


Max-margin Methods

Notation

	Notation	Example
Inputs	\mathbf{x}^i	 images $(\mathbf{v}_1^T, \mathbf{v}_2^T, \dots, \mathbf{v}_5^T)^T$, where (coded) \mathbf{v}_1 is a vector of grey level for letter 'b', etc.
Number of Input <i>micro-labels</i> (length)	L_i	5 (can be tricky for CFG). Assumption: segmented.
Input Space	\mathcal{X}	The space of images.
True outputs aka <i>macro-label</i>	\mathbf{y}^i	the word 'brace'
Output Space	\mathcal{Y}	{all possible words} = $\{A^n \mid n \text{ is a natural number}\}$, where A is the alphabet. Note: \mathcal{Y} is NOT restricted to 5-letter words. n can be 10, 1, etc.
Any output (candidate)	\mathbf{y}	any element of \mathcal{Y} , e.g., 'bcace', 'brare', 'lrace', or even 'is', 'country' which are shorter or longer than 5 letters.

We have a set of training examples $\{(\mathbf{x}^i, \mathbf{y}^i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \dots, l\}$. For multi-class tasks which do NOT have structured outputs, we also conveniently adopt the above notation (viewing as if $L_i \equiv 1$). The connotation of \mathcal{Y} and \mathbf{y} are independent of \mathbf{x}^i . But in the following we implicitly assume they are restricted once \mathbf{x}^i is given. For example, in the above 'brace' case, we will restrict \mathcal{Y} to 5-letter words. So it is better to write \mathcal{Y}^i .

Part I. Data representation and model assumption

Representing the data in a disposable form is a critical design task. For binary classification, we simply write the input \mathbf{x} as a real vector. For multi-class classification (including structured output tasks), the data representation is rather tricky and depends on model assumption.

Model assumption means the pool of models we want to investigate. If the optimal separator is $\mathbf{w}^T \mathbf{x}$, then we can never achieve good performance if we restrict our attention only to functions in $\{\sin(\mathbf{w}^T \mathbf{x})\}$. Below we want to restrict our classifier to the following form:

$$f(\mathbf{x}^i; \mathbf{w}) = \arg \max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}^i, \mathbf{y}; \mathbf{w}) = \arg \max_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, f(\mathbf{x}^i, \mathbf{y}) \rangle$$

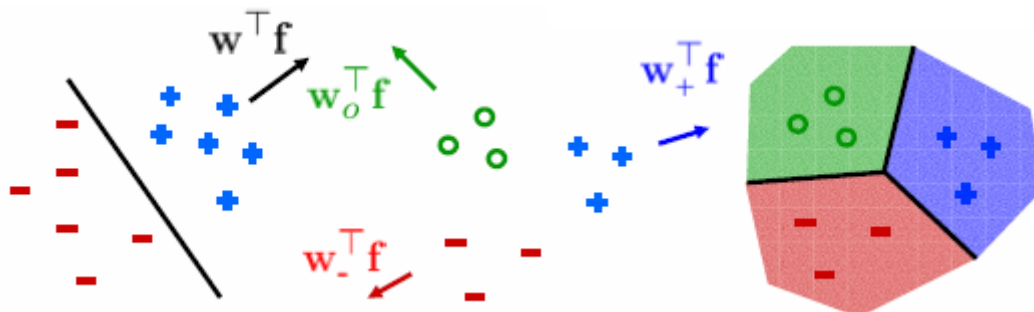
\mathbf{w} is a parameter vector. This means the following process:

1. Map each pair of $(\mathbf{x}^i, \mathbf{y})$ maps to a feature vector $f(\mathbf{x}^i, \mathbf{y})$ according to a certain predefined rule (this rule is problem specific and is not to be learned *here*);
2. Compute a score $F(\mathbf{x}^i, \mathbf{y}; \mathbf{w}): \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$, which is the sum of features values weighted by \mathbf{w} , i.e., $\langle \mathbf{w}, f(\mathbf{x}^i, \mathbf{y}) \rangle$ (the \mathbf{w} is to be learned from training examples);
3. the classifier's prediction is the \mathbf{y} which yields the highest score.

Example.

$$\begin{aligned}
 \mathbf{f}_i(\text{POLITICS}) &= [0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0] \\
 \mathbf{f}_i(\text{SPORTS}) &= [1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0] \\
 \mathbf{f}_i(\text{OTHER}) &= [0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0] \\
 \mathbf{w} &= [1 & 1 & -1 & -2 & 1 & -1 & 1 & -2 & -2 & -1 & -1 & 1] \\
 \\
 \mathbf{f}(\mathbf{x}, \mathbf{y}) &= [0 & 0 & \dots & \mathbf{f}(\mathbf{x}) & \dots & 0] \\
 \mathbf{w} &= [\mathbf{w}_0 & \mathbf{w}_1 & \dots & \mathbf{w}_y & \dots & \mathbf{w}_k]
 \end{aligned}$$

This will result in the actual rule $f(\mathbf{x}^i; \mathbf{w}) = \arg \max_{y \in \mathcal{Y}} \langle \mathbf{w}_y, f(\mathbf{x}^i) \rangle$. Intuitively:



If the task is binary, we usually only need feature vector $f(x)$ and compare $w^T f(x)$ with 0.

We also take use $c_i = 1$ or -1 to represent that x_i is positive or negative respectively.

More about representation in CFG...

Part II. Max-margin philosophy [copied partly from Bousquet et al., 2003]

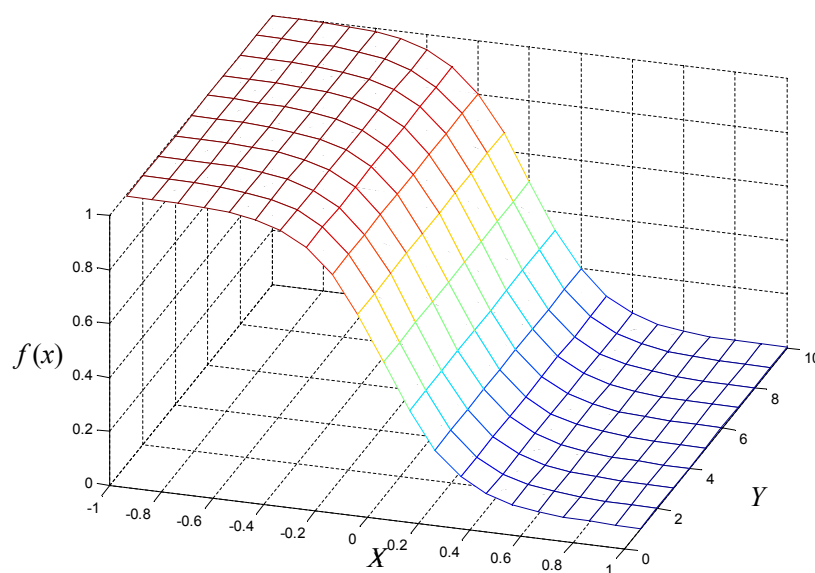
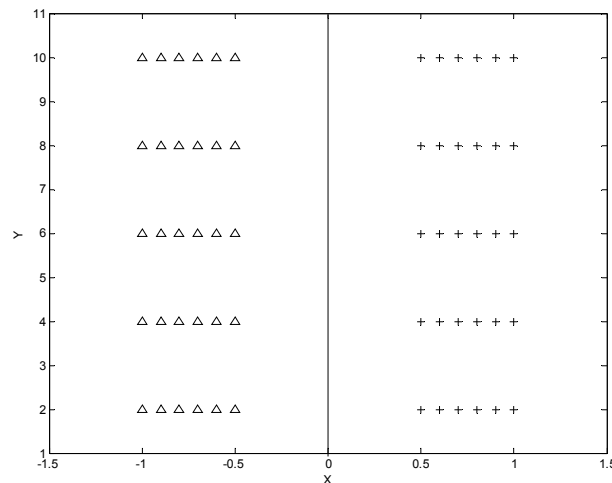
Most existing learning algorithms perform a trade-off between fit of the data (to be discussed in Part III) and 'complexity' of the solution. The way this complexity is defined varies from one algorithm to another and is usually referred to as a prior probability or a regularizer. The choice of this term amounts to having a preference for certain solutions and there is no a priori best such choice since it depends on the learning problem to be addressed. This means that the right choice should be dictated by prior knowledge or assumptions about the problem or the class of problems to which the algorithm is to be applied. Let us consider the binary

classification setting. A typical assumption that is (at least implicitly) used in many learning algorithms is the following

Two points that are close in input space should have the same label.

