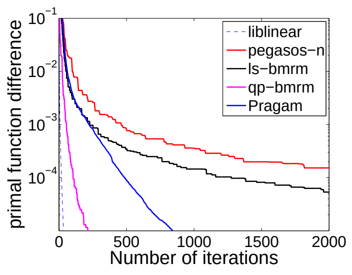
In this chapter, we described a new lower bound for the number of iterations required by BMRM and similar algorithms which are widely used solvers for the regularized risk minimization problem. This shows that the iteration bounds shown for these solvers are optimum. Our lower bounds are somewhat surprising because the empirical performance of these solvers indicates that they converge linearly to an ϵ accurate solution on a large number of datasets. Perhaps a more refined analysis is needed to explain this behavior.

The SVM problem has received significant research attention recently. For instance, Shalev-Shwartz et al. (2007) proposed a stochastic subgradient algorithm pegasos. The convergence of pegasos is analyzed in a stochastic setting and it was shown that it converges in $O(1/\epsilon)$ iterations. We believe that our lower bounds can be extended to any arbitrary subgradient based solvers in the primal including pegasos. This is part of ongoing research.

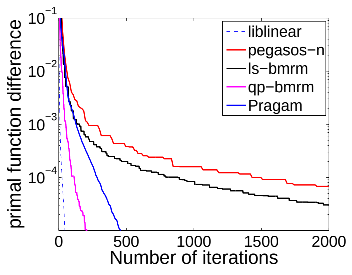
Our technique of solving the dual optimization problem is not new. A number of solvers including SVM-Light (Joachims, 1999) and SMO (Platt, 1999) work on the dual problem. Even though linear convergence is established for these solvers, their rates have $n^{\geq 2}$ dependence which renders the analysis unusable for practical purposes. Other possible approaches include the interior-point method of (Ferris & Munson, 2002) which costs $O(nd^2 \log(\log(1/\epsilon)))$ time and $O(d^2)$ space where d refers to the dimension of the features. liblinear (Hsieh et al., 2008a) performs coordinate descent in the dual, and has $O(nd \log(1/\epsilon))$ complexity but only after more than $O(n^2)$ steps. Mirror descent



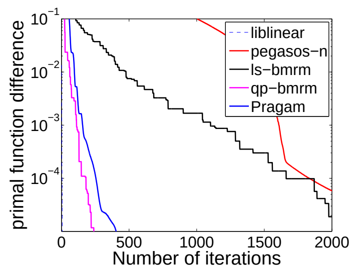
(a) adult9 (pegasos diverged)



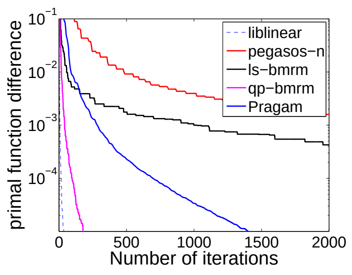
(b) astro-ph



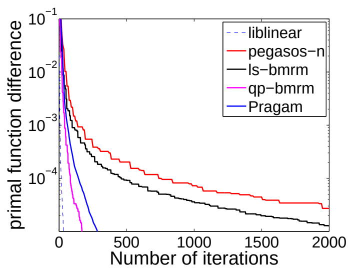
(c) aut-avn



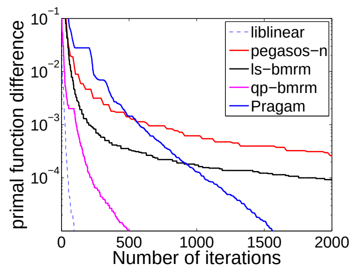
(d) covertype



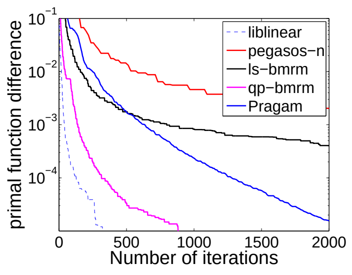
(e) news20



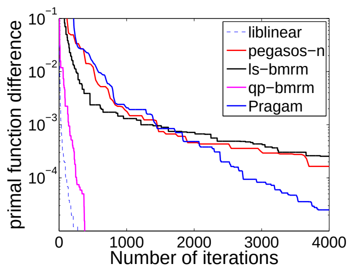
(f) real-sim



(g) reuters-c11



(h) reuters-ccat

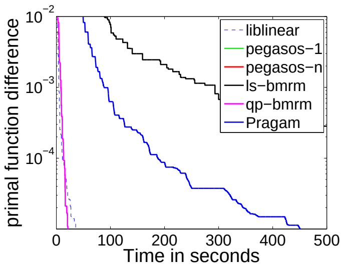


(i) web8

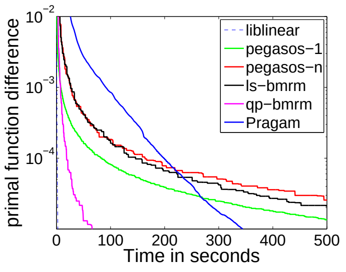
Figure 5.2: Primal function error versus number of iterations.

algorithms (Beck & Teboulle, 2003) cost $O(nd)$ per iteration, but their convergence rate is $1/\epsilon^2$. These rates are prohibitively expensive when n is very large.

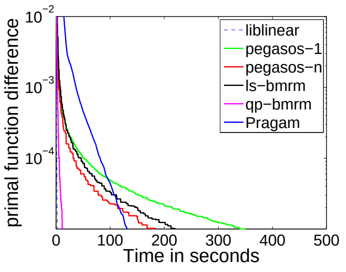
The $O(1/\sqrt{\epsilon})$ rates for the new SVM algorithm we described in this chapter has a favorable dependence on n as well as λ. Although our emphasis has been largely theoretical, the empirical experiments indicate that our solver is competitive with the state of the art. Finding an efficient solver with fast rates of convergence and good empirical performance remains a holy grail of optimization for machine learning.



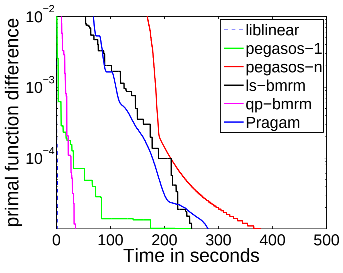
(a) adult9 (pegasos diverged)



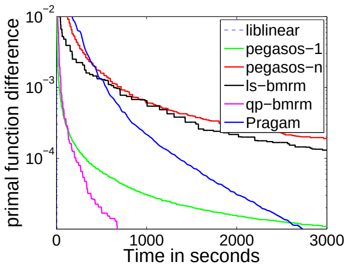
(b) astro-ph



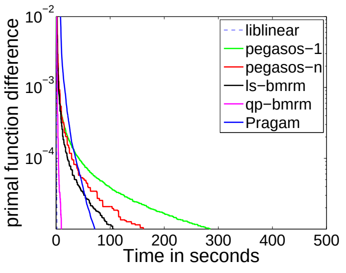
(c) aut-avn



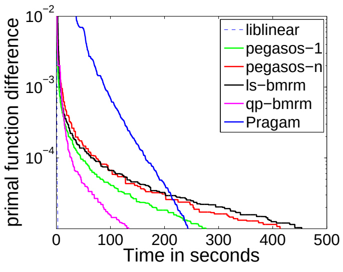
(d) covertype



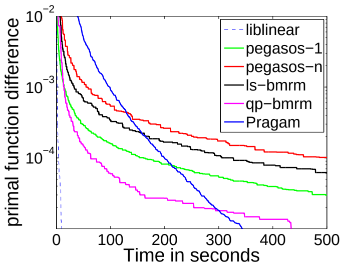
(e) news20



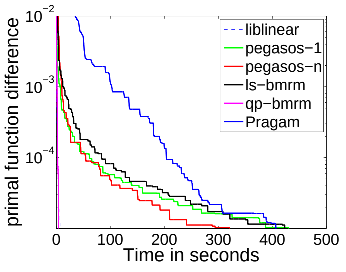
(f) real-sim



(g) reuters-c11

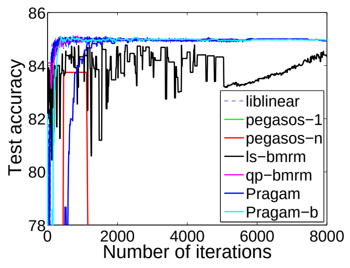


(h) reuters-ccat

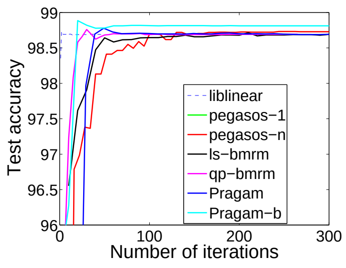


(i) web8

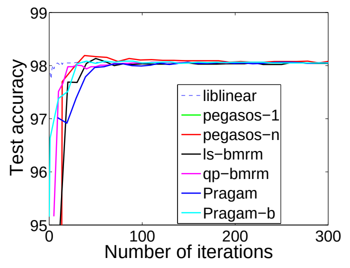
Figure 5.3: Primal function error versus time.



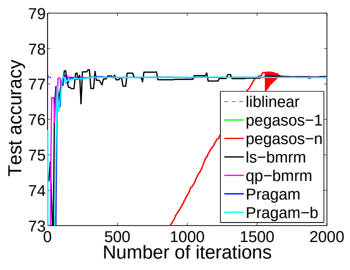
(a) adult9



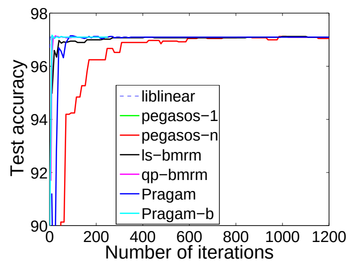
(b) astro-ph



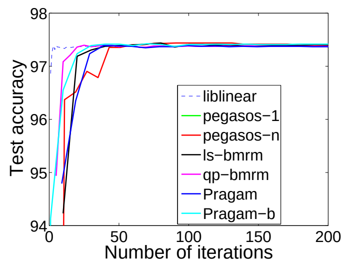
(c) aut-avn



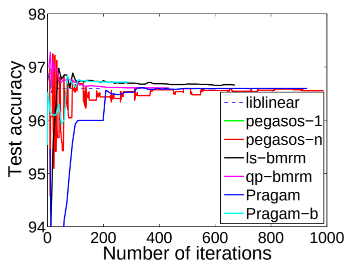
(d) covertype



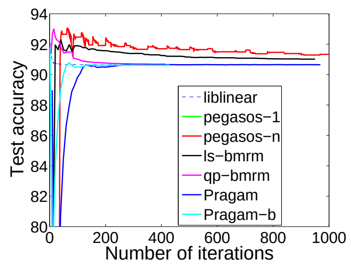
(e) news20



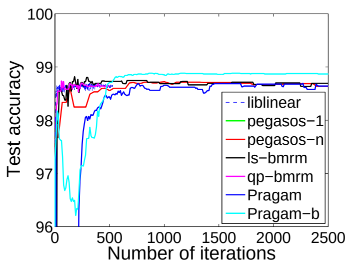
(f) real-sim



(g) reuters-c11

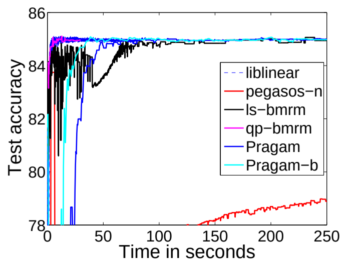


(h) reuters-ccat

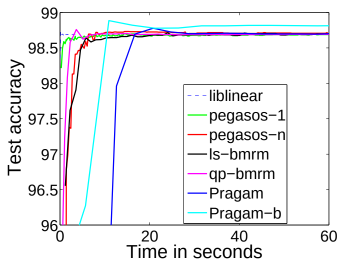


(i) web8

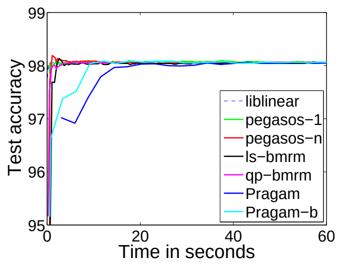
Figure 5.4: Test accuracy versus number of iterations.



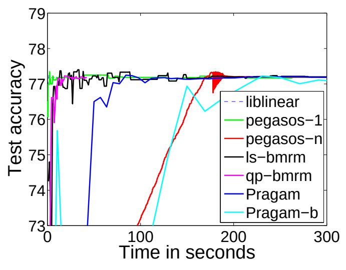
(a) adult9



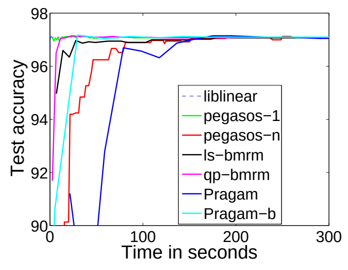
(b) astro-ph



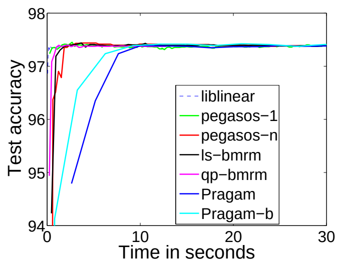
(c) aut-avn



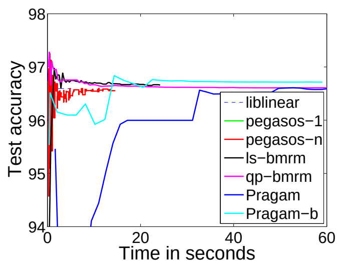
(d) covertype



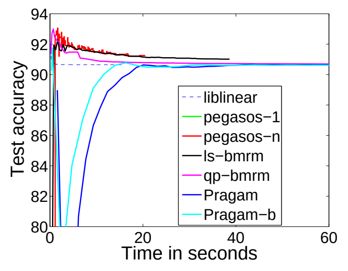
(e) news20



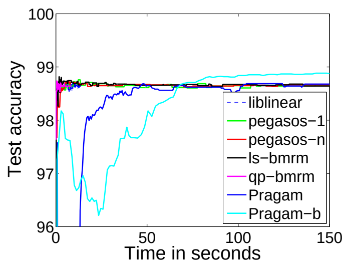
(f) real-sim



(g) reuters-c11



(h) reuters-ccat



(i) web8

Figure 5.5: Test accuracy versus time.

Appendix A

Fundamentals of Convex Analysis

In this appendix, we provide an introduction to convex analysis which is used in this thesis. All definitions and most properties can be found in (Hiriart-Urruty & Lemaréchal, 1993b).

A.1 Convex set and convex function

Definition 62 (Convex set) *A set $C \subseteq \mathbb{R}^d$ is convex if for any two points $\mathbf{x}_1, \mathbf{x}_2 \in C$ and any $\lambda \in (0, 1)$, we have*

$$\lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in C.$$

In other words, the line segement between any two points must lie in C.

Definition 63 (Open set) *A set $C \subseteq \mathbb{R}^d$ is open if for any point $\mathbf{x} \in C$, there exists an $\epsilon > 0$, such that $\mathbf{z} \in C$ for all $\mathbf{z} : \|\mathbf{z} - \mathbf{x}\| < \epsilon$. In other words, there is an ϵ-ball around $\mathbf{x}$: $B_\epsilon(\mathbf{x}) := \{\mathbf{z} : \mathbf{z} : \|\mathbf{z} - \mathbf{x}\| < \epsilon\}$ which is contained in C.*

Definition 64 (Convex hull) *For any nonempty set S in $\mathbb{R}^d$, its convex hull $\mathrm{co}S$ is defined as:*

$$\mathrm{co}S := \left\{ \sum_{i=1}^{k} \lambda_i \mathbf{x}_i : \mathbf{x}_i \in S, \lambda_i \geq 0, \sum_i \lambda_i = 1, k \in \mathbb{N} \right\}.$$

It can be shown to be the smallest convex set that subsumes S.

Definition 65 (Convex function) *Given a convex set C, a function $f : C \mapsto \mathbb{R}$ is convex if for any two points $\mathbf{x}_1, \mathbf{x}_2 \in C$ and any $\lambda \in (0, 1)$, we have*

$$f(\lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1 - \lambda) f(\mathbf{x}_2).$$

In general, we can define a generalized function $f : \mathbb{R}^d \mapsto \overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$, such that $f(\mathbf{x})$ is $+\infty$ for all $\mathbf{x} \notin C$. And we call C, on which f is finite, the domain of f

$$\text{dom } f := \left\{ \mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) < \infty \right\}.$$

Definition 66 (Subgradient and subdifferential) *Given a function f and a point $\mathbf{x}$ with $f(\mathbf{x}) < \infty$, a vector $\mathbf{u}$ is called a subgradient of f at $\mathbf{x}$ if*

$$f(\mathbf{y}) - f(\mathbf{x}) \geq \langle \mathbf{y} - \mathbf{x}, \mathbf{u} \rangle, \qquad \forall\, \mathbf{y} \in \mathbb{R}^d.$$

The set of all such $\mathbf{u}$ is called the subdifferential of f at $\mathbf{x}$, and is denoted by $\partial f(\mathbf{x})$.

Function f is convex iff $\partial f(\mathbf{x})$ is not empty for all $\mathbf{x}$ where $f(\mathbf{x}) < \infty$. If f is further differentiable at $\mathbf{x}$, then $\partial f(\mathbf{x})$ is a singleton comprised of the gradient of f at $\mathbf{x}$: $\nabla f(\mathbf{x})$.

Property 2 (Calculus rules for subgradient)

- ***Linearity:*** *if f and g are convex functions from $\mathbb{R}^d$ to $\mathbb{R}$, then $\partial(f + g)(\mathbf{x}) = \partial f(\mathbf{x}) + \partial g(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^d$.*
- ***Affine transformation:*** *for any linear transform A and offset $\mathbf{b}$, define $g(\mathbf{x}) := f(A\mathbf{x} + \mathbf{b})$. Then $\partial g(\mathbf{x}) = A^\top \partial f(A\mathbf{x} + \mathbf{b})$ for all $\mathbf{x} \in \mathbb{R}^d$.*
- ***Point-wise maximization:*** *Suppose $g(\mathbf{x}) := \max_{i \in \mathcal{I}} f_i(\mathbf{x})$, then $\partial f(\mathbf{x})$ is the convex hull of the union $\cup_{i \in \mathcal{I}^*_{\mathbf{x}}} \partial f_i(\mathbf{x})$, where $\mathcal{I}^*_{\mathbf{x}}$ is the index set which attains the max: $\mathcal{I}^*_{\mathbf{x}} := \{i \in \mathcal{I} : f_i(\mathbf{x}) = f(\mathbf{x})\}$.*

Definition 67 (Strong convexity) *Given a norm $\|\cdot\|$ on $\mathbb{R}^d$, a convex function f is called strongly convex with modulus σ wrt $\|\cdot\|$ if for all $\mathbf{x}_1, \mathbf{x}_2 \in \text{dom } f$ and $\lambda \in (0, 1)$ we have*

$$f(\lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1 - \lambda) f(\mathbf{x}_2) - \frac{1}{2}\lambda(1 - \lambda) \|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

Strong convexity can be equivalently defined in the following ways depending on the differentiability:

Property 3 *f is σ-strongly convex wrt $\|\cdot\|$ iff:*

- *$f(\mathbf{x}) - \frac{\sigma}{2} \|\mathbf{x}\|$ is convex.*
- *$f(\mathbf{x}_2) \geq f(\mathbf{x}_1) + \langle \mathbf{g}, \mathbf{x}_2 - \mathbf{x}_1 \rangle + \frac{1}{2}\sigma \|\mathbf{x}_1 - \mathbf{x}_2\|^2$ for all $\mathbf{x}_1, \mathbf{x}_2 \in \text{dom} f$ and $\mathbf{g} \in \partial f(\mathbf{x}_1)$.*

- *If f is differentiable, then* $\langle \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle \geq \sigma \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2$ *for all* $\mathbf{x}_1, \mathbf{x}_2 \in \mathrm{dom} f$.
- *If f is twice differentiable, then* $\left\langle \nabla^2 f(\mathbf{x}), \mathbf{x} \right\rangle \geq \sigma \left\| \mathbf{x} \right\|^2$ *for all* $\mathbf{x} \in \mathrm{dom} f$. *If the norm is chosen as the Euclidean norm, then this is equivalent to the Hessian's eigenvalues being lower bounded by σ.*

Definition 68 (Lipschitz continuity) *Given a norm $\|\cdot\|$ on $\mathbb{R}^d$ and a norm $\|\cdot\|_*$ on $\mathbb{R}^s$, a function $f : \mathbb{R}^d \mapsto \mathbb{R}^s$ is called Lipschitz continuous with modulus L wrt $\|\cdot\|$ and $\|\cdot\|_*$ if*

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)|_* \leq L \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\| \qquad \forall \mathbf{x}_1, \mathbf{x}_2 : f(\mathbf{x}_1) < +\infty, f(\mathbf{x}_2) < +\infty.$$

Lipschitz continuity characterizes the rate of change of f, and is stronger than continuity but weaker than differentiability.

If a convex function $f : \mathbb{R}^d \mapsto \overline{\mathbb{R}}$ is differentiable and its gradient $\nabla f : \mathrm{int}\ \mathrm{dom} f \to \mathbb{R}^d$ is Lipschitz continuous with modulus L wrt $\|\cdot\|$ (setting $\|\cdot\|_* = \|\cdot\|$), then we call f as L-l.c.g.

A.2 Fenchel conjugate

Definition 69 (Fenchel dual) *Given a function $f : \mathbb{R}^d \to \overline{\mathbb{R}}$, its Fenchel dual is defined as*

$$f^\star(\boldsymbol{\mu}) := \sup_{\mathbf{x} \in \mathbb{R}^d} \langle \mathbf{x}, \boldsymbol{\mu} \rangle - f(\mathbf{x}).$$

Example 1 (Affine functions) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be defined as $f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle + b$ where $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Then*

$$f^\star(\boldsymbol{\mu}) = \begin{cases} -b & \text{if } \boldsymbol{\mu} = \mathbf{a} \\ +\infty & \text{otherwise} \end{cases}.$$

Example 2 (Hinge loss) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be defined as $f(\mathbf{x}) = [\rho - \langle \mathbf{w}, \mathbf{x} \rangle]_+$ where $\rho \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^d$, and $[\cdot]_+ := \max\{\cdot, 0\}$. Then*

$$f^\star(\boldsymbol{\mu}) = \begin{cases} \rho\lambda & \text{if } \boldsymbol{\mu} = \lambda \mathbf{w}, \text{ and } \lambda \in [-1, 0] \\ +\infty & \text{otherwise} \end{cases}.$$

Example 3 *Let us consider the Fenchel dual of the function*

$$f(\mathbf{x}) = \begin{cases} -\sum_{i=1}^{d} x_i & \textit{if } \mathbf{x} \in Q \\ +\infty & \textit{otherwise} \end{cases},$$

where $Q = \left\{\mathbf{x} \in \mathbb{R}^d : x_i \in [0, d^{-1}], \sum_i y_i x_i = 0\right\}$ *and* $y_i \in \{-1, +1\}$. *Below we show*

$$f^\star(\boldsymbol{\mu}) = \min_{b \in \mathbb{R}} \frac{1}{d} \sum_i [1 + \mu_i - y_i b]_+ ,$$

where $[x]_+ := \max\{x, 0\}$. *To this end, it suffices to show that for all* $\boldsymbol{\mu} \in \mathbb{R}^d$*:*

$$\sup_{\mathbf{z} \in Q} \langle \mathbf{z}, \boldsymbol{\mu} \rangle + \sum_i z_i = \min_{b \in \mathbb{R}} \frac{1}{d} \sum_i [1 + \mu_i - y_i b]_+ . \tag{A.1}$$

Posing the latter optimization as:

$$\min_{\xi_i, b} \frac{1}{d} \sum_i \xi_i \qquad s.t. \quad 1 + \mu_i - y_i b \le \xi_i, \quad \xi_i \ge 0.$$

Write out the Lagrangian:

$$\mathcal{L} = \frac{1}{d} \sum_i \xi_i + \sum_i \rho_i (1 + z_i - y_i b - \xi_i) - \sum_i \beta_i \xi_i .$$

Taking partial derivatives:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \xi_i} &= \frac{1}{d} - z_i - \beta_i = 0 & \Rightarrow \quad & z_i \in [0, d^{-1}], \\ \frac{\partial \mathcal{L}}{\partial b} &= -\sum_i \rho_i y_i = 0 & \Rightarrow \quad & \sum_i z_i y_i = 0. \end{aligned}$$

Plugging back into L,

$$\mathcal{L} = \sum_i z_i (1 + \mu_i), \qquad s.t. \quad z_i \in [0, d^{-1}], \quad \sum_i y_i z_i = 0.$$

Maximizing $\mathcal{L}$ *wrt* $\mathbf{z}$ *is exactly the LHS of* (A.1)*.*

Example 4 (Relative entropy) *Suppose*

$$f(\mathbf{x}) = \begin{cases} \sum_{i=1}^{d} w_i \ln \frac{w_i}{1/n} & \textit{if } \mathbf{x} \in \Delta_d \\ +\infty & \textit{otherwise} \end{cases}.$$

where Δ_d is the d-dimensional simplex: $\left\{\mathbf{x} \in [0,1]^d : \sum_i x_i = 1\right\}$. Then

$$f^\star(\boldsymbol{\mu}) = \ln\left(\frac{1}{d}\sum_{i=1}^{d} \exp \mu_)\right).$$

Property 4 (Dual connection) *If f is convex and closed, then*

$$f(\mathbf{x}) + f^\star(\boldsymbol{\mu}) - \langle \mathbf{x}, \boldsymbol{\mu}\rangle \geq 0.$$

And the equality is attained iff $\boldsymbol{\mu} \in \partial f(\mathbf{x})$ iff $\mathbf{x} \in \partial f^\star(\boldsymbol{\mu})$.

Property 5 *$f^\star$, as the supremum of linear functions, is convex. It is also closed. When f is closed and convex, $f^{\star\star} = f$.*

Property 6 (Calculus rules)

1. *If $g(\mathbf{x}) = f(\mathbf{x}) + a$, then $g^\star(\boldsymbol{\mu}) = f^\star(\boldsymbol{\mu}) - a$.*
2. *If $g(\mathbf{x}) = af(\mathbf{x})$ with $a > 0$, then $g^\star(\boldsymbol{\mu}) = af^\star(\boldsymbol{\mu}/a)$.*
3. *If A is an invertible linear operator, and $g(\mathbf{x}) = f(A\mathbf{x})$, then $g^\star(\boldsymbol{\mu}) = f^\star(A^{-1}\boldsymbol{\mu})$.*
4. *If $g(\mathbf{w}) = f(\mathbf{x} - \mathbf{x}_0)$, then $g^\star(\boldsymbol{\mu}) = f^\star(\boldsymbol{\mu}) + \langle \boldsymbol{\mu}, \mathbf{x}_0\rangle$.*
5. *If $g(\mathbf{x}) = f(\mathbf{x}) + \langle \boldsymbol{\mu}_0, \mathbf{x}\rangle$, then $g^\star(\boldsymbol{\mu}) = f^\star(\boldsymbol{\mu} - \boldsymbol{\mu}_0)$.*

Property 7 (max **rule)** *(Hiriart-Urruty & Lemaréchal, 1993a, Theorem 2.4.7) Let $f_1, \ldots, f_n$ be finitely many convex functions from $\mathbb{R}^d$ to $\mathbb{R}$ and let $f := \max_i f_i$. Denote by $m := \min\{n, d+1\}$. For every*

$$\boldsymbol{\mu} \in \text{dom } f^\star = \text{co} \cup_{i \in [n]} \{\text{dom } f_i^\star\}.$$

there exists $\mathbf{s}_i \in \text{dom } f_i^\star$ and convex multipliers $\sigma_i \in \mathbb{R}$ $(i = 1, \ldots, m)$ such that

$$\boldsymbol{\mu} = \sum_i \sigma_i \mathbf{s}_i \qquad \textit{and} \qquad f^\star(\boldsymbol{\mu}) = \sum_i \sigma_i f_i^\star(\mathbf{s}_i).$$

The expansion of $\{\sigma_i\}$ may be not unique, but the value of $f^\star(\boldsymbol{\mu})$ is unique.

