# Conditional Bernoulli Mixtures for Multi-Label Classification

**Cheng Li** CHENGLI@CCS.NEU.EDU
**Bingyu Wang** RAINICY@CCS.NEU.EDU
**Virgil Pavlu** VIP@CCS.NEU.EDU
**Javed Aslam** JAA@CCS.NEU.EDU
College of Computer and Information Science, Northeastern University, Boston, MA 02115, USA

## Abstract

Multi-label classification is an important machine learning task wherein one assigns a subset of candidate labels to an object. In this paper, we propose a new multi-label classification method based on *Conditional Bernoulli Mixtures*. Our proposed method has several attractive properties: it captures label dependencies; it is efficient both in training and in prediction; it reduces the multi-label problem to several standard binary and multi-class problems; it subsumes the classic independent binary prediction and power-set subset prediction methods as special cases; and it exhibits accuracy and/or computational complexity advantages over sophisticated approaches such as classifier chains, conditional dependency networks, and conditional random fields.

We demonstrate two implementations using logistic regression and gradient boosted trees, together with training procedures based on Expectation Maximization. We further derive an efficient prediction procedure based on dynamic programming, thus avoiding the cost of examining an exponential number of potential label subsets. Experimental results show the effectiveness of the proposed method against competitive alternatives on benchmark multi-label datasets.

## 1. Introduction

Multi-label classification is an important machine learning task wherein one assigns a *subset* of candidate labels to an object. Such problems arise naturally in the context of medical diagnostics, text classification, image tagging, and a host of other areas. For example, in the medical domain, a patient may present multiple illnesses or undergo a procedure with multiple billing codes; in news categorization, an article can be associated with multiple categories; in image tagging, a photo may have multiple tags (see Figure 1).

*Figure 1.* An image from the NUS-WIDE multi-label dataset. Independent binary logistic regressions predict the labels {`clouds, lake, sky, sunset, water`}. Our proposed method correctly predicts {`clouds, lake, sky, sunset, water, reflection`}, because it captures the dependency between `reflection` and the other labels.

Multi-label classification is particularly difficult in the common case when the labels are *many* and *dependent*. In this case, simple approaches such as independent binary label prediction and Power-Set subset prediction fail due to label dependency and combinatorial explosion, respectively. More sophisticated approaches such as classifier chains, conditional dependency networks, and conditional random fields suffer due to accuracy/computational-complexity trade-offs in training and/or prediction. We introduce *Conditional Bernoulli Mixtures* for multi-label classification to address these challenges, yielding multi-label classifiers that are accurate and efficient.

Formally, in a multi-label classification problem, we are given a set of label candidates $\mathcal{L} = \{1, 2, ..., L\}$. Every data point $\mathbf{x} \in \mathbb{R}^D$ matches a subset of labels $\mathcal{Y} \subseteq \mathcal{L}$,

*Proceedings of the 33^rd International Conference on Machine Learning*, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

which is often also written in the form of a binary vector $\mathbf{y} \in \{0,1\}^L$, with each bit $y_\ell$ indicating the presence or absence of the corresponding label. The goal of learning is to build a classier $h : \mathbb{R}^D \mapsto \{0,1\}^L$ which maps an instance to a subset of labels. The label subset $\mathcal{Y}$ can be of arbitrary size; when the size is restricted to be 1, the problem is called multi-class classification. Furthermore, if the total number of label candidates $L$ is 2, the problem becomes binary classification.

Various measures can be used for evaluating multi-label classifiers. Among them, three commonly used measures are *Subset Accuracy*, *Jaccard Index* and *Hamming Loss*. Let $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ be a multi-label dataset with ground truth labels, and $\{\hat{\mathbf{y}}_n\}_{n=1}^N$ be the predictions made by a classifier. Subset Accuracy generalizes the conventional multi-class accuracy notion:

$$\frac{1}{N}\sum_{n=1}^{N} \mathbb{1}[\mathbf{y}_n = \hat{\mathbf{y}}_n], \tag{1}$$

where $\mathbb{1}[\cdot]$ is the indicator function. In computing Subset Accuracy, a predicted subset is considered correct only when it matches the true subset exactly. Admittedly, this metric is very stringent in evaluation. However, optimizing for Subset Accuracy is a very interesting algorithmic design and research problem, as it encourages the classifiers to output coherent and complete predictions. The optimal classification for Subset Accuracy is achieved by predicting the mode of the conditional joint probability

$$h^*(\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}). \tag{2}$$

Jaccard Index measures the overlap between the true set and the predicted set using the size of their intersection divided by the size of their union, and Hamming Loss measures bit-wise errors. The theoretical analysis in (Dembczyński et al., 2012) shows that a multi-label classifier designed for optimizing one measure may perform poorly under other measures. We focus on the multi-label classification task with Subset Accuracy as the learning objective and primary evaluation measure, but for completeness we also report the Jaccard Index and Hamming Loss results in supplementary material.

According to (2), optimizing subset accuracy requires estimating the conditional joint probability $p(\mathbf{y}|\mathbf{x})$, which captures conditional label dependencies given features. One naive approach is to assume conditional independence among labels $p(\mathbf{y}|\mathbf{x}) = \prod_{\ell=1}^L p(y_\ell|\mathbf{x})$, which reduces a multi-label problem into $L$ binary classification problems. This approach is called *Binary Relevance* (BinRel) (Tsoumakas & Katakis, 2007), and is widely used due to its simplicity. One obvious disadvantage is that the individual label predictions can often be conflicting, such as tagging `cat` but not `animal` for an image.