One possible way to enforce this assumption is to look for a decision function which is consistent with the training data and which *does not change too much between neighboring points*. This can be done in a regularization setting, using the Lipschitz norm as a regularizer. For differentiable functions, the Lipschitz norm of a function is the supremum of the norm of the gradient. We want a classifier function f , which produces low training loss and is smooth at training points. That is, we do not want $\|\nabla f\|$ (∇f stands for the gradient of f) be large at training points. It is thus natural to consider algorithms of the form

$$\min_f \sup_x \|\nabla f(x)\|^2, \text{ under the constraint that training data is well fit} \quad (1)$$



Performing such a minimization on the set of linear functions leads to the maximum margin solution, whereas the 1-nearest neighbor decision function is one of the solutions of the above optimization problem when the set of functions is unconstrained.

If we restrict f to linear functions $w^T x + b$. Then it means we want to minimize

$$\sup_i \|\nabla f(x_i)\|^2 + C \sum_{i=1}^l \text{loss}(w^T x_i + b, c_i)$$

But as $\nabla f \equiv w$ (independent of x), this is equivalent to minimize

$$\frac{1}{2} w^T w + C \sum_{i=1}^l \text{loss}(w^T x_i + b, c_i)$$

Substituting hinge loss (or any other loss) for $\text{loss}(\cdot)$, we derive SVM.

Although very useful because widely applicable, the above assumption is sometimes too weak. Indeed, most 'real-world' learning problems have more structure than what this assumption captures. For example, most data is located in regions where the label is constant (clusters) and regions where the label is not well-defined are typically of low density. This can be formulated via the so-called cluster assumption:

Two points that are connected by a line that goes through high density regions should have the same label.

Another related way of stating this assumption is to say that the decision boundary should lie in regions of low density. Assumption 1 slightly implies Assumption 2.

How do we implement these assumptions? Now suppose we restrict our classifiers to linear forms. Intuitively, the assumptions imply that data points should be as far away from decision boundary as possible.

1. We know a directed hyperplane (a hyperplane with normal direction) is uniquely decided by $w^T x + b = 0$ where $\|w\| = 1$. I.e., there is a bijection between {directed hyperplanes} and $\{w^T x + b = 0 \mid \|w\| = 1\}$. So restricting ourselves in the world of $\|w\| = 1$ will not cause any loss of candidate hyperplane.

2. We also know that a point x 's distance to a hyperplane $w^T x + b = 0$ is $|w^T x + b| / \|w\|$.

Combining 1 and 2, we can safely add the constraint $\|w\| = 1$ and say that the larger

$|w^T x_i + b|$ is, the farther is x_i from the decision boundary. So similar to (1), we want to

maximize $\min_i |w^T x_i + b|$ over w and b . Also note that besides x_i we also have labels c_i ,

which will decide on which side of the hyperplane x_i is. In the best case when the dataset is linear separable, we can maximize $\min_i |w^T x_i + b|$ under the constraint that $c_i (w^T x_i + b) \geq 0$. We call $2 \cdot \min_i |w^T x_i + b|$ the **margin** of dataset \mathcal{X} with respect to hyperplane $w^T x + b = 0$ ($\|w\|=1$), usually denoted as ρ . Maximizing the margin leads to the following formulation:

1. Fix/Constrain the norm of weight vector w (QCQP)

$$\begin{aligned} \text{Max (wrt } w, b, \gamma) \quad & \gamma & (2) \\ \text{s.t.} \quad & \|w\| \leq 1 \text{ (or } \|w\| = 1) \\ & c_i (w^T x_i + b) \geq \gamma & \text{for } \forall i \end{aligned}$$

Finally, $\|w\|$ must be 1 (why?). Note if the dataset is linearly separable, γ will be automatically optimized to non-negative (otherwise, flip the sign of w and b).

2. Fix the functional margin to 1, then minimize the norm of the weight vector w (QP)

Mathematically, (2) is equivalent to (QP) (recall $\text{distance}(x, \text{plane}) = |w^T x + b| / \|w\|$):

$$\begin{aligned} \text{Min (wrt } w, b): \quad & \frac{1}{2} w^T w \\ \text{s.t.} \quad & c_i (w^T x_i + b) \geq 1 \text{ for } \forall i \end{aligned}$$

Note although in the linearly separable case, there always exist w and b which satisfies $c_i (w^T x_i + b) \geq 0$ under the constraint $\|w\|=1$, there may not be any w and b which satisfy $c_i (w^T x_i + b) \geq 1$ for $\forall i$. But in the new form, we no longer restrict $\|w\|=1$.

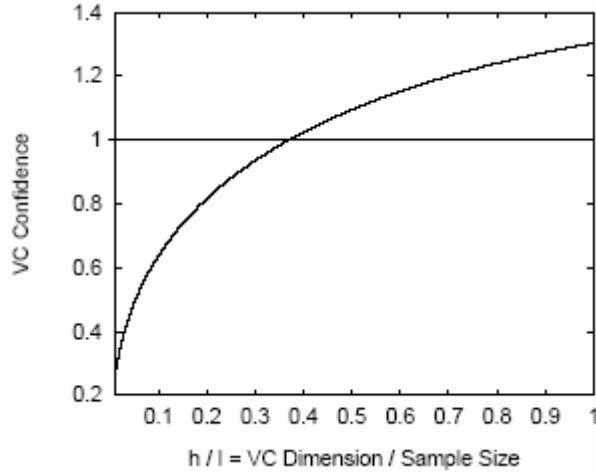
In Vapnik's statistical machine learning theory, it says that the optimal solution has VC

dimension h bounded from above $h \leq \min \left\{ \left\lceil \frac{(\text{diameter of dataset})^2}{\rho^2} \right\rceil \right\} + 1$.

Let $R(w, b) = \int \frac{1}{2} |y - f(x, w, b)| dP(x, y)$, $R_{emp}(w, b) = \frac{1}{2l} \sum_{i=1}^l |y_i - f(x_i, w, b)|$

Then, $R(w, b) \leq R_{emp}(w, b) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}}$ with probability

$1 - \eta$.

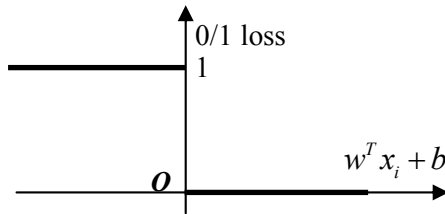


So the larger ρ , the smaller h , and the smaller $R(w, b)$.

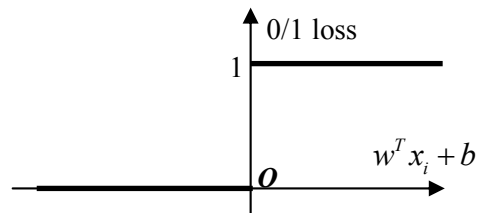
Finally we realize we can not count on the linear separability. So once there does not exist such a perfect linear boundary, we must find a trade-off between maximizing the

margin and fitting the training data (soft-margin): $\frac{1}{2} w^T w + C \sum_{i=1}^l \text{loss}(w^T x_i + b, c_i)$. (3)

One hard loss is 0/1 loss: $\text{loss}(w^T x_i + b, c_i) = \mathbf{1}(c_i \cdot (w^T x_i + b) \geq 0)$, where $\mathbf{1}(\text{true}) = 1$, $\mathbf{1}(\text{false}) = 0$ (indicator function).

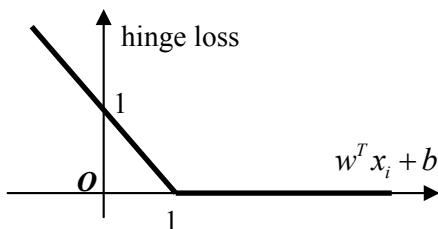


$c_i = 1$ (true class is positive)

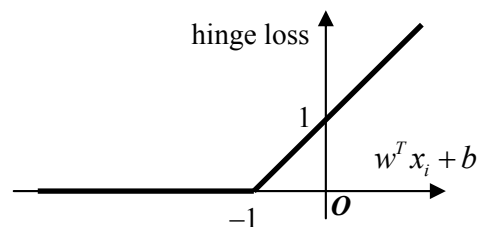


$c_i = -1$ (true class is negative)

However this will make optimization problem (3) NP-complete, so we have to approximate the 0/1 loss by other choices, such as hinge loss l^H .