As an application of the max rule, we have

Example 5 (Maximum of affine functions) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be defined as $f(\mathbf{x}) = \max_{i\in[n]} \langle \mathbf{a}_i, \mathbf{x}\rangle + b_i$ where $b_i \in \mathbb{R}$, and $\mathbf{a}_i \in \mathbb{R}^d$. Then*

$$f^\star(\boldsymbol{\mu}) = \begin{cases} -\sum_{i=1}^n \lambda_i b_i & \textit{if } \boldsymbol{\mu} = \sum_{i=1}^n \lambda_i \mathbf{a}_i \textit{ with } \lambda_i \geq 0, \sum_i \lambda_i \leq 1 \\ +\infty & \textit{otherwise} \end{cases}.$$

Example 6 (Maximum of hinge loss) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be defined as $f(\mathbf{x}) = \max_{i\in[n]}[\rho_i - \langle \mathbf{w}_i, \mathbf{x}\rangle]_+$ where $\rho_i \in \mathbb{R}$, and $\mathbf{w}_i \in \mathbb{R}^d$. Then*

$$f^\star(\boldsymbol{\mu}) = \begin{cases} -\sum_{i=1}^n \lambda_i \rho_i & \text{if } \boldsymbol{\mu} \in \left\{-\sum_{i=1}^k \lambda_i \mathbf{w}_i, \lambda_i \geq 0, \sum_i \lambda_i \leq 1\right\} \\ +\infty & \text{otherwise} \end{cases}.$$

Property 8 (Strong convexity and L-l.c.g under Fenchel dual) *(Hiriart-Urruty & Lemaréchal, 1993b, Theorem 4.2.1 and 4.2.2)*

1. *If $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is strongly convex with modulus $c > 0$, then $\operatorname{dom} f^\star = \mathbb{R}^n$, and $\nabla f^\star$ is Lipschitz continuous with constant σ^{-1}.*

2. *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex and have L-Lipschitz continuous gradient mapping ($L > 0$), then $f^\star$ is strongly convex with modulus L^{-1}.*

Property 9 *(Borwein & Lewis, 2000, Theorem 3.3.5) Let E_1 be a subset of $\mathbb{R}^d$, and E_2 be a subset of $\mathbb{R}^s$. Let A be a linear map from E_1 to E_2. Given functions $f : E_1 \to \overline{\mathbb{R}}$ and $g : E_2 \to \overline{\mathbb{R}}$, we have*

$$\inf_{\mathbf{x}\in E_1} f(\mathbf{x}) + g(A\mathbf{x}) \geq \sup_{\boldsymbol{\mu}\in E_2} -f^\star(A^\star \boldsymbol{\mu}) - g^\star(-\boldsymbol{\mu}).$$

If f and g are convex and $\mathbf{0} \in \operatorname{core}(\operatorname{dom} g - A \operatorname{dom} f)$[1], then the equality holds and the supremum is attained if finite.

A.3 Convex analysis for the log partition function

In this section, we prove some important fundamental properties of the log partition function of exponential family distributions.

Proposition 70 *$g(\boldsymbol{\theta})$ strongly convex if, and only if, $\boldsymbol{\phi}$ is minimal.*

Proof **Strong convexity $\Rightarrow$ minimality:** Suppose there exits an $\boldsymbol{\alpha}$, which is not necessarily in Θ, such that $\langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle = C$ ν-almost everywhere. For any $\boldsymbol{\theta}$ in the interior of Θ, there must exist a $\delta > 0$ such that $\boldsymbol{\theta} + \lambda\boldsymbol{\alpha} \in \Theta$ for all $\lambda \in [0, \delta]$. Let $f(x) := \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}/2\rangle)$ and $g(x) := \exp(\langle \boldsymbol{\phi}(x), (\boldsymbol{\theta} + \delta\boldsymbol{\alpha})/2\rangle)$. As $g(x)/f(x) = \exp(\langle \boldsymbol{\phi}(x), (\delta\boldsymbol{\alpha})/2\rangle) = \exp(\delta C/2)$ ν-almost everywhere, so $\|f\|_2 \|g\|_2 = \|fg\|_1$, *i.e.*

$$\sqrt{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)} \cdot \sqrt{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} + \delta\boldsymbol{\alpha}\rangle)\nu(\mathrm{d}x)} = \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} + \delta\boldsymbol{\alpha}/2\rangle)\nu(\mathrm{d}x).$$

[1] The core of a set C in a space E is the set of points $\mathbf{x} \in C$ such that for any direction $\mathbf{d} \in E$, $\mathbf{x} + t\mathbf{d} \in C$ for all small real t.

Therefore, taking log of both sides,

$$\frac{1}{2}g(\boldsymbol{\theta}) + \frac{1}{2}g(\boldsymbol{\theta} + \delta\boldsymbol{\alpha}) = g(\boldsymbol{\theta} + \delta\boldsymbol{\alpha}/2),$$

which contracts with the strong convexity (in fact contracts with the strict convexity which is implied by strong convexity).

Minimality $\Rightarrow$ strong convexity: As the Hessian is the covariance matrix and Θ is open, strong convexity follows if we can show the covariance matrix is positive definite very where. Suppose otherwise, there exists a $\boldsymbol{\theta}$ such that the covariance matrix under $p(x; \boldsymbol{\theta})$ is just positive semi-definition, *i.e.* there is a vector $\boldsymbol{\alpha}$ satisfying

$$\boldsymbol{\alpha}^\top (\mathbb{E}[\boldsymbol{\phi}\boldsymbol{\phi}^\top] - \mathbb{E}[\boldsymbol{\phi}]\mathbb{E}[\boldsymbol{\phi}]^\top)\boldsymbol{\alpha} = 0 \qquad \Rightarrow \qquad \mathbb{E}[\langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle^2] = (\mathbb{E}[\langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle])^2.$$

This means

$$\begin{aligned}
&\int \langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle^2 \frac{\exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)}{\int \exp(\langle \boldsymbol{\phi}(y), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}y)} = \left(\int \langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle \frac{\exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)}{\int \exp(\langle \boldsymbol{\phi}(y), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}y)}\right)^2 \\
\Rightarrow &\int \langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle^2 \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}x) \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}x) \\
&\quad = \left(\int \langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)\right)^2.
\end{aligned}$$

By Cauchy-Schwartz inequality, we derive $\langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}/2\rangle) / \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}/2\rangle) = \langle \boldsymbol{\phi}(x), \boldsymbol{\alpha}\rangle$ is constant ν-almost everythere. ■

Proposition 71 *Irrespective of whether* Θ *is open, the log partition function* $g(\boldsymbol{\theta})$ *is lower semi-coninuous.*

Proof Let $\boldsymbol{\theta} \in \Theta$, and let a sequence in Θ $\{\boldsymbol{\theta}_n\}$ converge to $\boldsymbol{\theta}$. Since $\exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle) = \lim_{n\to\infty} \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}_n\rangle)$ for all x, so by Fatou's lemma, we have

$$\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}x) \le \liminf_{n\to\infty} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}_n\rangle)\nu(\mathrm{d}x).$$

So $h(\boldsymbol{\theta}) := \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)$ is a lower semi-continuous function in $\boldsymbol{\theta}$. As log is monotonically increasing, hence $g(\boldsymbol{\theta}) = \log h(\boldsymbol{\theta})$ must be lower semi-coninuous. ■

When Θ is open, the lower semi-continuity is direct from the convexity of $g(\boldsymbol{\theta})$ because any convex function must be continuous on the interior of its domain.

Below we assume that Θ is open.

Proposition 72 *For any distribution $p(x;\boldsymbol{\theta})$ from an exponential family, the expectation of the absolute value of sufficient statistics and their arbitrary power is in $\mathbb{R}$:*

$$\mathop{\mathbb{E}}_{x\sim p(x;\boldsymbol{\theta})}\left[\left|\prod_i \phi_i^{\alpha_i}(x)\right|\right] \in \mathbb{R}.$$

for all $\alpha_i \in \mathbb{N}\cup\{0\}$. As a result,

$$\mathop{\mathbb{E}}_{x\sim p(x;\boldsymbol{\theta})}\left[\prod_i \phi_i^{\alpha_i}(x)\right] \in \mathbb{R}.$$

Proof For any $z \in \mathbb{R}$, $\delta > 0$, by elementary math we have $|z| \le \delta^{-1}(e^{z\delta} + e^{-z\delta})$. So for any $n \in \mathbb{N}\cup\{0\}$, we have

$$|z|^n \le \delta^{-n}(e^{z\delta}+e^{-z\delta})^n = \delta^{-n}\sum_{i=0}^{n}\binom{n}{i}e^{(n-2i)z\delta} \le \left(\frac{2}{\delta}\right)^n (e^{nz\delta}+e^{-nz\delta}).$$

Therefore

$$\begin{aligned}\left|\prod_i \phi_i^{\alpha_i}(x)\right| &\le \left(\frac{2}{\delta}\right)^{\sum_i \alpha_i}\prod_i (e^{\delta\alpha_i\phi_i(x)} + e^{-\delta\alpha_i\phi_i(x)}) \\ &= \left(\frac{2}{\delta}\right)^{\sum_i \alpha_i}\sum_{b_i\in\{-1,1\}}\exp\left(\delta\sum_i b_i\alpha_i\phi_i(x)\right)\end{aligned}$$

For any $\boldsymbol{\theta} \in \Theta$, due to the openness of Θ, there must exist a $\delta > 0$ such that $\boldsymbol{\theta} + \delta\,\mathrm{vec}_i\,\{b_i\alpha_i\} \in \Theta$ for all $b_i \in \{-1,1\}$. Here $\mathrm{vec}_i\,\{b_i\alpha_i\}$ means assembling all $b_i\alpha_i$ into one vector. So

$$\begin{aligned}&\int\left|\prod_i \phi_i^{\alpha_i}(x)\right|\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\nu(\mathrm{d}x) \\ \le\; &\left(\frac{2}{\delta}\right)^{\sum_i \alpha_i}\sum_{b_i\in\{-1,1\}}\int\exp\left(\left\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}+\delta\mathop{\mathrm{vec}}_i\{b_i\alpha_i\}\right\rangle\right)\nu(\mathrm{d}x) \\ <\; &+\infty. \end{aligned} \tag{A.2}$$

For improper integral, absolute convergence implies normal convergence. ■

Proposition 73 *For any distribution $p(x;\boldsymbol{\theta})$ from an exponential family, $g(\boldsymbol{\theta})$ must be differentiable at $\boldsymbol{\theta}$ and the derivative is equal to the mean of suffficient statistics,* i.e.

$$\nabla g(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{x\sim p(x;\boldsymbol{\theta})}[\boldsymbol{\phi}(x)].$$

Proof First,

$$\frac{\partial}{\partial \theta_i} g(\boldsymbol{\theta}) = \frac{\frac{\partial}{\partial \theta_i} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \mathrm{d}x}{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x)},$$

and we show that the direvative and integral can be interchanged

$$\frac{\partial}{\partial \theta_i} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x) = \int \frac{\partial}{\partial \theta_i} \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x).$$

By definition, letting $\mathbf{e}_i$ be the i-th coordinate unit vector, we have

$$\begin{aligned} \frac{\partial}{\partial \theta_i} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x) &= \lim_{t \to 0} \frac{1}{t} \left\{ \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} + t\mathbf{e}_i \rangle) \nu(\mathrm{d}x) - \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x) \right\} \\ &= \lim_{t \to 0} \int \underbrace{\frac{1}{t} \left[\exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} + t\mathbf{e}_i \rangle) - \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \right]}_{:=h(x,t)} \nu(\mathrm{d}x) \end{aligned}$$

As Θ is open, there exists a $\delta > 0$ such that both $\boldsymbol{\theta} - \delta \mathbf{e}_i$ and $\boldsymbol{\theta} + \delta \mathbf{e}_i$ are in Θ. For any $z \in \mathbb{R}$, it is elementary to show that $|e^z - 1| \leq e^{|z|} - 1$. And if $a \in \overline{\mathbb{R}}_+$ and $z \in (0, \delta)$, then $\frac{e^{az}-1}{z} \leq \frac{e^{a\delta}-1}{\delta}$. So for any $t \in (-\delta, \delta)$,

$$\left| \frac{\exp(\phi_i(x)t) - 1}{t} \right| \leq \frac{e^{|\phi_i(x)t|} - 1}{|t|} \leq \frac{e^{|\phi_i(x)|\delta} - 1}{\delta} \leq \frac{e^{\phi_i(x)\delta} + e^{-\phi_i(x)\delta}}{\delta}.$$

Hence

$$\begin{aligned} |h(x,t)| &= \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \left| \frac{\exp(\phi_i(x)t) - 1}{t} \right| \\ &\leq \delta^{-1} \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)(\exp(\phi_i(x)t) + \exp(-\phi_i(x)t)) \\ &= \delta^{-1} \left(\exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} + \delta \mathbf{e}_i \rangle) + \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} - \delta \mathbf{e}_i \rangle) \right). \end{aligned}$$

Since both $\theta - \delta \mathbf{e}_i$ and $\theta + \delta \mathbf{e}_i$ are in Θ, so $\int |h(x,t)| \, \nu(\mathrm{d}x) \leq \delta^{-1}(g(\boldsymbol{\theta} + \delta \mathbf{e}_i) + g(\boldsymbol{\theta} - \delta \mathbf{e}_i))$. Therefore by the dominated convergence theorem, we have

$$\begin{aligned} \frac{\partial}{\partial \theta_i} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x) &= \lim_{t \to 0} \int h(x,t) \nu(\mathrm{d}x) = \int \lim_{t \to 0} h(x,t) \nu(\mathrm{d}x) \\ &= \int \frac{\partial}{\partial \theta_i} \exp(\langle \boldsymbol{\phi}(x), \theta \rangle) \nu(\mathrm{d}x) = \int \phi_i(x) \exp(\langle \boldsymbol{\phi}(x), \theta \rangle) \nu(\mathrm{d}x). \end{aligned}$$

So

$$\frac{\partial}{\partial \theta_i} g(\boldsymbol{\theta}) = \frac{\frac{\partial}{\partial \theta_i} \int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x)}{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x)} = \int \phi_i(x) \frac{\exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle)}{\int \exp(\langle \boldsymbol{\phi}(x), \boldsymbol{\theta} \rangle) \nu(\mathrm{d}x)} \nu(\mathrm{d}x) = \mathop{\mathbb{E}}_{x \sim p(x; \boldsymbol{\theta})} [\phi_i(x)].$$

■

Proposition 74 *The log partition function $g(\boldsymbol{\theta})$ is C^∞ on Θ.*

Proof It is not hard to see that we only need to prove, without loss of generality,

$$\frac{\partial}{\partial\theta_1}\int\prod_i \phi_i^{\alpha_i}(x)\cdot\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)=\int\frac{\partial}{\partial\theta_1}\prod_i\phi_i^{\alpha_i}(x)\cdot\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)\in\mathbb{R}$$

for all $\alpha_i\ge 0$ and for all $\boldsymbol{\theta}\in\Theta$. We can proceed in the same way as the proof of Proposition 73, but we prefer writing out the details. First,

$$\begin{aligned}&\frac{\partial}{\partial\theta_1}\int\prod_i\phi_i^{\alpha_i}(x)\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)\\&=\lim_{t\to0}\frac{1}{t}\left\{\int\prod_i\phi_i^{\alpha_i}(x)\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}+t\mathbf{e}_1\rangle)\nu(\mathrm{d}x)-\int\prod_i\phi_i^{\alpha_i}(x)\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\nu(\mathrm{d}x)\right\}\\&=\lim_{t\to0}\int\underbrace{\frac{1}{t}\prod_i\phi_i^{\alpha_i}(x)\left[\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}+t\mathbf{e}_1\rangle)-\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\right]}_{:=h(x,t)}\nu(\mathrm{d}x).\end{aligned}$$

As Θ is open, there exists a $\delta>0$ such that both $\boldsymbol{\theta}-\delta\mathbf{e}_1$ and $\boldsymbol{\theta}+\delta\mathbf{e}_1$ are in Θ. So for any $t\in(-\delta,\delta)$, we have

$$\begin{aligned}|h(x,t)|&=\left|\prod_i\phi_i^{\alpha_i}(x)\right|\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\left|\frac{\exp(\phi_1(x)t)-1}{t}\right|\\&\le\delta^{-1}\left|\prod_i\phi_i^{\alpha_i}(x)\right|\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)(\exp(\phi_1(x)t)+\exp(-\phi_1(x)t))\\&=\delta^{-1}\left|\prod_i\phi_i^{\alpha_i}(x)\right|(\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}+\delta\mathbf{e}_1\rangle)+\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}-\delta\mathbf{e}_1\rangle)).\end{aligned}$$

By Eq. (A.2), we have

$$\begin{aligned}\int|h(x,t)|\,\nu(\mathrm{d}x)\le\delta^{-1}&\int\left|\prod_i\phi_i^{\alpha_i}(x)\right|\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}+\delta\mathbf{e}_1\rangle)\nu(\mathrm{d}x)\\&+\delta^{-1}\int\left|\prod_i\phi_i^{\alpha_i}(x)\right|\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}-\delta\mathbf{e}_1\rangle)\nu(\mathrm{d}x)\\&<+\infty.\end{aligned}$$

Therefore by the dominated convergence theorem, we have

$$\frac{\partial}{\partial\theta_i}\int\prod_i \phi_i^{\alpha_i}(x)\cdot\exp(\langle\boldsymbol{\phi}(x),\boldsymbol{\theta}\rangle)\nu(\mathrm{d}x) = \lim_{t\to 0}\int h(x,t)\nu(\mathrm{d}x) = \int \lim_{t\to 0} h(x,t)\nu(\mathrm{d}x)$$
$$= \int \frac{\partial}{\partial\theta_i}\prod_i \phi_i^{\alpha_i}(x)\cdot\exp(\langle\boldsymbol{\phi}(x),\theta\rangle)\nu(\mathrm{d}x) = \int \phi_1(x)\prod_i \phi_i^{\alpha_i}(x)\cdot\exp(\langle\boldsymbol{\phi}(x),\theta\rangle)\nu(\mathrm{d}x)$$
$$< +\infty,$$

where the last step is due to Proposition 72. ■

Appendix B

Message Update Formulae of Expectation Propagation

In this appendix, we give the detailed derivation of the messages used in EP updates in Chapter 3.

B.1 Preliminaries: canonical parametrization of multi-variate Gaussians

The multi-variate Gaussian distributions (MVGs) are commonly expressed in terms of *moment parametrization*:

$$\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\tilde{\pi})^{n/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{B.1}$$

where $\tilde{\pi} = 3.14159...$[1] The mean and variance can be easily read off from Eq (B.1), hence its name. However, inconvenience arises when we study the product of two MVGs' density function. Although the result still assumes the exponential form of Eq. (B.1), the new mean and variance can not be expressed in simple forms of the original mean and variance. Therefore the *canonical representation* is often adopted:

$$\boldsymbol{\Pi} := \boldsymbol{\Sigma}^{-1}, \qquad \boldsymbol{\Gamma} := \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu},$$

and we obtain an equivalent formulation of Eq (B.1) denoted by $\mathcal{N}'$:

$$\mathcal{N}'(\mathbf{x};\boldsymbol{\Gamma},\boldsymbol{\Pi}) := \exp\left(a + \boldsymbol{\Gamma}^{\top}\mathbf{x} - \frac{1}{2}\mathbf{x}^{\top}\boldsymbol{\Pi}\mathbf{x}\right), \tag{B.2}$$

[1]We will use π later for the precision of univariate Gaussians as a standard notation, so here we introduce $\tilde{\pi}$ to avoid symbol conflict.

where $a := \left(-n \log(2\tilde{\pi}) + \log|\mathbf{\Pi}| - \mathbf{\Gamma}^\top \mathbf{\Pi} \mathbf{\Gamma}\right)/2$. $\mathbf{\Pi}$ is often called *precision* and $\mathbf{\Gamma}$ is called *precision-mean*. For one dimensional Gaussians, the canonical parametrization can be simplified into

$$\pi = \frac{1}{\sigma^2}, \qquad \tau = \pi\mu = \frac{\mu}{\sigma^2} \qquad \Leftrightarrow \qquad \sigma = \sqrt{\frac{1}{\pi}}, \qquad \mu = \frac{\tau}{\pi}.$$

So whenever we see $p(x) \propto \exp\left(-\frac{1}{2}ax^2 + bx\right)$, we can immediately read off $\pi = a$ and $\tau = b$ from the quadratic form. The representation of Eq. (B.2) is also useful when we multiply or divide two Gaussians. Suppose $p_i(\mathbf{x}) \sim \mathcal{N}'(\mathbf{\Gamma}_i, \mathbf{\Pi}_i)$ for $i = 1, 2$, then

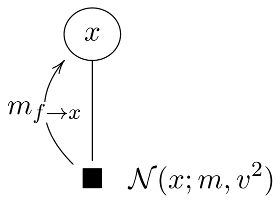
$$p_1(\mathbf{x})p_2(\mathbf{x}) \sim \mathcal{N}'(\mathbf{\Gamma}_1 + \mathbf{\Gamma}_2, \mathbf{\Pi}_1 + \mathbf{\Pi}_2), \qquad \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} \sim \mathcal{N}'(\mathbf{\Gamma}_1 - \mathbf{\Gamma}_2, \mathbf{\Pi}_1 - \mathbf{\Pi}_2). \tag{B.3}$$

Notice that $\frac{p_1(\mathbf{x})}{p_2(\mathbf{x})}$ can have a negative definite "covariance matrix". Anyway, $\frac{p_1(\mathbf{x})}{p_2(\mathbf{x})}$ is obviously NOT the density function of the random variable $\frac{\mathbf{x}_1}{\mathbf{x}_2}$.

B.2 EP updates for all factors in Figure 3.1

We now derive the update formulae in Figure 3.1, which can also be found in Table 1 of (Herbrich et al., 2007). We only need messages from factor to node. We use $m_{f\to x}$ to denote the message sent from factor f to node x in the *last* iteration, and the current iteration will send $m^{\text{new}}_{f\to x}$ which is what we are computing.

Prior factor

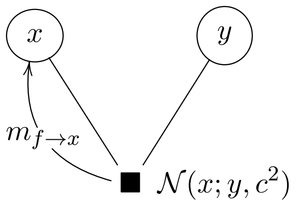


The factor is $f(x) = \mathcal{N}(x; m, v^2)$, and the update equation is

$$\pi^{\text{new}}_x \leftarrow \pi_x + \frac{1}{v^2}, \qquad \tau^{\text{new}}_x \leftarrow \tau_x + \frac{m}{v^2}. \tag{B.4}$$

This update equation should be taken with care. It does *not* mean that if we run EP on the whole graph repeatedly, then the formulae (B.4) should be applied again and again. In fact, that would push the π_x and τ_x to infinity. Because the factor is Gaussian and is attached only to x, the message $m_{f\to x}$ is always exact and equals $\mathcal{N}'(\frac{m}{v^2}, \frac{1}{v^2})$. If $m_{f\to x}$ is initialized to $\mathcal{N}'(0,0)$, then the first message from f to x will update the marginal $p(x)$ according to Eq. (B.4). Afterwards, $p(x)$ will never be changed by $m_{f\to x}$.

Noise factor



Here the factor is $f(x,y) = \exp\left(-(x-y)^2/(2c^2)\right)$. Suppose the message $m_{f\to y} \sim \mathcal{N}'(\tau_{f\to y}, \pi_{f\to y})$ and current marginal $p(y) \sim \mathcal{N}'(\tau_y, \pi_y)$. Then by Eq. (B.3),

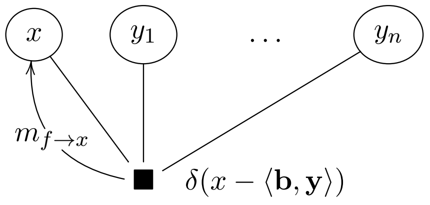
$$m_{y\to f} = \frac{p(y)}{m_{f\to y}(y)} \sim \mathcal{N}'(\tau_y - \tau_{f\to y}, \pi_y - \pi_{f\to y}).$$

So the message $m^{\text{new}}_{f\to x}$ is:

$$\begin{aligned} m^{\text{new}}_{f\to x}(x) &= \int f(x,y) m_{y\to f}(y) \mathrm{d}y \\ &\propto \int \exp\left(-\frac{1}{2c^2}(x-y)^2\right) \exp\left(-\frac{1}{2}(\pi_y - \pi_{f\to y})y^2 + (\tau_y - \tau_{f\to y})y\right) \mathrm{d}y \\ &\propto \exp\left(-\frac{1}{2}\frac{\pi_y - \pi_{f\to y}}{1 + c^2(\pi_y - \pi_{f\to y})}x^2 + \frac{\tau_y - \tau_{f\to y}}{1 + c^2(\pi_y - \pi_{f\to y})}x\right), \end{aligned}$$

i.e., $m^{\text{new}}_{f\to x} \sim \mathcal{N}'(a(\tau_y - \tau_{f\to y}), a(\pi_y - \pi_{f\to y}))$, where $a = \left(1 + c^2(\pi_y - \pi_{f\to y})\right)^{-1}$.

Linear combination factor 1



Here the factor is $f(x, \mathbf{y}) = \delta(x - \mathbf{a}^\top \mathbf{y})$, where δ is the Dirac impulse function. So

$$\begin{aligned} m_{f\to x}(x) &= \int f(x, \mathbf{y}) \prod_{i=1}^{n} m_{y_i \to f}(y_i) \mathrm{d}\mathbf{y} \\ &= \int f(x, \mathbf{y}) \prod_{i=1}^{n} \frac{p(y_i)}{m_{f\to y_i}(y_i)} \mathrm{d}\mathbf{y} \\ &= \int \delta(x - \mathbf{a}^\top \mathbf{y}) \underbrace{\prod_i \mathcal{N}'\left(y_i; \tau_{y_i} - \tau_{f\to y_i}, \pi_{y_i} - \pi_{f\to y_i}\right)}_{:=p(\mathbf{y})} \mathrm{d}\mathbf{y} \end{aligned} \tag{B.5}$$

To proceed, we need a property of the Dirac function:

$$\delta(x) = \lim_{\varepsilon\to 0} f_\varepsilon(x), \qquad \text{where } f_\varepsilon(x) := \begin{cases} \frac{1}{\varepsilon} & x \in [0,\varepsilon] \\ 0 & x \overline{\in} [0,\varepsilon] \end{cases} . \tag{B.6}$$

So

$$\begin{aligned} m_{f\to x}(x) &= \lim_{\varepsilon\to 0} \int f_\varepsilon(x - \mathbf{a}^\top \mathbf{y}) p(\mathbf{y}) \mathrm{d}\mathbf{y} = \lim_{\varepsilon\to 0} \frac{1}{\varepsilon} \Pr_{\mathbf{y}\sim p(\mathbf{y})} \left\{ \mathbf{a}^\top \mathbf{y} - x \in [0,\varepsilon] \right\} \\ &= p_z(x), \qquad \text{where } z := \mathbf{a}^\top \mathbf{y}. \end{aligned}$$

In general, if $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}})$, then

$$z = \mathbf{a}^\top \mathbf{y} \sim \mathcal{N}(\mathbf{a}^\top \mu_{\mathbf{y}}, \mathbf{a}^\top \Sigma_{\mathbf{y}} \mathbf{a}) = \mathcal{N}' \left(\frac{\mathbf{a}^\top \mu_{\mathbf{y}}}{\mathbf{a}^\top \Sigma_{\mathbf{y}} \mathbf{a}}, (\mathbf{a}^\top \Sigma_{\mathbf{y}} \mathbf{a})^{-1} \right).$$

By definition of $p(\mathbf{y})$ in Eq. (B.5), we have

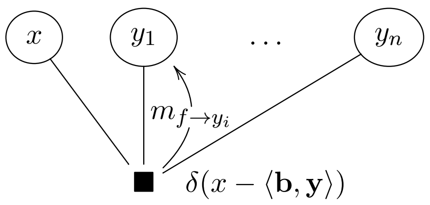
$$\mathbf{y} \sim \mathcal{N}' \left(\operatorname{vec}\left(\frac{\tau_{y_i} - \tau_{f\to y_i}}{\pi_{y_i} - \pi_{f\to y_i}} \right), \quad \operatorname{diag}\left(\frac{1}{\pi_{y_i} - \pi_{f\to y_i}} \right) \right),$$

where $\operatorname{vec}(a_i)$ stands for $(a_1, \ldots, a_n)^\top$ and diag yields a diagonal matrix. We now have

$$m_{f\to x} \sim \mathbf{a}^\top \mathbf{y} \sim \mathcal{N}' \left(\left(\sum_i \frac{a_i^2}{\pi_{y_i} - \pi_{f\to y_i}} \right)^{-1} \sum_i a_i \frac{\tau_{y_i} - \tau_{f\to y_i}}{\pi_{y_i} - \pi_{f\to y_i}}, \quad \left(\sum_i \frac{a_i^2}{\pi_{y_i} - \pi_{f\to y_i}} \right)^{-1} \right).$$

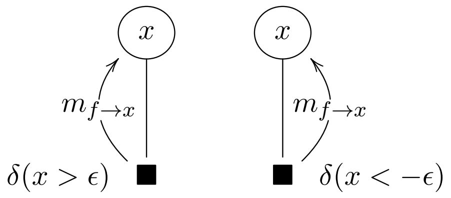
If fact, this result is obvious, because this factor enforces that $x = \mathbf{a}^\top \mathbf{y}$, and hence x should have the same distribution as $\mathbf{a}^\top \mathbf{y}$.