At the other extreme is the *Power-Set* (PowSet) approach (Tsoumakas & Katakis, 2007), treating each label subset as a class, and trains a multi-class classifier. As a consequence, one would be restricted in practice to predicting only label subsets seen in the training data and always assigning zero probabilities to unseen subsets; even for many of the subsets observed there would likely be a scarcity of training data. Power-Set is handling label dependencies and its predictions are coherent, but the method is often infeasible on the exponential number of label sets.

**Bernoulli Mixtures.** Mixture models offer a flexible and powerful framework for many multivariate density estimation problems. A mixture generally takes the form

$$p(\mathbf{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi^k p(\mathbf{y}|\boldsymbol{\beta}^k), \tag{3}$$

which approximates the complex joint $p(\mathbf{y}|\boldsymbol{\theta})$ as a weighted combination of $K$ component densities $p(\mathbf{y}|\boldsymbol{\beta}^k)$, each of which is typically much simpler. The EM algorithm can iterate between estimating the mixture coefficients $\boldsymbol{\pi}$ and fitting $p(\mathbf{y}|\boldsymbol{\beta}^k)$ until convergence.

Bernoulli Mixtures (BM) is a classic model for multi-dimensional binary variable density estimation (Lazarsfeld et al., 1968; McLachlan & Peel, 2004; Bishop, 2006) where the learnability of component densities is realized by assuming independence of variables within each mixture component. Thus each component density $p(\mathbf{y}|\boldsymbol{\beta}^k)$ becomes a product of Bernoulli densities and this leads to

$$p(\mathbf{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi^k \prod_{\ell=1}^{L} \text{Ber}(y_\ell|\mu_\ell^k), \tag{4}$$

where $\boldsymbol{\theta} = \{\pi^k, \mu_\ell^k, \ell = 1,2,...,L; k = 1,2,...,K\}$ denotes the set of parameters. BM has two attractive properties. First, *dependencies*: although the $L$ variables are assumed to be independent inside each component, they are in general dependent in the mixture (they are forced independent only when $K = 1$). This can be verified by computing the following covariance matrix and observing that it is non-diagonal for $K \geq 2$ (Bishop, 2006).

$$\text{cov}[\mathbf{y}] = \sum_{k=1}^{K} \pi^k[\boldsymbol{\Sigma}^k + \boldsymbol{\mu}^k(\boldsymbol{\mu}^k)^T] - \mathbb{E}(\mathbf{y})\mathbb{E}(\mathbf{y})^T, \tag{5}$$

where $\mathbb{E}(\mathbf{y}) = \sum_{k=1}^K \pi^k \boldsymbol{\mu}^{\mathbf{k}}$, $\boldsymbol{\Sigma}^{\mathbf{k}} = \text{diag}\{\mu_\ell^k(1-\mu_\ell^k)\}$ and $\boldsymbol{\mu}^k = (\mu_1^k, \mu_2^k, ..., \mu_L^k)^T$.

Second, *computational cost*: the full factorization for each component makes BM fit efficient via the EM algorithm.

In this paper, we propose a new multi-label classification method which approximates $p(\mathbf{y}|\mathbf{x})$ based on the Bernoulli

Mixtures. The method simultaneously *learns* the label dependencies from data and *trains* classifiers to account for such dependencies.

## 2. Conditional Bernoulli Mixtures

For multi-label classification, we model the conditional probability $p(\mathbf{y}|\mathbf{x})$ with a discriminative extension of BM, capturing conditional dependencies among binary labels given features. (Dembczyński et al., 2012) shows that labels could be largely conditionally independent given features, namely $p(\mathbf{y}|\mathbf{x}) \approx \prod_{\ell=1}^{L} p(y_\ell|\mathbf{x})$, even when labels are strongly marginally dependent ($p(\mathbf{y}) \neq \prod_{\ell=1}^{L} p(y_\ell)$), as long as each label is highly predictable from features. It suffices to model only the correlations among errors, i.e. the part not (easily) predictable from features. Thus conditioning on features greatly reduces the need to estimate label correlations and makes learning much easier; *the conditional on* $\mathbf{x}$ *is essential for good approximation and effective training*, without making prior assumptions of the form of label dependencies (as many other methods do).

Making both mixture coefficients and Bernoulli probabilities conditional on $\mathbf{x}$, we obtain our proposed model, Conditional Bernoulli Mixtures (CBM):

$$p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi(z^k = 1|\mathbf{x},\boldsymbol{\alpha}) \prod_{\ell=1}^{L} b(y_\ell|\mathbf{x},\boldsymbol{\beta}_\ell^k) \qquad (6)$$

where $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}_\ell^k, \ell = 1,2,...,L; k = 1,2,...,K\}$ are model parameters to be learned, and $\mathbf{z} = (z^1,...,z^K)$ is a hidden categorical indicator variable, such that $z^k = 1$ if the data point is assigned to component $k$. We use $\pi$ and $b$ to represent categorical distributions and Bernoulli distributions, respectively.

