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

Xinhua Zhang
University of Alberta
xinhua.zhang.cs@gmail.com

Thore Graepel
Microsoft Research, Cambridge
thoreg@microsoft.com

Ralf Herbrich
Microsoft Research, Cambridge
rherb@microsoft.com

Abstract

Many real world applications employ multi-variate performance measures and each example can belong to multiple classes. The currently most popular approaches train an SVM for each class, followed by ad hoc thresholding. Probabilistic models using Bayesian decision theory are also commonly adopted. In this paper, we propose a Bayesian online multi-label classification framework (BOMC) which learns a probabilistic linear classifier. The likelihood is modeled by a graphical model similar to TrueSkill$^{\text{TM}}$, and inference is based on Gaussian density filtering with expectation propagation. Using samples from the posterior, we label the testing data by maximizing the expected F_1-score. Our experiments on Reuters1-v2 dataset show BOMC compares favorably to the state-of-the-art online learners in macro-averaged F_1-score and training time.

1 Introduction

Real world applications often involve a large number of classes and each example can be associated with multiple classes. For instance, many web related objects such as blogs, bookmarks, RSS feeds are attached with tags which are essentially forms of categorization. A news article on "Obama supported the $170 billion AIG bailout after intense debate" can be associated with `insurance`, `economics`, and `politics`. In the search industry, revenue comes from clicks on the ads embedded in the result page. The selection and placement of ads can be significantly improved if ads are automatically tagged, or further categorized into a hierarchy or ontology. This setting is referred to as multi-label classification in machine learning, which is also useful in many other applications such as bioinformatics (Seki & Mostafa, 2005) and video retrieval (Qi et al., 2007).

Appearing in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9. Copyright 2010 by the authors.

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

1. The problem scale is large in the number of data points n, number of features D, and number of classes C. Usually, we can afford at most $O(nDC)$ computations. Hence efficiency is critical and expensive operations such as pairwise comparison must be avoided.

2. Often multi-variate performance measures are used, *e.g.* macro-average F_β-score and area under the ROC. They couple the labels of all data points and/or classes in a nondecomposable way. As a result, models learned by minimizing the training error often perform poorly under this new measure, and it is important to calibrate the trained model according to the testing data.

3. Labels can be highly correlated and many applications employ a tree structured ontology. One example is the Pascal challenge on large scale hierarchical text classification which is based on the ODP web directory data: `lshtc.iit.demokritos.gr`.

Existing algorithms for multi-label classification can be categorized into three dimensions: a) batch v.s. online, b) frequentist v.s. Bayesian, and c) using structures in the label space v.s. treating the labels as independent. These dimensions help us to analyze how much a learning algorithm fits the above three challenges, and to eventually motivate our new algorithm.

A typical max-margin frequentist method generalizes the binary hinge loss to the maximum inconsistency (Elisseeff & Weston, 2001). Intuitively, for each pair of associated label c and non-associated label c', the linear score of c is expected to exceed that of c' by a certain margin. However, this method may take $O(C^2)$ time, and is hence inapplicable when the number of classes is large. Among the probabilistic methods, mixture models are the most natural. 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 expanding the latent topic space to the power set of the topics.

Unfortunately, all of these batch methods are very expensive in training, and hence do not scale well. For large datasets, online learning becomes effective. It employs cheap updates and works well for stream data. Although it has been widely used for binary classification, it is less studied for the multi-label scenario. For example, the additive online category ranking (Crammer & Singer, 2003) uses pairwise class comparison but is made efficient by precomputation. Bayesian online learning (Opper, 1998) has also been studied. They essentially perform assumed density filtering, where at each step the posterior of the model is updated based on the likelihood of a single data point, followed by approximations, *e.g.* using Gaussians (Minka, 2001). We are unaware of any published Bayesian online learner for multi-label classification.

Bayesian methods learn a *distribution* over a family of models. They are both useful and commonly adopted. Although learning and applying distributions over models is generally more computationally expensive that point estimates, they provide more flexibility in decision making and allow the model to be used for different purposes. A case in point is the second challenge above, multi-variate performance measures. Joachims (2005) tailored the training of SVM for the multi-variate measure, however the testing examples were still labeled by applying the learned model independently. Other frequentist methods also rely on ad hoc thresholding. In contrast, with a distribution of models available, the Bayesian method provides a principled framework for labeling the test data by optimizing the posterior expectation of the multi-variate measure in a batch fashion. Also, the model can be estimated independent of the performance measure. This is especially useful for online learning where data points are intrinsically decoupled.

Finally, to make use of the structure in the label space as desired from the third challenge, some 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 interdependence is through 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, the CRF was trained in a batch fashion and it is not clear whether it can also be learned efficiently in the stochastic online setting for the multi-label data.

We propose a Bayesian online multi-label classification framework (BOMC) which learns a probabilistic model of the linear classifier (Section 2). The labels are loosely coupled via a global bias for the multi-label scenario. The training labels are incorporated to update the posterior of the classifiers via a graphical model similar to TrueSkill$^{\mathrm{TM}}$ (Herbrich et al., 2007). Inference is based on assumed density filtering through the stream of training data, and expectation propagation (Minka, 2001) is applied on each training example (Section 3). This allows us to efficiently learn from a large amount of training data. Using samples from the posterior of the model, we label the testing examples by maximizing the expected F_β-score (Section 4). Encouraging experimental results are presented in Section 5, including the comparison in macro-average F_β-score and training time. Section 6 concludes the whole paper with future work.