Linear combination factor 2



Here the factor is $f(x, y) = \delta(x = \mathbf{b}^\top \mathbf{y})$. Update equations for this factor can be derived by converting the it into the case in row 3. In row 3, the receiving node (x) appears in the factor $\delta(x = \mathbf{a}^\top \mathbf{y})$ as the linear combination ($\mathbf{a}^\top \mathbf{y}$) of the rest nodes. Here in row 4, the receiving node y_n does not appear in the factor potential as the linear combination of the rest nodes. However, we can re-write the potential $\delta(\mathbf{x} = \mathbf{b}^\top \mathbf{y})$ into $\delta(y_n = \mathbf{a}^\top [y_1, \ldots, y_{n-1}, x]^\top)$, where $\mathbf{a} = b_n^{-1}(-b_1, \ldots, -b_{n-1}, 1)^\top$. If $b_n = 0$, then in fact y_n is not connected to the factor, and hence we can ignore y_n.

Margin comparison factor



Here the factor is $f(x) = \delta(x > \varepsilon)$ and $f(x) = \delta(|x| > \varepsilon)$. Suppose the message $m_{f\to x}$ in the last time step is $\mathcal{N}'(\pi_{f\to x}, \tau_{f\to x})$. Then the product of the latest messages that f has received from other adjacent nodes is $m_{\text{rest}\to f} = \mathcal{N}'(\underbrace{\pi_x - \pi_{f\to x}}_{:=c}, \underbrace{\tau_x - \tau_{f\to x}}_{:=d})$, where τ_x and π_x are the precision-mean and precision of the current marginal $p(x)$. The message $m_{f\to x}$ in the current time step, if calculated by standard belief propagation formula, should be exactly $\delta(x > \varepsilon)$. Hence the new marginal $p^{\text{new}}(x) = m_{\text{rest}\to f}(x)\delta(x > \varepsilon)$. However, in EP we want to approximate it by a Gaussian. In particular, we approximate the new marginal $p^{\text{new}}(x)$ by a Gaussian which preserves the mean and (co)variance[2]. This means we only require the mean and (co)variance of $p^{\text{new}}(x)$, as computed below.

In fact, this is a standard truncated Gaussian. In general, if $X \sim \mathcal{N}(\mu, \sigma^2)$ is restricted into the interval $X \in (a, b)$ where $-\infty \le a < b \le \infty$, then the truncated Gaussian distribution has normalized probability density function (pdf)

$$h(x; \mu, \sigma, a, b) = \frac{\frac{1}{\sigma}\mathcal{N}\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)},$$

where Φ is the cumulative distribution function of the standard Gaussian distribution. Then one has (by http://en.wikipedia.org/wiki/Truncated_normal_distribution):

$$\mathbb{E}[X|a < X < b] = \mu + \sigma \frac{\mathcal{N}\left(\frac{a-\mu}{\sigma}\right) - \mathcal{N}\left(\frac{b-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \tag{B.7a}$$

$$\mathrm{Var}[X|a < X < b] = \sigma^2 \left[1 + \frac{\frac{a-\mu}{\sigma}\mathcal{N}\left(\frac{a-\mu}{\sigma}\right) - \frac{b-\mu}{\sigma}\mathcal{N}\left(\frac{b-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} - \left(\frac{\mathcal{N}\left(\frac{a-\mu}{\sigma}\right) - \mathcal{N}\left(\frac{b-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}\right)^2\right] \tag{B.7b}$$

[2]Notice that row 5 is the only approximate message of EP in the whole graph.

In our case, $\sigma = \frac{1}{\sqrt{c}}$, $\mu = \frac{d}{c}$. If $f = \delta(x > \varepsilon)$, then $a = \varepsilon, b = \infty$. Then

$$\mathbb{E}[X|a < X < b] = \frac{d}{c} + \frac{1}{\sqrt{c}} \frac{\mathcal{N}\left(\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}}\right) - 0}{1 - \Phi\left(\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}}\right)} = \frac{d}{c} + \frac{1}{\sqrt{c}} \frac{\mathcal{N}\left(\frac{d-\varepsilon c}{\sqrt{c}}\right)}{\Phi\left(\frac{d-\varepsilon c}{\sqrt{c}}\right)},$$

$$\begin{aligned}\mathrm{Var}[X|a < X < b] &= \frac{1}{c}\left[1 + \frac{\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}} \mathcal{N}\left(\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}}\right) - 0}{1 - \Phi\left(\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}}\right)} - \left(\frac{\mathcal{N}\left(\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}}\right) - 0}{1 - \Phi\left(\frac{\varepsilon - \frac{d}{c}}{1/\sqrt{c}}\right)}\right)^2\right] \\ &= \frac{1}{c}\left[1 - \frac{\mathcal{N}\left(\frac{d-\varepsilon c}{\sqrt{c}}\right)}{\Phi\left(\frac{d-\varepsilon c}{\sqrt{c}}\right)} \left(\frac{\mathcal{N}\left(\frac{d-\varepsilon c}{\sqrt{c}}\right)}{\Phi\left(\frac{d-\varepsilon c}{\sqrt{c}}\right)} + \frac{d - \varepsilon c}{\sqrt{c}}\right)\right].\end{aligned}$$

By introducing functions

$$V_{\delta(\cdot > \varepsilon)}(t, \varepsilon) := \frac{\mathcal{N}(t - \varepsilon)}{\Phi(t - \varepsilon)}, \quad W_{\delta(\cdot > \varepsilon)}(t, \varepsilon) := V_{\delta(\cdot > \varepsilon)}(t, \varepsilon)(V_{\delta(\cdot > \varepsilon)}(t, \varepsilon) + t - \varepsilon),$$

we arrive at

$$\begin{aligned}\frac{\tau^{\mathrm{new}}}{\pi^{\mathrm{new}}} &= \mathbb{E}[X|a < X < b] = \frac{d}{c} + \frac{1}{\sqrt{c}} V_{\delta(\cdot > \varepsilon)}\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{c}\right), \\ \frac{1}{\pi^{\mathrm{new}}} &= \mathrm{Var}[X|a < X < b] = \frac{1}{c}\left(1 - W_{\delta(\cdot > \varepsilon)}\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{c}\right)\right).\end{aligned}$$

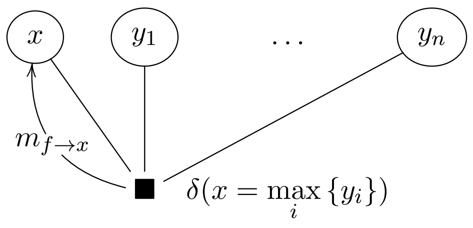
The case of $f(x) = \delta(x < \varepsilon)$ can be dealt with similarly by using Eq. (B.7).

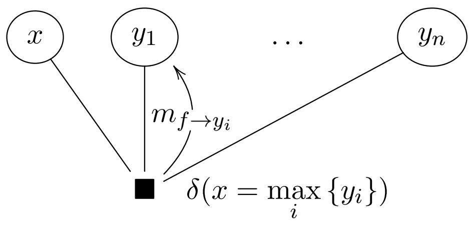
B.3 Message passing for the max factor

There turns out no closed form formula for the messages related to the max factor. We will first give the mathematical form of the messages, and then show how it can be reduced to the cumulative distribution functions of multi-variate Gaussians. Finally, we provide an efficient approximate algorithm.

B.3.1 Mathematical formulation of messages

Suppose the factor is $f(x, y_1, \ldots, y_n) := \delta(x = \max_i \{y_i\})$. We are interested in the update formulae for this max node: $m_{f \to x}$ and $m_{f \to y_i}$. Similar to the case of linear combination, we proceed as follows:





$$\begin{aligned} m_{f \to x}(x) &= \int f(x, \mathbf{y}) \prod_{i=1}^{n} m_{y_i \to f}(y_i) \mathrm{d}\mathbf{y} = \int f(x, \mathbf{y}) \prod_{i=1}^{n} \frac{p(y_i)}{m_{f \to y_i}(y_i)} \mathrm{d}\mathbf{y} \\ &= \int \delta(x = \max\{y_i\}) \underbrace{\prod_i \mathcal{N}'(y_i; \tau_{y_i} - \tau_{f \to y_i}, \pi_{y_i} - \pi_{f \to y_i})}_{:= p(\mathbf{y})} \mathrm{d}\mathbf{y} \end{aligned} \tag{B.8}$$

Again using the definition in Eq. (B.6), we have

$$\begin{aligned} m_{f \to x}(x) &= \lim_{\varepsilon \to 0} \int f_\varepsilon(x - \max\{y_i\}) p(\mathbf{y}) \mathrm{d}\mathbf{y} \\ &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \Pr_{\mathbf{y} \sim p(\mathbf{y})} \{\max\{y_i\} - x \in [0, \varepsilon]\} \\ &= p_z(x), \qquad \text{where } z := \max\{y_i\}. \end{aligned}$$

So the message $m_{f \to x}$ is exactly the distribution of the max of multiple Gaussians $\{y_i\}$, where $y_i \sim \mathcal{N}'(y_i; \tau_{y_i} - \tau_{f \to y_i}, \pi_{y_i} - \pi_{f \to y_i})$.

Finally, we compute $m_{f \to y_1}$. Different from factor $\delta(x = \mathbf{a}^\top \mathbf{y})$, we cannot convert the derivation of $m_{f \to y_1}$ into $m_{f \to x}$. First similar to Eq. (B.8), for a fixed value of y_1 we have:

$$\begin{aligned} m_{f \to y_1}(y_1) &= \int f(x, \mathbf{y}) \underbrace{\prod_{i=2}^{n} \overbrace{m_{y_i \to f}(y_i)}^{:= p_i(y_i)}}_{p_{2:n}(\mathbf{y}_{2:n})} \overbrace{m_{f \to x}(x)}^{p_x(x)} \mathrm{d}y_2 \dots \mathrm{d}y_n \mathrm{d}x \\ &= \int \delta\left(x - \max_{i=1\dots n} \{y_i\}\right) p_{2:n}(\mathbf{y}_{2:n}) p_x(x) \mathrm{d}y_2 \dots \mathrm{d}y_n \mathrm{d}x \end{aligned} \tag{B.9}$$

Using the definition in Eq. (B.6), we have

$$\begin{aligned} m_{f\to y_1}(y_1) &= \lim_{\varepsilon\to 0} \int f_\varepsilon \left(x - \max_{i=1\dots n}\{y_i\}\right) p_{2:n}(\mathbf{y}_{2:n}) p_x(x) \mathrm{d}y_2 \dots \mathrm{d}y_n \mathrm{d}x \\ &= \lim_{\varepsilon\to 0} \frac{1}{\varepsilon} \Pr_{\mathbf{y}_{2:n}\sim p_{2:n}, x\sim p_x} \left\{ \max\left\{ y_1, \max_{i=2,\dots,n} y_i \right\} - x \in [0,\varepsilon] \right\}. \end{aligned}$$

Here $y_2, \dots, y_n$ are involved only in terms of $\max_{i=2,\dots,n} y_i$. So we can treat them in whole by introducing $z := \max_{i=2\dots n} y_i$ and denoting its distribution as p_z, which is again the max of $n-1$ Gaussians. We now study $\Pr_{z\sim p_z, x\sim p_x}\{\max\{y_1, z\} - x \in [0,\varepsilon]\}$ in more detail.

$$\begin{aligned} &\Pr_{z\sim p_z, x\sim p_x} \{\max\{y_1,z\} - x \in [0,\varepsilon]\} \\ &= \Pr(\max\{y_1,z\} \le x+\varepsilon) - \Pr(\max\{y_1,z\} \le x) \\ &= \Pr\{z \le y_1 \text{ and } y_1 \le x+\varepsilon\} + \Pr\{z \ge y_1 \text{ and } z \le x+\varepsilon\} \\ &\quad - (\Pr\{z \le y_1 \text{ and } y_1 \le x\} + \Pr\{z \ge y_1 \text{ and } z \le x\}) \\ &\overset{*}{=} \Pr_z(z \le y_1) \Pr_x(y_1 \le x+\varepsilon) + \int_{y_1}^{+\infty} \Pr_x(x \ge z-\varepsilon) p_z(z) \mathrm{d}z \\ &\quad - \Pr_z(z \le y_1) \Pr_x(y_1 \le x) - \int_{y_1}^{+\infty} \Pr_x(z \le x) p_z(z) \mathrm{d}z \\ &= \Pr_z(z \le y_1) \Pr_x(y_1 - \varepsilon \le x \le y_1) + \int_{y_1}^{+\infty} \Pr_x(z-\varepsilon \le x \le z) p_z(z) \mathrm{d}z \end{aligned}$$

where the step (*) makes use of the independence between x and z (do not get confused by $x = \max\{y_1, z\}$). So now

$$\begin{aligned} m_{f\to y_1}(y_1) &= \lim_{\varepsilon\to 0} \frac{1}{\varepsilon} \Pr_{z\sim p_z, x\sim p_x} \{\max\{y_1,z\} - x \in [0,\varepsilon]\} \\ &= \Pr_z(z \le y_1) \lim_{\varepsilon\to 0} \frac{\Pr_x(y_1 - \varepsilon \le x \le y_1)}{\varepsilon} + \int_{y_1}^{+\infty} p_z(z) \lim_{\varepsilon\to 0} \frac{\Pr_x(z-\varepsilon \le x \le z)}{\varepsilon} \mathrm{d}z \\ &= \Pr_z(z \le y_1) p_x(y_1) + \int_{y_1}^{+\infty} p_z(z) p_x(z) \mathrm{d}z, \end{aligned}$$

which makes a lot of sense. $m_{f\to y_1}$ involves the cumulative distribution function (cdf) of z, the max of $n-1$ independent Gaussians. If $p_z(z)$ is approximated by a Gaussian via moment matching up to the second order, then the second term (integral) is just the cdf of a one dimensional Gaussian.

The mean and variance of $m_{f\to y_1}$ are not trivial. Even after $p_z(z)$ is approximated by a Gaussian, they are still not trivial. In the next subsection, we will discuss how to compute the mean and variance of the maximum of multiple Gaussians.

B.3.2 Moments of the maximum of multiple Gaussians

In this subsection, we show how the mean and variance of the maximum of multiple Gaussians can be expressed in terms of the cumulative distribution functions (cdf) of multi-variable Gaussians (MVGs). This result is completely from (Afonja, 1972). We fixed some typos and polished the symbols. In the whole subsection, we use $i, j \neq p, q, r$ to represent that $i \neq p$, $i \neq q$, $i \neq r$, $j \neq p$, $j \neq q$, and $j \neq r$.

Max of Two Gaussians

In the simplest case of only two Gaussians $x_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ with correlation ρ, we have an exact expression of the pdf of $x = \max\{x_1, x_2\}$: $p(x) = m_1(-x) + m_2(-x)$, where

$$f_1(x) = \frac{1}{\sigma_1}\phi\left(\frac{x+\mu_1}{\sigma_1}\right) \times \Phi\left(\frac{\rho(x+\mu_1)}{\sigma_1\sqrt{1-\rho^2}} - \frac{x+\mu_2}{\sigma_2\sqrt{1-\rho^2}}\right)$$

$$f_2(x) = \frac{1}{\sigma_2}\phi\left(\frac{x+\mu_2}{\sigma_2}\right) \times \Phi\left(\frac{\rho(x+\mu_2)}{\sigma_2\sqrt{1-\rho^2}} - \frac{x+\mu_1}{\sigma_1\sqrt{1-\rho^2}}\right),$$

and $\phi(\cdot)$ and $\Phi(\cdot)$ are, respectively, the pdf and cdf of the standard normal distribution. The moments can also be easily derived:

$$\mathbb{E}[x] = \mu_1\Phi\left(\frac{\mu_1-\mu_2}{\theta}\right) + \mu_2\Phi\left(\frac{\mu_2-\mu_1}{\theta}\right) + \theta\phi\left(\frac{\mu_1-\mu_2}{\theta}\right)$$

$$\mathbb{E}\left[x^2\right] = (\sigma_1^2+\mu_1^2)\Phi\left(\frac{\mu_1-\mu_2}{\theta}\right) + (\sigma_2^2+\mu_2^2)\Phi\left(\frac{\mu_2-\mu_1}{\theta}\right) + (\mu_1+\mu_2)\theta\phi\left(\frac{\mu_1-\mu_2}{\theta}\right),$$

where $\theta := \sqrt{\sigma_1^2+\sigma_2^2-2\rho\sigma_1\sigma_2}$.

Recursively raply the max of two Gaussians

It is obvious that the max of two Gaussians is not Gaussian. But knowing its mean and variance allows us to approximate it by a Gaussian with the same mean and variance. This also allows us to recursively apply the procedure of taking the max of two Gaussians. For example, Figure B.1 uses a chain style where $t_1 = \max\{x_1, x_2\}$, and $t_i = \max\{t_{i-1}, x_{i+1}\}$ for $i = 2, 3, \ldots, n-1$.

Figure B.2 uses a tree structure for recursion. Intuitively, the topology in Figure B.2 may deliver lower approximation error than Figure B.1 because the highest layer of Figure B.2 introduce independent noises while the error of t_1 in Figure B.1 will be propagated through the whole chain. This intuition is confirmed by (Sinha et al., 2006). Finally, neither of the approaches makes use of the correlation of the n Gaussians, so they work only for independent Gaussians.

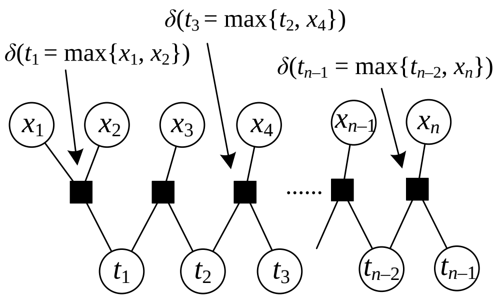


Figure B.1: Recursively apply the max of two Gaussians in a chain structure to compute the max of n Gaussians.

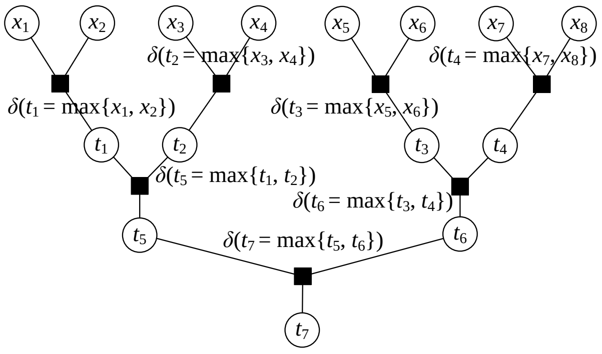


Figure B.2: Recursively apply the max of two Gaussians in a tree structure to compute the max of n Gaussians.

General results for correlated multi-dimensional random variables

Suppose a k-dimensional multivariate random variable $\mathbf{x}$ has p.d.f. $\phi_k(\mathbf{x};\boldsymbol{\theta},\Sigma)$ (not necessarily Gaussian), where $\boldsymbol{\theta}=\mathbb{E}[\mathbf{x}]$ is the mean and $\Sigma=\{\sigma_{ij}\}=\{\mathrm{Cov}(x_i x_j)\}$ is the covariance matrix. Notice this notation does not imply that $\mathbf{x}$ must be a multivariate *Gaussian* distribution with mean $\boldsymbol{\theta}$ and covariance matrix Σ. To avoid introducing more symbols, we denote $\phi_k(\mathbf{z};\mathbf{R})$ as the standardized form of $\phi_k(\mathbf{x};\boldsymbol{\theta},\Sigma)$ with $z_i=\frac{x_i-\theta_i}{\sigma_i}$ and $\sigma_i=\sqrt{\sigma_{ii}}$, *i.e.*, only translation and scaling are applied, with no rotation. Furthermore, we define $\Phi_k(\mathbf{b};\mathbf{R}):=\int_{b_1}^{\infty}\dots\int_{b_2}^{\infty}\phi_k(\mathbf{z};\mathbf{R})\mathrm{d}\mathbf{z}=\int_{\mathbf{b}}^{\infty}\phi_k(\mathbf{z};\mathbf{R})\mathrm{d}\mathbf{z}$. See the last part of this appendix section for a summary of the symbols.

Denote $y:=\max\{x_1,\dots,x_k\}$. The general results is:

Theorem 75 (Theorem 1 of (Afonja, 1972)) *The r-th moment of y about the origin is given by*

$$\mu'_r(y)=\sum_{i=1}^{k}\int_{\mathbf{a}_i^+}(\theta_i+\sigma_i z_i)^r\phi_k(\mathbf{z};\mathbf{R}_i^+)\mathrm{d}\mathbf{z}=\sum_{i=1}^{k}\sum_{j=1}^{r}\binom{r}{j}\theta_i^{r-j}\sigma_i^j\mu_j(z_i) \tag{B.10}$$

where

$$\begin{aligned}
\mu_j(z_i) &= \int_{\mathbf{a}_i^+}^{\infty} z_i^j\phi_k(\mathbf{z};\mathbf{R}_i^+)\mathrm{d}\mathbf{z}\\
\mathbf{a}_i^+ &= (a_i(1),\dots,a_i(k)) =: \{a_i(j)\}_{j=1}^k\in\mathbb{R}^k\\
a_i(j) &= \begin{cases}\frac{\theta_j-\theta_i}{\sqrt{\mathrm{Var}(x_i-x_j)}} & \text{if } j\neq i\\ -\infty & \text{if } j=i\end{cases}\\
\mathbf{R}_i^+ &= \{r_i(s,t)\}_{s,t=1}^k\in\mathbb{R}^{k\times k}\\
r_i(s,t) &= \begin{cases}\mathrm{Corr}(x_i-x_s,x_i-x_t) & \text{if } s,t\neq i\\ \mathrm{Corr}(x_i,x_i-x_s) \text{ or } \mathrm{Corr}(x_i,x_i-x_t) & \text{if } s\neq i=t \text{ or } t\neq i=s \text{ resp.}\\ 1 & \text{if } s=t=i\end{cases}
\end{aligned}$$

Note we have changed the notation from (Afonja, 1972), in order to make the subscript indices clearer. The notations in the original paper are extremely hairy. For clarity, we write all vectors or matrices in boldface. All variables denoted with (), *e.g.*, $a_i(j)$, $r_{i,j}(p,q)$ are real values, and those without are vectors or matrices except for simple ground symbols like θ_i, σ_{ij}. Vectors and matrices are denoted by removing () from their elements, *e.g.*, $\mathbf{R}_{i,j}$ is a matrix made up of $\{r_{i,j}(s,t)\}_{s,t\neq i,j}$. We also try not to use superscript in order to avoid confusion with exponents unless double indices are used like $r_i^{s,t}(p,q)$.

We will focus on the Gaussian distributions of ϕ_k, and compute the 1^{st} and 2^{nd} moments of $y = \max\{x_i\}$. Attention will be paid to special cases like: equal mean of marginals of x_i, equal variance and covariance, and independent $\{x_i\}$ which is of our particular interest.

First order moments of the max of Gaussians

We first derive the general formula for the first order moment (mean) of $y = \max\{x_i\}$ where $\mathbf{x}$ is a multivariate normal distribution.

$$\mu_1'(y) = \sum_{i=1}^{k} \theta_i \Phi_{k-1}(\mathbf{a}_i; \mathbf{R}_i) + \sum_{i=1}^{k} \sum_{j\neq i} \frac{\sigma_{ii} - \sigma_{ij}}{\sqrt{\sigma_{ii} + \sigma_{jj} - 2\sigma_{ij}}} \phi_1(a_i(j)) \Phi_{k-2}(\boldsymbol{\alpha}_{i,j}; \mathbf{R}_{i,j}) \tag{B.11}$$

where

$$\begin{array}{rcll}
\mathbf{a}_i & = & \{a_i(j)\}_{j\neq i} \in \mathbb{R}^{k-1} & \\
\mathbf{R}_i & = & \{r_i(s,t)\}_{s,t\neq i} \in \mathbb{R}^{(k-1)\times(k-1)} & \\
\boldsymbol{\alpha}_{i,j} & = & \{\alpha_{i,j}(t)\}_{t\neq i,j} \in \mathbb{R}^{k-2} & \\
\mathbf{R}_{i,j} & = & \{r_{i,j}(s,t)\}_{s,t\neq i,j} \in \mathbb{R}^{(k-2)\times(k-2)} & \\
a_i(j) & = & \frac{\theta_j - \theta_i}{\sqrt{\mathrm{Var}(x_i - x_j)}} & i \neq j \text{ by definition} \\
r_i(s,t) & = & \mathrm{Corr}(x_i - x_s, x_i - x_t) & s,t \neq i \text{ by definition} \\
r_{i,j}(s,t) & = & \text{partial correlation between } x_i - x_s \text{ and } x_i - x_t \text{ given } x_i - x_j & \\
\alpha_{i,j}(t) & = & \frac{a_i(t) - a_i(j)\cdot r_i(j,t)}{\sqrt{1 - r_i(j,t)^2}} & i \neq j \neq t \neq i \text{ by definition} \\
\phi_1(a) & = & \text{standard normal p.d.f. evaluated at } a & \\
\Phi_0(\cdot;\cdot) & = & 1. &
\end{array}$$

A short note on partial correlation. Given random variables X, Y and set of random variables $\mathbf{Z}^n := \{Z_1, \ldots, Z_n\}$, the ($n$-th order) partial correlation between X and Y given $\mathbf{Z}$ can be computed recursively based on three $(n-1)$-th order partial correlations:

$$\rho_{XY|\mathbf{Z}} = \frac{\rho_{XY|\mathbf{Z}^{n-1}} - \rho_{XZ_n|\mathbf{Z}^{n-1}} \rho_{YZ_n|\mathbf{Z}^{n-1}}}{\sqrt{1 - \rho^2_{XZ_n|\mathbf{Z}^{n-1}}} \sqrt{1 - \rho^2_{YZ_n|\mathbf{Z}^{n-1}}}} \tag{B.12}$$

where $\mathbf{Z}^{n-1} := \{Z_1, \ldots, Z_{n-1}\}$, and the base of recursion is that $\rho_{XY|\emptyset} := \rho_{XY}$ which is the regular correlation between X and Y (we also denoted it as $\mathrm{Corr}(X, Y)$). Naively applying Eq. (B.12) costs exponential time complexity in n (order), and dynamic programming can reduce the complexity to $O(n^3)$. However, our problem only involves $n = 2$, so we just implemented the recursive formula.