CBM reduces a multi-label problem to a multi-class problem and several binary problems, approaching $p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$ akin to *divide and conquer*: the categorical distribution $\pi(z^k = 1|\mathbf{x},\boldsymbol{\alpha})$ assigns each instance to 1 of $K$ components probabilistically. Its goal is to divide the feature space into several regions such that each region only has weak conditional label correlations and can be approximated by a simple component. $\pi(z^k = 1|\mathbf{x},\boldsymbol{\alpha})$ can be instantiated by any multi-class classifier which provides probabilistic estimations, such as a Multinomial Logistic Regression.

Inside each region $k$, the local conditional joint density is approximated by a product of conditional marginal densities. Each local binary label predictor $b(y_\ell|\mathbf{x},\boldsymbol{\beta}_\ell^k)$ estimates the probability of getting label $y_\ell$ from component $k$ for the data point $\mathbf{x}$, and can be instantiated by any binary classifier which provides probabilistic estimations, such as a binary Logistic Regression. All $K$ components together with the space-partition function are learned jointly so as to break the global label correlation into simple forms.

On one extreme, if we only use one component (and hence $\pi(z^k = 1|\mathbf{x},\boldsymbol{\alpha})$ plays no role), all labels are conditionally independent and the CBM degenerates to Binary Relevance. On the other extreme, we can assign one component for each unique label subset and make each $b(y_\ell|\mathbf{x},\boldsymbol{\beta}_\ell^k)$ the corresponding binary constant, then the overall model simply selects one label subset from all possible subsets, which is conceptually the same as the Power-Set approach. By varying the number of components and the complexity of each component, CBM can provide a continuous spectrum between these two extremes. The main advantage of these reduction methods compared with models specifically designed for multi-label problem is that the reduction approach makes it easier to reuse many well-developed multi-class and binary classifiers. We demonstrate two instantiations of the CBM model, one with Logistic Regressions, and the other with Gradient Boosted Trees.

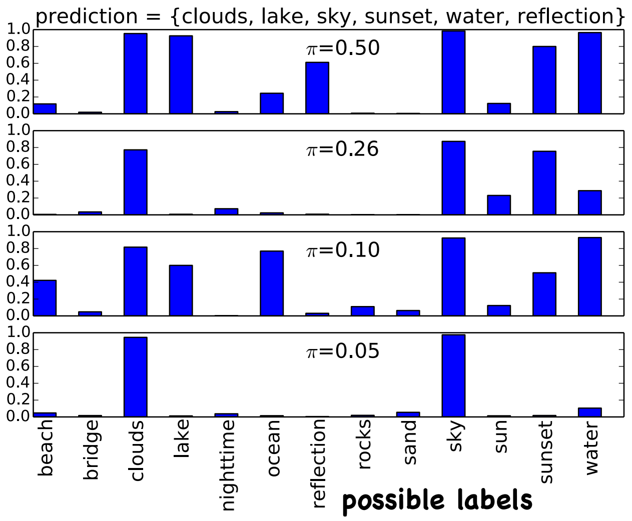

*Figure 2.* The top 4 most influential components for the test image in Fig. 1. $\pi$ values indicate component mixing coefficients. Each bar represents an individual label probability in one component. Labels with near zero probabilities are not displayed here.

**A Case Study.** Before diving into model training details, we illustrate how CBM makes joint label classification with an example. Figure 1 shows a test image in the NUS-WIDE dataset, for which the matched label `reflection` is missed by BinRel but is captured by our CBM. For this image, the most influential components produced by CBM are shown in Figure 2. Simply averaging individual label probabilities by component weights gives the conditional marginals $p(\texttt{lake} \in \mathcal{Y}|\mathbf{x},\boldsymbol{\theta}) = 0.56$, $p(\texttt{water} \in \mathcal{Y}|\mathbf{x},\boldsymbol{\theta}) = 0.69$, $p(\texttt{sunset} \in \mathcal{Y}|\mathbf{x},\boldsymbol{\theta}) = 0.66$, and $p(\texttt{reflection} \in \mathcal{Y}|\mathbf{x},\boldsymbol{\theta}) = 0.32$, which indicates that `reflection` by itself is not a likely label. However from the CBM joint density $p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$ we can also compute Pearson correlation coefficients $\rho_{\texttt{reflection},\texttt{lake}} = 0.50$,

$\rho_{\texttt{reflection,water}} = 0.40$, $\rho_{\texttt{reflection,sunset}} = 0.17$ and observe that `reflection` is positively correlated with `lake`, `water`, and `sunset`. In fact, the joint probability asserts that the subset {`clouds, lake, sky, sunset, water, reflection`} is the most probable one, with probability 0.09. By contrast, the subset {`clouds, lake, sky, sunset, water`} has a lower probability 0.06. Therefore, although individually unlikely, the label `reflection` is correctly added to the mostly likely subset, due to label correlations.

## 3. Training CBM with EM

CBM can be trained by maximum likelihood estimation. Below we first derive a generic EM algorithm that works for any instantiation of the components, and then consider two concrete instantiations.