2 A Bayesian model for multi-label data

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 $\{1, \ldots, C\} =: [C]$, and the label vector $\mathbf{y}^i \in \{0,1\}^C$ encodes in the multi-label setting that $y_c^i = 1$ if example $\mathbf{x}^i$ is relevant to class c, and 0 otherwise.

Our model uses a probabilistic linear discriminant $\mathbf{w}_c$ for each class c, and $\{\mathbf{w}_c\}_c$ are independent diagonal Gaussians whose mean and variance are estimated from the training data. We start from a special case of multi-label: multi-class where exactly one label is relevant. Our key model is the likelihood $p(\mathbf{y}| \{\mathbf{w}_c\}_c, \mathbf{x})$, the probability of label $\mathbf{y}$ given the weights $\{\mathbf{w}_c\}_c$. By Bayes' rule, the posterior of $\{\mathbf{w}_c\}_c$ can be computed by $p(\{\mathbf{w}\}_c |\mathbf{y}, \mathbf{x}) \propto p(\mathbf{y}| \{\mathbf{w}_c\}_c, \mathbf{x}) p(\{\mathbf{w}_c\}_c |\mathbf{x})$.

2.1 Multi-class case

We model the likelihood using a factor graph shown in Figure 1 (below the dashed line), where class 2 is assumed to be the correct class. $a_c = \langle \mathbf{w}_c, \mathbf{x} \rangle$ (inner product) is simply a linear discriminant, which is encoded by the factor $F_{wa}(\mathbf{w}_c, a_c) := \delta(a_c - \langle \mathbf{w}_c, \mathbf{x} \rangle)$ where δ is the Dirac/impulse function. To model the noise for practical purposes, Gaussian noise $\mathcal{N}(0, \beta^2)$ is added to a_c yielding f_c, which is represented by the factor $F_{af}(a_c, f_c) := \mathcal{N}(f_c - a_c, \beta^2)$. Our key assumption on the labeling mechanism is that the likelihood is non-zero only when f_2 is greater than all other f_c by a margin ε. This rule is implemented by first introducing a difference node $d_c = f_2 - f_c$ via the factor $F_{fd}(f_c, f_2, d_c) := \delta(d_c - (f_c - f_2))$. And then we check whether d_c is greater than ε: $F_d(d_c) := \mathbb{I}(d_c > \varepsilon)$, where here $\mathbb{I}(x) := 1$ if x is true and 0 otherwise.

By definition, the product of the factors below the dashed line in Figure 1 is $\alpha p(\mathbf{y}, \mathbf{a}, \mathbf{f}, \mathbf{d}|\mathbf{w}, \mathbf{x})$ where α

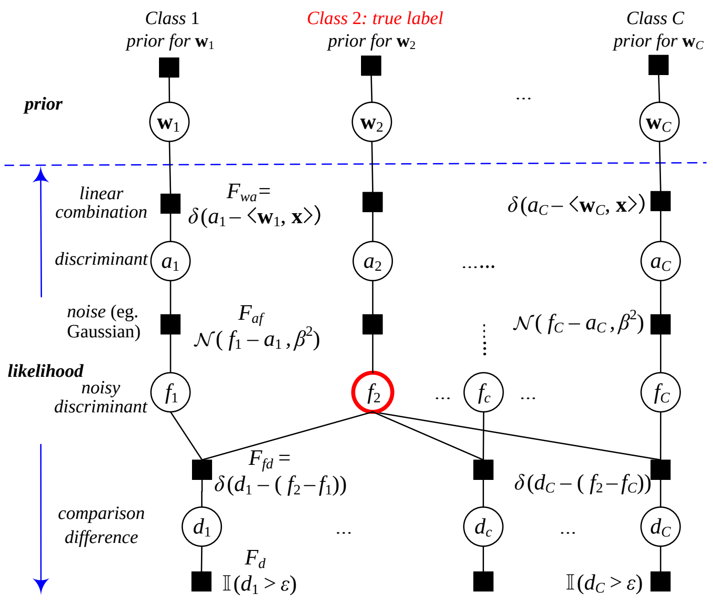


Figure 1: A factor graph for multi-class classification. The graph corresponds to an example $\mathbf{x}$ whose label is 2.

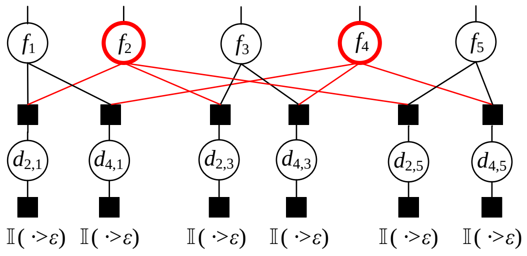

f_1 f_2 f_3 f_4 f_5
$d_{2,1}$ $d_{4,1}$ $d_{2,3}$ $d_{4,3}$ $d_{2,5}$ $d_{4,5}$
$\mathbb{I}(\cdot>\varepsilon)$

Figure 2: Multi-label classification via pairwise comparison. Class 2 and 4 are relevant. $d_{ij} = f_i - f_j$, where i is relevant and j is not.