Special case 1: equivariance and equicorrelated case Assuming

$$\sigma_{ii} = \sigma^2 \qquad \text{and} \qquad \sigma_{ij} = \sigma^2\rho \ (i \neq j),$$

then we have

$$\begin{aligned}
a_i(j) &= \frac{\theta_j - \theta_i}{\sigma\sqrt{2(1-\rho)}} \\
r_i(s,t) &= 1/2 \\
r_{i,j}(s,t) &= 1/3 \\
\alpha_{i,j}(t) &= \frac{2\theta_t - \theta_i - \theta_j}{\sigma\sqrt{6(1-\rho)}}.
\end{aligned}$$

The derivations are simple:

$$\begin{aligned}
a_i(j) &= \frac{\theta_j - \theta_i}{\sqrt{\mathrm{Var}(x_i - x_j)}} = \frac{\theta_j - \theta_i}{\sqrt{2\sigma^2 - 2\sigma^2\rho}} = \frac{\theta_j - \theta_i}{\sigma\sqrt{2(1-\rho)}} \\
r_i(s,t) &= \mathrm{Corr}(x_i - x_s, x_i - x_t) = \frac{\mathrm{Cov}(x_i - x_s, x_i - x_t)}{\sqrt{\mathrm{Var}(x_i - x_s)}\sqrt{\mathrm{Var}(x_i - x_t)}} \\
&= \frac{\sigma^2 - 2\sigma^2\rho + \sigma^2\rho}{\sqrt{2\sigma^2 - 2\sigma^2\rho}\sqrt{2\sigma^2 - 2\sigma^2\rho}} = \frac{1}{2} \\
\alpha_{i,j}(t) &= \frac{\frac{\theta_t - \theta_i}{\sigma\sqrt{2(1-\rho)}} - \frac{\theta_j - \theta_i}{\sigma\sqrt{2(1-\rho)}}\frac{1}{2}}{\sqrt{1 - 1/4}} = \frac{2\theta_t - \theta_i - \theta_j}{\sigma\sqrt{6(1-\rho)}}.
\end{aligned}$$

Denoting $A := x_i - x_s$, $B := x_i - x_t$, and $C := x_i - x_j$, by Eq. (B.12) we have:

$$r_{i,j}(s,t) = \frac{\rho_{AB} - \rho_{AC}\rho_{BC}}{\sqrt{1 - \rho_{AC}^2}\sqrt{1 - \rho_{BC}^2}} = \frac{\frac{1}{2} - \frac{1}{2}\frac{1}{2}}{1 - \frac{1}{2}\frac{1}{2}} = \frac{1}{3}. \tag{B.13}$$

Special case 2: equal means Suppose $\theta_i = \theta$. In this case, $a_i(j) = 0$ and $\alpha_{i,j}(t) = 0$. Using Eq. (B.11), we have

$$\mu_1'(y) = \theta + \sum_{i=1}^{k} \sum_{j \neq i} \frac{\sigma_{ii} - \sigma_{ij}}{\sqrt{\sigma_{ii} + \sigma_{jj} - 2\sigma_{ij}}} (2\pi)^{-1/2} \Phi_{k-2}(\mathbf{0}; \mathbf{R}_{i,j}),$$

where $\sum_{i=1}^{k} \Phi_{k-1}(\mathbf{0}; \mathbf{R}_i) = 1$ is from the main theorem in Eq. (B.10) setting $r = 0$.

Special case 3: $\{x_i\}$ are independent (covariance $\sigma_{ij} = 0$ for $i \neq j$) In this case $\sigma_{ij} = 0$ for $i \neq j$. Then

$$a_i(j) = \frac{\theta_j - \theta_i}{\sqrt{\mathrm{Var}(x_i - x_j)}} = \frac{\theta_j - \theta_i}{\sqrt{\sigma_i^2 + \sigma_j^2}},$$

$$r_i(s,t) = \mathrm{Corr}(x_i - x_s, x_i - x_t) = \frac{\mathrm{Cov}(x_i - x_s, x_i - x_t)}{\sqrt{\mathrm{Var}(x_i - x_s)}\sqrt{\mathrm{Var}(x_i - x_t)}} = \frac{\sigma_i^2}{\sqrt{\sigma_i^2 + \sigma_s^2}\sqrt{\sigma_i^2 + \sigma_t^2}}.$$

Using same notation in Eq. (B.13), we can compute $r_{i,j}(s,t)$ with

$$\rho_{AB} = \frac{\sigma_i^2}{\sqrt{\sigma_i^2 + \sigma_s^2}\sqrt{\sigma_i^2 + \sigma_t^2}}, \quad \rho_{AC} = \frac{\sigma_i^2}{\sqrt{\sigma_i^2 + \sigma_s^2}\sqrt{\sigma_i^2 + \sigma_j^2}}, \quad \rho_{BC} = \frac{\sigma_i^2}{\sqrt{\sigma_i^2 + \sigma_t^2}\sqrt{\sigma_i^2 + \sigma_j^2}}.$$

And the rest $\alpha_{i,j}(t)$ does not admit any simple form.

Second order moments of the max of Gaussians

Finally, the second order moment of $y = \max\{x_i\}$ where $\mathbf{x}$ is multivariate Gaussian is

$$\begin{aligned}
\mu_2'(y) = &\sum_{i=1}^{k} \sigma_i \sum_{j \neq i} r_i(j,i) \left\{\theta_i \phi_1(a_i(j)) + \sigma_i a_i(j) \cdot r_i(j,i)\right\} \Phi_{k-2}(\boldsymbol{\alpha}_{i,j}; \mathbf{R}_{i,j}) \qquad \text{(B.14)} \\
&+ \sum_{i=1}^{k} \sigma_i^2 \sum_{s \neq i} r_i(s,i) \sum_{t \neq i,s} \left\{ \phi_2 \left(\begin{pmatrix} a_i(s) \\ a_i(t) \end{pmatrix}; r_i(s,t) \right) \right. \\
&\left. \times \Phi_{k-3}\left(\boldsymbol{\alpha}_i^{s,t}; \mathbf{R}_i^{s,t}\right) \left[r_i(t,i) - r_i(s,t) \cdot r_i(s,i)\right] \right\}
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{\alpha}_i^{s,t} &= \left\{\alpha_i^{s,t}(p)\right\}_{p \neq i,s,t} \in \mathbb{R}^{k-3} && i,s,t \text{ mutually different by def} \\
\mathbf{R}_i^{s,t} &= \left\{r_i^{s,t}(p,q)\right\}_{p,q \neq i,s,t} \in \mathbb{R}^{(k-3)\times(k-3)} && i,s,t \text{ mutually different by def} \\
r_i^{s,t}(p,q) &= \text{partial correlation between } x_i - x_p \text{ and } x_i - x_q \text{ given } x_i - x_s \text{ and } x_i - x_t \\
\alpha_i^{s,t}(p) &= \tfrac{a_i(p) - \beta_{i,t}(p,s) a_i(s) - \beta_{i,s}(p,t) a_i(t)}{\sqrt{1 - r_i(p,s)^2}\sqrt{1 - r_{i,s}(p,t)^2}} && i,s,t,p \text{ mutually different by def} \\
\beta_{i,j}(s,t) &= \tfrac{r_i(s,t) - r_i(s,j) r_i(t,j)}{1 - r_i(t,j)^2} && i,j,s,t \text{ mutually different by def} \\
\Phi_{-1}(\cdot;\cdot) &= 0 && \text{if } k = 2.
\end{aligned}$$

Special case 1: $\{x_i\}$ are independent (covariance $\sigma_{ij} = 0$ for $i \neq j$) It only affects $r_i^{s,t}(p,q)$. The only convenience provided by independence is that $\mathrm{Corr}(x_i -$

$x_s, x_i - x_t) = \frac{\sigma_i^2}{\sqrt{\sigma_i^2+\sigma_s^2}\sqrt{\sigma_i^2+\sigma_t^2}}$. The rest terms still do not admit simple forms.

Summary of symbols

In Figure B.3 we summarize the relationship among symbols which is helpful for implementation. Table B.1 summarizes the correspondence between symbols in this report and in (Afonja, 1972). Finally, we list all the symbols used in this section of appendix.

$$\begin{tikzcd}
r_i^{s,t}(p,q) \arrow[dr] & & r_i(s,t) \arrow[dl] \arrow[dr] \arrow[ddr] & & a_i(j) \arrow[dl] \arrow[ddl] & r_{i,j}(s,t) \arrow[ddll] \\
& \beta_{i,j}(s,t) \arrow[drr] & & \alpha_{i,j}(t) & & \\
& & & \alpha_i^{s,t}(p) & &
\end{tikzcd}$$

Figure B.3: Relationship among symbols. Those without inbound arrows can be computed directly from $\boldsymbol{\theta}$ and Σ.

(Afonja, 1972)		here with original indices	here with new indices
$a_{i,j}$	$\Leftrightarrow$	$a_i(j)$	$a_i(j)$
$\alpha_{i,jj'}$	$\Leftrightarrow$	$\alpha_{i,j}(j')$	$\alpha_{i,j}(t)$
$\alpha_{i,jj'j''}$	$\Leftrightarrow$	$\alpha_i^{j,j'}(j'')$	$\alpha_i^{s,t}(p)$
$\beta_{i,qs.j}$	$\Leftrightarrow$	$\beta_{i,j}(q,s)$	$\beta_{i,j}(s,t)$
$r_{i,qs.j}$	$\Leftrightarrow$	$r_{i,j}(q,s)$	$r_{i,j}(s,t)$
$r_{i,jj'}$	$\Leftrightarrow$	$r_i(j,j')$	$r_i(s,t)$
$r_{i,qs.jj'}$	$\Leftrightarrow$	$r_i^{j,j'}(q,s)$	$r_i^{s,t}(p,q)$

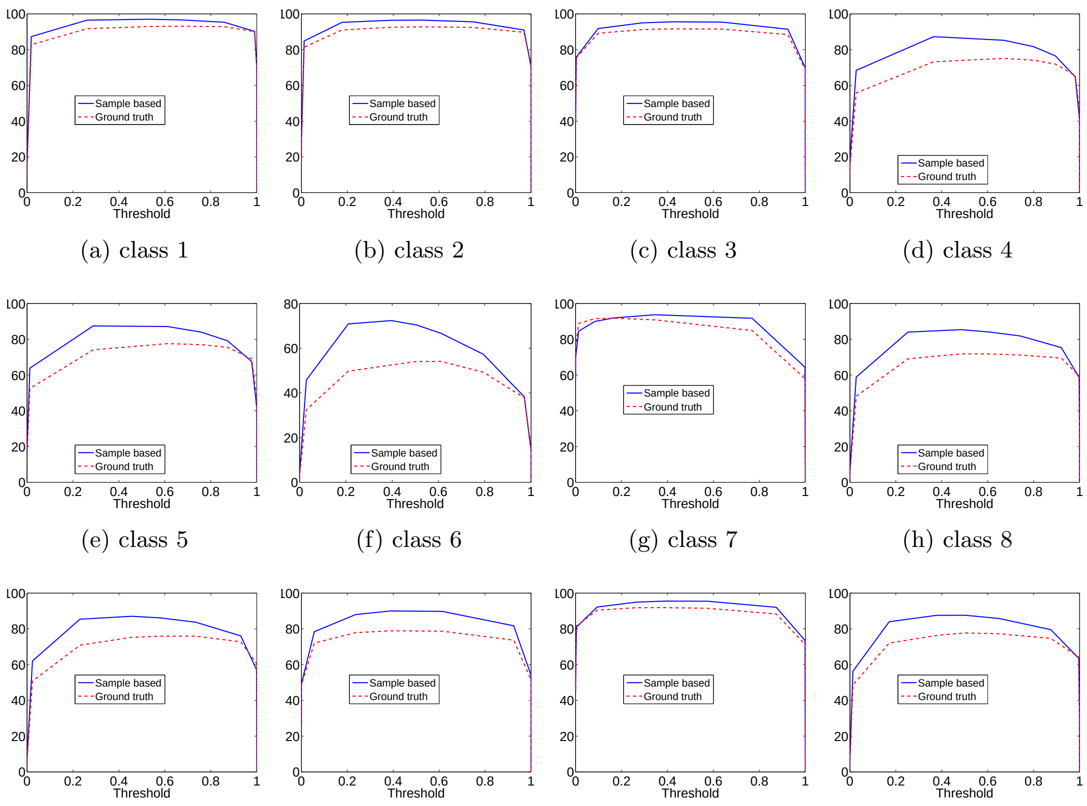
Table B.1: Correspondence Between Symbols in This Report and in (Afonja, 1972).

List of symbols

$\phi_k(\mathbf{x};\boldsymbol{\theta},\Sigma)$	$=$	p.d.f. of k-dimensional multivariate random variable $\mathbf{x}$	
$\phi_k(\mathbf{z},\mathbf{R})$	$=$	standardized form of $\phi_k(\mathbf{x},\boldsymbol{\theta},\Sigma)$	
$\Phi_k(b;\mathbf{R})$	$=$	$\int_{b_1}^{\infty}\dots\int_{b_2}^{\infty}\phi_k(\mathbf{z};\mathbf{R})\mathrm{d}\mathbf{z}=\int_{\mathbf{b}}^{\infty}\phi_k(\mathbf{z};\mathbf{R})\mathrm{d}\mathbf{z}$	
y	$=$	$\max\{x_1,\dots,x_k\}$	
$\mu_r'(y)$	$=$	r-th moment of y (about the origin)	
$\mu_j(z_i)$	$=$	$\int_{\mathbf{a}_i^+}^{\infty} z_i^j\phi_k(\mathbf{z};\mathbf{R}_i^+)\mathrm{d}\mathbf{z}$	
$\mathbf{a}_i^+$	$=$	$(a_i(1),\dots,a_i(k)) =: \{a_i(j)\}_{j=1}^k\in\mathbb{R}^k$	
$a_i(j)$	$=$	$\begin{cases}\frac{\theta_j-\theta_i}{\sqrt{\mathrm{Var}(x_i-x_j)}} & \text{if } j\neq i\\ -\infty & \text{if } j=i\end{cases}$	
$\mathbf{R}_i^+$	$=$	$\{r_i(s,t)\}_{s,t=1}^k\in\mathbb{R}^{k\times k}$	
$r_i(s,t)$	$=$	$\begin{cases}\mathrm{Corr}(x_i-x_s,x_i-x_t) & \text{if } s,t\neq i\\ \mathrm{Corr}(x_i,x_i-x_s)\text{ or }\mathrm{Corr}(x_i,x_i-x_t) & \text{if } s\neq i=t\text{ or }t\neq i=s\text{ resp.}\\ 1 & \text{if } s=t=i\end{cases}$	
$\mathbf{a}_i$	$=$	$\{a_i(j)\}_{j\neq i}\in\mathbb{R}^{k-1}$	
$\mathbf{R}_i$	$=$	$\{r_i(s,t)\}_{s,t\neq i}\in\mathbb{R}^{(k-1)\times(k-1)}$	
$\boldsymbol{\alpha}_{i,j}$	$=$	$\{\alpha_{i,j}(t)\}_{t\neq i,j}\in\mathbb{R}^{k-2}$	
$\mathbf{R}_{i,j}$	$=$	$\{r_{i,j}(s,t)\}_{s,t\neq i,j}\in\mathbb{R}^{(k-2)\times(k-2)}$	
$a_i(j)$	$=$	$\frac{\theta_j-\theta_i}{\sqrt{\mathrm{Var}(x_i-x_j)}}$	$i\neq j$ by definition
$r_i(s,t)$	$=$	$\mathrm{Corr}(x_i-x_s,x_i-x_t)$	$s,t\neq i$ by definition
$r_{i,j}(s,t)$	$=$	partial correlation between x_i-x_s and x_i-x_t given x_i-x_j	
$\alpha_{i,j}(t)$	$=$	$\frac{a_i(t)-a_i(j)\cdot r_i(j,t)}{\sqrt{1-r_i(j,t)^2}}$	$i\neq j\neq t\neq i$ by definition
$\phi_1(a)$	$=$	standard normal p.d.f. evaluated at a	
$\Phi_0(\cdot;\cdot)$	$=$	$1.$	
$\boldsymbol{\alpha}_i^{s,t}$	$=$	$\left\{\alpha_i^{s,t}(p)\right\}_{p\neq i,s,t}\in\mathbb{R}^{k-3}$	i,s,t mutually different by def
$\mathbf{R}_i^{s,t}$	$=$	$\left\{r_i^{s,t}(p,q)\right\}_{p,q\neq i,s,t}\in\mathbb{R}^{(k-3)\times(k-3)}$	i,s,t mutually different by def
$r_i^{s,t}(p,q)$	$=$	partial correlation between x_i-x_p and x_i-x_q given x_i-x_s and x_i-x_t	
$\alpha_i^{s,t}(p)$	$=$	$\frac{a_i(p)-\beta_{i,t}(p,s)a_i(s)-\beta_{i,s}(p,t)a_i(t)}{\sqrt{1-r_i(p,s)^2}\sqrt{1-r_{i,s}(p,t)^2}}$	i,s,t,p mutually different by def
$\beta_{i,j}(s,t)$	$=$	$\frac{r_i(s,t)-r_i(s,j)r_i(t,j)}{1-r_i(t,j)^2}$	i,j,s,t mutually different by def
$\Phi_{-1}(\cdot;\cdot)$	$=$	0	if $k=2$.

Detailed Result of Empirical Optimal Threshold

In Section 3.3.3, we showed six examples of how $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ vary as functions of θ. In this appendix, we show the same result for all the 101 classes in the group `topics`.

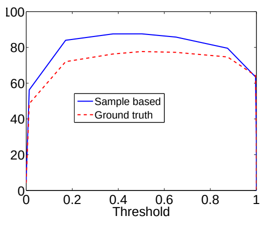


(a) class 1 (b) class 2 (c) class 3 (d) class 4

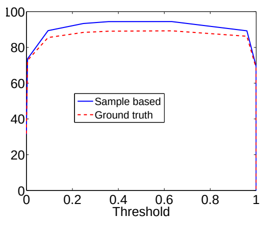
(e) class 5 (f) class 6 (g) class 7 (h) class 8

(i) class 9 (j) class 10 (k) class 11 (l) class 12

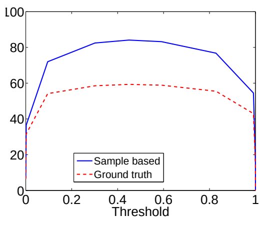
Figure C.1: Example curves of $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ (blue) and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (red) v.s. θ.



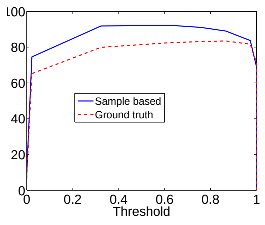
(a) class 13



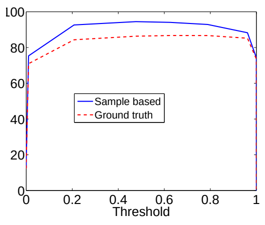
(b) class 14



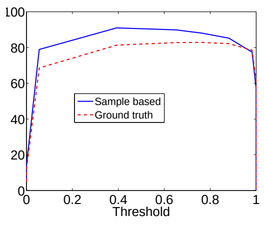
(c) class 15



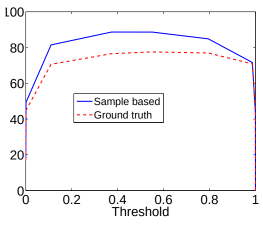
(d) class 16



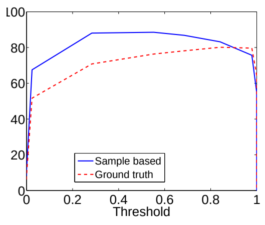
(e) class 17



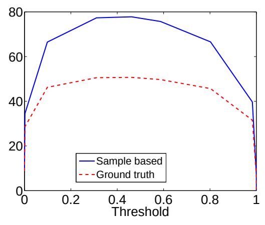
(f) class 18



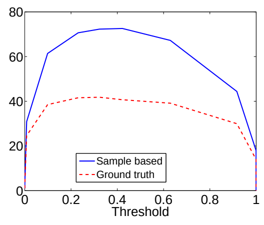
(g) class 19



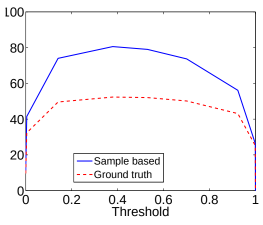
(h) class 20



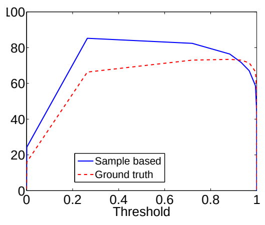
(i) class 21



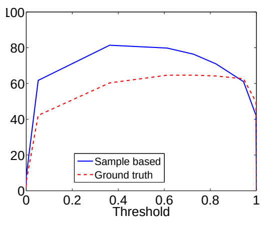
(j) class 22



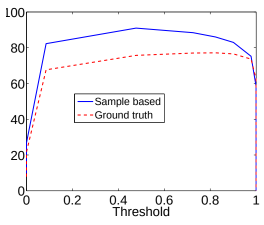
(k) class 23



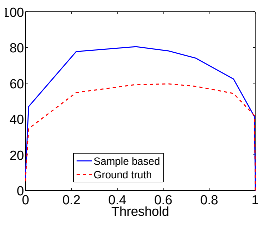
(l) class 24



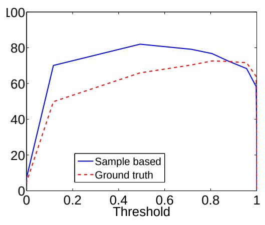
(m) class 25



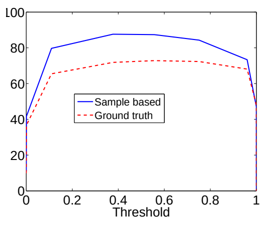
(n) class 26



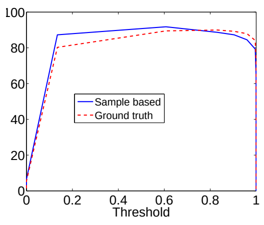
(o) class 27



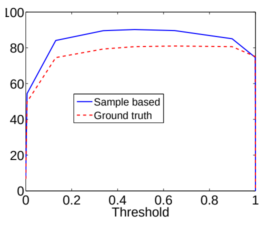
(p) class 28



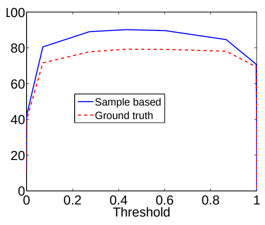
(q) class 29



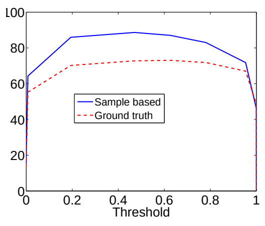
(r) class 30



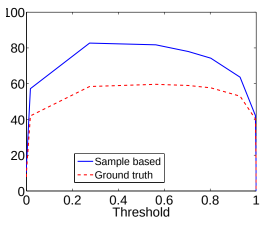
(s) class 31



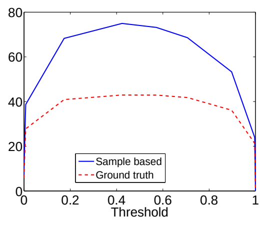
(t) class 32



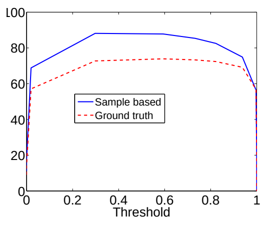
(u) class 33



(v) class 34

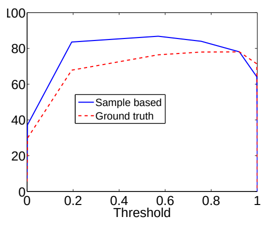


(w) class 35

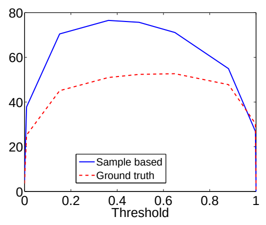


(x) class 36

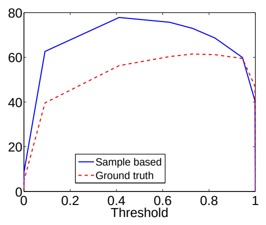
Figure C.2: Example curves of $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ (blue) and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (red) v.s. θ.



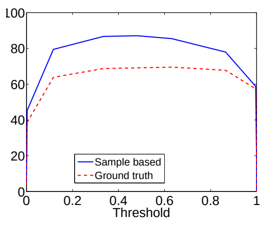
(a) class 37



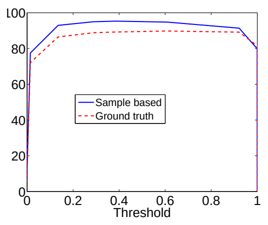
(b) class 38



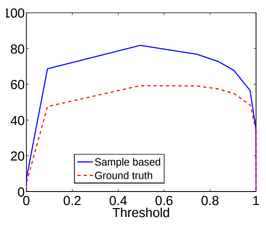
(c) class 39



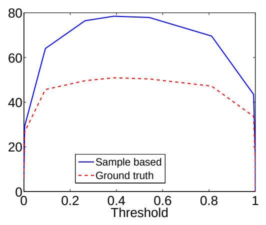
(d) class 40



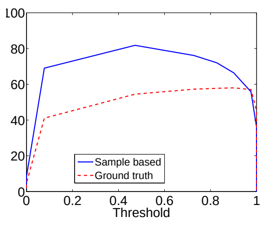
(e) class 41



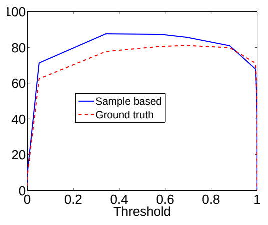
(f) class 42



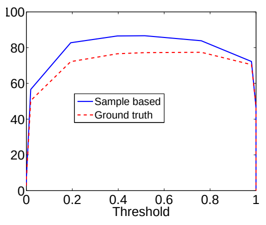
(g) class 43



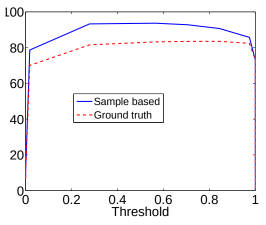
(h) class 44



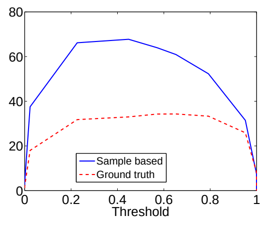
(i) class 45



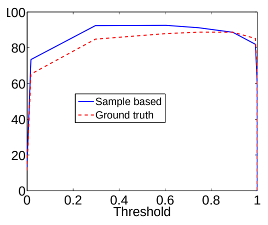
(j) class 46



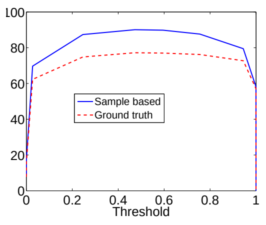
(k) class 47



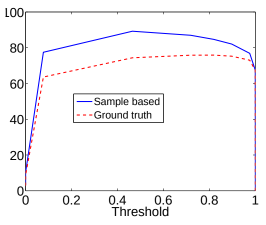
(l) class 48



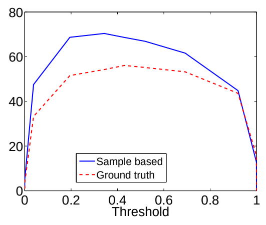
(m) class 49



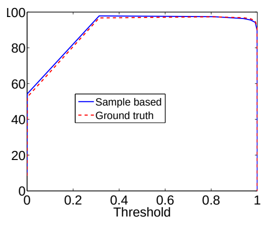
(n) class 50



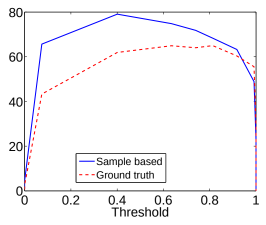
(o) class 51



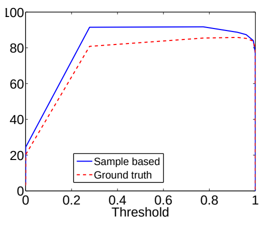
(p) class 52



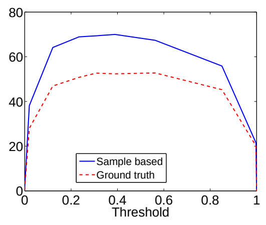
(q) class 53



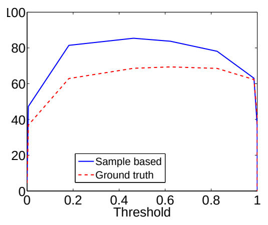
(r) class 54



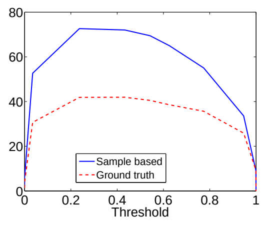
(s) class 55



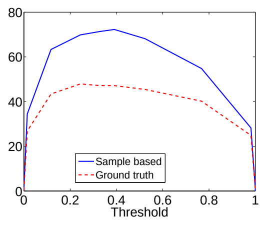
(t) class 56



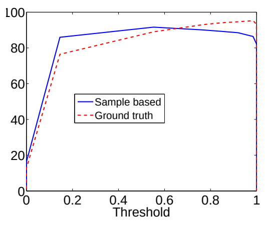
(u) class 57



(v) class 58



(w) class 59



(x) class 60

Figure C.3: Example curves of $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ (blue) and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (red) v.s. θ.