To make the mathematical derivation clear, we use different symbols for random variable and values of random variables. We use $\mathbf{Y}_n$ when treating the labeling of $\mathbf{x}_n$ as an unknown random variable and $\mathbf{y}_n$ when referring to its specific labeling assignment given in the training set. The likelihood for the data set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ is

$$\prod_{n=1}^{N}\{\sum_{k=1}^{K}[\pi(z_n^k = 1|\mathbf{x}_n, \boldsymbol{\alpha}) \prod_{\ell=1}^{L} b(y_{n\ell}|\mathbf{x}_n, \boldsymbol{\beta}_\ell^k)]\}.$$

Since the model contains hidden variables, we use EM to minimize an upper bound of the negative log likelihood. Denoting the posterior categorical distribution $p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{y}_n, \boldsymbol{\theta})$ as $\Gamma(\mathbf{z}_n) = (\gamma_n^1, \gamma_n^2, ..., \gamma_n^K)$, the upper bound can be written as

$$J = \sum_{n=1}^{N} \mathbb{KL}(\Gamma(\mathbf{z}_n)||\pi(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\alpha})) + \sum_{k=1}^{K}\sum_{\ell=1}^{L}\sum_{n=1}^{N} \gamma_n^k \mathbb{KL}(\text{Ber}(Y_{n\ell}|y_{n\ell})||b(Y_{n\ell}|\mathbf{x}_n, \boldsymbol{\beta}_\ell^k)), \tag{7}$$

where $\text{Ber}(Y_{n\ell}|y_{n\ell})$ denotes the Bernoulli distribution with head probability $y_{n\ell}$.

**E Step** re-estimates the posterior membership probability of each data point belonging to each component.

$$\gamma_n^k = \frac{\pi(z_n^k = 1|\mathbf{x}_n, \boldsymbol{\alpha}) \prod_{\ell=1}^{L} b(y_{n\ell}|\mathbf{x}_n, \boldsymbol{\beta}_\ell^k)}{\sum_{k=1}^{K} \pi(z_n^k = 1|\mathbf{x}_n, \boldsymbol{\alpha}) \prod_{\ell=1}^{L} b(y_{n\ell}|\mathbf{x}_n, \boldsymbol{\beta}_\ell^k)}. \tag{8}$$

**M Step** updates all model parameters. The bound (7) shows that the optimization of all parameters can be nicely decomposed into a series of separate optimization problems, one for each component:

$$\boldsymbol{\alpha}_{new} = \underset{\boldsymbol{\alpha}}{\text{argmin}} \sum_{n=1}^{N} \mathbb{KL}(\Gamma(\mathbf{z}_n)||\pi(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\alpha})), \tag{9}$$

$$\boldsymbol{\beta}_{\ell\ new}^k = \underset{\boldsymbol{\beta}_\ell^k}{\text{argmin}} \sum_{n=1}^{N} \gamma_n^k \mathbb{KL}(\text{Ber}(Y_{n\ell}|y_{n\ell})||b(Y_{n\ell}|\mathbf{x}_n, \boldsymbol{\beta}_\ell^k)). \tag{10}$$

The optimization problem defined in (9) is a multi-class classification problem with target class distribution (soft labels) $\Gamma(\mathbf{z}_n) = (\gamma_n^1, \gamma_n^2, ..., \gamma_n^K)$. The training goal for the $\pi(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\alpha})$ is to assign each data point to certain components based only on its features, such that the assignment matches the posterior component membership based on both features and labels.

The optimization problem defined in (10) is a weighted binary classification problem. $\mathbf{x}_n$ has target label $y_{n\ell}$ and weight $\gamma_n^k$. The prediction component $b(Y_{n\ell}|\mathbf{x}_n, \boldsymbol{\beta}_\ell^k)$ is trained as a binary classier for label $\ell$ in component $k$, based only on training data within (soft) component $k$.

To train CBM, we iterate between the E step and the M step, until the upper bound (7) converges (see Algorithm 1). In

**Algorithm 1** Generic Training for CBM

```
1: repeat
2:    E Step
3:    for $n = 1, 2, ..., N$; $k = 1, 2, ..., K$ do
4:       update $\gamma_n^k$ as in (8)
5:    M Step
6:    update $\boldsymbol{\alpha}$ as in (9)
7:    for $K = 1, 2, ..., K$; $\ell = 1, 2, ..., L$ do
8:       update $\boldsymbol{\beta}_\ell^k$ as in (10)
9: until converge
```

practice, the EM algorithm can get stuck in local optima, so careful initializations are often necessary for good performance. Due to the natural connection between CBM and the BM, one simple way of initializing CBM is to first fit a BM just for labels without looking at features. BM can be trained very quickly by a simpler EM algorithm (Bishop, 2006). We use several random starts for a set of BM models, and select the one with the best training objective to initialize the CBM procedure.

### 3.1. CBM with Logistic Regression learners

In the simplest form, all models are linear, so we employ a multinomial logistic regression for the mixture $\pi$, and a binary logistic regressions for each predictor $b$. The outer loop of the training procedure is the EM algorithm described above. In the M step, to optimize each logistic regression, we take derivatives of the corresponding objective w.r.t. its parameters and update all parameters with L-BFGS method (Nocedal & Wright, 2006), which is the state-of-the-art optimization method for large scale logistic regression. All objectives defined in (9) and (10) are convex (even though the overall EM objective $J$ is non-convex). In each M step, we can start with the existing model, and perform a few incremental update steps until it converges. To avoid over fitting, we also add an L2 regularization (Gaussian priors) to all the coefficients. Overall LR makes CBM quite fast.