$c_i = 1$ (true class is positive)



$c_i = -1$ (true class is negative)

Formally, we can write $\frac{1}{2}w^T w + C \sum_{i=1}^l l^H(w^T x_i + b, c_i)$ into nicer form: (interpret it!)

Primal: (4)

$$\text{Min (wrt } w, b, \xi): \frac{1}{2}w^T w + C \sum_{i=1}^N \xi_i$$

$$s.t. \quad c_i (w^T x_i + b) \geq 1 - \xi_i \quad \text{for } \forall i$$

$$\xi_i \geq 0 \quad \text{for } \forall i$$

Dual: (5)

$$\text{Max (wrt } \alpha): \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j d_i d_j x_i^T x_j$$

$$s.t. \quad \sum_{i=1}^l \alpha_i d_i = 0, \quad \alpha_i \in [0, C] \quad \text{for } \forall i$$

Once we solve the dual problem with α_i^* , we will have $w^* = \sum_{i=1}^l \alpha_i^* d_i x_i$, and the determi-

nant hyperplane is $w^{*T} x + b = \sum_{i=1}^l \alpha_i^* d_i x_i^T x + b$.

3. Motivating from the dual directly. Kernel interpolation (not the Mercer Kernel).

Assume that the discriminant function is $f_\alpha(x) = \sum_{i=1}^l \alpha_i c_i k(x_i, x)$ ($c_i \in \{-1, 1\}$, binary for

simplicity), where k can be a Gaussian kernel: $k(x_i, x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\|x_i - x\|^2 / \sigma^2)$, re-

stricting that $\alpha_i > 0$. If the cost function is: $\min_{\alpha \geq 0} \sum_{i=1}^l (f_\alpha(x_i) - c_i)^2$, then it becomes Ra-

dial Basis Neural Network. It is know to be able to fit all c_i perfectly, as Gaussian kernel matrix $k(x_i, x_j)$ is always positive definite (thus invertible). So people add regularization

terms like $\min_{\alpha \geq 0} \left(\|\alpha\|^2 + C \sum_{i=1}^l (f_\alpha(x_i) - c_i)^2 \right)$. Then people say, maybe hinge loss is better

than square loss and they change the form into: $\min_{\alpha \geq 0} \left(\|\alpha\|^2 + C \sum_{i=1}^l l^H(f_\alpha(x_i), c_i) \right)$, which is

formally as: $\min \|\alpha\|^2 + C \sum_{i=1}^l \xi_i$

$$s.t. \quad c_i \cdot \sum_{j=1}^l \alpha_j d_j k(x_i, x_j) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, \quad \alpha_i \geq 0 \quad \text{for all } i.$$

This is already very similar to SVM, except that it used $\|\alpha\|^2$ as regularizer. The story that

finally introduces $\sum_{i,j=1}^l \alpha_i \alpha_j c_i c_j k(x_i, x_j)$ is as follows. $\|\alpha\|^2$ means that we adopt a Gaussian prior of α 's distribution (zero mean, covariance matrix being identity matrix). Now we use a new prior of α which kind-of reflects the conformability of $k(x_i, x_j)$ and their labels c_i and c_j . We assume that the prior of α is a zero-mean Gaussian distribution with inverse covariance matrix V : $V_{ij} = c_i c_j k(x_i, x_j)$. The purpose of inversion will be shown later. Adding another trivial constraint $\alpha_i \in [0, C]$, the optimization problem is now formulated as:

$$\begin{aligned} \text{Min} \quad & \sum_{i,j=1}^l \alpha_i \alpha_j c_i c_j k(x_i, x_j) + C \sum_{i=1}^l \xi_i \\ \text{s.t.} \quad & c_i \cdot \sum_{j=1}^l \alpha_j d_j k(x_i, x_j) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, \alpha_i \in [0, C] \quad \text{for all } i. \end{aligned}$$

This is exactly the same as (4) mathematically (except a bias term).

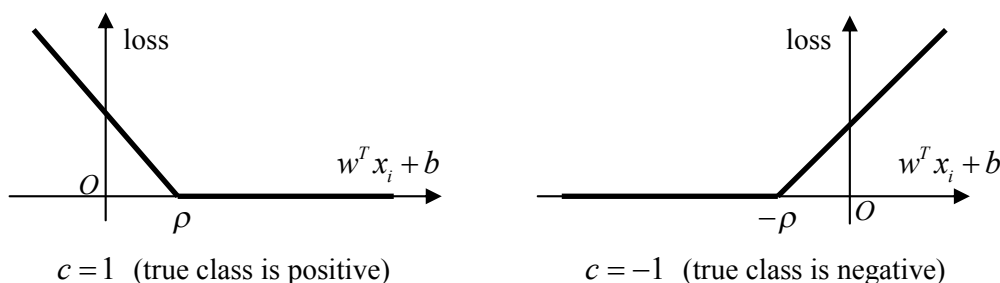
Note the regularization term is usually exerting opposite influence of loss function. If we interpret $\sum_{i,j=1}^l \alpha_i \alpha_j c_i c_j k(x_i, x_j) = \sum_{i=1}^l \alpha_i \cdot c_i f_\alpha(x_i)$, then minimizing it means encouraging the model prediction $f_\alpha(x_i)$ to be of different sign from c_i (the correct label).

Another way to view dual cost function: training error + regularizer.

4. One important variant

$$(\nu\text{-SVM}) \quad \min_{w,b,\rho \geq 0} \frac{1}{2} w^T w + C \sum_{i=1}^N l^H(w^T x_i + b, c_i, \rho) - \nu \rho \quad (C, \nu \text{ are user specified})$$

The new hinge loss function is:



ν -SVM allows changeable function margin. ν is an upper bound on the fraction of margin

errors, a lower bound on the fraction of support vectors, and that both of these quantities approach ν asymptotically. Formally,

$$\begin{aligned} \text{Min (wrt } w, b, \xi, \rho): \quad & \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i \\ \text{s.t.} \quad & c_i (w^T x_i + b) \geq \rho - \xi_i \quad \text{for } \forall i \\ & \xi_i \geq 0 \quad \text{for } \forall i \quad \text{and} \quad \rho \geq 0 \end{aligned}$$

ν is an important parameter, and there is also some algorithms that learn ν .

Part III. Choice of loss function and multi-class SVM

Loss function is often used in a confusing way because people use it to refer to two different things: goodness of sequence label probability distribution given by the model (e.g. hinge loss) vs. difference measure of two sequences (e.g., Hamming distance). The most distinctive difference is the latter requires the w vector and a probability model (e.g., CRF, logistic regression) to calculate, whereas the former only depends on the tuple (x^i, y^i, y) . Usually, the

latter is positive when $y \neq y^i$ and 0 if $y = y^i$. For mathematical convenience, it is desirable to be decomposable on cliques. This measure is relatively simple and thus we focus on the former so-called loss function.

Since for multi-class SVM, the regularizer is still $w^T w$ as in binary case, the only difference is the loss function. So discussing the loss function is exactly discussing multi-class SVM. We will also see some loss functions for non-max-margin classifiers (e.g., CRF) for comparison. This part borrows a lot from (Altun 03 EMNLP).

1. What is the numerical form of the loss function ?

log loss, hinge loss, exp loss, 0/1 loss, ...

2. What is graph components (e.g., sequence) are involved in loss function calculation?

sequence, cliques, single node, sub-tree, ...

We first look at some loss functions defined on the whole sequence.

1. 0/1 loss for multi-class: $\sum_{i=1}^l \text{step} \left(w^T f(x^i, y^i) - \max_{y \neq y^i} w^T f(x^i, y) \right)$

Corresponding margin = $w^T f(x^i, y^i) - \max_{y \neq y^i} w^T f(x^i, y)$ Discontinuous, NP-complete.

2. hinge loss. $\sum_{i=1}^l \max \left(0, 1 - \max_{y \neq y^i} \left(w^T f(x^i, y^i) - w^T f(x^i, y) \right) \right)$, which checks whether

the $w^T f(x^i, y^i)$ is larger than the max of the rest $w^T f(x^i, y)$ by more than 1.