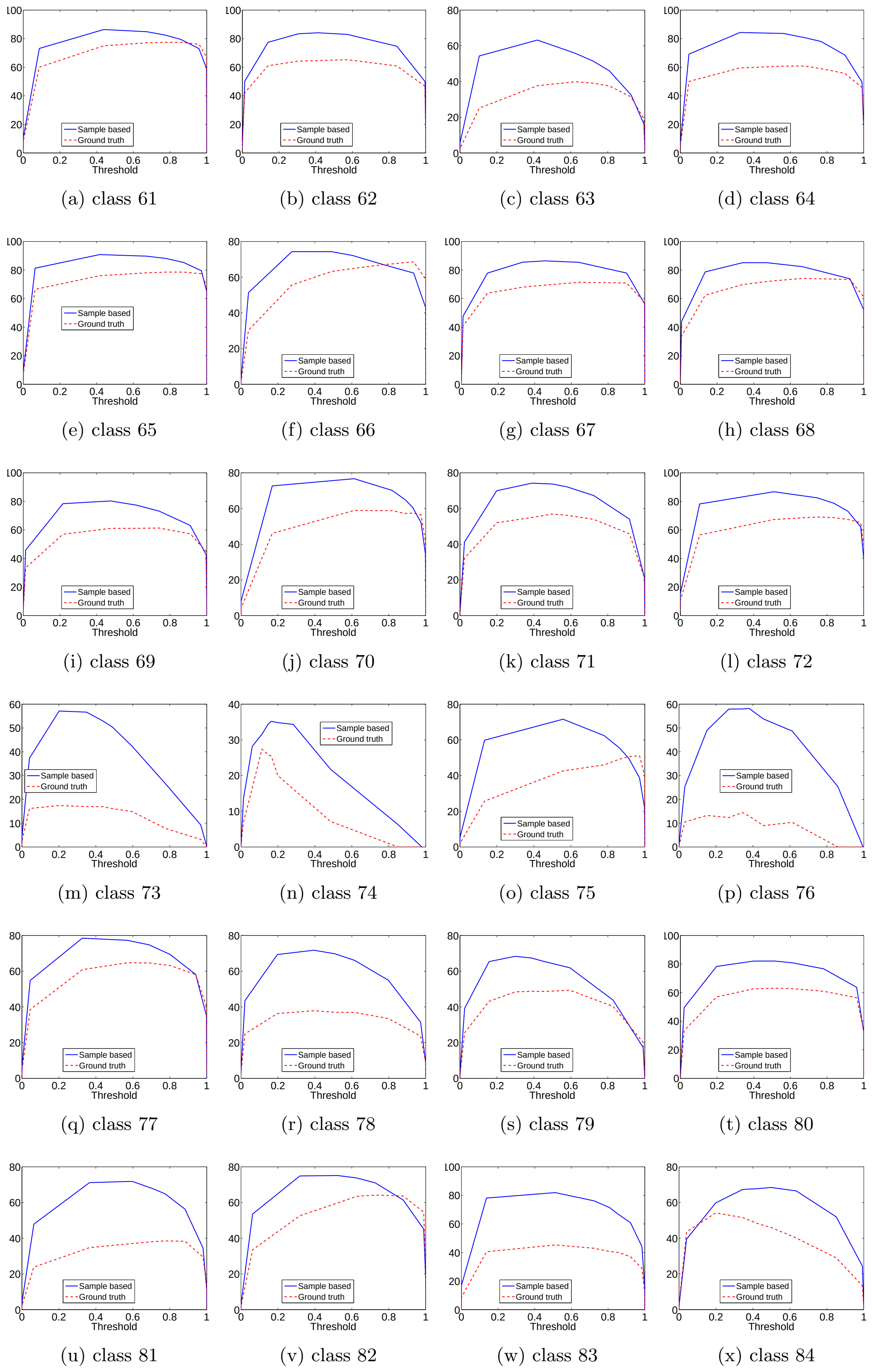
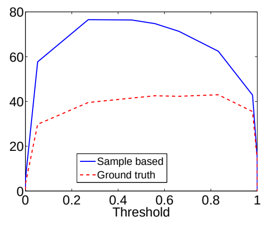
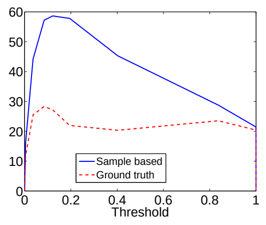


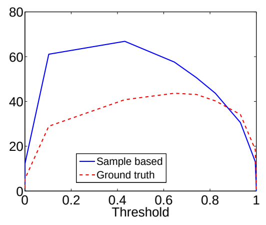
Figure C.4: Example curves of $\widetilde{\mathrm{ExpFs}}(\mathbf{l}(\theta))$ (blue) and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (red) v.s. θ.



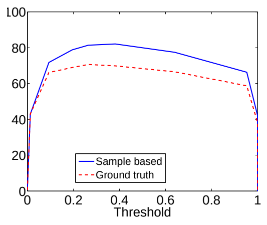
(a) class 85



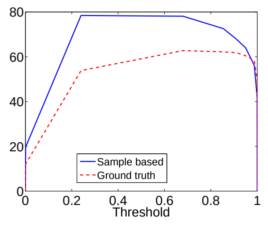
(b) class 86



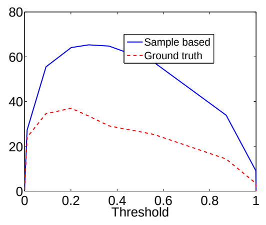
(c) class 87



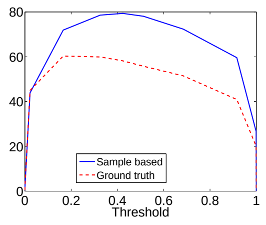
(d) class 88



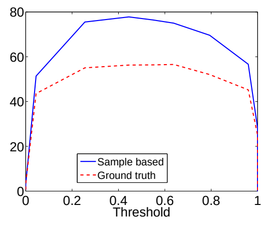
(e) class 89



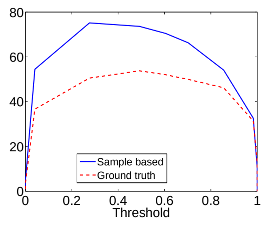
(f) class 90



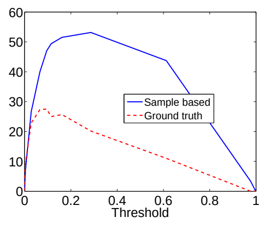
(g) class 91



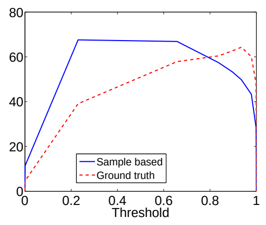
(h) class 92



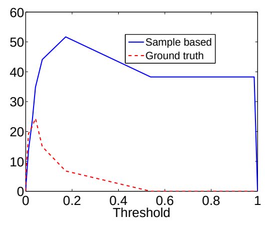
(i) class 93



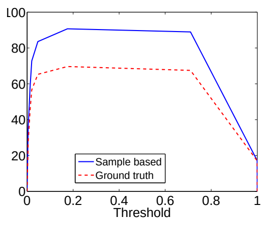
(j) class 94



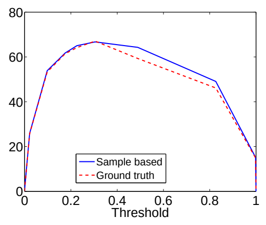
(k) class 95



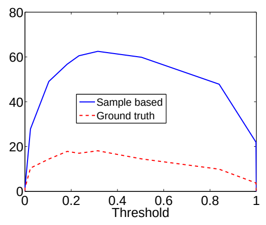
(l) class 96



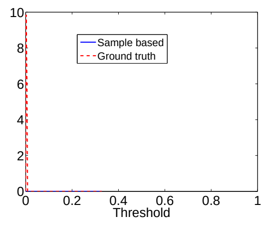
(m) class 97



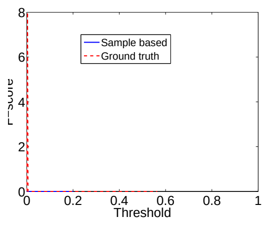
(n) class 98



(o) class 99



(p) class 100



(q) class 101

Figure C.5: Example curves of $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ (blue) and F-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (red) v.s. θ.

Appendix D

Modeling Hierarchies in Labels

Hierarchy is often modeled as a tree composed of all possible labels, and extension to forest is immediate. The root is the most general class and by going down to the leaves, the labels become more and more specific. It encodes the fact that if a child label is relevant, then its parent label must be relevant as well (but *not* vice versa). In the sequel, we will use the word "node" and "label" interchangeably. Another example relationship is co-occurrence, where two labels must be relevant or irrelevant at the same time. Both forms can be unified in the framework of propositional logic. Using $l_c \in \{1, 0\}$ to represent whether label c is relevant, the tree hierarchy encodes $l_p \to l_c$ if class c is a child of class p, and co-occurrence can be denoted by $l_a \Leftrightarrow l_b$. Clearly propositional logic can model more general relations like $l_a \wedge \neg l_b \to l_c$.

In the simplest case, the relations are provided as prior information. A more complicated task is to induce relations from the data, and associate them with some confidence. This will be similar to association rule mining. Here we assume that a hierarchy of classes is available a priori in terms of a tree, whose connotation is:

Every data point is associated with a (possibly empty) set of labels. Whenever an instance is labeled with a certain label c, it must also be labeled with all the nodes on the path from the root down to c.

This is exactly the setting used in (Rousu et al., 2006), which utilizes the hierarchy mainly through the definition of loss function: if the node c is misclassified, then further mistakes made in the subtree rooted at c are not penalized. Formally, suppose the true label on a tree is $\mathbf{y}$ and the predicted label is $\mathbf{l}$, then the so-called H-loss defined by Rousu et al. (2006) is:

$$l_H(\mathbf{l}, \mathbf{y}) = \sum_c a_c \cdot \delta(l_c \neq y_c \wedge l_h = y_h \quad \forall h \in \text{ancestor of } c).$$

where the summation goes through all the classes, and $a_c \in [0, 1]$ downscales the loss when going deeper in the tree.

Inference can be performed on directed trees, and the conditional probability $p(l_c|l_p)$

is important where l_p is the parent node and l_c is the child node. By definition, $p(l_c = 1|l_p = 0) = 0$ and $p(l_c = 0|l_p = 0) = 1$ so we only need to learn $p(l_c|l_p = 1)$. Similar to the marginal probability models, we also assume a linear model for $p(l_c|l_p = 1)$, and its weights are updated only when the ground truth satisfies $l_p = 1$. However, as the marginals and the conditionals are learned separately, their consistency is not guaranteed, *i.e.*

$$p(l_c) \neq \sum_{l_p} p(l_c|l_p) \cdot p(l_p).$$

To solve this problem, we may resort to undirected trees, where we learn a linear model for the node and edge potentials. However, all the factors in undirected graphical models are globally coupled and learning becomes hard.

So we would rather patch the directed tree model in two different ways:

1. Learn the model of marginal only for the root label in the tree. The rest non-root nodes will be inferred only via the conditionals (edges). This will immediately rule out over-parametrization and inconsistency.

2. Still learn the marginals of all nodes and conditionals on all edges, but treat these probabilities as node and edge potentials of the undirected graph, and samples can be drawn easily from this undirected tree.

The second approach might be better because it uses more information, and enforces consistency between different parts of the models learned separately. And this approach is also general enough to model propositional logic. For each proposition such as $l_p \to l_c$ and $l_a \Leftrightarrow l_b$, we introduce a factor with linear parametrization, and the resulting factor graph can be loopy.

Appendix E

Statistical Estimation and Concentration of Measure

Parameter estimation is one of the key problems in statistics. Given an unknown distribution F, we wish to estimate a parameter of F: $\theta = \theta(F)$, *e.g.*, moments, quantiles, and Gini differences. To this end, suppose we have observations (a stochastic process) $X_1, X_2, \ldots$ obtained from *certain statistical experiment*, and then a *statistic* $g(X_1, X_2, \ldots)$[1] is applied as an *estimator* of θ.

When X_i are independently and identically distributed (*iid*) according to F, the behavior of the statistics has been well studied. However, the *iid* assumption is often unrealistic in practice and the non-*iid* setting is of particular interest in Chapter 4. This appendix will first list some desirable statistical properties of estimators, then state the properties of U-statistics under *iid* observations. Finally, we will briefly survey some fairly recent results on U-statistics under non-*iid* observations.

E.1 Desirable statistical properties of estimators

In this section, we describe some general intuitions on what makes a good estimator. We will emphasize the intuition, and more rigorous statements will be given in the subsequent sections under more specific settings. We assume $X := (X_1, X_2, \ldots)$ is the random variables of observation from a stochastic process related to F but not necessarily *iid*, and $g(X)$ is an estimator.

Unbiasedness. It is natural to desire that the estimator $g(X)$ be centered around the true parameter θ. $g(X)$ is said to be an unbiased estimator of θ if

$$\text{Bias}(g(X), \theta) := \mathbb{E}[g(X) - \theta] = 0.$$

[1] Most literature use the symbol h for statistics. We use g here in order to avoid confusion with the kernel function of U-statistics later.

Mean square error. The mean square error of an estimator $g(X)$ is defined by

$$\mathrm{MSE}(g(X), \theta) := \mathbb{E}\left[(g(X) - \theta)^2\right] = \mathrm{Var}(g(X)) + \mathrm{Bias}(g(X), \theta)^2.$$

The lower the MSE, the better. Incidently, this shows the bias-variance tradeoff, which is related to a similar issue in statistical machine learning.

Consistency. A sequence of estimators $g_n(X)$ are consistent for θ if $g_n(X) \xrightarrow{\mathbb{P}} \theta$ as $n \to \infty$, *i.e.*,

$$\lim_{n\to\infty} \Pr\left(|g_n(X) - \theta| > \epsilon\right) = 0 \quad \text{for all } \epsilon > 0. \tag{E.1}$$

Here and henceforth, the symbol Pr means the implicit probability measure is clear from the context. Same is true when discussing convergence of random variables. Note that consistency does not imply unbiasedness since it allows $\mathrm{Bias}(g_n(X), \theta) \to 0$ as $n \to \infty$. An example is BUGBUG (HSIC_b). When g_n is based on n observations $X_1, \ldots, X_n$, consistency simply means that the estimation approaches the true parameter with more observations. This is often formulated as the law of large numbers (LLN), whose simplest form says when we have more and more *iid* observations, the average of the observations converges to the mean ($\theta = \mu$) of the distribution. Loosely speaking, it has two forms: letting $g_n(X) = \bar{X}_n := \frac{1}{n}\sum_{i=1}^n X_i$,

$$\begin{aligned} &\text{weak law of large numbers (WLLN):} && \bar{X}_n \xrightarrow{\mathbb{P}} \theta && \text{as } n \to \infty; \\ &\text{strong law of large numbers (SLLN):} && \bar{X}_n \xrightarrow{a.s.} \theta && \text{as } n \to \infty. \end{aligned}$$

The Borel-Cantelli lemma allows one to convert a sufficiently rapidly converging WLLN into a SLLN.

Mode of convergence. The consistency Eq (E.1) is defined in terms of convergence in probability ($\xrightarrow{\mathbb{P}}$). Stronger modes such as converging almost surely ($\xrightarrow{a.s.}$) and in moment/mean ($\xrightarrow{L_p}$), or weaker modes like converging in distribution ($\xrightarrow{\mathcal{D}}$) are also useful. All these modes will be used later.

Concentration of measure and rate of convergence. When the sequence of estimators $g_n(X)$ are consistent (*i.e.*, Eq (E.1)), it is furthermore desirable that a) the probability mass of $g_n(X)$ concentrates tightly around θ, and b) the concentration tightens fast with the growth of n, or the number of observations. These are typically characterized by two kinds of bounds: uniform bounds and asymptotic bounds which are defined by converging in probability and in distribution, respectively. Two typical forms are:

$$\begin{aligned} \text{uniform bounds:} \quad & \Pr\{|g_n(X) - \theta| > \epsilon\} \le c\exp(-\eta n\epsilon^2) \quad \text{for all } \epsilon > 0; \\ \text{asymptotic bounds:} \quad & \sqrt{n}(g_n(X) - \theta) \overset{\mathcal{D}}{\to} \mathcal{N}(0, \sigma), \quad \text{as } n \to \infty, \end{aligned}$$

where c, η, σ are some constants. The asymptotic bounds are also known as central limit theorem (CLT).

In practice, computational efficiency is also an important issue.

In the next two sections, we will specialize the discussion to a common class of parameters: U-parameters. They can be estimated by U-statistics which are applied in Chapter 4. Before branching to U-statistics with *iid* and non-*iid* observations, we first give the (common) definition of U-parameters.

Definition 76 (U-parameters) *(Hoeffding, 1963) $\theta = \theta(F)$ is a U-parameter of distribution F if there is a measurable function $h : \mathbb{R}^d \to \mathbb{R}$, called the kernel, so that*

$$\theta(F) = \theta_h(F) = \int_{\mathbb{R}^d} h \, \mathrm{d}F^{(d)} \quad \textit{for all } F \textit{ for which the integral is defined,}$$

where $F^{(d)}$ is the product measure $F \times \ldots \times F$ on $\mathbb{R}^d$. The positive integer d is called the order of the kernel.

For example, when $d = 1$ and $h(x) = x$, $\theta_h(F)$ is the mean of F. When $d = 2$ and $h(x_1, x_2) = |x_1 - x_2|^\alpha$, $\theta_h(F)$ is the generalized Gini difference of F.

U-parameter is often called an *estimable* parameter, in the sense that a parameter can be estimated in an unbiased matter if, and only if, it is a U-statistics (Lehmann, 1983). For simplicity, we assume that the kernel h is symmetric, *i.e.*, invariant to the permutation of its arguments. This is enough for our use in Chapter 4.

E.2 U-Statistics for *iid* observations

In this section, assume we have a sequence of observations $(X_1, X_2, \ldots, X_n)$ drawn *iid* from a distribution F. For convenience, let $X_i^j := (X_i, X_{i+1}, \ldots, X_j)$. We now define the U-statistics which can be used to estimate the corresponding U-parameter.

Definition 77 (U-statistics) *(Hoeffding, 1963) Given a U-parameter θ_h where h is symmetric, the U-statistic for θ_h based on a sequence $X_1, \ldots, X_n$ is defined by*

$$U_h^n := U_h(X_1, \ldots, X_n) = \begin{pmatrix} n \\ d \end{pmatrix}^{-1} \sum \left\{ h(X_{i_1}, \ldots, X_{i_d}) : 1 \le i_1 < i_2 < \ldots < i_d \le n \right\}.$$

Closely related is a V-statistic for θ_h and $\{X_i\}_{i=1}^n$:

$$V_h^n := V_h(X_1, \ldots, X_n) = n^{-d} \sum \{h(X_{i_1}, \ldots, X_{i_d}) : 1 \leq i_j \leq n \text{ for all } j\}\,.$$

U-statistics have a number of desirable properties under *iid* observations, and we list a few important ones below.

Theorem 78 (Unbiasedness) *The U-statistic U_h^n is an unbiased estimator of θ_h:*

$$\mathbb{E}_{X_1^n}[U_h^n] = \theta_h, \qquad \textit{for all } n.$$

Theorem 79 (Minimality of variance) *(Serfling, 1980) U_h^n has the lowest variance among all unbiased estimators of θ_h (for a fixed n).*

Theorem 80 (Concentration of measure, a.k.a. Hoeffding) *(Hoeffding, 1963) If h is of order d and $h \in [a, b]$, then*

$$\Pr\left\{U_h^n - \mathbb{E}_{X_1^n}[U_h^n] \geq \epsilon\right\} \leq \exp\left(-\frac{2\lfloor n/d \rfloor \epsilon^2}{(b-a)^2}\right) \qquad \textit{for all } \epsilon > 0.$$

We can further characterize the variance of U-statistics, based on which an asymptotic bound on the concentration of measure can be established (Serfling, 1980).

Assume $\mathbb{E}_{X_1^d}[h(X_1^d)^2]$ is finite. For any $t \in [1, d]$, define

$$h_t(X_1^t) := \mathbb{E}_{X_{t+1}^d}[h(X_1^d)], \tag{E.2}$$

i.e., expecting out the tail $d - t$ variables X_{t+1}^d. Define $\zeta_0 = 0$ and

$$\zeta_t := \mathrm{Var}_{X_1^t}[h_t(X_1^t)] \in \mathbb{R}.$$

Then

Theorem 81 (Variance of U-statistics) *(Serfling, 1980, pp. 183)*

$$\mathrm{Var}[U_h^n] = \binom{n}{d}^{-1} \sum_{t=1}^{d} \binom{d}{t}\binom{n-d}{d-t} \zeta_t.$$

Now we can state the asymptotic bound, which branches into two cases depending on whether $\zeta_1 = 0$ or not.

Theorem 82 (CLT 1) *(Serfling, 1980, pp. 192) If $\mathbb{E}[h^2] < \infty$ and $\zeta_1 > 0$, then U_h^n converges in distribution to a Gaussian with mean θ_h and variance $d^2\zeta_1/n$:*

$$\sqrt{n}(U_h^n - \theta_h) \xrightarrow{\mathcal{D}} \mathcal{N}(0, d^2\zeta_1) \quad \textit{as } n \to \infty.$$

Theorem 83 (CLT 1) *(Serfling, 1980, pp. 194) If* $\mathbb{E}[h^2] < \infty$ *and* $\zeta_1 = 0 < \zeta_2$*, then*

$$n(U_h^n - \theta_h) \overset{\mathcal{D}}{\to} \frac{d(d-1)}{2} \sum_{i=1}^{\infty} \lambda_i(\chi_i^2(1) - 1),$$

where $\chi_i^2(1)$ *are* iid $\chi^2(1)$ *random variables, and* λ_i *are the eigenvalues of*

$$\int_{\mathbb{R}} (h_2(x_1, x_2) - \theta_h)\Phi_i(x_1)\mathrm{d}F(x_2) = \lambda_i \Phi_i(x_1).$$

Φ_i *are the corresponding eigen-functions.*

E.3 Statistics for non-*iid* observations

In reality, the *iid* assumption almost never holds. So it is crucial to study the non-*iid* setting, which motivates Chapter 4.

Let $(\Omega^n, \mathcal{F}, \mathbf{P})^2$ be a probability space and $\{X_i\}_{i=1}^n$ be its associated real valued stochastic process. If the process is stationary[3], we will also assume that the probability space of X_1 to be $(\Omega, \mathcal{A}, P)$.

This section again focuses on U-statistics. The definition of U-parameter in Definition 76 keeps intact. The U-statistics, however, needs to be generalized using stochastic process.

Definition 84 (U-statistics with stochastic process) *Given a U-parameter* θ_h *where* h *is symmetric, the U-statistic for* θ_h *based on the sequence* $X_1, \ldots, X_n$ *is*

$$U_h^n = U_h(X_1, \ldots, X_n) = \begin{pmatrix} n \\ d \end{pmatrix}^{-1} \sum \{h(X_{i_1}, \ldots, X_{i_d}) : 1 \le i_1 < i_2 < \ldots < i_d \le n\}.$$

To state the convergence and concentration results, some definitions pertaining to stochastic process is necessary. Intuitively, a stochastic process is "nice" if it is close to *iid*, where "closeness" is characterized by concepts such as stationarity, ergodicity, and in particular quantified by mixing coefficients.

Definition 85 (Ergodic stationary process) *A real valued* ergodic stationary process *(ESP) with sample space* $(\Omega, \mathcal{A}, P)$ *is a stochastic sequence* $(X_1, X_2, \ldots)$ *of form*

[2]Ω^n means Ω to the power of n, and $\mathbf{P}$ is not necessarily a product measure.

[3]Stationary means if $F_{t_1,\ldots,t_k}$ is a distribution function of the the joint distribution $X_{t_1}, \ldots, X_{t_k}$, then for all k and all $\tau \in \mathbb{Z}$, we have $F_{t_1,\ldots,t_k} = F_{t_1+\tau,\ldots,t_k+\tau}$.

$X_k = f \circ T^k$ *where* T *is an ergodic*[4]*, probability preserving transformation*[5] *of the probability space* $(\Omega, \mathcal{A}, P)$*, and* $f : \Omega \to \mathbb{R}$ *is a measurable function. The* marginal *of the ESP is the distribution of* X_1*, and the ESP is called* integrable *if* X_1 *is integrable, and* bounded *if* X_1 *is essentially bounded.*

An important characterization of the mixing property for stochastic processes is the absolute regularity.

Definition 86 (Absolute regularity) *A process* $\{X_i\}_{i=1}^{\infty}$ *is called absolutely regular (also known as weak Bernoulli) if* $\lim_{k\to\infty} \beta_k = 0$ *where*

$$\beta_k := 2 \sup_n \mathop{\mathbb{E}}_{B \in \mathcal{A}_1^n} \left[\sup_{A \in \mathcal{A}_{n+k}^{\infty}} |P(A|B) - P(A)| \right]. \tag{E.3}$$

Here $\mathcal{A}_n^m := \sigma(X_n, \ldots, X_m)$ *for* $n \le m$*,* i.e.*, the* σ*-algebra generated by* $\{X_i\}_{i=n}^{m}$[6]*.*

Intuitively, absolute regularity implies that for any two possible states of the system (realizations of the random variable), when given a sufficient amount of time between the two states, the occurrence of the states is *independent*. Different notions of mixing can be derived from different ways of characterizing independence, *e.g.*:

$$\begin{aligned}
\text{strong mixing}: \quad \alpha_k &= \sup_n \sup_{A \in \mathcal{A}_1^n} \sup_{B \in A_{n+k}^{\infty}} |P(A \cap B) - P(A)P(B)| \to 0 \text{ as } k \to \infty \\
\text{uniformly mixing}: \quad \phi_k &= \sup_n \sup_{A \in \mathcal{A}_1^n} \sup_{B \in A_{n+k}^{\infty}} |P(B|A) - P(B)| \to 0 \text{ as } k \to \infty \\
\psi\text{-mixing}: \quad \psi_k &= \sup_n \sup_{A \in \mathcal{A}_1^n} \sup_{B \in A_{n+k}^{\infty}} \left| \frac{P(A \cap B)}{P(A)P(B)} - 1 \right| \to 0 \text{ as } k \to \infty.
\end{aligned}$$

It is clear that $\psi_k \ge \phi_k \ge \beta_k \ge \alpha_k$. Under some mild regularity conditions, SLLN and WLLN can be proved for non-*iid* observations.

Theorem 87 (SLLN for U-statistics) *(Aaronson et al., 1996) Let* $\{X_i\}_{i=1}^{n}$ *be an ESP with marginal* F*, and let* $h : \mathbb{R}^d \to \mathbb{R}$ *be a measurable function bounded by an* F*-integrable product*[7]*. If any of the following three conditions hold:*

1. *F is discrete;*

[4] Ergodic transformation means $P(T^{-1}(A) \bigtriangleup A) = 0$ for some $A \in \mathcal{A}$ only if $P(A) = 0$ or $P(A) = 1$. Here $\bigtriangleup$ stands for symmetric difference: $A \bigtriangleup B := (A \backslash B) \cup (B \backslash A)$.

[5] A transformation $T : \Omega \to \Omega$ is called probability preserving if for any set $A \in \mathcal{A}$, $P(T^{-1}A) = P(A)$.

[6] The σ-algebra generated by a family of real valued random variables $\{X_i\}_{i \in \mathcal{I}}$ is defined to be the smallest σ-algebra for which all X_i are measurable, *i.e.*, the smallest σ-algebra that contains $\{X_i^{-1}(B) : i \in \mathcal{I}, B \in \mathcal{B} \text{ (the Borel } \sigma\text{-algebra)}\}$.

[7] Meaning that there exist measurable functions $f_i : \mathbb{R} \to \mathbb{R}$ with $\int |f_i| \, \mathrm{d}F < \infty$, such that $|h(x_1, \ldots, x_d)| \le \prod_{i=1}^{d} f_i(x_i)$.