**CBM+LR Complexity.** The overall complexity of the training procedure for CBM is dominated by the updates in M steps and depends on the complexities of the binary and multi-class learning algorithms used. Assuming the base classifiers are Logistic Regression (LR), we measure the complexity for each L-BFGS update for all model weights. Let $N$ be the number of instances, $D$ the number of features, $L$ the number of labels, $K$ the number of components, $C$ the number of unique label subsets in the training set, and $A$ the average label subset size. In CBM with LR, each update takes $\mathcal{O}(KLDN)$. For comparison, each update takes $\mathcal{O}(LDN)$ in Binary Relevance with LR, $\mathcal{O}(CDN)$ in Power-Set with LR, and $\mathcal{O}(C(AD+L^2)N)$ in pair-wise CRF with support approximate inference (Ghamrawi & McCallum, 2005). Since typically $KL < C$, CBM is slower than BinRel, but faster than PowSet and pairCRF.

## 3.2. CBM with Gradient Boosting learners

Many datasets require non-linear decision boundaries, in which case the CBM model with Logistic Regressors may not have enough explanation power. To make CBM non-linear, we use gradient boosted trees (GB) for both $\pi$ and $b$ (we also have tried CBM with GB-$\pi$ and LR-$b$, and vice versa). The original gradient boosted trees algorithm described in (Friedman, 2001) is designed for standard multi-class problems, and does not take label distributions or instance weights as inputs, thus some modifications are necessary. The target label distribution is easy to deal with: one still takes the functional gradient of the objective w.r.t. each scoring function, where the objective is defined by the target distribution. To handle instance weights, one ignores them when calculating functional gradients, and perform a weighted least square fit when fitting each regression tree to the gradients. Unlike logistic regression, where all parameters can be easily updated, gradient boosting introduces new trees to the ensemble while keeping old trees untouched. Thus our M step here is slightly different from before: rather than re-adjusting all parameters to optimize the objective, we improve the objective by adding a few more trees. This amounts to a partial M step, and the resulting EM algorithm is usually called the generalized EM algorithm (Gupta & Chen, 2011). We emphasize that the use of boosting as an underlying non-linear classifier here is just for demonstration purposes. Other classifiers such as neural networks could also be used. This is in direct contrast to AdaBoost.MH and Adaboost.MR (Schapire & Singer, 2000) which specifically employ boosting to optimize Hamming Loss and Rank Loss.

## 3.3. Validation of Training Procedure on Artificial Data

To validate the training procedure, we fit a CBM model on artificial data generated by a "true CBM" and see whether the training procedure could recover the true parameters. The "true CBM" has $K = 3$ components and uses Logistic Regressions for both $\pi$ and $b$. Each of the $N = 15,000$ data points $\mathbf{x}$ has $D = 7$ features generated independently from Gaussian distributions. We test two different ways of generating the true label subset $\mathbf{y}$: *argmax*, where the most likely $\mathbf{y}$ is always selected; and *sampling*, where the true $\mathbf{y}$ is sampled from the true CBM distribution. For each of these two datasets, we also make a *noise* variant with Gaussian noise added to Logistic Regression scores. Test Subset Accuracy results are presented in Table 1. It shows that if the true distribution is indeed generated by a CBM, then the proposed training procedure is quite effective in recovering the distribution, while other multi-label classifiers are less effective.

*Table 1.* Test Subset Accuracy on artificial data.

| Methods | y=argmax | y=argmax +noise | y=sample | y=sample +noise |
|---|---|---|---|---|
| upper bound | 100 | 76.4 | 65.2 | 48.1 |
| CBM+LR:K=2 | 89.9 | 69.3 | 60.2 | 45.8 |
| CBM+LR:K=3 | **98.4** | **75.8** | **65.0** | **48.0** |
| BinRel+LR | 82.0 | 49.9 | 48 | 40.0 |
| PowSet+LR | 97.5 | 74.1 | 64.6 | 47.3 |
| pairCRF | 96.7 | 73.6 | 64.7 | 45.9 |

## 4. Fast Prediction by Dynamic Programming

Make a prediction for a given $\mathbf{x}$ requires finding the label vector $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$. There are $2^L$ label set candidates, and it is intractable to evaluate the probability for each of them. Many multi-label methods suffer from this intractability (see Section 5). Fortunately the structure in CBM makes it possible to solve this problem efficiently, either with *sampling* or using *dynamic programming*. As a common preprocessing step, for a given $\mathbf{x}$, we can first compute $\pi^k = \pi(z^k = 1|\mathbf{x}, \boldsymbol{\alpha})$ and $\mu^k_\ell = b(y_\ell = 1|\mathbf{x}, \boldsymbol{\beta}^k_\ell)$, for all $k = 1, 2, .., K$ and $\ell = 1, 2, ..., L$.

**Prediction by Sampling.** The CBM density form (6) suggests a natural sampling strategy for a label vector $\mathbf{y}$. We first sample a component $k$ according to the mixture probabilities $\pi^1, ..., \pi^K$. Then from this component, we sample each label $y_\ell$ independently with probability $\mu_\ell^k$. The procedure can be repeated multiple times to generate a set of $\mathbf{y}$ candidates, from which we pick the most probable one. The sampling strategy works best when the most probable label vector has a high probability. Sampling is easy to implement, but does not guarantee that the predicted $\mathbf{y}$ is indeed the globally most probable one.