Max-margin models are built upon hinge loss.

$$3. \text{ log loss (soft-max): } \sum_{i=1}^l \left(-w^T f(x^i, y^i) + \log \sum_y \exp(w^T f(x^i, y)) \right) = \sum_{i=1}^l \log(1 + e^{-H_i}),$$

where

$$H_i = w^T f(x^i, y^i) - \log \sum_{y \neq y^i} \exp(w^T f(x^i, y)) = w^T f(x^i, y^i) - \text{softMax}(w^T f(x^i, y))$$

since $\log(\exp(x) + \exp(y)) \approx \max(x, y)$ called soft max. This loss function is motivated

by assuming that $P(y | x, w) = \frac{\exp(w^T f(x, y))}{\sum_{\bar{y}} \exp(w^T f(x, \bar{y}))}$. So *maximizing* the log likelihood

$$\log \prod_i P(y^i | x^i, w) = \sum_{i=1}^l \log \left(\frac{\exp(w^T f(x, y))}{\sum_{\bar{y}} \exp(w^T f(x, \bar{y}))} \right) = \sum_{i=1}^l (w^T f(x, y) - \log \sum_{\bar{y}} \exp(w^T f(x, \bar{y})))$$

. If $w^T f(x^i, y^i)$ is the largest, then it measures the relative value of $\exp(w^T f(x^i, y^i))$ and $\sum_{y \neq y^i} \exp(w^T f(x^i, y))$ (how much the latter can pull the former away). If

$w^T f(x^i, y^i)$ is not the largest, then compare the loss compares the (soft-max - $w^T f(x^i, y^i)$).

Log loss is used in CRF.

Preferable when uncertain on noisy data, penalty not that peaky.

4. rank loss. The 0/1 loss, log loss and hinge loss all count whether the desired label has been mistakenly ranked (not getting the highest score). However, most of the time, systems (particularly sequence labelling systems) are tested with respect to their error rate on test data, i.e., the fraction of times the function assigns a higher score the a label sequence $y \neq y^i$.

So the rank loss might be a more natural objective to minimize

$$\sum_{i=1}^l \sum_{y \neq y^i} \Theta(w^T f(x^i, y) - w^T f(x^i, y^i)),$$

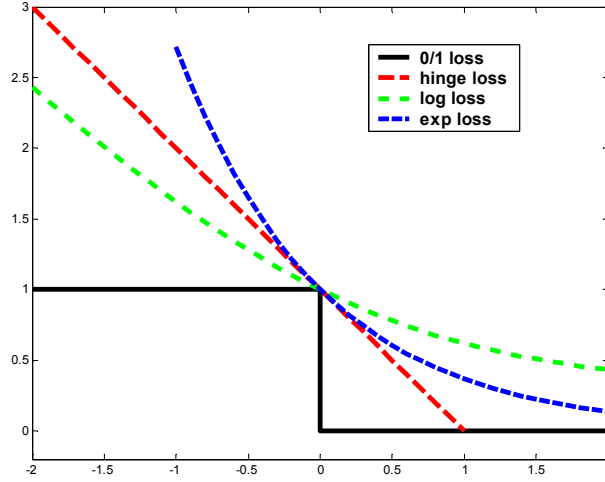
which is the total number of label sequences whose score ranks higher than the correct label sequences for the training instances in corpus C .

5. exp loss. Rank loss is NP-complete to optimize, so one can optimize an upper bound:

$$\sum_{i=1}^l \sum_{y \neq y^i} \exp(w^T f(x^i, y) - w^T f(x^i, y^i))$$

The advantage of the exp loss over the log loss is its property of penalizing incorrect labelings very severely, whereas it penalized almost nothing when the label sequence is correct. However, it also means the disadvantage of being sensitive to noisy data, since systems optimizing exp loss spends more effort on the outliers and tends to be vulnerable to noisy data.

The relationship of the 4 loss functions (excluding rank loss) is illustrated as:



The x-axis is $w^T f(x^i, y^i) - \max_{y \neq y^i} w^T f(x^i, y)$ for 0/1 loss and hinge loss and $w^T f(x^i, y^i) - \text{softMax}_{y \neq y^i} w^T f(x^i, y)$ for log loss and exp loss.

Besides defining on the complete sequence, it is useful to realize that in many applications it is very difficult to get the whole label sequence correct since most of the time classifiers are not perfect and as the sequences get longer, the probability of predicting every label in the sequence correctly decreases exponentially. For this reason performance is usually measured pointwise, i.e. in terms of the number of individual labels that are correctly predicted. Most common optimization functions in the literature, however, treat the whole label sequence as one label, penalizing a label sequence that has one error and a label sequence that is all wrong in the same manner. We may be able to develop better classifiers by using a loss function more similar to the evaluation function. One possible way of accomplishing this may be minimizing pointwise loss functions. Sequential optimizations optimize the joint conditional probability distribution $P(y|x;w)$, whereas pointwise optimizations that we propose opti-

mize the marginal conditional probability distribution, $P(y_t|x^i) = \sum_{y: y_t=y_t^i} P(y|x^i;w)$.

Then using the CRF probabilistic model, pointwise log loss function is

$$-\sum_{i=1}^l \sum_{t=1}^{L_i} \log P(y_t^i | x^i; w) = -\sum_{i=1}^l \sum_{t=1}^{L_i} \log \sum_{y: y_t=y_t^i} P(y | x^i; w).$$

Also using the CRF model, the pointwise exp loss is:

$$\sum_{i=1}^l \sum_{t=1}^{L_i} \sum_y \exp \left(w^T f(x^i, y) - \log \sum_{\bar{y}: \bar{y}_t=y_t^i} \exp(w^T \bar{y}) \right) = \sum_{i=1}^l \sum_{t=1}^{L_i} P(y_t^i | x^i; w)^{-1}$$

By the way, the sequential log loss is exactly what Lafferty et al. used. The sequential exp

loss is defined as $\sum_{i=1}^l \sum_{y \neq y^i} \exp(w^T f(x^i, y) - w^T f(x^i, y^i)) = \sum_{i=1}^l (P(y^i | x^i)^{-1} - 1)$.

Similarly, hinge loss 0/1 loss etc can be defined pointwise. Between pointwise and sequential, we can also define loss function on cliques, subtrees or other interesting sub-graphs.

Part IV. Variants of hinge loss function for structured output

In part III, we defined hinge loss $\sum_{i=1}^l \max\left(0, 1 - \max_{y \neq y^i} (w^T f(x^i, y) - w^T f(x^i, y^i))\right)$. In

ordinary multi-class classification, we do not emphasize the measurement of how different two classes are. We only say they are different. But once we have structured output space, how to define the difference between two sequences (or other structures) is a crucial concern.

We use $l_i(y)$ to denote the loss $loss(x^i, y^i, y)$, which can be Hamming distance of y^i and y , or 0/1 loss $\mathbf{1}(y^i \neq y)$, or any other loss function. So we change the original hinge loss

(specifically the 1) to $\max\left(0, \max_{y \neq y^i} (l_i(y) + w^T f(x^i, y) - w^T f(x^i, y^i))\right)$. This means if

there is a certain y which is very different from y^i and $l_i(y)$ is large, then even when $w^T f(x^i, y)$ is small, their sum is already possible to be larger than $w^T f(x^i, y^i)$ and may cause more loss (contribute to the max). As a result, the learner will try to make $w^T f(x^i, y)$ by far smaller than $w^T f(x^i, y^i)$ to ensure that serious mistakes will not be made. It is essentially re-scaling the margin. If we notice that $l_i(y^i) = 0$, we can write it in a compact form:

$$\max_y (l_i(y) + w^T f(x^i, y)) - w^T f(x^i, y^i). \quad (6)$$

Note when $l_i(y^i) = \mathbf{1}(y^i \neq y)$, (6) becomes normal hinge loss:

$$\max\left(0, 1 - w^T f(x^i, y^i) + \max_y w^T f(x^i, y)\right). \quad (7)$$

Another possible definition of hinge loss is

$$\max_y l_i(y) \cdot \left(1 - w^T (f(x^i, y^i) - f(x^i, y))\right). \quad (8)$$

This is essentially *re-scaling the slack variable* according to the loss. Intuitively, violating a margin constraint involving a $y \neq y^i$ with high loss $l_i(y^i)$ should be penalized more severely than a violation involving an output value with smaller loss. Note when

$l_i(y^i) = \mathbf{1}(y^i \neq y)$, (8) also becomes normal hinge loss (7).