2. h is continuous at $F^{(d)}$ almost every point;

3. $\{X_i\}_{i=1}^n$ is absolutely regular.

then

$$U_h^n \overset{a.s.}{\to} \theta_h(F), \qquad \text{as } n \to \infty.$$

Theorem 88 (WLLN for U-statistics) *(Borovkova et al., 1999, Thm 1) Let $\{X_i\}_{i=1}^n$ be a ESP with marginal F, and let $h : \mathbb{R}^d \to \mathbb{R}$ be a measurable function and $F^{(d)}$-a.e. continuous. Suppose moreover that the family of random variables $\{h(X_{i_1}, \dots, X_{i_d}) : i_j \geq 1 \text{ for all } 1 \leq j \leq d\}$ is uniformly integrable. Then*

$$U_h^n \overset{\mathbb{P}}{\to} \theta_h \quad \text{as } n \to \infty.$$

In particular this holds, if $\sup_{i_1,\dots,i_d} \mathbb{E}\left[|h(X_{i_1}, \dots, X_{i_d})|^{1+\delta}\right] < \infty$ for some $\delta > 0$.

Theorem 89 (LLN in moment for U-statistics) *(Borovkova et al., 1999, Thm 2) Let $\{X_i\}_{i=1}^n$ be a stationary and absolutely regular process with marginal F, and let $h : \mathbb{R}^d \to \mathbb{R}$ be a measurable function. Suppose moreover that the family of random variables $\{h(X_{i_1}, \dots, X_{i_d}) : i_j \geq 1 \text{ for all } 1 \leq j \leq d\}$ is uniformly integrable. Then*

$$U_h^n \overset{L_1}{\to} \theta_h \quad \text{as } n \to \infty.$$

and hence also in probability.

Theorem 90 (CLT for U-statistics) *(Borovkova et al., 2001, Thm 7) Let $\{X_i\}_{i=1}^n$ be an absolutely regular process with mixing coefficients β_k, and let h be a bounded 1-continuous[8] kernel. Suppose the sequences $\{\beta_k\}_{k\geq1}$, $\{\alpha_k\}_{k\geq1}$ and $\{\phi(\alpha_k)\}_{k\geq1}$[9] satisfy the following summability condition:*

$$\sum_{k=1}^{\infty} k^2(\beta_k + \alpha_k + \phi(\alpha_k)) < \infty.$$

Then the series

$$\sigma^2 = \mathrm{Var}(h_1(X_0))^2 + 2\sum_{k=1}^{\infty} \mathrm{Cov}(h_1(X_0), h_1(X_k))$$

[8]Definition of p-continuity (Borovkova et al., 2001, Def 2.13): Let $\{X_i\}_{i=1}^n$ be a stationary stochastic process. A measurable function $g : \mathbb{R}^I \to \mathbb{R}$ is called p-continuous if there exists a function $\phi : (0, \infty) \to (0, \infty)$ with $\phi(\epsilon) = o(1)$ as $\epsilon \to 0$ such that $\mathbb{E}\left[|g(\xi_{I_1}, \xi_{I_2}) - g(\xi_{I_1}, \xi'_{I_2})|^p \mathbf{1}_{\{\|\xi_{I_2} - \xi'_{I_2}\| \leq \epsilon\}}\right] \leq \phi(\epsilon)$ holds for all disjoint index sets I_1 and I_2 with $I_1 \cup I_2 = I$ and for all random vectors ξ_{I_1}, ξ_{I_2}, ξ'_{I_2} such that (ξ_{I_1}, ξ_{I_2}) has distribution $P_{X_{I_1}, X_{I_2}}$ or $P_{X_{I_1}} \times P_{X_{I_2}}$ and ξ'_{I_2} has the same distribution as X_{I_2}. In our case, the cardinality of I only needs to be d.

[9]This ϕ is from the definition of 1-continuity of h.

converges absolutely and

$$\sqrt{n}(U_h^n - \theta_h) \overset{\mathcal{D}}{\to} \mathcal{N}(0, 4\sigma^2), \quad \text{as } n \to \infty.$$

Here, h_1 is defined by Eq (E.2).

It is noteworthy that there is a large amount of literature on the concentration of measures for *iid* and non-*iid* processes. Many more results can be found from, *e.g.*, (Ledoux, 2001; Lugosi, 2006), and http://www.stats.org.uk/law-of-large-numbers for an online literature collection of LLN.

E.4 Mixing coefficients using Markov properties

Most results in the previous sections on non-*iid* process provide global characterization of the stochastic process and the estimators, while more useful insights may be derived by studying the Markov properties of the process (*e.g.*, first order Markov chain). To this end, we need some mixing coefficients which:

1. Can be decomposed onto the structure of the process, which yields more refined characterization and tighter bounds.

2. Can be used to bound the concentration of measure.

The η-mixing coefficient introduced by Kontorovich (2007) is a powerful tool for these two purposes. The main result of concentration is as follows. Let $\{X_i\}_{1 \le i \le n}$ ($X_i \in \Omega$) be a stochastic process defined on the probability space $(\Omega^n, \mathcal{F}, \mathbf{P})$, and $f : \Omega^n \to \mathbb{R}$ be a function satisfying some Lipschitz condition. Then we have

$$\Pr\{|f - \mathbb{E}f| > \epsilon\} \le 2\exp\left(-\frac{\epsilon^2}{2\,\|f\|^2_{\mathrm{Lip},\mathbf{w}}\,\|\triangle_n \mathbf{w}\|_2^2}\right), \tag{E.4}$$

where:

1. Pr is with respect to $\mathbf{P}$.

2. $\|f\|_{\mathrm{Lip},\mathbf{w}}$ is the Lipschitz constant of f with respect to the weighted Hamming metric $d_{\mathbf{w}}$: $d_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^n w_i \delta(x_i \ne y_i)$, where $\mathbf{w} \in \mathbb{R}^n$ with all $w_i \ge 0$. Lipschitz constant means that $|f(\mathbf{x}) - f(\mathbf{y})| \le \|f\|_{\mathrm{Lip},\mathbf{w}} \cdot d_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ for any $\mathbf{x}, \mathbf{y}$.

3. $\triangle_n$ is the n-by-n η-mixing matrix, which is the focus of following discussion.

The utility of Eq E.4 relies on bounding $\|f\|_{\text{Lip},\mathbf{w}}$ and $\|\triangle_n\|_2$. Restricting $\mathbf{w}$ to normalized Hamming (*i.e.*, $\sum_{i=1}^n w_i = 1$ with $w_i \geq 0$), we have

$$\|\triangle_n \mathbf{w}\|_2^2 \;\leq\; n \cdot \max_{1\leq i\leq n} (\triangle_n \mathbf{w})_i^2 \;\leq\; n\,\|\triangle_n\|_\infty^2 .$$

So the major challenge becomes controlling the quantity $\|\triangle_n\|_\infty$, and Kontorovich (2007) showed that by using the the Markov property of the process, $\|\triangle_n\|_\infty$ can be tightly bounded.

E.4.1 Bounds on $\|\triangle_n\|_\infty$ using Markov property

This section collects several important results from (Kontorovich, 2007, Chapter 4). We start with the definition of $\triangle_n$.

Definition 91 (η-mixing matrix) *(Kontorovich, 2007, Section 3.2.1) Let $(\Omega^n, \mathcal{F}, \mathbf{P})$ be a probability space and $\{X_i\}_{i=1}^n$ be its associated real valued stochastic process. For $1 \leq i \leq j \leq n$ and $x \in \Omega^i$, let $\mathcal{L}(X_j^n | X_1^i = x)$ be the law of X_j^n conditioned on $X_1^i = x$. For $\mathbf{y} \in \Omega^{i-1}$ and $w, w' \in \Omega$, define*

$$\eta_{ij}(y, w, w') := \left\|\mathcal{L}(X_j^n | X_1^i = [\mathbf{y}w]) - \mathcal{L}(X_j^n | X_1^i = [\mathbf{y}w'])\right\|_{\text{TV}},$$

$$\bar{\eta}_{ij} := \max_{\mathbf{y}\in\Omega^{i-1}} \max_{w,w'\in\Omega} \eta_{ij}(\mathbf{y}, w, w'),$$

where $\|\cdot - :\|_{\text{TV}}$ is the total variation distance of two probability measures p and q on $(\Omega, \mathcal{A})$ defined by $\|p - q\|_{\text{TV}} := \sup_{A\in\mathcal{A}} |p(A) - q(A)|$. Finally, $\triangle_n = \triangle_n(\mathbf{P})$ is defined as the upper-triangular matrix with $(\triangle_n)_{ii} = 1$ and $(\triangle_n)_{ij} = \bar{\eta}_{ij}$ for $1 \leq i < j \leq n$.

The key interesting property of $\|\triangle_n\|_\infty$ (or $\bar{\eta}_{ij}$) is that its upper bound can be much tightened by utilizing the Markov properties (conditional independence) of the process (if any). Results below are all directly quoted from (Kontorovich, 2007, Chapter 4).

Markov chains

Let μ be an inhomogeneous Markov measure on Ω^n, induced by the kernels p_0 and $p_i(\cdot|\cdot)$, $1 \leq i \leq n$. Thus

$$\mu(x) = p_0(x_1) \prod_{i=1}^{n-1} p_i(x_{i+1}|x_i).$$

Define the i^{th} contraction coefficient:

$$\theta_i \;=\; \max_{y,y'\in\Omega} \left\|p_i(\cdot|y) - p_i(\cdot|y')\right\|_{\text{TV}}; \tag{E.5}$$

then

$$\bar{\eta}_{ij} \le \theta_i \theta_{i+1} \dots \theta_{j-1}. \tag{E.6}$$

Undirected Markov chains

For any graph $G = (V, E)$, where $|V| = n$ and the maximal cliques have size 2 (are edges), we can define a measure on Ω^n as follows

$$\mu(x) = \mathbf{P}(X = x) = \frac{\prod_{(i,j)\in E} \psi_{ij}(x_i, x_j)}{\sum_{x'\in\Omega^n} \prod_{(i,j)\in E} \psi_{ij}(x'_i, x'_j)}.$$

for some $\psi_{ij} \ge 0$. We can relate the induced Markov transition kernel $p_i(\cdot|\cdot)$ to the random field measure μ by:

$$p_i(x|y) = \frac{\sum_{v_1^{i-1}} \sum_{z_{i+2}^n} \mu[vyxz]}{\sum_{x'\in\Omega} \sum_{v_1^{i-1}} \sum_{z_{i+2}^n} \mu[vyx'z]}, \qquad x, y \in \Omega$$

Then one can bound the i^{th} contraction coefficient θ_i of the Markov chain:

$$\theta_i \ = \ \max_{y,y'\in\Omega} \frac{1}{2} \sum_{x\in\Omega} |p_i(x|y) - p_i(x|y)| \ \le \ \frac{R_i - r_i}{R_i + r_i},$$

where $R_i := \max_{x,y\in\Omega} \psi_{i,i+1}(x, y)$ and $r_i := \min_{x,y\in\Omega} \psi_{i,i+1}(x, y)$. Eq (E.6) still applies.

Hidden Markov chains

Consider two finite sets $\hat{\Omega}$ (the hidden state space) and Ω (the observed state space). Let $(\hat{\Omega}^n, \mu)$ be a probability space, where μ is a Markov measure with transition kernels $p_i(\cdot|\cdot)$. Thus for $\hat{x} \in \hat{\Omega}^n$, we have:

$$\mu(\hat{x}) = p_0(\hat{x}_1) \prod_{k=1}^{n-1} p_k(\hat{x}_{k+1}|\hat{x}_k).$$

Suppose $(\hat{\Omega} \times \Omega, \nu)$ is a probability space whose measure ν is defined by

$$\mu(\hat{x}, x) = \mu(\hat{x}) \prod_{l=1}^{n} q_l(x_l|\hat{x}_l),$$

where $q_l(\cdot|\hat{x}_l)$ is a probability measure on Ω for each $\hat{x} \in \hat{\Omega}$ and $1 \le l \le n$. On this product space, we define the random process $(\hat{X}_i, X_i)_{1\le i\le n}$, which is clearly Markov. The marginal projection of $(\hat{X}_i, X_i)$ onto X_i results in a random process on the probability space (Ω^n, ρ), where

$$\rho(x) = \mathbf{P}(X = x) = \sum_{\hat{x}\in\hat{\Omega}^n} \nu(\hat{x}, x).$$

The process $(X_i)_{1\le i\le n}$ with measure ρ is called hidden Markov process, which need not be Markov to any order. Define the k^{th} contraction coefficient θ_k by

$$\theta_k \;=\; \sup_{\hat{x},\hat{x}'\in\hat{\Omega}} \left\|p_k(\cdot|\hat{x}) - p_k(\cdot|\hat{x}')\right\|_{\mathrm{TV}}.$$

Then for the hidden Markov process $(X_i)_{1\le i\le n}$, we have

$$\bar{\eta}_{ij} \le \theta_i\theta_{i+1}\ldots\theta_{j-1}, \qquad \text{for } 1 \le i < j \le n.$$

Markov tree

If Ω is a finite set, a Markov tree measure μ is defined on Ω^n by a tree $T = (V, E)$ and transition kernels p_0, $\{p_{ij}(\cdot|\cdot)\}_{(i,j)\in E}$. Take $V = [n] := \{1, \ldots, n\}$. The topology of T and the transition kernels determine the measure μ on Ω^n:

$$\mu(x) = p_0(x_1) \prod_{(i,j)\in E} p_{ij}(x_j|x_i). \tag{E.7}$$

A measure on Ω^n satisfying Eq (E.7) for some tree T and $\{p_{ij}\}$ is said to be compatible with tree T; a measure is a Markov tree measure if it is compatible with some tree.

Suppose $\{X_i\}_{i\in\mathbb{N}}$, $(X_i \in \Omega)$ is a stochastic process defined on $(\Omega^{\mathbb{N}}, \mathbf{P})$. If for each $n > 0$ there is a tree $T^{(n)} = ([n], E^{(n)})$ and a Markov tree measure μ_n compatible with $T^{(n)}$ such that for all $x \in \Omega^n$ we have

$$\mathbf{P}\left\{X_1^n = x\right\} = \mu_n(x),$$

then we call X a Markov tree process.

For all $(u, v) \in E$, define (u, v)-contraction coefficient θ_{uv} by

$$\theta_{uv} = \max_{y,y'\in\Omega} \left\|p_{uv}(\cdot|y) - p_{uv}(\cdot|y')\right\|_{\mathrm{TV}}.$$

Suppose $\max_{(u,v)\in E}\theta_{uv} \le \theta < 1$ for some θ and the width[10] of the tree T is at most L[11]. Then for the Markov tree process X we have

$$\bar{\eta}_{ij} \;\le\; \left(1 - (1-\theta)^L\right)^{\lfloor (j-i)/L \rfloor}, \qquad \text{for all } 1 \le i < j \le n.$$

[10]The width of a tree is defined as the greatest number of nodes in each level/depth.

[11]In theory, the root of the tree can be arbitrarily chosen, which does affect the width. However, the sharpest bound is attained when the choice of root minimizes the width.

E.5 Concentration of measure with function space

So far, we have characterized random processes by using only one *fixed* function. However, machine learning often involves optimization over a space of functions, and hence it is crucial to study the concentration properties in conjunction with function spaces. This section will motivate the very fundamentals for *iid* observations, and the non-*iid* case is far from well studied in machine learning.

Suppose the instance space is $\mathcal{X}$ and the label space is $\mathcal{Y} = \{-1, 1\}$. We are given a sequence of labeled instances $(X_i, Y_i)_{1 \le i \le n}$ which are assumed to be *iid* for now. Let P be a distribution of $Z_i = (X_i, Y_i)$. Let $\mathcal{G}$ be a space of functions $g : \mathcal{X} \to \mathcal{Y}$ from which we search for an optimal classifier. The selection criteria is to minimize the expected risk:

$$R(g) = P\{g(X) \neq Y\}, \tag{E.8}$$

which can be estimated by the empirical risk

$$\bar{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \delta(f(X_i) \neq Y_i). \tag{E.9}$$

Treating $\delta(g(X_i) \neq Y_i)$ as a function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ of $Z_i = (X_i, Y_i)$, *i.e.*, $f(Z_i) = \delta(g(X_i) \neq Y_i)$, then Eq (E.8) and Eq (E.9) can be rewritten as

$$R(f) = \mathbb{E}[f(Z)] \qquad \text{and} \qquad \bar{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} f(Z_i). \tag{E.10}$$

and suppose the corresponding space of f is $\mathcal{F}$. Our goal is to minimize $R(f)$ with respect to f. But in practice, this can only be accomplished approximately via minimizing the empirical risk. Let f_n^* be the function (classifier/hypothesis) which minimizes the empirical risk

$$f_n^* = \operatorname*{arginf}_{f \in \mathcal{F}} \bar{R}_n(f).$$

A natural question to ask is how much $\bar{R}_n(f_n^*)$ deviates from $R(f_n^*)$. So we study its upper bound (uniform deviation)

$$\psi(Z_1^n) = \sup_{f \in \mathcal{F}} \left| R(f) - \bar{R}_n(f) \right| = \max \Big\{ \underbrace{\sup_{f \in \mathcal{F}} \{R(f) - \bar{R}_n(f)\}}_{\psi_+(Z_1^n)}, \ \underbrace{\sup_{f \in \mathcal{F}} \{\bar{R}_n(f) - R(f)\}}_{\psi_-(Z_1^n)} \Big\}.$$

We first bound $\psi_+(Z_1^n)$, and $\psi_-(Z_1^n)$ can be bounded similarly. A standard procedure which is also taken by Bousquet et al. (2005) is to invoke the McDiarmid's inequality noticing that Z_i causes at most $1/n$ change in $\psi_+(Z_1^n)$:

$$P\left\{\left|\psi_+(Z_1^n) - \mathbb{E}_{Z_1^n}\left[\psi_+(Z_1^n)\right]\right| > \epsilon\right\} \leq \exp\left(-2n\epsilon^2\right), \tag{E.11}$$

i.e., with probability at least $1-\delta$ we have

$$\left|\psi_+(Z_1^n) - \mathbb{E}_{Z_1^n}\left[\psi_+(Z_1^n)\right]\right| \leq \sqrt{\frac{\log(1/\delta)}{2n}}. \tag{E.12}$$

So the problem becomes bounding $\mathbb{E}_{Z_1^n}\left[\psi_+(Z_1^n)\right]$, which can be done by using Rademacher averages as in (Bousquet et al., 2005).

E.5.1 Rademacher averages

Define n *iid* binary random variables $\{\sigma_i\}_{i=1}^n$, where each σ_i has $\Pr(\sigma_i = 1) = \Pr(\sigma_i = -1) = 0.5$. Then a simple application of Jensen's inequality yields

$$\mathbb{E}_{Z_1^n}\left[\psi_+(Z_1^n)\right] = \mathbb{E}_{Z_1^n}\left[\sup_{f\in\mathcal{F}}\left\{\mathbb{E}\left[f(Z)\right] - \frac{1}{n}\sum_{i=1}^n f(Z_i)\right\}\right] \leq 2\,\underbrace{\mathbb{E}_{Z_1^n}\,\overbrace{\mathbb{E}_{\sigma_1^n}\left[\sup_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^n \sigma_i f(Z_i)\right]}^{:=R_n(\mathcal{F},Z_1^n)}}_{:=R_n(\mathcal{F})}, \tag{E.13}$$

where $R_n(\mathcal{F}, Z_1^n)$ and $R_n(\mathcal{F})$ are the empirical Rademacher average and (population) Rademacher average respectively (Mendelson, 2003).

Notice that in $R_n(\mathcal{F}, Z_1^n)$, the change of Z_i again causes at most $1/n$ variation, so McDiarmid's inequality can be applied again. Combining with Eq. (E.12), we conclude that with probability at least $1-\delta$:

$$\psi_+(Z_1^n) \leq 2\,R_n(\mathcal{F}) + \sqrt{\frac{\log(1/\delta)}{2n}} \quad \text{and} \quad \psi_+(Z_1^n) \leq 2\,R_n(\mathcal{F}, Z_1^n) + \sqrt{\frac{2\log(2/\delta)}{n}}.$$

As for $\psi_-(Z_1^n)$, Eq. (E.12) keeps intact (except changing $\psi_+(Z_1^n)$ to $\psi_-(Z_1^n)$). It is also not hard to show that Eq. (E.13) still holds for $\psi_-(Z_1^n)$. Since $0 \leq \psi(Z_1^n) = \max\left\{\psi_+(Z_1^n), \psi_-(Z_1^n)\right\}$, we conclude that with probability at least $1-\delta$:

$$\psi(Z_1^n) \leq 2\,R_n(\mathcal{F}) + \sqrt{\frac{\log(2/\delta)}{2n}}, \tag{E.14}$$

$$\psi(Z_1^n) \leq 2\,R_n(\mathcal{F}, Z_1^n) + \sqrt{\frac{2\log(4/\delta)}{n}}. \tag{E.15}$$

The bound in Eq. (E.14) uses $R_n(\mathcal{F})$ which depends solely on the function class $\mathcal{F}$, and can be bounded via the VC-dimension of $\mathcal{F}$. Bound in Eq. (E.15) is data dependent, which could be tighter than that in Eq. (E.14) if the samples are "typical".

It is interesting to examine $R_n(\mathcal{F}, Z_1^n)$ in more depth. First of all, it is independent of Y_i because

$$\sigma_i f(Z_i) = \sigma_i \delta(g(X_i) \neq Y_i) = \delta(g(X_i) \neq \sigma_i Y_i) + \frac{\sigma_i - 1}{2},$$

therefore

$$R_n(\mathcal{F}, Z_1^n) = \mathbb{E}_{\sigma_1^n}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_i \delta(g(X_i) \neq \sigma_i Y_i)\right] - \frac{1}{2} \tag{E.16}$$

$$= \mathbb{E}_{\sigma_1^n}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_i \delta(g(X_i) = \sigma_i Y_i)\right] - \frac{1}{2}. \tag{E.17}$$

In effect, σ_i flips the sign of Y_i. The expectation over all $\sigma_1^n \in \{-1, 1\}^n$ enumerates all the possible labelings of $X_1, \ldots, X_n$. Secondly, Eq. (E.16) shows that for each labeling one looks for the classifier g which maximizes the classification error. On the other hand, Eq. (E.17) shows that for each labeling one looks for the classifier g which maximizes the accuracy. These two interpretations look contradictory, but indeed they both characterize the complexity of the function space $\mathcal{F}$. Obviously, $\mathcal{F} \subseteq \mathcal{F}'$ implies $R_n(\mathcal{F}, Z_1^n) \leq R_n(\mathcal{F}', Z_1^n)$. Therefore, although a larger $\mathcal{F}$ lends more possibility in reducing the empirical risk, the Rademacher average grows and Eq. (E.14) and (E.15) show the higher risk that the empirical risk deviates from the true expected risk.

In general, f does not need to be restricted to the form of $\delta(g(X_i) \neq Y_i)$, and it can map from any space to $\mathbb{R}$. $\sum_{i=1}^n \sigma_i f(Z_i)$ measures how well the vector/direction $(\sigma_1, \ldots, \sigma_n)$ aligns with $(f(Z_1), \ldots, f(Z_n))$. This alignment is similar to our discussion in Section BUGBUG. More details can be found in (Bartlett & Mendelson, 2002).

E.5.2 Extension to non-*iid* observations

Unfortunately, little has been about the deviation $\psi(Z_1^n) = \sup_{f\in\mathcal{F}}\left|R(f) - \bar{R}_n(f)\right|$ when the process (samples) is non-*iid*. First, the analog of Eq. (E.11) can be simply derived from Eq. (E.4):

$$\mathbf{P}\left\{\psi_+(Z_1^n) - \mathbb{E}\psi_+(Z_1^n) > \epsilon\right\} \leq \exp\left(-\frac{n\epsilon^2}{4\left\|\triangle_n\right\|_\infty^2}\right).$$

Now the key difficulty is to generalize Eq. (E.13) to the non-*iid* case, *i.e.*, some kind of tailoring the definition of Rademacher average to the non-*iid* process. This is still an open question. Once this can be done, the bounds in Eq. (E.14) and (E.15) carry over directly.

Appendix F

Proof for Proposition 44

In this appendix, we give the detailed proof of why the function defined by Eq. (4.13) is not in the RKHS of k. For self-containedness, we copy to here the definitions in Proposition 44.

Consider a simple three node chain on $\mathbb{R}^3$ with cliques $\{\{Z_1, Z_2\}, \{Z_2, Z_3\}\}$,

$$Z_1 \text{ —— } Z_2 \text{ —— } Z_3$$

and let the kernel on the cliques be Gaussian:

$$k_{ij}((z_i, z_j), (z'_i, z'_j)) = \exp(-(z_i - z'_i)^2 - (z_j - z'_j)^2) \qquad \text{for } \{i, j\} = \{1, 2\} \text{ or } \{2, 3\},$$

and the joint kernel be $k_{12} + k_{23}$:

$$k(\mathbf{z}, \mathbf{z}') = k_{12}((z_1, z_2), (z'_1, z'_2)) + k_{23}((z_2, z_3), (z'_2, z'_3)).$$

Pick two functions $f_{12} \in \mathcal{H}_{12}$, $f_{23} \in \mathcal{H}_{23}$, and define f by

$$\begin{aligned} f_{12}(x_1, x_2) &:= 0, \\ f_{23}(x_2, x_3) &:= k_{23}(\mathbf{0}, (z_2, z_3)) = \exp(-z_2^2 - z_3^2), \\ f(\mathbf{z}) = f(z_1, z_2, z_3) &:= f_{12}(x_1, x_2) + f_{23}(x_2, x_3) = \exp(-z_2^2 - z_3^2). \end{aligned}$$

We show in the rest of the appendix that f is not in $\mathcal{H}$. To this end, we need the following lemma which will be proved in Section F.3 later.

Lemma 92 *Let k be a Gaussian kernel on $\mathbb{R}^2$ with RKHS $\mathcal{H}$. If $f \in \mathcal{H}$ and $f(x, y)$ is independent of x for all y, then $f = 0$.*

F.1 Finite linear combination

We first show that $f(\mathbf{z})$ is not in the linear span of $\{k((a, b, c), \mathbf{z}) : a, b, c \in \mathbb{R}\}$. Otherwise suppose that $f(\mathbf{z}) = \sum_{i=1}^n \alpha_i k((a_i, b_i, c_i), \mathbf{z})$, $\alpha_i \in \mathbb{R}$. Then

$$f(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i k((a_i, b_i, c_i), \mathbf{z}) = \underbrace{\sum_{i=1}^{n} \alpha_i k_{12}((a_i, b_i), (z_1, z_2))}_{:= f_{12}(z_1, z_2) \in \mathcal{H}_{12}} + \underbrace{\sum_{i=1}^{n} \alpha_i k_{23}((b_i, c_i), (z_2, z_3))}_{:= f_{23}(z_2, z_3) \in \mathcal{H}_{23}}$$

Since both $f(\mathbf{z})$ and f_{23} are independent of z_1, so is f_{12}. By Lemma 92, we must have $f_{12} = 0$, *i.e.*,

$$\sum_{i=1}^{n} \alpha_i k_{12}((a_i, b_i), (z_1, z_2)) = 0 \qquad \forall z_1, z_2 \qquad \text{(F.1)}$$

$$\sum_{i=1}^{n} \alpha_i k_{23}((b_i, c_i), (z_2, z_3)) = \exp(-z_2^2 - z_3^2) \qquad \forall z_2, z_3 \qquad \text{(F.2)}$$

Integrating out z_1, z_2 on both sides of Eq. (F.1) throughout $\mathbb{R}^2$, we derive $\sum_i a_i = 0$. Integrating out z_2, z_3 on both sides of Eq. (F.2) throughout $\mathbb{R}^2$, we derive $\sum_i a_i \neq 0$. Contradiction.