**Prediction by Dynamic Programming and Pruning.** In order for the overall probability $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ to be high, at least one component $k$ for which $\prod_{\ell=1}^{L} b(y_\ell|\mathbf{x}, \boldsymbol{\beta}_\ell^k)$ must be high. On the other hand, we can show that the one maximizing overall probability does not necessarily maximize any component probability. To find the most probable $\mathbf{y}$, we design a dynamic programing procedure FIND-NEXT-HIGHEST() that enumerates label subsets in a decreasing probability order (per component), and iterate round-robin across components until we are certain that the unchecked subsets will never produce a higher overall probability.

**Algorithm 2** Prediction w/ Dynamic Prog. and Pruning

```
 1: Input: $\pi^k$, $\mu_\ell^k$, $k = 1, 2, ..., K$, $\ell = 1, 2, ..., L$
 2: Initialize candidate component set $\mathcal{S} = \{1, 2, ..., K\}$
 3: Initialize the maximum overall probability $M = -\infty$
 4: for $k = 1, 2, ..., K$ do
 5:     Initialize the maximum component probability $G^k$
 6:     Initialize the priority queue $Q^k$ using parameters $\mu_\ell^k$
 7: while $\mathcal{S} \neq \Phi$ do
 8:     for $k \in \mathcal{S}$ do
 9:         $\mathbf{y} = Q^k$.FIND-NEXT-HIGHEST()
10:         Let $p = \sum_{m=1}^{K} \pi^m \prod_{\ell=1}^{L} \text{Ber}(y_\ell|\mu_\ell^m)$ and $q = \prod_{\ell=1}^{L} \mu_\ell^k$
11:         if $p > M$ then
12:             set $M = p$ and $\mathbf{y}^* = \mathbf{y}$
13:         if $\pi^k q \leq M/K$ or $\pi^k q + \sum_{r \neq k} \pi^r G^r \leq M$ then
14:             Remove $k$ from $\mathcal{S}$
15: Output: $\mathbf{y}^*$

16: function FIND-NEXT-HIGHEST()
17:     $\mathbf{y} = Q^k.deque()$
18:     for $\ell = 1, 2, ..., L$ do
19:         Generate $\mathbf{y}'$ by flipping the $\ell$-th bit of $\mathbf{y}$
20:         if $\mathbf{y}'$ is unseen then
21:             $Q^k.enqueue(\mathbf{y}')$
22:     Output: $\mathbf{y}$
```

For a component $k$, the $\mathbf{y}$ with the highest component probability is the set containing precisely the $y_\ell$ with $\mu_\ell^k \geq 1/2$. To produce a ranked list, we use a max priority queue $Q^k$ to store candidate label vectors and their associated component probabilities; initially, $Q^k$ contains the $\mathbf{y}$ with the highest component probability. The $t$-th call to $Q^k$.FIND-NEXT-HIGHEST() returns the label vector with the $t$-th highest component probability. The overall prediction algorithm (Algorithm 2) iterates over all components, checks the next label candidate (Lines 9-10), updates the best label vector found so far (Lines 11-12), and prunes the remaining label candidates in a component's ranked list (Lines 13-14). The algorithm terminates when all components' ranked lists have been pruned, and surely returns the most probable $\mathbf{y}^*$. In practice, we observe the algorithm rarely visits elements deeper than rank 10 in the component lists.

## 5. Related Work

Modeling label dependencies has been long regarded as a central theme in multi-label classification. Estimating the high-dimensional conditional densities $p(\mathbf{y}|\mathbf{x})$ is very challenging and various approaches have been proposed to tackle this problem based on different approximations.

**Classifier-chains**. According to Chain Rule, the joint probability $p(\mathbf{y}|\mathbf{x})$ can be written as a product of conditionals

$$p(y_1|\mathbf{x})p(y_2|\mathbf{x}, y_1) \cdots p(y_L|\mathbf{x}, y_1, .., y_{L-1}). \quad (11)$$

This reduces a multi-label learning problem to $L$ binary learning problems, each of which learns a new label given all previous labels. During prediction, Classifier-chains (CC) (Read et al., 2011) classify labels greedily in a sequence: label $\ell$ is decided by maximizing $p(\mathbf{y}_\ell|\mathbf{x}, \mathbf{y}_1, .., \mathbf{y}_{\ell-1})$, and becomes a feature to be use in the prediction for label $\ell + 1$. This greedy prediction procedure has three issues: 1) the predicted subset can be quite different from the joint mode (Dembczyński et al., 2011); 2) the overall prediction depends on the chain order; 3) errors in previous label predictions propagate to later label predictions. To address the first issue, Probabilistic Classifier Chains (PCC) replaces the greedy search strategy with some more accurate search strategies, such as exhaustive search (Cheng et al., 2010), $\epsilon$-approximate search (Dembczynski et al., 2012), Beam Search (Kumar et al., 2012; 2013), and A* search (Mena et al., 2015). To address the latter two issues, Ensemble of Classifier Chains (ECC) (Read et al., 2011) averages several predictions made by different chains, wherein the averaging can take place at either the individual label level (ECC-label) or the label subset level (ECC-subset). Using dynamic programming to find the optimal chain orders (Liu & Tsang, 2015) has been proposed recently.