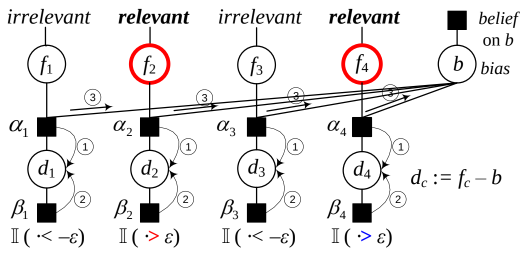

irrelevant
relevant
irrelevant
relevant
belief on b
bias
α_1 α_2 α_3 α_4
β_1 β_2 β_3 β_4
$d_c := f_c - b$
$\mathbb{I}(\cdot<-\varepsilon)$
$\mathbb{I}(\cdot>\varepsilon)$

Figure 3: Multi-label classification via total ordering and a global bias.

is independent of $\mathbf{w}$. So the product of all the factors in Figure 1 is proportional to $p(\mathbf{y}|\mathbf{w},\mathbf{x})p(\mathbf{w}|\mathbf{x})$ in $\mathbf{w}$. Therefore, the posterior $p(\{\mathbf{w}\}_c|\mathbf{y},\mathbf{x})$ can be obtained by simply marginalizing out $\mathbf{a},\mathbf{f}$, and $\mathbf{d}$ in the graph.

It is noteworthy that our likelihood model and factor graph are very similar to the TrueSkill$^{\text{TM}}$ diagram (Herbrich et al., 2007). Our factor graph corresponds to a fixed example $\mathbf{x}$ while in TrueSkill$^{\text{TM}}$ it corresponds to a match. The classes in our setting correspond to the teams. The parameter we estimate is the weights which are linearly combined using $\mathbf{x}$, while TrueSkill$^{\text{TM}}$ learns the players' skills, which are combined according to how teams are formed.

2.2 Multi-label case

In the multi-label scenario, the likelihood model can be extended using the pairwise comparison principle as in (Elisseeff & Weston, 2001). Figure 2 illustrates this idea where the noisy discriminant value f_c of relevant classes is enforced to exceed that of the irrelevant classes. Unfortunately, this method may cost $O(C^2)$ computations, which is not affordable. As a simplification, we assume a total order underlying the relevance of the labels. This translates to thresholding the discriminant of the classes by a global bias, as illustrated by Figure 3. One can further incorporate a "local" bias for each class by, *e.g.*, adding an artificial constant feature. This could even eliminate the need for global bias and decouple all the classes. We will compare these two models in experiment.

3 Online learning and inference

To estimate the model $\mathbf{w}$ and b from the training data, we adopt the Gaussian density filtering scheme (GDF, Maybeck, 1982) which is also used in (Herbrich et al., 2007). It employs a Gaussian prior $p_0(\mathbf{w})$, and at each iteration "absorbs" the likelihood of one training example $(\mathbf{x}^i,\mathbf{y}^i)$, computes the posterior

$$p_i(\mathbf{w}) := p(\mathbf{w}|\mathbf{x}^i,\mathbf{y}^i) \propto p_{i-1}(\mathbf{w})p(\mathbf{y}^i|\mathbf{w},\mathbf{x}^i),$$

and approximates it by a Gaussian that is closest in the sense of the Kullback-Leibler divergence.

In this paper, we restrict the prior and posterior to diagonal Gaussians though covariance could be incorporated at a much higher computational cost. As analyzed before, the posterior can be computed by marginalizing out $\mathbf{a},\mathbf{f}$, and $\mathbf{d}$ in the factor graph in Figure 3. This marginalization and subsequent Gaussian approximation can be effectively performed by expectation propagation (EP, Minka, 2001).

3.1 EP and message passing schedule

Intuitively, EP is similar to loopy belief propagation, but further approximates the messages as much as possible. In particular, it approximates the marginals of the factors by Gaussians via matching the first and second moments. Consequently, the posterior is also approximated by a Gaussian. Since the set of factors used in our model is the same as in TrueSkill$^{\text{TM}}$, the message formulae can be found in Table 1 of (Herbrich et al., 2007).

One important issue in implementing EP is the message passing schedule. There is no loop in all the graphical models from Figure 1 to 3. However, they all have non-Gaussian factors $\mathbb{I}(\cdot > \varepsilon)$, which necessitates running EP repeatedly on the graph. Observe that the shortest paths between these factors only involve factors $\{\alpha_c,\beta_c\}$, variables $\{d_c\}$ and bias b (see Figure 3).

Hence we only need to run EP iteratively over b and $\{\alpha_c, d_c, \beta_c\}_c$ as the arrows show. This significantly reduces the cost of each EP iteration from $O(DC)$ (for all weights) to $O(C)$.[1] In practice, suppose we only send messages from factors to variables, then we just need to repeatedly perform:

$$①\colon \{\alpha_c \to d_c\}_{c=1}^{C}; \;\; ②\colon \{\beta_c \to d_c\}_{c=1}^{C}; \;\; ③\colon \{\alpha_c \to b\}_{c=1}^{C}.$$

3.2 Dynamic learning

Our model is static, while many real world applications benefit from modeling temporal and spatial evolutions. For example, the categorization rule of news wire may vary with time. Besides, GDF depends on the random order of training examples and the belief of our model only propagates in the forward direction of the data stream. In the batch setting, we may add dynamic factors between the weight nodes of the factor graphs of adjacent time steps to allow smooth temporal variation. Dangauthier et al. (2008) extended TrueSkill™ to dynamic scenarios, where EP is performed back and forth over the whole dataset. While theoretically appealing, it is very expensive in both time and space, and hence we stick to GDF in this paper.