F.2 Limit of linear combination

Next, we need to show that including the limit point of the linear span does not solve the problem either. Suppose there is a sequence of functions in $\mathcal{H}$:

$$\left\{ f^n := \sum_{i=1}^{N_n} \alpha_i^n k((a_i^n, b_i^n, c_i^n), \cdot) \right\}_{n \in \mathbb{N}}$$

such that $f^n \xrightarrow{\mathcal{H}} f$. We rewrite

$$\begin{aligned} f^n(z_1, z_2, z_3) &= \sum_{i=1}^{N_n} \alpha_i^n k((a_i^n, b_i^n, c_i^n), (z_1, z_2, z_3)) \\ &= \underbrace{\sum_{i=1}^{N_n} \alpha_i^n k_{12}((a_i^n, b_i^n), (z_1, z_2))}_{:= f_{12}^n(z_1, z_2) \in \mathcal{H}_{12}} + \underbrace{\sum_{i=1}^{N_n} \alpha_i^n k_{23}((b_i^n, c_i^n), (z_2, z_3))}_{:= f_{23}^n(z_2, z_3) \in \mathcal{H}_{23}} \end{aligned}$$

It is easy to check that $\|f^n\|^2 = \|f_{12}^n\|^2 + \|f_{23}^n\|^2$. As $\{\|f^n\|\}_n$ is bounded, so is $\{\|f_{12}^n\|\}_n$. Hence $\{f_{12}^n\}_n$ must have a cluster point $f_{12}^* \in \mathcal{H}_{12}$, and a subsequence $\{f_{12}^{n_k}\}_k$ that converges to it. Without loss of generality, we assume that this subsequence is $\{f_{12}^n\}_n$ itself. Similarly, we can assume that $\{f_{23}^n\}_n$ converges to $f_{23}^* \in \mathcal{H}_{23}$. In the limit, we have $f_{12}^*(z_1, z_2) + f_{23}^*(z_2, z_3) = f(z_1, z_2, z_3)$. By Lemma 92, we must have

$f^*_{12} = 0$ and $f^*_{23} = \exp(-z_2^2 - z_3^2)$. In conjunction with Proposition 13, we obtain uniform convergence

$$\sum_{i=1}^{n} \alpha_i^n k_{12}((a_i^n, b_i^n), (z_1, z_2)) \rightrightarrows 0 \qquad \text{as } n \to \infty, \tag{F.3}$$

$$\sum_{i=1}^{n} \alpha_i^n k_{23}((b_i^n, c_i^n), (z_2, z_3)) \rightrightarrows \exp(-z_2^2 - z_3^2) \qquad \text{as } n \to \infty. \tag{F.4}$$

Uniform convergence allows us to integrate both sides of Eq. (F.3) and (F.4) throughout $\mathbb{R}^2$, which yields $\lim_{n\to\infty} \sum_{i=1}^n \alpha_i^n = 0$ and $\lim_{n\to\infty} \sum_{i=1}^n \alpha_i^n \neq 0$ respectively. Contradiction.

F.3 Proof of Lemma 92

Finally, we turn to prove Lemma 92. Our proof relies on an important theorem on the orthonormal basis of real Gaussian RKHS (Steinwart & Christmann, 2008, Theorem 4.42). It requires that the kernel be defined on a domain which contains an open set, and this is of course true for $\mathbb{R}^2$. Applying to our special case, the theorem says for all f in $\mathcal{H}$, there must exist a double-indexed array $\mathbf{b} = \{b_{i,j} \in \mathbb{R}\}_{i,j\in\mathbb{N}_0}$, such that

$$f(z_1, z_2) = \sum_{i,j\in\mathbb{N}_0} b_{i,j} \sqrt{\frac{2^{i+j}}{i! \cdot j!}} z_1^i z_2^j \exp\left(-z_1^2 - z_2^2\right), \tag{F.5}$$

$$\|\mathbf{b}\|_2^2 := \sum_{i,j\in\mathbb{N}_0} b_{i,j}^2 < \infty, \tag{F.6}$$

$$\|f\|_{\mathcal{H}} = \|\mathbf{b}\|_2 . \tag{F.7}$$

Denoting $a_i := \sqrt{\frac{2^i}{i!}}$, Eq. (F.5) gives:

$$\begin{aligned} f(z_1, z_2) \cdot \exp(z_2^2) &= \sum_{i,j\in\mathbb{N}_0} b_{i,j} a_i a_j z_1^i z_2^j \sum_{k=0}^{\infty} \frac{(-1)^k z_1^{2k}}{k!} \\ &= \sum_{i,j,k\in\mathbb{N}_0} b_{i,j} a_i a_j \frac{(-1)^k}{k!} z_1^{i+2k} z_2^j, \end{aligned}$$

Since the left hand side is independent of z_1, the identity theorem for the power series on the right hand side implies that the coefficient of $z_1^p z_2^q$ be 0 for any $p \geq 1$ and $q \geq 0$. This means that for all $j \in \mathbb{N}_0$, we have:

$z_1^0 z_2^j$: coefficient is $c_{0,j} := b_{0,j} a_0 a_j$,

$z_1^1 z_2^j$: coefficient is $c_{1,j} := b_{1,j} a_1 a_j = 0$

$z_1^2 z_2^j$: coefficient is $b_{2,j}a_2a_j - b_{0,j}a_0a_j = 0$, hence $c_{2,j} := b_{2,j}a_2a_j = b_{0,j}a_0a_j = c_{0,j}$.
$z_1^3 z_2^j$: coefficient is $b_{3,j}a_3a_j - b_{1,j}a_1a_j = 0$, hence $c_{3,j} := b_{3,j}a_3a_j = 0$
$z_1^4 z_2^j$: coefficient is $b_{4,j}a_4a_j - b_{2,j}a_2a_j + \frac{1}{2}b_{0,j}a_0a_j = 0$, hence $c_{4,j} := b_{4,j}a_4a_j = \frac{1}{2}c_{0,j}$.

Letting $c_{k,j} := b_{k,j}a_ka_j$, we will prove by induction that $c_{k,j} = 0$ if k is odd, and $c_{2k,j} = \frac{1}{k!}c_{0,j}$ for all $k \geq 0$.

Suppose $c_{2i-1,j} = 0$ holds for $i = 1, 2, \ldots, k$. Then we check the coefficient of $z_1^{2k+1}z_2^j$:

$$\begin{aligned}\text{coefficient} &= b_{2k+1,j}a_{2k+1}a_j - b_{2k-1,j}a_{2k-1}a_j + \frac{1}{2!}b_{2k-3,j}a_{2k-3}a_j - \frac{1}{3!}b_{2k-5,j}a_{2k-5}a_j + \ldots \\ &= c_{2k+1,j} - c_{2k-1,j} + \frac{1}{2!}c_{2k-3,j} - \frac{1}{3!}c_{2k-5,j} + \ldots \\ &= 0.\end{aligned}$$

So $c_{2k+1,j} = 0$.

Now we prove the result for $c_{2k,j}$. Suppose $c_{2i,j} = \frac{1}{i!}c_{0,j}$ for all $i \leq k$. Then we check the coefficient of $z_1^{2k+2}z_2^j$:

$$\begin{aligned}\text{coefficient} &= b_{2k+2,j}a_{2k+2}a_j - b_{2k,j}a_{2k}a_j + \frac{1}{2!}b_{2k-2,j}a_{2k-2}a_j - \frac{1}{3!}b_{2k-4,j}a_{2k-4}a_j + \ldots \\ &= c_{2k+2,j} - c_{2k,j} + \frac{1}{2!}c_{2k-2,j} - \frac{1}{3!}c_{2k-4,j} + \ldots \\ &= 0.\end{aligned}$$

Hence

$$\begin{aligned}c_{2k+2,j} &= c_{2k,j} - \frac{1}{2!}c_{2k-2,j} + \frac{1}{3!}c_{2k-4,j} - \ldots \\ &= c_{0,j} \cdot \left(\frac{1}{1!}\frac{1}{k!} - \frac{1}{2!}\frac{1}{(k-1)!} + \frac{1}{3!}\frac{1}{(k-2)!} - \ldots\right) \\ &= \frac{-c_{0,j}}{(k+1)!} \cdot \sum_{i=1}^{k+1}(-1)^i\binom{k+1}{i} \\ &= \frac{c_{0,j}}{(k+1)!},\end{aligned}$$

where the last step is based on the binomial expansion of $(1-1)^{k+1}$. Now we check the square norm of f: $\|f\|_{\mathcal{H}}^2 = \sum_{i,j} b_{i,j}^2$. First fix $j \in \mathbb{N}_0$, we have

$$\sum_{i \in \mathbb{N}_0} b_{i,j}^2 = \frac{c_{0,j}^2}{a_j^2}\sum_{i=0}^{\infty}\frac{(2i)!}{(i!)^2 2^{2i}}. \tag{F.8}$$

Unfortunately, the series diverges, which can be seen by using exactly the same argument in the proof of (Steinwart & Christmann, 2008, Theorem 4.42): denote $\alpha_i :=$

$\frac{(2i)!}{(i!)^2 2^{2i}}$, then

$$\frac{\alpha_{i+1}}{\alpha_i} = \frac{2i+1}{2i+2} > \frac{i}{i+1},$$

for all $i \geq 1$. So $i\alpha_i$ is an increasing positive sequence, and therefore there exists a positive constant C such that $\alpha_i \geq \frac{C}{i}$ for all $i \geq 1$. Hence $\sum_{i=0}^{\infty} \alpha_i = \infty$. Since $\|f\| < \infty$ and $0 < a_j < \infty$ for fixed j, Eq. (F.8) requires that $c_{0,j} = 0$ for all $j \in \mathbb{N}_0$, which means $b_{i,j} = 0$ for all $i, j \in \mathbb{N}_0$. So $f = 0$.

Appendix G

Incomplete Cholesky Decomposition for HSIC-Struct

With T observations and a $\omega \times \omega$ mixing matrix M, a naïve implementation of (4.27) costs $O(T^2\omega^2)$ computations and $O(T^2)$ memory to obtain $K^\star$. This is beyond normal capacity in computation and storage when T is large (*e.g.*, 67000 in our experiment). We resort to a low rank approximation of kernel matrix $K^\star$ via incomplete Cholesky decomposition Bach & Jordan (2002), *i.e.* approximating $K^\star$ by $PP^\top$ where matrix P is $T \times D$ with $D \ll T$. At each iteration d ($1 \leq d \leq D$), one column of $K^\star$ is needed which costs $O(T\omega^2)$, and the update of P costs $O(Td)$. So the total cost is $O(TD\omega^2 + TD^2)$ for computation and $O(TD)$ for storage. The short Matlab routine for incomplete Cholesky decomposition is given below:

```
 Input:
        Kstar : a n-by-n positive semi-definite matrix (we will only
                          query a small fraction of its elements)
        max_d : the maximum rank of the approximating matrix P
        tol   : the approximation tolerance
 Output:
        A matrix P (n-by-d) such that PP' approximates K.
        Either d = max_d, or the approximation error is below tol.

 d = 0;
 I = [];
 n = size(Kstar, 1);     % Kstar matrix is sized n-by-n

*buf = the diagonal of Kstar (n-by-1 vector);
 sum_buf = sum(buf);
 [nu, I(j+1)] = max(buf);
 P = zeros(1, n);
 while (sum_buf > tol && d < max_d)
     d = d + 1;
     nu = nu^(0.5);

*    K_col = the I(j)-th column of Kstar;
     P(:, d) = ( K_col - P * P( I(d), : )' ) ./ nu;

     buf = buf - P(:, d).^ 2;
     [nu, I(d+1)] = max(buf);

     sum_buf = sum(buf);
 end
```

In the two steps marked with ($\star$), we either need a column or a diagonal of the matrix. For $K^\star = K \star M$ (K convolved with M), each element $K^\star_{st}$ only depends on $\omega \times \omega$ values in K. This locality property of the convolution operation is critical for our efficient approximation algorithm to work.

Appendix H

Detailed proof for Theorem 57

For self-containedness, we repeat the notation used by Teo et al. (2010):

$$
\begin{aligned}
a_t &\in \partial_w R_{\text{emp}}(w_{t-1}) \\
b_t &= R_{\text{emp}}(w_{t-1}) - \langle w_{t-1}, a_t \rangle \\
A_t &= (a_1, \ldots, a_t) \\
\bar{b}_t &= (b_1, \ldots, b_t) \\
R_t^{\text{cp}}(w) &= \max_{1 \le i \le t} \langle w, a_i \rangle + b_i \\
J_t(w) &= \lambda \Omega(w) + R_t^{\text{cp}}(w) \\
J_t^*(\alpha) &= \lambda \Omega^*(-\lambda^{-1} A_t^\top \alpha) - \bar{b}_t \alpha \\
w_t &= \operatorname*{argmin}_w J_t(w) \\
\alpha_t &= \operatorname*{argmin}_{\alpha \in \Delta_t} J_t^*(\alpha).
\end{aligned}
$$

Note the dual connection is $w_t = -\lambda^{-1} A_t \alpha_t$.

In line search BMRM, the inner optimization problem

$$J_t^*(\alpha) = \lambda \Omega(-\lambda^{-1} A_t \alpha) - \bar{b}_t \alpha,$$

is subject to $\alpha = \alpha(\eta) = \left(\eta \alpha_{t-1}^\top, 1 - \eta\right)^\top$ with $\eta \in [0, 1]$. Let $\eta_t = \operatorname{argmin} J_t^*(\alpha(\eta))$, and $\alpha_t = \alpha(\eta_t)$. With some abuse of notation, we write $J_t^*(\eta) = J_t^*(\alpha(\eta))$.

H.1 Example Dataset and Initial Few Steps

Consider the following training examples in 1-d space. Let $x_i \in \mathbb{R}$ be features and $y_i \in \{-1, 1\}$ be labels. Pick $(x_1, y_1) = (1, 1)$, $(x_2, y_2) = (-1, -1)$, $(x_3, y_3) = (\frac{1}{2}, 1)$, and

$(x_4, y_4) = (-\frac{1}{2}, -1)$. Set $\lambda = \frac{1}{16}$. Then the objective function of SVM is:

$$J(w) = \frac{1}{32}w^2 + \frac{1}{2}[1-w]_+ + \frac{1}{2}\left[1 - \frac{w}{2}\right]_+ . \tag{H.1}$$

First we confirm that the optimal solution is $w^* = 2$. This can be seen by checking the subgradient of $J(w)$ at $w^* = 2$:

$$\partial J(2) = \frac{1}{16}2 - \frac{1}{2}\frac{1}{2}[0,1] \qquad \Rightarrow \qquad 0 \in \partial J(2).$$

And $J(w^*) = \frac{1}{8}$. Now the dual connection is $w_t = -16 A_t \alpha_t$, and $J_t^*(\alpha) = 8\left\|A_t \alpha\right\|^2 - \bar{b}_t \alpha$.

Let $w_0 = 0$. We now do a few iterations.

Step 1 $A_1 = a_1 = \left(-\frac{3}{4}\right)$, $\bar{b}_1 = b_1 = 1$, $\alpha_1 = 1$, $w_1 = -16\frac{-3}{4}1 = 12$.

Step 2 $a_2 = 0$, $A_2 = \left(-\frac{3}{4}, 0\right)$, $b_2 = 0$, $\bar{b}_2 = (1, 0)$. Let $\alpha = (\eta, 1-\eta)^\top$. Then

$$J_2^*(\eta) = 8\left(-\frac{3}{4}\eta\right)^2 - \eta = \frac{9}{2}\eta^2 - \eta$$

$$\Rightarrow \quad \eta_2 = \operatorname*{argmin}_{\eta\in[0,1]} J_2^*(\eta) = \frac{1}{9}, \qquad \alpha_2 = \left(\frac{1}{9}, \frac{8}{9}\right), \qquad w_2 = -16\frac{-3}{4}\frac{1}{9} = \frac{4}{3} \in (1, 2).$$

Step 3 $a_3 = -\frac{1}{4}$, $A_3 = \left(-\frac{3}{4}, 0, -\frac{1}{4}\right)$, $b_3 = \frac{1}{2}$, $\bar{b}_3 = (1, 0, \frac{1}{2})$. Let $\alpha = \left(\frac{1}{9}\eta, \frac{8}{9}\eta, 1-\eta\right)^\top$. Then

$$J_3^*(\eta) = 8\left(-\frac{3}{4}\frac{1}{9}\eta - \frac{1}{4}(1-\eta)\right)^2 - \frac{1}{9}\eta - \frac{1}{2}(1-\eta) = \frac{2}{9}\eta^2 - \frac{5}{18}\eta$$

$$\Rightarrow \quad \eta_3 = \operatorname*{argmin}_{\eta\in[0,1]} J_3^*(\eta) = \frac{5}{8}, \qquad \alpha_3 = \left(\frac{5}{72}, \frac{5}{9}, \frac{3}{8}\right)^\top, \qquad w_3 = -16\left(\frac{-3}{4}\frac{5}{72} + \frac{-1}{4}\frac{3}{8}\right) = \frac{7}{3} > 2.$$

Step 4 $a_4 = 0$, $A_4 = \left(-\frac{3}{4}, 0, -\frac{1}{4}, 0\right)$, $b_4 = 0$, $\bar{b}_4 = (1, 0, \frac{1}{2}, 0)$. Let $\alpha = \left(\frac{5}{72}\eta, \frac{5}{9}\eta, \frac{3}{8}\eta, 1-\eta\right)^\top$. Then

$$J_4^*(\eta) = 8\left(-\frac{3}{4}\frac{5}{72}\eta - \frac{1}{4}\frac{3}{8}\eta\right)^2 - \frac{5}{72}\eta - \frac{1}{2}\frac{3}{8}\eta = \frac{49}{288}\eta^2 - \frac{37}{144}\eta$$

$$\Rightarrow \quad \eta_4 = \operatorname*{argmin}_{\eta\in[0,1]} J_4^*(\eta) = \frac{37}{49}, \qquad \alpha_4 = \left(\frac{185}{3528}, \frac{185}{441}, \frac{111}{392}, \frac{12}{49}\right)^\top,$$

$$w_4 = -16\left(\frac{-3}{4}\frac{185}{3528} + \frac{-1}{4}\frac{111}{392}\right) = \frac{37}{21} \in (1, 2).$$

Step 5 $a_5 = -\frac{1}{4}$, $b_5 = 1$.

In general, if $w_{t-1} \in (1, 2)$, then $a_t = -\frac{1}{4}$, and $b_t = 1$. If $w_{t-1} > 2$, then $a_t = 0$, and $b_t = 0$. As we have seen, $w_t \in (1, 2)$ for t being even, and $w_t > 2$ for t being odd. We will show that this is true indeed: w_t oscillates around 2 and approaches 2 in both directions.

H.2 Asymptotic Rates

Theorem 93 gives recursive formulae of w_t and $\alpha_{t,1}$ (the first element of α_t).

Theorem 93 (Theorem 59 in text) *For $k \geq 1$, we have*

$$w_{2k+1} = 2\frac{w_{2k-1}^3 + 12\alpha_{2k-1,1}w_{2k-1}^2 + 16w_{2k-1}\alpha_{2k-1,1}^2 - 64\alpha_{2k-1,1}^3}{w_{2k-1}\left(w_{2k-1} + 4\alpha_{2k-1,1}\right)^2} > 2, \tag{H.2}$$

$$\alpha_{2k+1,1} = \frac{w_{2k-1}^2 + 16\alpha_{2k-1,1}^2}{\left(w_{2k-1} + 4\alpha_{2k-1,1}\right)^2}\alpha_{2k-1,1}, \tag{H.3}$$

$$w_{2k} = 2 - \frac{8\alpha_{2k-1,1}}{w_{2k-1}} \in (1, 2). \tag{H.4}$$

(H.2) and (H.3) provide recursive formulae to compute w_{2k+1} and $\alpha_{2k+1,1}$ based on w_{2k-1} and $\alpha_{2k-1,1}$, and (H.4) gives w_{2k}.

The proof will be given in the section H.3. By (H.2) and (H.4), we have:

$$2 - w_{2k} = 8\frac{\alpha_{2k-1,1}}{w_{2k-1}} \tag{H.5}$$

$$w_{2k+1} - 2 = 8\frac{\alpha_{2k-1,1}(w_{2k-1} - 4\alpha_{2k-1,1})}{w_{2k-1}(w_{2k-1} + 4\alpha_{2k-1,1})}. \tag{H.6}$$

BMRM guarantees

$$\lim_{k\to\infty} w_k = 2, \qquad \text{hence} \qquad \lim_{k\to\infty} \alpha_{2k-1,1} = 0.$$

We will show in Theorem 94 that $\alpha_{2k-1,1}$ tends to 0 at the rate of $1/k$, and hence by (H.5) and (H.6), $|2 - w_k|$ approaches 0 at the same rate. This further implies, as will be shown in Theorem 95, that $J(w_k) - J(w^*)$ goes to 0 at the rate of $1/k$ as well.

Theorem 94 (Theorem 58 in text) $\lim_{k\to\infty} k\alpha_{2k-1,1} = \frac{1}{4}$. *Hence by (H.5) and (H.6), we have* $\lim_{k\to\infty} k\,|2 - w_k| = 2$.

Proof The proof is based on (H.3). Let $\beta_k = 1/\alpha_{2k-1,1}$, then $\lim_{k\to\infty} \beta_k = \infty$ because $\lim_{k\to\infty} \alpha_{2k-1,1} = 0$. Now

$$\lim_{k\to\infty} k\alpha_{2k-1,1} = \left(\lim_{k\to\infty} \frac{1}{k\alpha_{2k-1,1}}\right)^{-1} = \left(\lim_{k\to\infty} \frac{\beta_k}{k}\right)^{-1} = \left(\lim_{k\to\infty} \beta_{k+1} - \beta_k\right)^{-1},$$

where the last step is by the discrete version of L'Hospital's rule[1].

$\lim_{k\to\infty} \beta_{k+1} - \beta_k$ can be computed by plugging definition $\beta_k = 1/\alpha_{2k-1,1}$ into (H.3), which gives:

$$\frac{1}{\beta_{k+1}} = \frac{w_{2k}^2 + 16\frac{1}{\beta_k^2}}{\left(w_{2k} + 4\frac{1}{\beta_k}\right)^2}\frac{1}{\beta_k} \quad \Rightarrow \quad \beta_{k+1} - \beta_k = 8\frac{w_{2k}\beta_k^2}{w_{2k}^2\beta_k^2 + 16} = 8\frac{w_{2k}}{w_{2k}^2 + \frac{16}{\beta_k^2}}.$$

Since $\lim_{k\to\infty} w_k = 2$ and $\lim_{k\to\infty} \beta_k = \infty$, so

$$\lim_{k\to\infty} k\alpha_{2k-1,1} = \left(\lim_{k\to\infty} \beta_{k+1} - \beta_k\right)^{-1} = \frac{1}{4}.$$

■

Theorem 95 (Theorem 57 in text) $\lim_{k\to\infty} k(J(w_k) - J(w^*)) = \frac{1}{4}$.

Proof Let $\epsilon_k = 2 - w_k$, then $\lim_{k\to\infty} k\,|\epsilon_k| = 2$ by Theorem 94.

If $\epsilon_k > 0$, then $J(w_k) - J(w^*) = \frac{1}{32}(2-\epsilon_k)^2 + \frac{1}{2}\frac{\epsilon_k}{2} - \frac{1}{8} = \frac{1}{8}\epsilon_k + \frac{1}{32}\epsilon_k^2 = \frac{1}{8}\,|\epsilon_k| + \frac{1}{32}\epsilon_k^2$.

If $\epsilon_k \leq 0$, then $J(w_k) - J(w^*) = \frac{1}{32}(2-\epsilon_k)^2 - \frac{1}{8} = -\frac{1}{8}\epsilon_k + \frac{1}{32}\epsilon_k^2 = \frac{1}{8}\,|\epsilon_k| + \frac{1}{32}\epsilon_k^2$.

Combining these two cases, we conclude $\lim_{k\to\infty} k(J(w_k) - J(w^*)) = \frac{1}{4}$. ■

H.3 Proof of Theorem 93

We prove Theorem 93 by induction. Obviously, Theorem 93 holds for $k = 1$. Suppose Theorem 93 holds for indices up to some $k-1$ ($k \geq 2$). Let $p = 2k-1$ ($p \geq 3$). Then

$$A_p = \left(-\frac{3}{4}, 0, -\frac{1}{4}, \ldots, 0, -\frac{1}{4}\right), \qquad \bar{b}_p = \left(1, 0, \frac{1}{2}, \ldots, 0, \frac{1}{2}\right).$$

[1]http://hermes.aei.mpg.de/arxiv/05/04/034/article.xhtml

So

$$w_p = -16A_p\alpha_p = (-16)\left(-\frac{3}{4}\alpha_{p,1} - \frac{1}{4}\alpha_{p,3} - \frac{1}{4}\alpha_{p,5} - \ldots - \frac{1}{4}\alpha_{p,p-2} - \frac{1}{4}\alpha_{p,p}\right)$$
$$\Rightarrow \quad \alpha_{p,3} + \ldots + \alpha_{p,p-2} + \alpha_{p,p} = \frac{w_p}{4} - 3\alpha_{p,1}.$$

So

$$\bar{b}_p\alpha_p = \alpha_{p,1} + \frac{1}{2}\alpha_{p,3} + \frac{1}{2}\alpha_{p,5} + \ldots + \frac{1}{2}\alpha_{p,p-2} + \frac{1}{2}\alpha_{p,p} = \frac{1}{8}w_p - \frac{1}{2}\alpha_{p,1}$$

Since $w_p > 2$, so $a_{p+1} = 0$, $b_{p+1} = 0$. So $A_{p+1} = (A_p, 0)$, $\bar{b}_{p+1} = (\bar{b}_p, 0)$. Let $\alpha_{p+1} = (\eta\alpha_p, 1-\eta)$, then $J^*_{p+1}(\eta) = 8\eta^2(A_p^\top\alpha_p)^2 - \eta\bar{b}_p\alpha_p$. So

$$\eta_{p+1} = \frac{\bar{b}_p\alpha_p}{16\,(A_p\alpha_p)^2} = \frac{2w_p - 8\alpha_{p,1}}{w_p^2}, \tag{H.7}$$
$$w_{p+1} = -16A_p\alpha_p\eta_{p+1} = w_p\eta_{p+1} = 2 - \frac{8\alpha_{p,1}}{w_p} < 2.$$

Since $\alpha_{2,1} = \frac{1}{9}$, $p \geq 3$, and $\alpha_{k,1} \geq \alpha_{k+1,1}$ due to the update rule of ls-bmrm, we have

$$8\alpha_{p,1} \leq \frac{8}{9} < 2 < w_p, \tag{H.8}$$

hence $w_{p+1} > 1$.

Next step, since $w_{p+1} \in (1, 2)$, so $a_{p+2} = -\frac{1}{4}$, $b_{p+2} = \frac{1}{2}$, $A_{p+2} = \left(A_p, 0, -\frac{1}{4}\right)$, $\bar{b}_{p+1} = \left(\bar{b}_p, 0, \frac{1}{2}\right)$. Let $\alpha_{p+2}(\eta) = (\eta\eta_{p+1}\alpha_t, \eta(1-\eta_{p+1}), 1-\eta)$. Then

$$A_{p+2}\alpha_{p+2} = \eta\eta_{p+1}A_p\alpha_p - \frac{1}{4}(1-\eta), \qquad \bar{b}_{p+2}\alpha_{p+2} = \eta\eta_{p+1}\bar{b}_p\alpha_p + \frac{1}{2}(1-\eta).$$

So

$$\begin{aligned} J^*_{p+2}(\eta) &= 8(A_{p+2}\alpha_{p+2})^2 - \bar{b}_{p+2}\alpha_{p+2} \\ &= \frac{1}{2}\left(4\eta_{p+1}A_p\alpha_p + 1\right)^2\eta^2 - \left(4\eta_{p+1}A_p\alpha_p + \eta_{p+1}\bar{b}_p\alpha_p + \frac{1}{2}\right)\eta + const, \end{aligned}$$

where the const means terms independent of η. So

$$\eta_{p+2} = \operatorname*{argmin}_{\eta\in[0,1]} J^*_{p+2}(\eta) = \frac{4\eta_{p+1}A_p\alpha_p + \eta_{p+1}\bar{b}_p\alpha_p + \frac{1}{2}}{\left(4\eta_{p+1}A_p\alpha_p + 1\right)^2\eta_{p+1}^2} = \frac{w_p^2 + 16\alpha_{p,1}^2}{(w_p + 4\alpha_{p,1})^2}. \tag{H.9}$$

and

$$\begin{aligned} w_{p+2} &= -16A_{p+2}\alpha_{p+2} = -16\eta_{p+2}\eta_{p+1}A_p\alpha_p + 4(1-\eta_{p+2}) \\ &= 2\frac{w_p^3 + 12\alpha_{p,1}w_p^2 + 16w_p\alpha_{p,1}^2 - 64\alpha_{p,1}^3}{w_p\left(w_p + 4\alpha_{p,1}\right)^2}, \end{aligned}$$

where the last step is by plugging (H.7) and (H.9). Now check

$$w_{p+2} - 2 = \frac{8\alpha_{p,1}(w_p - 4\alpha_{p,1})}{w_p(w_p + 4\alpha_{p,1})} > 0,$$

where the last step is due to (H.8).