The potential disadvantage of the margin re-scaling (6) is that it may give significant weight to output values $y \in \mathcal{Y}$ that are not even close to being confusable with the target values y^i , because every increase in the loss increases the required margin. Putting (6) and (8) mathematically:

$$\underset{w, \xi}{\text{Min}} \quad \frac{1}{2} \|w\|^2 + \frac{C}{l} \sum_{i=1}^l \xi_i \quad (6)'$$

$$\text{s.t. } w^T (f(x^i, y^i) - f(x^i, y)) \geq l_i(y) - \xi_i, \quad \xi_i \geq 0 \quad \text{for } \forall i, \forall y \in \mathcal{Y} \setminus y^i$$

and
$$\underset{w, \xi}{\text{Min}} \quad \frac{1}{2} \|w\|^2 + \frac{C}{l} \sum_{i=1}^l \xi_i \quad (8)'$$

$$\text{s.t. } w^T (f(x^i, y^i) - f(x^i, y)) \geq 1 - \frac{\xi_i}{l_i(y)}, \quad \xi_i \geq 0 \quad \text{for } \forall i, \forall y \in \mathcal{Y} \setminus y^i$$

Part V. Solving the optimization problem

1. Dual formulation.

We re-write the formulation (6)' after some equivalent simplification ($f_i(y) \triangleq f(x_i, y)$).

$$\underset{w, \xi}{\text{Min}} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i \quad (9)$$

$$\text{s.t. } w^T f_i(y^i) + \xi_i \geq w^T f_i(y) + l_i(y) \quad \text{for } \forall i, \forall y \in \mathcal{Y}$$

The number of constraints can go exponentially with L_i , thus intractable to solve. We can use Lagrange multiplier methods.

Primal problem:
$$\underset{w: g(w) \geq 0}{\text{Min}} \quad f(w) \quad (10)$$

Lagrangian:
$$L(w, \alpha) \triangleq f(w) - \alpha^T g(w)$$

Let (primal)
$$L(w) \triangleq \max_{\alpha \geq 0} L(w, \alpha) = \max_{\alpha \geq 0} (f(w) - \alpha^T g(w)), \quad \tilde{p} \triangleq \min_w L(w),$$

(dual)
$$L(\alpha) \triangleq \min_w L(w, \alpha) = \min_w (f(w) - \alpha^T g(w)), \quad \tilde{d} \triangleq \max_{\alpha \geq 0} L(\alpha).$$

Proposition 1. $\tilde{p} = \underset{w: g(w) \geq 0}{\text{Min}} f(w)$. This is because

$$\tilde{p} \triangleq \min_w \max_{\alpha \geq 0} (f(w) - \alpha^T g(w))$$

$$\begin{aligned}
&= \min \left(\min_{w: g(w) \geq 0} \max_{\alpha \geq 0} (f(w) - \alpha^T g(w)), \min_{w: g(w) < 0} \max_{\alpha \geq 0} (f(w) - \alpha^T g(w)) \right) \\
&= \min \left(\min_{w: g(w) \geq 0} f(w), +\infty \right) \\
&= \min_{w: g(w) \geq 0} f(w)
\end{aligned}$$

Proposition 2. $\tilde{d} = \max_{\alpha \geq 0} \min_w L(w, \alpha) \leq \min_w \max_{\alpha \geq 0} L(w, \alpha) = \tilde{p}$

Proposition 3. (Slater's condition) If the primal problem (10) is convex and is **strictly feasible**, i.e., there exists $u_0 : f_i(u_0) < 0$, then $\tilde{p} = \tilde{d}$.

Combining Proposition 1 and 3, we conclude that we can solve $\underset{w: g(w) \geq 0}{\text{Min}} f(w)$ by solving

$$\max_{\alpha \geq 0} \min_w (f(w) - \alpha^T g(w)).$$

So the Lagrangian of (9) is

$$\min_{w, \xi} \max_{\alpha \geq 0} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i,y} \alpha_i(y) (w^T f_i(y^i) + \xi_i - w^T f_i(y) - l_i(y)) \quad (11)$$

which has the same value as

$$\max_{\alpha \geq 0} \min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i,y} \alpha_i(y) (w^T f_i(y^i) + \xi_i - w^T f_i(y) - l_i(y)). \quad (12)$$

The significant difference is that the inner optimization of (11) is constrained, while that of (12) is unconstrained, which can be solved analytically. Also, the outer optimization only involves a bound constraint. Now we optimize over α .

To do the inner optimization, we only need to apply normal unconstrained optimization,

viewing α as fixed constant. Setting $\frac{\partial L(w, \xi; \alpha)}{\partial w} = \frac{\partial L(w, \xi; \alpha)}{\partial \xi} = 0$, we finally derive

the dual problem:

$$\begin{aligned}
&\underset{w, \xi}{\text{Min}} \quad -\frac{1}{2} \left\| \sum_{i,y} \alpha_i(y) \Delta f_i(y) \right\|^2 + \sum_{i,y} \alpha_i(y) l_i(y) \\
&s.t. \quad \sum_y \alpha_i(y) = C, \alpha_i(y) \geq 0 \text{ for } \forall i, y
\end{aligned}$$

where $\Delta f_i(y) \triangleq f_i(y^i) - f_i(y)$. Without loss of generality, we normalize α_i by C and divide objective by C , then the resulting dual is given by

$$\underset{w, \xi}{\text{Min}} \quad -\frac{1}{2} C \left\| \sum_{i,y} \alpha_i(y) \Delta f_i(y) \right\|^2 + \sum_{i,y} \alpha_i(y) l_i(y) \quad (13)$$

$$s.t. \quad \sum_y \alpha_i(y) = 1, \alpha_i(y) \geq 0 \quad \text{for } \forall i, y.$$

The original w is given by $w = C \sum_{i,y} \alpha_i(y) \Delta f_i(y)$ and the score function is

$$w^T f_x(y) = C \sum_{i,y} \alpha_i(y) \Delta f_i(y) f_x(y) \quad (14)$$

So we can stay in the space of α . The main insight is that $\alpha_i(y)$ can be interpreted as a kind of distribution over y , since they lie in the simplex:

$$\sum_y \alpha_i(y) = 1, \alpha_i(y) \geq 0 \quad \text{for } \forall i, y.$$

This dual distribution does not represent the probability that the model assigns to an instantiation, but the importance of the constraint associated with the instantiation to the solution. The dual objective is a function of expectations of $l_i(y)$ and $\Delta f_i(y)$ with respect to $\alpha_i(y)$.

This is totally different from the dual form of binary classification. Max (wrt α):

$$\sum_{i=1}^l \alpha_i - \frac{1}{2} C \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j d_i d_j x_i^T x_j, \quad s.t. \quad \sum_{i=1}^l \alpha_i d_i = 0, \quad \alpha_i \in [0,1] \quad \text{for } \forall i. \quad (\text{opposite sign}).$$

Non-zero $\alpha_i(y)$ are called support vectors. However, even solving the dual problem (13) may be expensive for large datasets or large output spaces. Special algorithms like Sequential Minimal Optimization have been proposed to this end.

2. Factorization.

This part is the most exciting part of M^3N . However, the solution is more mathematical than machine learning. We will try to interpret it more from machine learning angle.

The biggest problem in structured output learning is that the number of possible labelling is usually exponential to the number of variables. This usually makes a model mathematically intractable. Sometimes, it is relatively easy to design a dynamic programming algorithm to do exact inference. In the worse cases, approximate inference algorithms are abundant. However, learning a model (estimating parameters) is usually more troubled by the computational cost. Most algorithms either use approximate algorithms, or cleverly decompose the structured model into some parts and learn the intra-part and inter-part model separately. For CRF, the exponential computational cost stems from the calculation of model expectation. Fortunately, there is a Baum-Welch algorithm which can compute the expectation in polynomial time. However, this comes with the price that CRF is a chain. Changing it into higher order r will cause computational complexity to grow exponentially with respect to r .