4 Generalization for multi-variate performance measure

Given a set of test data $X_{\text{test}} := \{\mathbf{x}^i \in \mathbb{R}^D : i \in [n]\}$, our task is to label $\mathbf{x}^i$ with a subset of $[C]$ using the posterior of weights $\{\mathbf{w}_c \sim \mathcal{N}(\boldsymbol{\mu}_c, \Sigma_c) : \text{class } c \in [C]\}$ where $\Sigma_c = \text{diag}(\sigma_{c,1}^2, \ldots, \sigma_{c,D}^2)$ and a global bias $b \sim \mathcal{N}(\mu_0, \sigma_0^2)$. Our objective is to optimize some multi-variate performance measure as found, *e.g.*, in many applications like information retrieval. In the case of binary classification, let $\mathbf{l} \in \{0,1\}^n$ be the reference label, and $\mathbf{y} \in \{0,1\}^n$ be a predicted label. Multi-variate measures such as F_β-score and area under ROC require that the predicted labels on the test set be optimized as a whole. For example, the F_1-score is defined as the harmonic mean of precision and recall:

$$F_1\text{-score}(\mathbf{y}, \mathbf{l}) := \frac{2\sum_{i=1}^n y^i \cdot l^i}{\sum_{i=1}^n y^i + \sum_{i=1}^n l^i}. \qquad (1)$$

Furthermore, in the multi-label scenario, Eq. (1) defines for each class c an F_1-score $F_1(c)$, and the overall macro-average F_1-score (Lewis et al., 2004) is defined as the average of $F_1(c)$ given by $\sum_c F_1(c)/C$. Clearly, it can be optimized by maximizing $F_1(c)$ for each c independently. So in the sequel, we will focus on binary classification and omit the class index c when it is clear from the context. Following the principles of Bayesian decision theory, we will optimize the expected value of the multi-variate performance measure under the posterior model.

Let y^i be a Bernoulli random variable, with $y^i = 1$ indicating $\mathbf{x}^i$ belongs to class c according to our model and 0 otherwise. *Given* an instantiation of $\mathbf{w}$ and b, we define the label y as $p(y = 1|\mathbf{w}, b) := \mathbb{I}(\langle \mathbf{w}, \mathbf{x}\rangle - b > 0)$. Therefore using the posterior of $\mathbf{w}$ and b, we have

$$p(y = 1) = \mathop{\mathbb{E}}_{\mathbf{w},b}[p(y = 1|\mathbf{w}, b)] = \Phi\left(\frac{\langle \boldsymbol{\mu}, \mathbf{x}\rangle - \mu_0}{\sqrt{\sigma_0^2 + \mathbf{x}^\top \Sigma \mathbf{x}}}\right), \quad (2)$$

where Φ is the cumulative distribution of the standard normal distribution. Since there is no theoretical guarantee that thresholding $p(y = 1)$ at 0.5 optimizes the multi-variate performance measure, we will label X_{test} in a much more principled Bayesian fashion, which is based on the joint distribution of all labels $\mathbf{y} := (y^1, \ldots, y^n)^\top$. *Given the model*, all labels $\{y^i\}$ are assumed to be independent. However after integrating out $\mathbf{w}$ and b, the independence is lost under

$$p(\mathbf{y}|X_{\text{test}}) := \mathop{\mathbb{E}}_{\mathbf{w},b}\left[\prod_{i=1}^n p\left(y^i|\mathbf{x}^i, \mathbf{w}, b\right)\right]. \qquad (3)$$

4.1 Expected F_1-score and optimization

Suppose we label X_{test} by $\mathbf{l} \in \{0,1\}^n$. Then the expected F_1-score will be

$$\text{ExpFs}(\mathbf{l}) := \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y})}\left[F_1\text{-score}(\mathbf{y}, \mathbf{l})\right],$$

and it is natural to choose the $\mathbf{l}$ which maximizes it:

$$\mathbf{l}^* := \mathop{\text{argmax}}_{\mathbf{l} \in \{0,1\}^n} \text{ExpFs}(\mathbf{l}) = \mathop{\text{argmax}}_{\mathbf{l} \in \{0,1\}^n} \mathop{\mathbb{E}}_{\mathbf{y}}[F_1\text{-score}(\mathbf{y}, \mathbf{l})]. \quad (4)$$

In general, closed form solutions rarely exist for maximizing expected multi-variate measures, and computing the expectation in (4) is intractable. So we resort to approximations. The most related algorithm that tackles the problem (4) is by Jansche (2007). Intuitively, for a fixed value of $\sum_i l^i$, $\mathbf{l}$ appears only in the numerator of the objective and optimization gets easier. However, in order to solve $\text{argmax}_{\mathbf{l}:\sum_i l^i = r} \text{ExpFs}(\mathbf{l})$ by simply sorting $\mathbb{E}[y^i]$, Jansche (2007) assumed that the y^i are independent which is unrealistic in both theory and practice.