A similar reduction method named **Conditional Dependency Networks** (CDN) estimates $p(\mathbf{y}|\mathbf{x})$ based on full conditionals and Gibbs sampling (Guo & Gu, 2011). During learning, one binary classifier is trained for each full conditional $p(y_\ell|\mathbf{x}, y_1, .., y_{\ell-1}, y_{\ell+1}, ..., y_L)$. During prediction, Gibbs sampling is used to find the mode of the joint. A subtle issue is that the method cannot handle perfectly correlated or anti-correlated labels: consider binary classification as multi-label problem with only 2 exhaustive and exclusive labels. A perfect model for $p(y_1|\mathbf{x}, y_2)$ is $1-y_2$; in other words, the feature information is completely ignored. The same applies to $p(y_2|\mathbf{x}, y_1)$. So the prediction will inevitably fail. In general, Gibbs sampling may fail in the presence of perfect correlations or anti-correlation since the resulting stochastic process is not ergodic; when relations are very strong, Gibbs sampling may still need a long time to converge. (Gasse et al., 2015) factorizes labels into independent factors based on some statistical conditional independence tests and a Markov boundary learning algorithm. The limitation, as pointed out by the authors, is that the statistical tests are ineffective when the ratio between samples and labels is not big enough. Their method beats the Power-Set baseline on synthetic datasets, but not on any real dataset. (Zhang & Zhang, 2010) propose to use a Bayesian Network to encode the conditional dependencies of the labels as well as the feature set, with the features as the common parent of all labels.

**Conditional Random Fields** (CRF) (Sutton & McCallum, 2006) offer a general framework for structured prediction problems based on undirected graphical models. In multi-label classifications labels can form densely connected graphs, so restrictions are imposed in order to make training and inference tractable. For problems involving only hard label relations (exclusive or hierarchical), a special CRF model is proposed (Deng et al., 2014); this works only when label dependencies are *strict* and *a priori* known.

**pair-CRF** limits the scope of potential functions to label pairs, and does not model higher order label interactions (Ghamrawi & McCallum, 2005). The exact inference and prediction in pair-wise CRF requires checking all possible label subsets, which is intractable for datasets with many labels. As shown in their experiment results, the most effective way of doing approximate inference and prediction is to consider only label subsets that occur in the training set. But this eliminates the possibility of predicting unseen subsets. Our CBM model does not have this limitation (see Section 4).

There are also algorithms which exploit label structures but do not estimate $p(\mathbf{y}|\mathbf{x})$ explicitly. Examples include Multi-label k-Nearest Neighbors (Zhang & Zhou, 2007), Compressed Sensing based method (Hsu et al., 2009), Principle Label Space Transformation (Tai & Lin, 2012), Conditional Principal Label Space Transformation (Chen & Lin, 2012) and Compressed Labeling (Zhou et al., 2012).

Various **mixture models** exist for multi-label document classification or supervised topical modeling (McCallum, 1999; Ueda & Saito, 2002; McAuliffe & Blei, 2008; Yang et al., 2009; Ramage et al., 2009; Puurula, 2011; Kim et al., 2012). Our CBM model differs from these models in two aspects. 1) These models are generative in nature, i.e. they model $p(\mathbf{x})$, where $\mathbf{x}$ is typically a bag-of-words representation of a document. By contrast, our CBM is purely discriminative, since it targets $p(\mathbf{y}|\mathbf{x})$ directly. The principled advantage of discriminative approach over generative approach, as stated in (Sutton & McCallum, 2011), is that the former does not make overly simplistic independence assumptions among features, and is thus better suited to including rich, overlapping features. 2) Most of these models are designed specifically for text data, while our method can be applied to any multi-label classification problem. A Conditional Multinomial Mixture model has been proposed for Superset Label Learning (Liu & Dietterich, 2012).

The architecture of CBM also mimics that of Mixture of Experts (ME) (Jacobs et al., 1991; Jordan & Jacobs, 1994), in which a gate model divides the input space into disjoint regions probabilistically and an expert model generates the output in each region. ME has been mainly used in regression and multi-class problems (Yuksel et al., 2012). CBM can be viewed as a multi-label extension of ME with a particular factorization of labels inside each expert.

## 6. Experiments

We perform experiments on five commonly used and relatively large multi-label datasets: SCENE, TMC2007, MEDIAMILL, NUS-WIDE from Mulan[1] and RCV1 (topics subset 1) from LIBSVM[2]. For the sake of reproducibility, we adopt the train/test splits provided by Mulan and LIBSVM. Datasets details are shown at the top of Table 2.

We compare Conditional Bernoulli Mixtures (CBM) with the following methods which estimate $p(\mathbf{y}|\mathbf{x})$: Binary Relevance (BinRel), Power-Set (PowSet), Classifier Chains (CC), Probabilistic Classifier Chains with Beam Search (PCC), Ensemble of Classifier Chains with label voting (ECC-label) and subset voting (ECC-subset), Conditional Dependency Networks (CDN) and pair-wise Conditional Random Fields (pairCRF). CDN is taken from the package MEKA[3]; the rest are based on our implementations.[4] For all methods which require a base learner, we employ Logistic Regression as a common base learner. Additionally, we test Gradient Boosted Trees (GB) as a non-linear base learner in conjunction with BinRel, PowSet and CBM. Both LR and pairCRF are L2 regularized. Hyper parameters tuning are done by cross-validation on the training set (see supplementary material). For methods involving random initializations or sampling, the reported results are av-

[1]`http://mulan.sourceforge.net`

[2]`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`

[3]`http://meka.sourceforge.net`

[4]Our implementations of CBM and several baselines (PowSet, PCC, CRF, etc.) are available at `https://github.com/cheng-li/pyramid`.