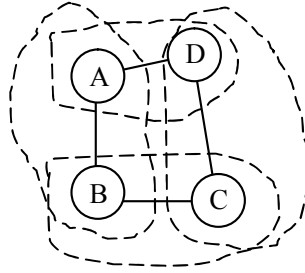
M^3N deals with the efficiency problem by decomposing the feature vector and loss function over the cliques of the Markov network. It assumes that $l_i(y) = \sum_{c \in \mathcal{C}} l_{i,c}(y_c)$ and

$\Delta f_i(y) = \sum_{c \in \mathcal{C}} \Delta f_{i,c}(y_c)$. Correspondingly, we need clique marginals of the distribution $\alpha_i(y)$. So we define the marginal dual variables as:

$$\mu_{i,c}(y_c) = \sum_{\bar{y} \sim y_c} \alpha_i(\bar{y}) \quad \forall i, \forall c \in \mathcal{C}^i, \forall y_c \quad (15)$$

(15) defines a map from $\alpha_i(y)$ space to $\mu_{i,c}(y_c)$ space. The intuitive objective of the transformation is variable reduction. Compared with the exponential number of $\alpha_i(y)$ variables, the number of $\mu_{i,c}(y_c)$ is just exponential to the tree-width of the graph (W), and proportional to the number of cliques (n_c): $O(\ln_c e^W)$. Hopefully the tree-width we need to deal with is small. The major objective is that the optimization problem (13) can be re-expressed in terms of $\mu_{i,c}(y_c)$. After the optimal $\mu_{i,c}(y_c)$'s are found, we can map the value back to $\alpha_i(y)$ efficiently. To make this work, we must pay attention to 2 problems:

1. The validity of translation between $\alpha_i(y)$ and $\mu_{i,c}(y_c)$. We denote the map as $\mathcal{M}: \alpha \mapsto \mu$. The domain of \mathcal{M} , $D(\mathcal{M})$, is the product of simplices of the l examples. But there is no guarantee that the domain of \mathcal{M} is the set of all legal marginals, i.e., not all free marginals $\mu_{i,c}(y_c)$ can be translated back to $\alpha_i(y)$. For examples:



$$\mu_{AB}(0,0) = \mu_{AB}(1,1) = 0.5$$

$$\mu_{AB}(1,0) = \mu_{AB}(0,1) = 0$$

$$\mu_{BC}(0,0) = \mu_{BC}(1,1) = 0.5$$

$$\mu_{BC}(1,0) = \mu_{BC}(0,1) = 0$$

$$\mu_{CD}(0,0) = \mu_{CD}(1,1) = 0.5$$

$$\mu_{CD}(1,0) = \mu_{CD}(0,1) = 0$$

$$\mu_{DA}(0,0) = \mu_{DA}(1,1) = 0$$

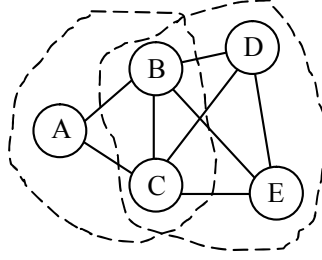
$$\mu_{DA}(1,0) = \mu_{DA}(0,1) = 0.5$$

The zeros on the edges AB, BC, CD only allows 0000 or 1111. But the edges DA disallows either. Finally, there is no valid corresponding α on y .

So we must restrict the range of \mathcal{M} , $R(\mathcal{M})$, so that the map from $D(\mathcal{M})$ to $R(\mathcal{M})$ is a *surjection*. If the graph \mathcal{G}^i is triangulated, then the following equations will guarantee that the \mathcal{M} is surjection.

$$\begin{aligned}\sum_{y_c} \mu_{i,c}(y_c) &= 1, \quad \forall i, \forall c \in \mathcal{C}^i \\ \mu_{i,c}(y_c) &\geq 0, \quad \forall i, \forall c \in \mathcal{C}^i, \forall y_c \\ \mu_{i,s}(y_s) &= \sum_{\bar{y}_c \sim y_s} \mu_{i,c}(\bar{y}_c), \quad \forall i, \forall s, c \in \mathcal{C}^i, s \subset c, \forall y_c\end{aligned}$$

The third condition is stating the consistency between cliques that share variables. It means for any two cliques C_1 and C_2 , if they share the same variables C , then the marginals on C derived from C_1 should be equal to the marginals on C derived from C_2 . The real value of $\mu_{i,c}(y_c)$ is not important and it only makes notation more concise.



For example, in the above figure, suppose all nodes are binary. Then it requires

$$\begin{aligned}\mu_{ABC}(1,1,1) + \mu_{ABC}(0,1,1) &= \mu_{BCDE}(1,1,0,0) + \mu_{BCDE}(1,1,0,1) + \mu_{BCDE}(1,1,1,0) + \mu_{BCDE}(1,1,1,1) \\ \mu_{ABC}(1,1,0) + \mu_{ABC}(0,1,0) &= \sum_{d,e \in \{0,1\}} \mu_{BCDE}(1,0,d,e) \\ \mu_{ABC}(1,0,1) + \mu_{ABC}(0,0,1) &= \sum_{d,e \in \{0,1\}} \mu_{BCDE}(0,1,d,e) \\ \mu_{ABC}(1,0,0) + \mu_{ABC}(0,0,0) &= \sum_{d,e \in \{0,1\}} \mu_{BCDE}(0,0,d,e).\end{aligned}$$

Note the \mathcal{C}^i may contain non-maximum cliques. If the graph is not triangulated, then we must triangulate it and then define the $\mu_{i,c}(y_c)$. Otherwise, the three conditions are not sufficient to guarantee that \mathcal{M} is surjection. The first example illustrates the case. Note the node marginals are all set to 0.5.

Once the three conditions are met, we wish to find a decoding from $\mu_{i,c}(y_c)$ back to

$\alpha_i(y)$. As \mathcal{M} is a many-to-one map, the $\alpha_i(y)$ will not be uniquely determined. We can just pick one, which can be the maximum-entropy distribution.

2. We should reformulate the entire QP (13) in terms of these marginal dual variables.

$$\begin{aligned}\sum_y \alpha_i(y) l_i(y) &= \sum_y \alpha_i(y) \sum_c l_{i,c}(y_c) = \sum_{c,y_c} l_{i,c}(y_c) \sum_{\bar{y} \sim y_c} \alpha_i(\bar{y}) = \sum_{c,y_c} \mu_{i,c}(y_c) l_{i,c}(y_c) \\ \sum_y \alpha_i(y) \Delta f_i(y) &= \sum_{c,y_c} \Delta f_{i,c}(y_c) \sum_{\bar{y} \sim y_c} \alpha_i(\bar{y}) = \sum_{c,y_c} \mu_{i,c}(y_c) \Delta f_{i,c}(y_c)\end{aligned}$$

Putting 1 and 2 together, we have the dual structured dual QP:

$$\begin{aligned} \text{Max}_{\mu_{i,c}(y_c)} \quad & \sum_{i,c,y_c} \mu_{i,c}(y_c) l_{i,c}(y_c) - \frac{1}{2} C \left\| \sum_{i,c,y_c} \mu_{i,c}(y_c) \Delta f_{i,c}(y_c) \right\|^2 \\ \text{s.t.} \quad & \sum_{y_c} \mu_{i,c}(y_c) = 1, \quad \forall i, \forall c \in \mathcal{C}^i \\ & \mu_{i,c}(y_c) \geq 0, \quad \forall i, \forall c \in \mathcal{C}^i, \forall y_c \\ & \mu_{i,s}(y_s) = \sum_{\bar{y}_c \sim y_s} \mu_{i,c}(\bar{y}_c), \quad \forall i, \forall s, c \in \mathcal{C}^i, s \subset c, \forall y_c \end{aligned} \quad (16)$$

The solution to the structured dual μ^* will give us the primal solution

$$w^* = C \sum_{i,c,y_c} \mu_{i,c}^*(y_c) \Delta f_{i,c}(y_c)$$

Part VI. Kernelization

We go back to the binary SVM dual optimization problem (5):

$$\begin{aligned} \text{Max (wrt } \alpha): \quad & \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j d_i d_j x_i^T x_j \\ \text{s.t.} \quad & \sum_{i=1}^l \alpha_i d_i = 0, \quad \alpha_i \in [0, C] \text{ for } \forall i \end{aligned}$$

Note input x_i and x_j appear only in the form of inner product. For non-linearly separable dataset, we can always map it to a high dimensional space where they become linearly separable. So if we map all x to another feature space $\varphi(x)$, then it still involves inner product in feature space only:

$$\text{Max (wrt } \alpha): \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j d_i d_j \varphi(x_i)^T \varphi(x_j)$$

$$s.t. \quad \sum_{i=1}^l \alpha_i d_i = 0, \quad \alpha_i \in [0, C] \text{ for } \forall i$$

and the determinant boundary is $w^{*T}x + b = \sum_{i=1}^l \alpha_i^* d_i \varphi(x_i)^T \varphi(x) + b$.

So we can define kernel $K(x_i, x) = \varphi(x_i)^T \varphi(x)$. $\varphi(\cdot)$ may be a high dimensional space (or even infinite dimension), but we can still define $K(x_i, x)$. So it offers us a convenient way to implicitly deal with rich feature spaces. Commonly used kernels include polynomial kernel $(x^T x_i + 1)^p$, radial basis function kernel $\exp(-\|x - x_i\|^2 / 2\sigma^2)$, linear kernel $x^T x_i$, $\tanh(\beta_0 x^T x_i + \beta_1)$.