Although we do recognize the importance of the correlation between $\{y^i\}$, it is too expensive to compute and store. We resort to a heuristic which respects the order of $p(y^i = 1)$ but tunes the threshold: given a certain threshold $\theta \in [0,1]$, we consider the class to be relevant if, and only if, $p(y^i = 1) > \theta$. So

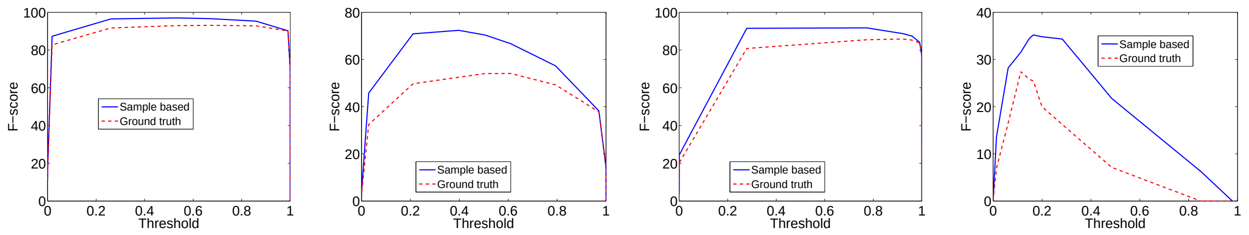
$$\mathbf{l}(\theta) := (\mathbb{I}(p(y^1 = 1) > \theta), \ldots, \mathbb{I}(p(y^n = 1) > \theta))^\top. \quad (5)$$

We now find the deterministic labeling by maximizing the expected F_1-score of $\mathbf{l}(\theta)$ wrt $\theta \in [0,1]$:

$$\theta^* := \text{argmax}_{\theta \in [0,1]} \; \text{ExpFs}(\mathbf{l}(\theta)). \qquad (6)$$

This reduction of search space from $\{0,1\}^n$ to $[0,1]$ significantly simplifies optimization, although $\mathbf{l}(\theta^*)$ is not

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



(a) class 4 (b) class 6 (c) class 55 (d) class 74

Figure 4: Example curves of $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ (solid blue) and F_1-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (dotted red) as functions of θ.

guaranteed to recover $\mathbf{l}^*$. Given $\mathbf{l} = \mathbf{l}(\theta)$, there is still no closed form to compute ExpFs$(\mathbf{l})$, so we evaluate it approximately based on samples $\{\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_S\}$ drawn independently and identically (*iid*) from $p(\mathbf{y})$ (via *iid* samples of $\mathbf{w}$ and b, then thresholding $\langle \mathbf{w}, \mathbf{x} \rangle - b$ at 0):

$$\widetilde{\text{ExpFs}}(\mathbf{l}) := \frac{1}{S} \sum_{s=1}^{S} \frac{\sum_{i=1}^{n} \tilde{y}_s^i l^i}{\sum_{i=1}^{n} \tilde{y}_s^i + \sum_{i=1}^{n} l^i}. \qquad (7)$$

The concentration of $\widetilde{\text{ExpFs}}(\mathbf{l})$ around ExpFs$(\mathbf{l})$ can be easily quantified in probability by McDiarmid's inequality (Herbrich, 2002, Theorem A.119). Unfortunately, a naïve application of (7) costs $O(nSCD)$ time, which is impractical for large datasets. We will design a more efficient algorithm in Section 4.3 using the "sufficient statistics". Before that, we first justify the labeling criteria in (6) and (5).

4.2 Soundness of Bayesian labeling criteria

Our labeling criteria $\mathbf{l}^* := \operatorname{argmax}_{\mathbf{l} \in \{0,1\}^n} \text{ExpFs}(\mathbf{l})$ is deemed as *sound* if $\mathbf{l}^*$ is "close" to the ground truth $\mathbf{y}^*$, as long as $p(\mathbf{w}, b)$ has been well estimated. However, the intractability of finding $\mathbf{l}^*$ precludes direct check. Fortunately, we can indirectly check the soundness of $\max_{\theta \in [0,1]} \text{ExpFs}(\mathbf{l}(\theta))$ by comparing two curves:

1. Expected F_1-score: ExpFs$(\mathbf{l}(\theta))$ versus θ.
2. True F_1-score: F_1-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ versus θ.

If these two curves are "similar", then it suggests that optimizing ExpFs$(\mathbf{l}(\theta))$ over θ is a good proxy to maximizing the real testing F_1-score against the ground truth. In practice, we can only use the sample based estimates $\widetilde{\text{ExpFs}}(\mathbf{l})$, $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ and its maximizer $\tilde{\theta}^*$.

Figure 4 shows an experimental result on comparing $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ and F_1-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ as functions of θ. It uses the `topics` group of Reuters dataset with 5 random samples. Due to space constraints, only 4 typical plots are shown. It can be observed that both curves follow roughly similar trend. Indeed, we only need the maximizer of $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$ (solid) to give approximately the max of F_1-score$(\mathbf{l}(\theta), \mathbf{y}^*)$ (dotted), *i.e.*

F_1-score$(\mathbf{l}(\tilde{\theta}^*), \mathbf{y}^*)$ be close to $\max_\theta F_1$-score$(\mathbf{l}(\theta), \mathbf{y}^*)$.

In this example, this is actually pretty much the case: the first term is 60.97 after summing up all the 101 classes, while the second term is 63.26.

4.3 Efficient calculation of empirical expected F_1-score

We design an efficient algorithm to compute $\widetilde{\text{ExpFs}}(\mathbf{l}(\theta))$, and use the Reuters dataset as an example. Here $C = 300$, $D = 5 \cdot 10^4$, $n = 10^5$, average number of non-zero features per example $\bar{D} = 70$, and we use $G = 20$ candidate θ. Our key idea is to collect three "sufficient statistics" $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ derived from the definition:

$$\widetilde{\text{ExpFs}}_c(\mathbf{l}(\theta_g)) = \frac{1}{S} \sum_{s=1}^{S} \qquad \text{(for class } c \text{ threshold } \theta_g\text{)}$$

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

$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ are cheap in space, but it is challenging to compute $\boldsymbol{\alpha}, \boldsymbol{\beta}$ efficiently due to the following constraints. 1) Memory or privacy constraints force the testing data to be accessed as a stream, which cannot be stored or revisited. In some cases, although revisiting is allowed, we can only afford at most a dozen of passes due to the cost of IO and parsing. 2) Sampling is also expensive in time and space. For the Reuters dataset, $\mathbf{w}$ costs $8CD$ bytes $= 120$ MB, and it takes $O(nC\bar{D})$ time to apply one sample to all the testing data, which means 2×10^9 time cost. Hence we can neither compute nor store over a dozen samples of $\mathbf{w}$, and so we let $S = 10$.