Bibliography

Aaronson, J., Burton, R., Dehling, H., Gilat, D., Hill, T., & Weiss, B. (1996). Strong laws for L and U-statistics. *Transactions of the American Mathematical Society*, *348*, 2845–2865.

Afonja, B. (1972). The moments of the maximum of correlated normal and t-variates. *Journal of the Royal Statistical Society. Series B*, *34*(2), 251–262.

Altun, Y., Hofmann, T., & Smola, A. J. (2004a). Gaussian process classification for segmenting and annotating sequences. In *Proc. Intl. Conf. Machine Learning*, 25–32. New York, NY: ACM Press.

Altun, Y., & Smola, A. (2006). Unifying divergence minimization and statistical inference via convex duality. In H. Simon, & G. Lugosi, eds., *Proc. Annual Conf. Computational Learning Theory*, LNCS, 139–153. Springer.

Altun, Y., Smola, A. J., & Hofmann, T. (2004b). Exponential families for conditional random fields. In *Uncertainty in Artificial Intelligence (UAI)*, 2–9. Arlington, Virginia: AUAI Press.

Amari, S.-i. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*(2), 251–276.

Amdahl, G. (1967). Validity of the single processor approach to achieving large-scale computing capabilities. *30*, 483–485.

Andrew, G., & Gao, J. (2007). Scalable training of l_1-regularized log-linear models. In *Proc. Intl. Conf. Machine Learning*, 33–40. New York, NY, USA: ACM.

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine Learning*, *50*, 5–43.

Aronszajn, N. (1950). Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, *68*, 337–404.

Bach, F. R., & Jordan, M. I. (2002). Kernel independent component analysis. *J. Mach. Learn. Res.*, *3*, 1–48.

Bagnell, J. A., & Ng, A. Y. (2006). On local rewards and scaling distributed reinforcement learning. In *Proc. NIPS'2005*, vol. 18.

Barahona, F., & Mahjoub, A. R. (1986). On the cut polytope. *Mathematical Programming*, *36*(2), 157–173.

Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, *101*(473), 138–156.

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, *3*, 463–482.

Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Beck, A., & Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, *31*(3), 167–175.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2000). The complexity of decentralized control of markov decision processes. In *UAI 16*.

Bertsekas, D. (1976). On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, *21*(2), 174–184.

Bollen, J. A. M. (1984). Numerical stability of descent methods for solving linear equations. *Numerische Mathematik*.

Bordes, A., Bottou, L., & Gallinari, P. (2009). SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, *10*, 1737–1754.

Bordes, A., Ertekin, S., Weston, J., & Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, *6*, 1579–1619.

Borgwardt, K., & Ghahramani, Z. (2009). Bayesian two-sample tests. Tech. Rep. 04-104. Http://arxiv.org/abs/0906.4032.

Borovkova, S., Burton, R., & Dehling, H. (1999). Consistency of the takens estimator for the correlation dimension. *Annals of Applied Probability*, *9*(2), 376–390.

Borovkova, S., Burton, R., & Dehling, H. (2001). Limit theorems for functionals of mixing processes with applications to dimension estimation. *Transactions of the American Mathematical Society*, *353*(11), 4261–4318.

Borwein, J. M., & Lewis, A. S. (2000). *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS books in Mathematics. Canadian Mathematical Society.

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In D. Haussler, ed., *Proc. Annual Conf. Computational Learning Theory*, 144–152. Pittsburgh, PA: ACM Press.

Bottou, L. (1991). *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole.* Ph.D. thesis, Université de Paris XI, Orsay, France.

Bottou, L., & Bousquet, O. (2007). The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, & S. Roweis, eds., *NIPS*. MIT Press. http://books.nips.cc/papers/files/nips20/NIPS2007_0726.pdf.

Bottou, L., & LeCun, Y. (2004). Large scale online learning. In S. Thrun, L. Saul, & B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*, 217–224. Cambridge, MA: MIT Press.

Bousquet, O., Boucheron, S., & Lugosi, G. (2005). Theory of classification: a survey of recent advances. *ESAIM: Probab. Stat.*, *9*, 323– 375.

Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *IJCAI*, 478–485.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization.* Cambridge, England: Cambridge University Press.

Brown, L. D. (1986). *Fundamentals of Statistical Exponential Families*, vol. 9 of *Lecture notes-monograph series.* Hayward, Calif: Institute of Mathematical Statistics.

Candes, E., & Tao, T. (2005). Decoding by linear programming. *IEEE Trans. Info Theory*, *51(12)*, 4203–4215.

Casella, G., & Robert, C. P. (1996). Rao-blackwellisation of sampling schemes. *Biometrika*, *83*(1), 81–94.

Catanzaro, B., Sundaram, N., & Keutzer, K. (2008). Fast support vector machine training and classification on graphics processors. In *Proc. Intl. Conf. Machine Learning.*

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Incremental algorithms for hierarchical classification. *J. Mach. Learn. Res.*, *7*, 31–54.

Cheney, E. W., & Goldstein, A. A. (1959). Newton's method for convex programming and tchebycheff approximation. *Numerische Mathematik*, *1*(1), 253–268.

Cheng, L., Vishwanathan, S. V. N., Schuurmans, D., Wang, S., & Caelli, T. (2006). Implicit online learning with kernels. In B. Schölkopf, J. Platt, & T. Hofmann, eds., *Advances in Neural Information Processing Systems 19*. Cambridge MA: MIT Press.

Cheng, L., Vishwanathan, S. V. N., & Zhang, X. (2008). Consistent image analogies using semi-supervised learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. Anchorage, AK.

Chu, C., Kim, S., Lin, Y. A., Yu, Y. Y., Bradski, G., Ng, A., & Olukotun, K. (2007). Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt, & T. Hofmann, eds., *Advances in Neural Information Processing Systems 19*.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In T. G. Dietterich, S. Becker, & Z. Ghahramani, eds., *Advances in Neural Information Processing Systems 14*, 625–632. Cambridge, MA: MIT Press.

Collins, M., Globerson, A., Koo, T., Carreras, X., & Bartlett, P. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *J. Mach. Learn. Res.*, *9*, 1775–1822.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, *42*, 393–405.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, *7*, 551–585.

Crammer, K., Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2003). Online passive-aggressive algorithms. In S. Thrun, L. Saul, & B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*. MIT Press.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, *3*, 951–991.

Dai, Y.-H., & Fletcher, R. (2006). New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming: Series A and B archive*, *106*(3), 403–421.

Dangauthier, P., Herbrich, R., Minka, T., & Graepel, T. (2008). Trueskill through time: Revisiting the history of chess. In *NIPS*.

Dekel, O., Keshet, J., & Singer, Y. (2004). Large margin hierarchical classification. In *Proc. Intl. Conf. Machine Learning*.

Dobson, A. J., & Barnett, A. (2008). *Introduction to Generalized Linear Models.* Texts in Statistical Science. Chapman & Hall/CRC, 3 edn.

Doucet, A., de Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice.* Springer-Verlag.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandrae, T. (2008). Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. Intl. Conf. Machine Learning.*

Duchi, J., & Singer, Y. (2009). Online and batch learning using forward-backward splitting. *J. Mach. Learn. Res.* Accepted.

Dutech, A., et al. (2005). Proc of reinforcement learning benchmarks and bake-offs II. In *NIPS 2005 Workshops.*

Elisseeff, A., & Weston, J. (2001). A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 681–687. Cambridge, MA: MIT press.

Enflo, P. (1973). *A Counterexample to the Approximation Property in Banach Spaces.* Acta Mathematica.

Fan, R.-E., & Lin, C.-J. (2007). A study on threshold selection for multi-label classification. Tech. rep., National Taiwan University. Http://www.csie.ntu.edu.tw/ cjlin/papers/threshold.pdf.

Ferris, M. C., & Munson, T. S. (2000). Interior point methods for massive support vector machines. Data Mining Institute Technical Report 00-05, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.

Ferris, M. C., & Munson, T. S. (2002). Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, *13*(3), 783–804.

Fukumizu, K., Sriperumbudur, B., Gretton, A., & Schölkopf, B. (2009). Characteristic kernels on groups and semigroups. In *Advances in Neural Information Processing Systems 21.*

Ghamrawi, N., & McCallum, A. (2005). Collective multi-label classification. In *CIKM.*

Graepel, T., Herbrich, R., & Shawe-Taylor, J. (2000). Generalisation error bounds for sparse linear classifiers. In *Proc. Annual Conf. Computational Learning Theory*, 298–303.

Graf, H. P., Cosatto, E., Bottou, L., Durdanovic, I., & Vapnik, V. (2004). Parallel support vector machines: The cascade SVM. In *Neural Information Processing Systems*.

Gretton, A., Bousquet, O., Smola, A., & Schölkopf, B. (2005a). Measuring statistical dependence with Hilbert-Schmidt norms. In S. Jain, H. U. Simon, & E. Tomita, eds., *ALT*, 63–77. Springer-Verlag.

Gretton, A., Fukumizu, K., Teo, C.-H., Song, L., Schölkopf, B., & Smola, A. (2008). A kernel statistical test of independence. Tech. Rep. 168, MPI for Biological Cybernetics.

Gretton, A., Herbrich, R., Smola, A., Bousquet, O., & Schölkopf, B. (2005b). Kernel methods for measuring independence. *J. Mach. Learn. Res.*, *6*, 2075–2129.

Grünwald, P. D. (2007). *The Minimum Description Length Principle*. MIT Press.

Guestrin, C., Lagoudakis, M., & Parr, R. (2002). Coordinated reinforcement learning. In *ICML*.

Guo, Y. G., Bartlett, P. L., Shawe-Taylor, J., & Williamson, R. C. (1999). Covering numbers for support vector machines. In *Proc. Annual Conf. Computational Learning Theory*.

Hammersley, J. M., & Clifford, P. E. (1971). Markov fields on finite graphs and lattices. Unpublished manuscript.

Hamze, F., & de Freitas, N. (2004). From fields to trees. In *Uncertainty in Artificial Intelligence (UAI)*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. New York: Springer, 2 edn.

Haussler, D. (1999). Convolutional kernels on discrete structures. Tech. Rep. UCSC-CRL-99-10, Computer Science Department, UC Santa Cruz.

Herbrich, R., Minka, T., & Graepel, T. (2007). Trueskill$^{\text{TM}}$: A Bayesian skill ranking system. In *NIPS*.

Hestenes, M. R., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, *49*(6), 409–436.

Hildreth, C. (1957). A quadratic programming procedure. *Naval Research Logistics Quarterly*, *4*, 79–85.

Hiriart-Urruty, J., & Lemaréchal, C. (1993a). *Convex Analysis and Minimization Algorithms, I and II*, vol. 305 and 306. Springer-Verlag.

Hiriart-Urruty, J., & Lemaréchal, C. (1993b). *Convex Analysis and Minimization Algorithms, I and II*, vol. 305 and 306. Springer-Verlag.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

Hosseni, S., & Jutten, C. (2003). On the separability of nonlinear mixtures of temporally correlated sources. *IEEE Signal Processing Letters*, *10*(2), 43–46.

Hsieh, C. J., Chang, K. W., Lin, C. J., Keerthi, S. S., & Sundararajan, S. (2008a). A dual coordinate descent method for large-scale linear SVM. In W. Cohen, A. McCallum, & S. Roweis, eds., *ICML*, 408–415. ACM.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., & Sundararajan, S. (2008b). A dual coordinate descent method for large-scale linear SVM. In A. McCallum, & S. Roweis, eds., *ICML*, 408–415. Omnipress.

Ihler, A. T., III, J. W. F., & Willsky, A. S. (2005). Loopy belief propagation: Convergence and effects of message errors. *J. Mach. Learn. Res.*, *6*, 905–936.

Iusem, A. N., & Pierro, A. R. D. (1990). On the convergence properties of Hildreth's quadratic programming algorithm. *Mathematical Programming*, *47*, 37–51.

Jansche, M. (2007). A maximum expected utility framework for binary sequence labeling. In *ACL*.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, & A. J. Smola, eds., *Advances in Kernel Methods — Support Vector Learning*, 169–184. Cambridge, MA: MIT Press.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proc. Intl. Conf. Machine Learning*, 377–384. San Francisco, California: Morgan Kaufmann Publishers.

Joachims, T. (2006). Training linear SVMs in linear time. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM.

Joachims, T., Finley, T., & Yu, C.-N. (2009). Cutting-plane traning of structural SVMs. *Machine Learning*, *76*(1).

Karvanen, J. (2005). A resampling test for the total independence of stationary time series: Application to the performance evaluation of ICA algorithms. *Neural Processing Letters*, *22*(3), 311 – 324.

Kearns, M. (1998). Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, *45*(6), 983–1006.

Kearns, M., Littleman, M., & Singh, S. (2001). Graphical models for game theory. In *Conference on Uncertainty in Artificial Intelligence*.

Keerthi, S. S., & Gilbert, E. G. (2002). Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, *46*, 351–360.

Kelley, J. (1960). The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, *8*, 703–712.

Kim, H.-C., & Ghahramani, Z. (2006). Bayesian gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(12), 1948–1959.

Kimeldorf, G. S., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, *33*, 82–95.

Kivinen, J., & Warmuth, M. K. (1995). Additive versus exponentiated gradient updates for linear prediction. In *Proc. 27th Annual ACM Symposium on Theory of Computing*, 209–218. ACM Press, New York, NY.

Kiwiel, K. C. (1985). *Methods of Descent for Nondifferentiable Optimization*.

Kiwiel, K. C. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, *46*, 105–122.

Kiwiel, K. C. (2000). Efficiency of proximal bundle methods. *Journal of Optimization Theory and Applications*, *104*(3), 589–603.

Koh, K., Kim, S.-J., & Boyd, S. (2006). An interior-point method for large-scale ℓ_1-regularized logistic regression. *J. Mach. Learn. Res.* Submitted.

Kontorovich, L. (2007). *Measure Concentration of Strongly Mixing Processes with Applications*. Ph.D. thesis, CMU.

Kschischang, F., Frey, B. J., & Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2), 498–519.

Lafferty, J. D., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning*, vol. 18, 282–289. San Francisco, CA: Morgan Kaufmann.

Lauritzen, S. L. (1996). *Graphical Models.* Oxford, UK: Oxford University Press.

Learned-Miller, E. G. (2004). Hyperspacing and the estimation of information theoretic quantities. Tech. Rep. 04-104, Department of Computer Science, University of Massachusetts. Http://www.cs.umass.edu/ elm/papers/04-104.pdf.

Ledoux, M. (2001). *The Concentration of Measure Phenomenon.* Providence, RI: AMS.

Lehmann, E. (1983). *Theory of Point Estimation.* New York: John Wiley and Sons.

Lemaréchal, C., Nemirovskii, A., & Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, *69*, 111–147.

Lewis, D. (2001). Applying support vector machines to the TREC-2001 batch filtering and routing tasks. In *Proc. TREC.*

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, *5*, 361–397.

Lugosi, G. (2006). Concentration of measure inequalities. http://www.econ.upf.es/~lugosi/anu.ps.

Luo, Z. Q., & Tseng, P. (1992). On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, *72*(1), 7–35.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press.

Maybeck, P. S. (1982). *Stochastic Models, Estimation and Control.* Academic Press.

McCallum, A. (1999). Multi-label text classification with a mixture model trained by EM. In *AAAI Workshop.*

Mendelson, S. (2003). A few notes on statistical learning theory. In S. Mendelson, & A. J. Smola, eds., *Advanced Lectures on Machine Learning*, no. 2600 in LNAI, 1–40. Heidelberg, Germany: Springer-Verlag.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, *A 209*, 415–446.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., & Müller, K.-R. (1999). Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, & S. Douglas, eds., *Neural Networks for Signal Processing IX*, 41–48. IEEE.

Minka, T. (2001). *Expectation Propagation for approximative Bayesian inference.* Ph.D. thesis, MIT Media Labs, Cambridge, USA.

Minka, T. (2005). Divergence measures and message passing. Report 173, Microsoft Research.

Minka, T., & Winn, J. (2009). Gates. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou, eds., *Advances in Neural Information Processing Systems 21*, 1073–1080.

Moreau, J. J. (1965). Proximite et dualite dans un espace hilbertien. *Bull. Soc. Math. Fr.*, *93*, 273–299.

Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany.*

Musicant, D. R., Kumar, V., & Ozgur, A. (2003). Optimizing f-measure with support vector machines. In *International Florida Artificial Intelligence Research Society Conference.*

Nedic, A. (2002). *Subgradient Methods for Convex Minimization.* Ph.D. thesis, MIT.

Nemenman, I., Shafee, F., & Bialek, W. (2002). Entropy and inference, revisited. In *Neural Information Processing Systems*, vol. 14. Cambridge, MA: MIT Press.

Nemirovski, A. (2009). Personal communications.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Soviet Math. Docl.*, *269*, 543–547.

Nesterov, Y. (2003). *Introductory Lectures On Convex Optimization: A Basic Course.* Springer.

Nesterov, Y. (2005a). Excessive gap technique in nonsmooth convex minimization. *SIAM J. on Optimization*, *16*(1), 235–249. ISSN 1052-6234.

Nesterov, Y. (2005b). Smooth minimization of non-smooth functions. *Math. Program.*, *103*(1), 127–152.

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Tech. Rep. 76, CORE Discussion Paper, UCL.

Ng, A., Jordan, M., & Weiss, Y. (2002). Spectral clustering: Analysis and an algorithm (with appendix). In T. G. Dietterich, S. Becker, & Z. Ghahramani, eds., *Advances in Neural Information Processing Systems 14*.

Nguyen, X., Wainwright, M., & Jordan, M. (2008). Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *NIPS 20*. MIT Press.

Nocedal, J., & Wright, S. J. (1999). *Numerical Optimization*. Springer Series in Operations Research. Springer.

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization*. Springer Series in Operations Research. Springer, 2nd edn.

Ong, C. S., Mary, X., Canu, S., & Smola, A. J. (2004). Learning with non-positive kernels. In *Proc. Intl. Conf. Machine Learning*.

Opper, M. (1998). A Bayesian approach to online learning. In *On-line Learning in Neural Networks*, 363–378. Cambridge University Press.

Opper, M., & Winther, O. (2000). Gaussian processes and SVM: Mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, & D. Schuurmans, eds., *Advances in Large Margin Classifiers*, 311–326. Cambridge, MA: MIT Press.

Pardalos, P. M., & Kovoor, N. (1990). An algorithm for singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, *46*, 321–328.

Peshkin, L., Kim, K.-E., Meuleau, N., & Kaelbling, L. P. (2000). Learning to cooperate via policy search. In *UAI*.

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. In *Machine Learning: ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, 280–291. Springer.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, & A. J. Smola, eds., *Advances in Kernel Methods — Support Vector Learning*, 185–208. Cambridge, MA: MIT Press.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. Rep. MSR-TR-98-14, Microsoft Research.

Quadrianto, N., Song, L., & Smola, A. (2009). Kernelized sorting. In *Advances in Neural Information Processing Systems 22*.

Rao, C. R. (1973). *Linear Statistical Inference and its Applications*. New York: John Wiley and Sons.

Richter, S., Aberdeen, D., & Yu, J. (2007). Natural actor-critic for road traffic optimization. In B. Schölkopf, J. Platt, & T. Hofmann, eds., *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press.

Robbins, H. E., & Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, *22*, 400–407.

Robinson, S. M. (1999). Linear convergence of epsilon-subgradient descent methods for a class of convex functions. *Mathematical Programming*, *86*(1), 41–50.

Rockafellar, R. T. (1970). *Convex Analysis*, vol. 28 of *Princeton Mathematics Series*. Princeton, NJ: Princeton University Press.

Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence*, *82*, 273–302.

Rousu, J., Sunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification methods. *Journal of Machine Learning Research*, *7*, 1601–1626.

Schneider, J., Wong, W.-K., Moore, A., & Riedmiller, M. (1999). Distributed value functions. In *Proc. Intl. Conf. Machine Learning*, 371–378. Morgan Kaufmann, San Francisco, CA.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels*. Cambridge, MA: MIT Press.

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1996). Nonlinear component analysis as a kernel eigenvalue problem. Tech. Rep. 44, Max-Planck-Institut für biologische Kybernetik.

Schramm, H., & Zowe, J. (1992). A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM J. Optimization*, *2*, 121–152.

Serfling, R. (1980). *Approximation Theorems of Mathematical Statistics.* New York: Wiley.

Shalev-Schwartz, S., & Srebro, N. (2008). SVM optimization: Inverse dependence on training set size. In W. Cohen, A. McCallum, & S. Roweis, eds., *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008).* Omnipress.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM. In *Proc. Intl. Conf. Machine Learning.*

Shawe-Taylor, J., & Dolia, A. (2007). A framework for probability density estimation. In M. Meila, & X. Shen, eds., *Proceedings of International Workshop on Artificial Intelligence and Statistics.*

Shen, H., Jegelka, S., & Gretton, A. (2009). Fast kernel-based independent component analysis. *IEEE Transactions on Signal Processing.* In press.

Shor, N. (1985). *Minimization methods for non-differentiable functions.* Springer-Verlag, Berlin.

Sinha, D., Zhou, H., & Shenoy, N. V. (2006). Advances in computation of the maximum of a set of random variables. In *7th International Symposium on Quality Electronic Design.*

Smola, A., Gretton, A., Song, L., & Schölkopf, B. (2007a). A hilbert space embedding for distributions. In E. Takimoto, ed., *Algorithmic Learning Theory*, Lecture Notes on Computer Science. Springer.

Smola, A., Vishwanathan, S. V. N., & Le, Q. (2007b). Bundle methods for machine learning. In D. Koller, & Y. Singer, eds., *Advances in Neural Information Processing Systems 20.* Cambridge MA: MIT Press.

Smola, A. J., Gretton, A., Song, L., & Schölkopf, B. (2007c). A Hilbert space embedding for distributions. In *Proc. Intl. Conf. Algorithmic Learning Theory*, vol. 4754 of *LNAI*, 13–31. Springer-Verlag.

Song, L., Smola, A., Borgwardt, K., & Gretton, A. (2008a). Colored maximum variance unfolding. In *Advances in Neural Information Processing Systems 20*, 1385–1392. Cambridge, MA: MIT Press.

Song, L., Smola, A., Gretton, A., Bedo, J., & Borgwardt, K. (2007a). Feature selection via dependence maximization. *J. Mach. Learn. Res.* Accepted with minor revisions.

Song, L., Smola, A., Gretton, A., & Borgwardt, K. (2007b). A dependence maximization view of clustering. In *ICML*, 815–822. Omnipress.

Song, L., Smola, A., Gretton, A., Borgwardt, K., & Bedo, J. (2007c). Supervised feature selection via dependence estimation. In *ICML*, 823–830. Omnipress.

Song, L., Zhang, X., Smola, A., Gretton, A., & Schölkopf, B. (2008b). Tailoring density estimation via reproducing kernel moment matching. In *ICML*.

Sontag, D., & Jaakkola, T. (2007). New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems 21*.

Sriperumbudur, B., Gretton, A., Fukumizu, K., Lanckriet, G., & Schölkopf, B. (2008). Injective hilbert space embeddings of probability measures. In *Proceedings of the 21st Annual Conference on Learning Theory*, 111–122.

Steinwart, I. (2001). On the generalization ability of support vector machines. Tech. rep., University of Jena.

Steinwart, I., & Christmann, A. (2008). *Support Vector Machines.* Information Science and Statistics.

Stögbauer, H., Kraskov, A., Astakhov, S., & Grassberger, P. (2004). Least dependent component analysis based on mutual information. *Phys. Rev. E*, *70*(6), 066123.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, & K.-R. Müller, eds., *Advances in Neural Information Processing Systems 12*, 1057–1063. Cambridge, MA: MIT Press.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In S. Thrun, L. Saul, & B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*, 25–32. Cambridge, MA: MIT Press.

Taskar, B., Lacoste-Julien, S., & Jordan, M. (2006). Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, *7*, 1627–1653.

Teo, C., Le, Q., Smola, A., & Vishwanathan, S. (2007). A scalable modular convex solver for regularized risk minimization. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM.

Teo, C. H., & Vishwanathan, S. V. N. (2006). Fast and space efficient string kernels using suffix arrays. In *ICML '06: Proceedings of the 23rd international conference on*

Machine learning, 929–936. New York, NY, USA: ACM Press. ISBN 1-59593-383-2. doi:http://doi.acm.org/10.1145/1143844.1143961.

Teo, C. H., Vishwanthan, S. V. N., Smola, A. J., & Le, Q. V. (2010). Bundle methods for regularized risk minimization. *J. Mach. Learn. Res.*, *11*, 311–365.

Theocharous, G., Murphy, K., & Kaelbling, L. (2004). Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *ICRA*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, *58*, 267–288.

Tikhonov, A. N. (1943). On the stability of inverse problems. *Dokl. Akad. Nauk SSSR*, *39*(5), 195–198.

Tikhonov, A. N. (1963). Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, *4*, 1035–1038.

Toussaint, M. (2009). Probabilistic inference as a model of planned behavior. Tech. rep., T. U. Berlin.

Tseng, P., & Yun, S. (2008). A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications.* To Appear.

Tseng, P., & Yun, S. (2009). A block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, *140*(3), 513–535.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, *6*, 1453–1484.

Tsypkin, Y. Z. (1971). *Adaptation and Learning in Automatic Systems.* Academic Press.

Vanderbei, R. J. (2008). *Linear Programming: Foundations and Extensions.* 3 edn.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* New York: Springer.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, I. R., & Borgwardt, K. (2009). Graph kernels. *J. Mach. Learn. Res.* Submitted.

Wainwright, M. (2002). *Stochastic processes on graphs with cycles: geometric and variational approaches.* Ph.D. thesis, EECS MIT.

Wainwright, M., & Jordan, M. I. (2006). Log-determinant relaxation for approximate inference in discrete Markov random fields. *IEEE Transactions on Signal Processing*, *54*(6), 2099–2109.

Wainwright, M. J., Jaakkola, T. S., & Willsky, A. S. (2003). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, *49*(5), 1120–1146.

Wainwright, M. J., Jaakkola, T. S., & Willsky, A. S. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, *51*(7), 2313–2335.

Wainwright, M. J., & Jordan, M. I. (2003). Graphical models, exponential families, and variational inference. Tech. Rep. 649, UC Berkeley, Department of Statistics.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, *1*(1 – 2), 1 – 305.

Weiss, Y. (2000). Correctness of local probability propagation in graphical models with loops. *Neural Computation*, *12*, 1–41.

Weiss, Y. (2001). Comparing the mean field method and belief propagation for approximate inference in MRFs. In D. Saad, & M. Opper, eds., *Advanced Mean Field Methods*. MIT Press.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Yang, Y. (2001). A study on thresholding strategies for text categorization. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*.

Yosida, K. (1964). *Functional Analysis*.

Zhang, X., Aberdeen, D., & Vishwanathan, S. V. N. (2007). Conditional random fields for multi-agent reinforcement learning. In *Proc. Intl. Conf. Machine Learning*.

Zhang, X., Graepel, T., & Herbrich, R. (2010a). Bayesian online learning for multi-label and multi-variate performance measures. In Y. W. Teh, & M. Titterington, eds., *Proc. Intl. Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Zhang, X., Saha, A., & Vishwanathan, S. V. N. (2010b). Lower bounds on rate of convergence of cutting plane methods. Cambridge MA: MIT Press.

Zhang, X., Song, L., Gretton, A., & Smola, A. (2009). Kernel measures of independence for non-iid data. In *Advances in Neural Information Processing Systems 21*.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. Intl. Conf. Machine Learning*, 912–919.

Ziehe, A., & Müller, K.-R. (1998). TDSEP – an efficient algorithm for blind separation using time structure. In *Proc. Intl. Conf. Artificial Neural Networks*, 675–680.