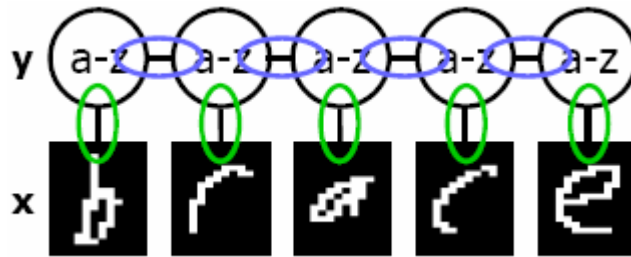
Note in the dual optimization problem (16), local (clique) basis functions appear only in terms of inner products.

$$\Delta f_{i,c}(y_c)^T \Delta f_{j,\bar{c}}(y_{\bar{c}}) = \left[f(x_c^i, y_c^i) - f(x_c^j, y_c^j) \right]^T \left[f(x_{\bar{c}}^j, y_{\bar{c}}^j) - f(x_{\bar{c}}^i, y_{\bar{c}}^i) \right]$$

and the score function for unseen data is

$$\begin{aligned} w^{*T} f(y^{new}) &= C \sum_{i,c,y_c} \mu_{i,c}^*(y_c) \Delta f_{i,c}(y_c) \cdot \sum_{\bar{c}} f_{\bar{c}}(y_{\bar{c}}^{new}) \\ &= C \sum_{i,c,\bar{c},y_c} \mu_{i,c}^*(y_c) \left(f(x_c^i, y_c^i) - f(x_c^j, y_c^j) \right)^T f_{\bar{c}}(y_{\bar{c}}^{new}) \end{aligned}$$

So what really matters is the inner products of local basis function, which makes kernelization possible. The definition of kernel is problem specific. Note here the kernel is defined between cliques of (possibly) different sequences. We use the handwriting recognition as an example.



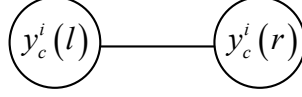
We first define a kernel on flat vector $k(x_c^i, x_{\bar{c}}^j)$, e.g. $(x_c^{iT} x_{\bar{c}}^j + 1)^p$. There are two types of cliques: node cliques and edge cliques. It is not straightforward to define kernel between node and edge cliques or between edge cliques. So we just define them to be 0 and focus on kernels between node cliques:

$$k\left((x_c^i, y_c^i), (x_{\bar{c}}^j, y_{\bar{c}}^j)\right) \triangleq \mathbf{1}(y_c^i = y_{\bar{c}}^j) k(x_c^i, x_{\bar{c}}^j)$$

Some other definitions on sequences may allow inter-edge kernels:

$$k\left(\left(x_c^i, y_c^i\right), \left(x_{\bar{c}}^j, y_{\bar{c}}^j\right)\right) \triangleq \mathbf{1}\left(y_c^i(l) = y_{\bar{c}}^j(l) \wedge y_c^i(r) = y_{\bar{c}}^j(r)\right)$$

where y_c^i is an edge clique:



There are also many types of kernel defined for structured data, say string kernels, tree kernels.

Part VII. Other two techniques

1. Linear programming for MAP decoding

The dual form gives one solution to the primal problem (9):

$$\underset{w, \xi}{\text{Min}} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i \quad (9)$$

$$\text{s.t. } w^T f_i(y^i) + \xi_i \geq w^T f_i(y) + l_i(y) \quad \text{for } \forall i, \forall y \in \mathcal{Y}$$

Another solution is to view the constraint as

$$w^T f_i(y^i) + \xi_i \geq \max_{y \in \mathcal{Y}^i} (w^T f_i(y) + l_i(y)) \quad \text{for } \forall i, \forall y \in \mathcal{Y}$$

If we have a polynomial time algorithm to infer $\max_{y \in \mathcal{Y}^i} (w^T f_i(y) + l_i(y))$ and substitute in

the constraint we will be able to circumvent the max over a exponentially many values. We formulate the inference problem by a linear programming problem. We still assume that the loss function and feature vector can be decomposed over cliques. We represent a possible assignment y by a set of binary variables $\mu_c(y_c)$, one for each clique c and each value of the clique y_c , which denotes whether the assignment has that value. So the inference problem can be formulated as

$$\begin{aligned} & \max_{\mu_{i,c}(y_c)} \sum_{c, y_c} \mu_{i,c}(y_c) (w^T f_{i,c}(y_c) + l_{i,c}(y_c)) \\ \text{s.t. } & \sum_{y_c} \mu_{i,c}(y_c) = 1, \quad \forall i, \forall c \in \mathcal{C}^i \\ & \mu_{i,c}(y_c) \geq 0, \quad \forall i, \forall c \in \mathcal{C}^i, \forall y_c \\ & \mu_{i,s}(y_s) = \sum_{\bar{y}_c \sim y_s} \mu_{i,c}(\bar{y}_c), \quad \forall i, \forall s, c \in \mathcal{C}^i, s \subset c, \forall y_c \end{aligned}$$

We should have added a constraint that $\mu_{i,c}(y_c)$ can only be integers. Fortunately, for a triangulated network with unique MAP assignment, the integrality constraint can be relaxed and the resulting LP is guaranteed to have integer solutions. If the MAP assignment is not

unique, the value of the LP is the same as the value of the integer program, and any linear combination of the MAP assignments maximizes the LP.

The dual is:

$$\begin{aligned} \min \quad & \sum_c \lambda_{i,c} \\ \text{s.t.} \quad & \lambda_{i,c} + \sum_{s \supset c} m_{i,s,c}(y_c) - \sum_{s \subset c, y'_s \sim y_c} m_{i,s,c}(y'_s) \geq w^T f_{i,c}(y_c) + l_{i,c}(y_c), \quad \forall c \in \mathcal{C}^i, \forall y_c \end{aligned}$$

Plugging the dual into (9), for each example i , we obtain

$$\begin{aligned} \underset{w, \xi, \lambda, m}{Min} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i \\ \text{s.t.} \quad & w^T f_i(y^i) + \xi_i \geq \sum_c \lambda_{i,c} \quad \forall i \\ & \lambda_{i,c} + \sum_{s \supset c} m_{i,s,c}(y_c) - \sum_{s \subset c, y'_s \sim y_c} m_{i,s,c}(y'_s) \geq w^T f_{i,c}(y_c) + l_{i,c}(y_c), \quad \forall c \in \mathcal{C}^i, \forall y_c \end{aligned}$$

This is already polynomial.

2. Certificate problem

This problem is crucial for matching problem. It is similar to verifying a NP problem by a polynomial-time algorithm, though we may not be able to solve it in polynomial time. E.g.

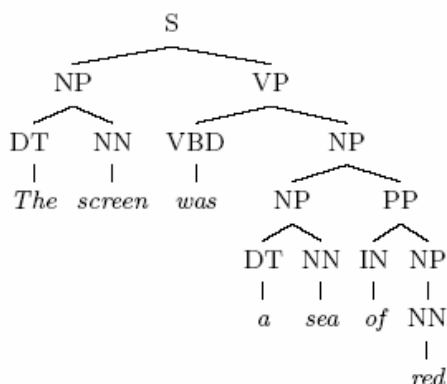
Consider the maximum weight spanning tree problem. A basic property of a spanning tree is that cutting any edge (j, k) in the tree creates two disconnected sets of nodes $(\mathcal{V}_j[jk], \mathcal{V}_k[jk])$, where $j \in \mathcal{V}_j[jk]$ and $k \in \mathcal{V}_k[jk]$. A spanning tree is optimal with respect to a set of edge weights if and only if for every edge (j, k) in the tree connecting $\mathcal{V}_j[jk]$ and $\mathcal{V}_k[jk]$, the weight of (j, k) is larger than (or equal to) the weight of any other edge (j', k') in the graph with $j' \in \mathcal{V}_j[jk], k' \in \mathcal{V}_k[jk]$ [Cormen *et al.*, 2001]. We discuss the conditions

are needed. In the spanning tree problem, suppose y_{jk} encodes whether edge (j, k) is in the tree and the score of the edge is given by $w^T \mathbf{f}_{i,jk}$ for some basis functions $\mathbf{f}(\mathbf{x}_{jk}^{(i)}, y_{jk})$. We also assume that the loss function decomposes into a sum of losses over the edges, with loss for each wrong edge given by $\ell_{i,jk}$. Then the optimality conditions are:

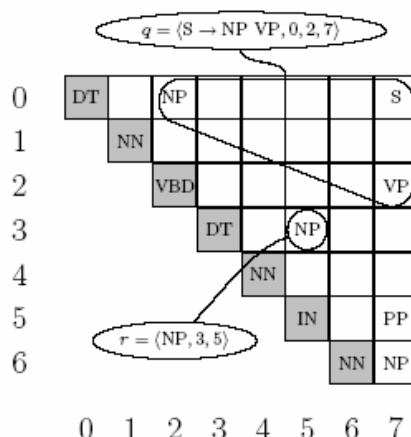
$$w^T \mathbf{f}_{i,jk} \geq w^T \mathbf{f}_{i,j'k'} + \ell_{i,j'k'}, \quad \forall jk, j'k' \text{ s.t. } y_{jk}^{(i)} = 1, j' \in \mathcal{V}_j[jk], k' \in \mathcal{V}_k[jk].$$

Part VIII. Application 1: Context Free Grammar Parsing

To discuss application, the most important consideration is how to formulate the application into the theoretical framework. We give the following correspondence between the concepts in this CFG parsing application and in M^3N :



(a)



(b)

Input vector	sentence
Output space	valid CFG parse trees
Cliques	parts (two types, see above). Let $R(x^i, y)$ be the set of parts belonging to a particular parse y . Because all rules are in binary-branching form, $ R(x^i, y) $ is constant across different derivations y for the same input sentence x^i .
Loss function	$L(x^i, y^i, y) = \sum_{r \in R(x^i, y)} l(x^i, y^i, r)$ <p>note the third operand is a clique. One approach would be to define $l(x^i, y^i, r)$ to be 0 only if the non-terminal A spans words $s \dots e$ in the derivation y^i and 1 otherwise. This would lead to $L(x^i, y^i, y)$ tracking the number of “constituent errors” in y, where a constituent is a tuple such as $\langle A, s, e, i \rangle$. Another, more strict definition would be to define $l(x^i, y^i, r)$ to be 0 if r of the type $\langle A \rightarrow B C, s, m, e, i \rangle$ is in the derivation y^i and 1 otherwise. This definition would lead to $L(x^i, y^i, y)$ being the number of CF-rule-tuples in y which are not seen in y^i.</p>
Feature vector	<p>any function mapping a rule production and its position in the sentence x^i, to some feature vector representation. For example, it could include features which identify the rule used in the production, or features which track the rule identity together with features of the words at positions s, m, e, and neighboring positions in the sentence x.</p> $f(x^i, y) = \sum_{r \in R(x^i, y)} f_r(x^i)$
dual joint distribution $\alpha_i(y)$	probability distributions over all possible valid parse trees for each sentence i
dual cost function	$\text{Min} - \frac{1}{2} C \left\ \sum_i \left(f(x^i, y^i) - \sum_y \alpha_i(y) f(x^i, y) \right) \right\ ^2 + \sum_{i,y} \alpha_i(y) l(x^i, y^i, y)$

	<p>Let $R(x^i) = \bigcup_y R(x^i, y)$, $\mu_{i,r} = \sum_y \alpha_{i,y} \mathbf{1}(r \in R(x, y))$, then minimize</p> $\sum_{i,r \in R(x^i)} \mu_{i,r} l(x^i, y^i, r) - \frac{1}{2} C \left\ \sum_{i,r \in R(x^i)} (\mathbf{1}[r \in R(x^i, y^i)] - \mu_{i,r}) f_r(x^i) \right\ ^2$
dual marginals $\mu_{i,c}(y_c)$	<p>As cliques are parts r, there is no different instantiations of r's. We write $\mu_{i,r}(\alpha_i)$ or $\mu_{i,r}$, meaning the proportion of parses that would contain part r if they were drawn from a distribution $\alpha_i(y)$.</p> $\mu_{i,r}(\alpha_i) = \sum_y \alpha_{i,y} \mathbf{1}(r \in R(x, y))$ <p>The number of such marginal terms is the number of parts, which is polynomial in the length of the sentence.</p>
dual marginals consistency	<p>Essentially, we need to enforce the condition that the expected proportions of parses having particular parts should be consistent with each other.</p> $\mu_{i,r} \geq 0, \sum_A \mu_{i,A,0,L_i} = 1, \text{ and } \mu_{i,A,s,e} = \sum_{B,C,s < m < e} \mu_{i,A \rightarrow BC,s,m,e}$ $\mu_{i,A,s,e} = \sum_{B,C,e < m \leq L_i} \mu_{i,B \rightarrow AC,s,m,e} + \sum_{B,C,0 \leq m \leq s} \mu_{i,B \rightarrow CA,s,m,e} \text{ (more in thesis?)}$

Part IX. Application 2: Protein Disulfide Connectivity Prediction

Abstracted problem:

We have a set of nodes N , $|N| = 2n$. We wish to partition N into two subsets of nodes A, B , such that $|A| = |B|$, $A \cap B = \emptyset$. Besides, we wish to find a matching between nodes in A and B (a bijection between A and B). This is called perfect matching.

The total number of possible match is $\frac{C_{2n}^n n!}{2^n} = \frac{(2n)!}{2^n \cdot n!} = O\left(\left(\frac{n}{2}\right)^n\right)$, super-exponential.

What is known: the attraction strength (similarity) between each pair of nodes. We represent each possible edge between nodes j and k ($j < k$) in the example x^i using a binary variable y_{jk}^i . Let \mathcal{Y}^i be the set of partition (bipartite) and matching $|\mathcal{Y}^i| = C_{2n_i}^{n_i} n! / 2^n$. Then we assume

1. The similarity between node j and k in example x is

$$s_{jk}(x) = \sum_d w_d f_d(x_{jk}) = w^T f(x_{jk}).$$

Here $f_d(x_{jk})$ is a real-valued basis function representing arbitrary information about the two cysteine (j, k) neighborhoods such as: the identity of the residues at specific positions around the two cysteines, or the predicted secondary structure in the neighborhood of each cysteine. We assume that the user provides the basis functions. w is what we will learn.

2. The hypothesis is a matching which maximizes weight of the matching edges:

$$h(x) = \arg \max_{y \in \mathcal{Y}} \sum_{jk} s_{jk}(x) y_{jk} = \arg \max_{y \in \mathcal{Y}} \sum_{jk} w^T f(x_{jk}) y_{jk}$$

We denote $w^T f(x, y) \triangleq \sum_{jk} w^T f(x_{jk}) y_{jk}$, $w^T f_i(y) \triangleq w^T f(x^i, y)$. So our max-margin optimization problem can be written as:

$$\begin{aligned} \text{Min } & \frac{1}{2} \|w\|^2 \\ \text{s.t. } & w^T f_i(y^i) \geq w^T f_i(y) + l_i(y), \quad \forall i, \forall y \in \mathcal{Y}^i \end{aligned}$$

Again, work must be done to reduce the complexity of constraints. We want a compact certificate of optimality that guarantees $y^i = \arg \max_y (w^T f_i(y) + l_i(y))$. Then in these satisfying w 's, we find the one with minimum norm.

Let M be a perfect matching for a complete undirected graph $G = (V; E)$. In an *alternating cycle/path* in G with respect to M , the edges alternate between those that belong to M and those that do not. An alternating cycle is *augmenting* with respect to M if the score of the edges in the matching M is smaller than the score of the edges not in the matching M .

Theorem 1. *A perfect matching M is a maximum weight perfect matching if and only if there are no augmenting alternating cycles.*

Theorem 2. *There exists a distance function $\{d_j^e\}$ satisfying the constraints below if and only if no augmenting alternating cycles exist.*

$$\begin{aligned} s_{jk} &\geq d_k^0 - d_j^1 & s_{jk} &\geq d_j^0 - d_k^1 & \forall jk \notin M \\ s_{jk} &\geq d_k^1 - d_j^0 & s_{jk} &\geq d_j^1 - d_k^0 & \forall jk \in M \end{aligned}$$

So we only need to introduce an auxiliary vector \mathbf{d} , leading to

$$\begin{aligned} \text{Min } & \frac{1}{2} \|w\|^2 \\ \text{s.t. } & H_i w + G_i d_i \geq q_i \quad \forall i \end{aligned}$$

The H_i, G_i, q_i are coefficients from equations in Theorem 2. They are constants and of polynomial size. So this means that among the w which satisfies

$y^i = \arg \max_y (w^T f_i(y) + l_i(y))$, we pick the one with smallest norm. If our basis functions are not rich enough to predict the training data perfectly, we can introduce a slack variable vector ξ_i to allow violations of the constraints.

Questions.

1. How to add bias? Add constant feature? Is it included in w and thus in the $\|w\|$ to be optimized?
2. ...