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. Furthermore, to ensure good statistical performance, we also store the seed of the random number generator for all the weight components, which allows us to apply the same samples of $\mathbf{w}$ to all the testing examples.

5 Empirical evaluation

In this section, we compare the empirical performance of several variants of our Bayesian online multi-label classifier (BOMC) with batch SVM and two state-of-the-art online learning classifiers. We focus on macro-average F_1-score and training time, and the dataset used is Reuters1-v2.

5.1 Dataset

The Reuters1-v2 dataset (Lewis et al., 2004) consists of three groups of categories: `topics`, `industries`, and `regions`, which contain 103, 354, and 366 categories (classes) respectively. It has 804,414 documents and every document is associated with zero or more labels from each of the three groups. In the experiment, the training and test sets were both sampled uniformly at random from the whole dataset.

`tf-idf` features (Salton & Buckley, 1988) were used for documents. On average each example has only about 77 non-zero features, although the whole training set contains about 35k features.

5.2 Algorithms

We compared different variants of BOMC with two state-of-the-art online learners. All these algorithms randomized the order of the training examples.

To train BOMC, the feature weights had prior $\mathcal{N}(0, 1)$, while the prior of the global bias was $\mathcal{N}(0, 10^4)$. The noise level was $\beta = 0.01$ and the margin $\varepsilon = 1$. EP was used for inference. On average, the relative change of mean and variance of the messages falls below 10^{-3} after only three iterations. All variants of BOMC were implemented in F#.

In practice, many classes only have very few positive examples, and this skewness is commonly dealt with by two heuristics. Yang (2001); Lewis et al. (2004) tune the threshold by cross validation (CV), which translates the separating hyperplane towards the negative region. However, CV is expensive and is intrinsically batch. 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 the less common classes. Using this heuristic with SVM$^{\text{light}}$, Lewis (2001) won the TREC-2001 batch filtering evaluation.

All algorithms under comparison perform very poorly when neither heuristic is used. Therefore we assume some prior knowledge such as the frequency ratio of positive and negative examples (denoted by r). BOMC can easily encode this prior by changing the factor $\mathbb{I}(\cdot > 1)$ to $\mathbb{I}(d > \ln(e + 1/r))$ for positive examples. The intuition behind this choice is that if the dataset has only a small fraction of positive examples, then the model is expected to correctly classify these positive examples with a higher margin (or confidence).

BMOC with sampling (`BOMC_Sample`) To label the test data, we drew 5 samples from the posterior of the learned model. 10 or 20 samples did not lead to any improvement.

BOMC with class mass normalization (`BOMC_CMN`) A much simpler but non-Bayesian heuristic for tuning the threshold is by matching the zero-th order moment (Zhu et al., 2003): sort $p(y^i = 1)$ and threshold by making the class ratio in the testing set identical to that in the training set.

BMOC: training all classes independently (`BOMC_IND_CMN` and `BOMC_IND_Sample`) We also tried training all classes independently, *i.e.* each class c has its own bias b_c without using the shared global bias. Now the posterior 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.

Batch SVM (`SVM_Batch`) Although online learning is the focus of this paper, we also tried out a batch SVM as a baseline. It has an unfair advantage over online learning because it revisits training examples. One SVM is trained for each class independently. To deal with class skewness, we tuned the threshold using nested CV (Yang, 2001) which outperformed the heuristic of reweighting false positive and false negative. We used the liblinear[2] implemented in C.

LaSVM (`LaSVM`) `LaSVM`[3] is an online solver for SVM, which, according to Bordes et al. (2005), takes a single pass to achieve similar generalization performance as the batch SVM. We tuned the bias using the CV based strategy because of its superior empirical performance.

Passive-Aggressive (`PA`) `PA` (Crammer et al., 2006) optimizes the regularized risk of the current example at each step. We tuned the threshold by CV, which can use `PA` or batch SVM. We call them `PA_OnlineCV` and `PA_BatchCV` respectively. `PA` is equivalent to running liblinear for one iteration.

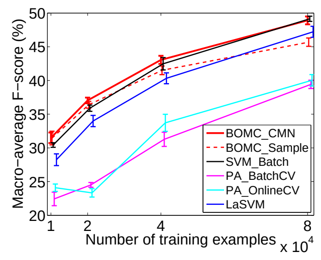
5.3 Results

We compared all algorithms with respect to the testing macro-average F_1-score, and the CPU time cost for training. We randomly sampled the training and test data five times which allowed us to plot error bars.

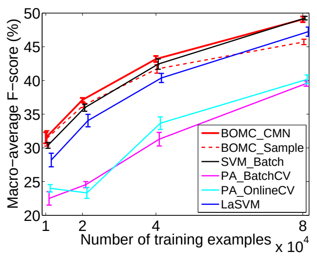
5.3.1 Macro-average F_1-score

Figure 5 shows the macro-average F_1-score as a function of the number of training examples. Among all online learners, `BOMC_CMN` achieves the highest macro-average F_1-score most of the time. `BOMC_Sample` is inferior to `BOMC_CMN`, but still competitive. Notice that CMN is also a method to choose the threshold, so it

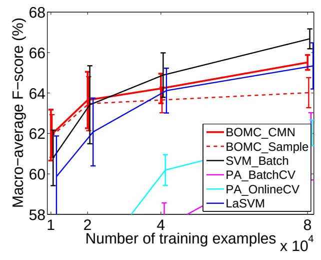
[2] `http://www.csie.ntu.edu.tw/~cjlin/liblinear`
[3] `http://leon.bottou.org/projects/lasvm`



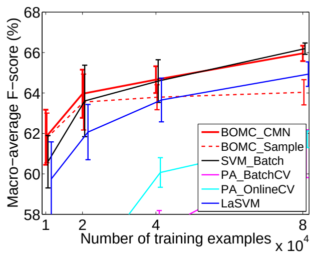
(a) #test = 200k, `industries`



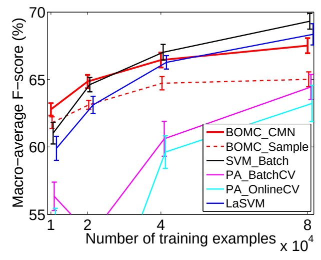
(b) #test = 700k, `industries`



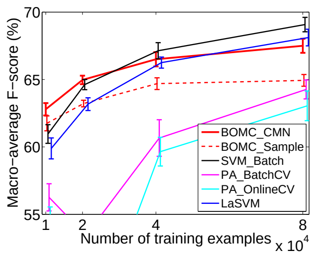
(c) #test = 200k, `regions`



(d) #test = 700k, `regions`

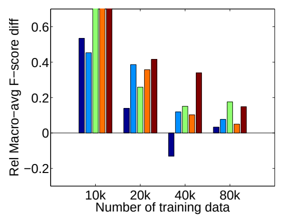


(e) #test = 200k, `topics`

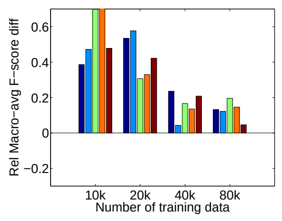


(f) #test = 700k, `topics`

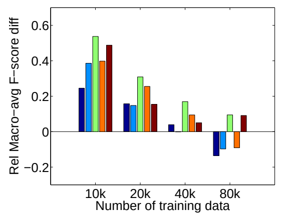
Figure 5: Comparison of F_1-score for the category groups `industries`, `regions`, and `topics`.



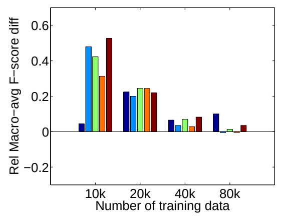
(a) #test = 200k, `industries`



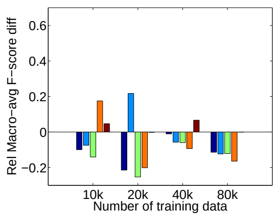
(b) #test = 700k, `industries`



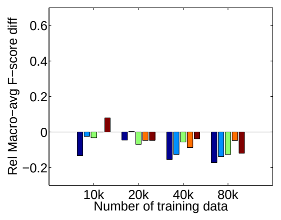
(c) #test = 200k, `regions`



(d) #test = 700k, `regions`



(e) #test = 200k, `topics`



(f) #test = 700k, `topics`

Figure 6: F_1-score of coupled models minus F_1-score of independent models. Height of bars represents relative difference. The bars in five colors correspond to five random draws of traning/testing examples.

suggests that the model is well trained, and our sample based method to find the threshold can be improved. Comparing Figure 5(c) and Figure 5(d) on the group `regions`, we observe that `BOMC_CMN` significantly benefits from a large test set. This is not surprising because the assumption made by CMN is more likely to hold when the test set is large.

Unsurprisingly, `SVM_Batch` usually yields the highest F_1-score. However, `BOMC_CMN` often performs as well as or even better than `SVM_Batch` by a single pass, especially on the dataset `industries`, or when the training set size is medium. `PA_OnlineCV` and `PA_BatchCV` perform worse than other algorithms probably due to being susceptible to noise. In contrast, `LaSVM` employs a removal step to handle noise, and converges to the true SVM solution if multiple passes are run. `LaSVM` is slightly worse than `BOMC_CMN`, but competitive.

5.3.2 Comparing coupled and decoupled BOMC

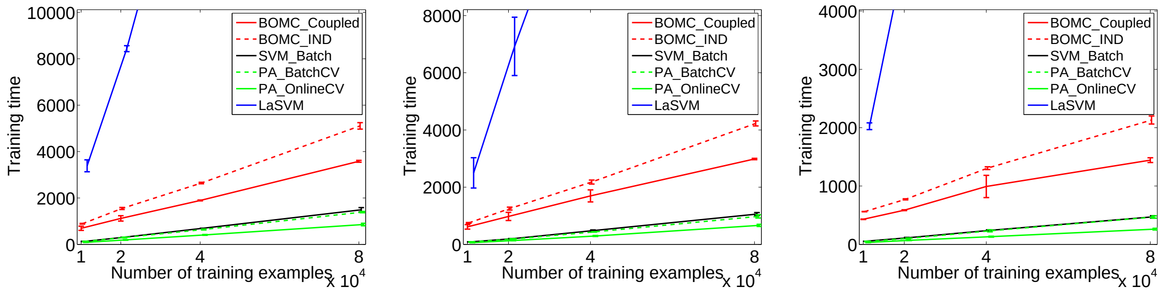
The benefit of modeling the interaction between labels has been confirmed by existing algorithms such as (Rousu et al., 2006; Ghamrawi & McCallum, 2005). Our multi-label model in Figure 3 loosely couples all the classes via the global bias. A natural question is why not introduce a "local" bias to all the classes and learn the model of all the classes independently. We now demonstrate in Figure 6 how much the macro-average F_1-score of `BOMC_CMN` (F_{coupled}) is relatively higher than that of `BOMC_IND_CMN` (F_{ind}) as quantified by the relative macro-average F_1-score difference:

$$200 \cdot (F_{\text{coupled}} - F_{\text{ind}})/(F_{\text{coupled}} + F_{\text{ind}}).$$

On the `industries` and `regions` groups, `BOMC_CMN` delivers significantly higher macro-average F_1-score than `BOMC_IND_CMN`. We observed that the global bias in `BOMC_CMN` is much more confidently learned (higher precision) than the feature weights and the local bias in `BOMC_IND_CMN`. This is because the global bias serves as a hub and is updated more often. On the `topics` group, `BOMC_IND_CMN` performs slightly better.

5.3.3 Training time

Figure 7 presents the CPU time cost for training with these algorithms. If an algorithm uses CV, then only the cost for training the final model is included.



(a) `industries` (b) `regions` (c) `topics`

Figure 7: CPU time for training.

The key observation is that the training time of all algorithms except `LaSVM` is linear in the number of training examples. This matches their algorithmic property. Training `BOMC_IND_CMN` takes slightly more time than `BOMC_CMN`, which suggests that the cost for updating local bias outweighs the savings from closed form posterior. `PA` and `SVM_Batch` can be trained faster than `BOMC` by a factor of 2–3, which could be attributed to the programming language (`C` vs F#).

Although `LaSVM` is the online learner which achieves closest testing F_1-score to `BOMC`, it takes much more training time. This is because `LaSVM` operates in the dual and has not been optimized for linear kernels.

6 Conclusion and future directions

We proposed a Bayesian online learning algorithm for multi-label classification. It uses Gaussian density filtering for efficient inference and can label unseen data in a principled manner, as opposed to the ad hoc thresholding schemes used in frequentist approaches. Empirically, it delivers favorable macro-average F_1-score compared with state-of-the-art online learners, and is even competitive with batch SVM.

This work can be extended in several directions. We are designing efficient algorithms to train the dynamic models briefed in Section 3.2, which is expected to yield a more accurate model. Label noise studied by Kim & Ghahramani (2006) can also be modeled in a straightforward way. For example, the common noise that flips the label by a probability ρ can be modeled by replacing the factor $\mathbb{I}(d > \varepsilon)$ with $\rho\mathbb{I}(d > \varepsilon) + (1 - \rho)\mathbb{I}(d < -\varepsilon)$. Finally, label hierarchies can also be conveniently incorporated using graphical models.

References

Bordes, A., Ertekin, S., Weston, J., & Bottou, L. (2005). Fast kernel classifiers with online and active learning. *JMLR*, *6*, 1579–1619.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *JMLR*, *7*, 551–585.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *JMLR*, *3*, 951–991.

Dangauthier, P., Herbrich, R., Minka, T., & Graepel, T. (2008). Trueskill through time: Revisiting the history of chess. In *NIPS*, 337–344.

Elisseeff, A., & Weston, J. (2001). A kernel method for multi-labelled classification. In *NIPS*, 681–688.

Ghamrawi, N., & McCallum, A. (2005). Collective multi-label classification. In *CIKM*, 195–200.

Herbrich, R. (2002). *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA: MIT Press.

Herbrich, R., Minka, T., & Graepel, T. (2007). Trueskill$^{\mathrm{TM}}$: A Bayesian skill ranking system. In *NIPS*, 569–576.

Jansche, M. (2007). A maximum expected utility framework for binary sequence labeling. In *ACL*, 736–743.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *ICML*, 377–384.

Kim, H.-C., & Ghahramani, Z. (2006). Bayesian gaussian process classification with the EM-EP algorithm. *IEEE PAMI*, *28*(12), 1948–1959.

Lewis, D. (2001). Applying support vector machines to the TREC-2001 batch filtering and routing tasks. In *Text REtrieval Conference*, 286–292.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *JMLR*, *5*, 361–397.

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 on Text Learning*.

Minka, T. (2001). *Expectation Propagation for approximative Bayesian inference*. Ph.D. thesis, MIT Media Labs.

Opper, M. (1998). A Bayesian approach to online learning. In *On-line Learning in Neural Networks*, 363–378. Cambridge University Press.

Qi, G.-J., Hua, X.-S., Rui, Y., Tang, J., Mei, T., & Zhang, H.-J. (2007). Correlative multi-label video annotation. In *International Conference on Multimedia*, 17–26.

Rousu, J., Sunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification methods. *JMLR*, *7*, 1601–1626.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523.

Seki, K., & Mostafa, J. (2005). An application of text categorization methods to gene ontology annotation. In *SIGIR*, 138–145.

Yang, Y. (2001). A study on thresholding strategies for text categorization. In *SIGIR*, 137–145.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, 912–919.