*Table 2.* Test Subset Accuracy of different methods on five datasets. All numbers are in percentages. Best performances are bolded.

| | dataset | SCENE | RCV1 | TMC2007 | MEDIAMILL | NUS-WIDE |
|---|---|---|---|---|---|---|
| | domain | image | text | text | video | image |
| | #labels / #label subsets | 6 / 15 | 103 / 799 | 22 / 1341 | 101 / 6555 | 81 / 18K |
| | #features / #datapoints | 294 / 2407 | 47K / 6000 | 49K / 29K | 120 / 44K | 128 / 270K |
| Method | Learner | | | | | |
| BinRel | LR | 51.5 | 40.4 | 25.3 | 9.6 | 24.7 |
| PowSet | LR | 68.1 | **50.2** | 28.2 | 9.0 | 26.6 |
| CC | LR | 62.9 | 48.2 | 26.2 | 10.9 | 26.0 |
| PCC | LR | 64.8 | 48.3 | 26.8 | 10.9 | 26.3 |
| ECC-label | LR | 60.6 | 46.5 | 26.0 | 11.3 | 26.0 |
| ECC-subset | LR | 63.1 | 49.2 | 25.9 | 11.5 | 26.0 |
| CDN | LR | 59.9 | 12.6 | 16.8 | 5.4 | 17.1 |
| pairCRF | linear | 68.8 | 46.4 | 28.1 | 10.3 | 26.4 |
| CBM | LR | 69.7 | 49.9 | **28.7** | 13.5 | **27.3** |
| BinRel | GB | 59.3 | 30.1 | 25.4 | 11.2 | 24.4 |
| PowSet | GB | **70.5** | 38.2 | 23.1 | 10.1 | 23.6 |
| CBM | GB | **70.5** | 43.0 | 27.5 | **14.1** | 26.5 |

eraged over 3 runs.

## 6.1. Results

Subset Accuracy (Test) measure on five datasets are shown in Table 2, grouped by the base learner. On four datasets, the highest Subset Accuracy is achieved by one of the CBM instantiations; and on the other dataset, CBM is close to the best one. This demonstrates the effectiveness of the CBM particular to optimizing Subset Accuracy. On Scene and Mediamill datasets, CBM+GB performs better than CBM+LR, which shows the benefit of being able to incorporate non-linear components. By contrast, the pairCRF is restricted work with linear functions, lacking the flexibility of the CBM model. For completeness, we present in supplementary materials the Jaccard Index and Hamming Loss results; CBM also achieve highest Jaccard Index on three out of five datasets, which is reasonable given the fact that CBM only targets Subset Accuracy and multi-label optimality is measure related.

## 6.2. Impact of Number of Components

Figure 3 shows how increasing the number of components $K$ affects test performance for CBM on TMC dataset. When $K = 1$, CBM and BinRel are mathematically equivalent; the slight difference in performance is due to the implementation details of CBM (see supplementary material). Increasing $K$ from 1 to 15 makes CBM quickly outperform BinRel. Increasing $K$ further gives smaller improvement for CBM and the performance asymptotes as $K$ reaches about 30. Other datasets show similar trends.

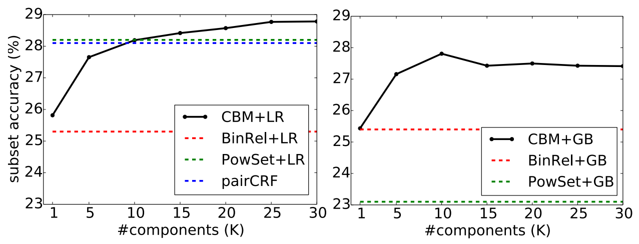

*Figure 3.* Test Subset Accuracy on TMC dataset with varying number of components $K$ for CBM+LR (left) and CBM+GB (right) compared with BinRel, PowSet, and pairCRF.

## 7. Conclusion

In this paper, we propose a new multi-label classification using Conditional Bernoulli Mixtures. It models conditional label dependence; it is purely discriminative; it reduces a multi-label problem to binary and multi-class problems; and generalizes methods already in use like Binary-Relevance. It is easy to train, and we can scale appropriately the number of components. CBM is also capable of predicting label subsets unseen during training. A training Expectation Maximization procedure is presented, together with an efficient prediction procedure based on dynamic programming.

## Acknowledgements

We thank reviewers for their helpful comments. We thank Shivani Agarwal for her suggestions on related work. This work has been generously supported through a grant from the Massachusetts General Hospital Physicians Organization, MGH 507222.

## References

Bishop, Christopher M. *Pattern recognition and machine learning*. Springer, 2006.

Chen, Yao-Nan and Lin, Hsuan-Tien. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*, pp. 1529–1537, 2012.

Cheng, Weiwei, Hüllermeier, Eyke, and Dembczynski, Krzysztof J. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 279–286, 2010.

Dembczyński, Krzysztof, Waegeman, Willem, and Hüllermeier, Eyke. Joint mode estimation in multi-label classification by chaining. In *CoLISD 2011 (Collective Learning and Inference on Structured Data 2011)*, 2011.

Dembczyński, Krzysztof, Waegeman, Willem, Cheng, Weiwei, and Hüllermeier, Eyke. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

Dembczynski, Krzysztof, Waegeman, Willem, and Hüllermeier, Eyke. An analysis of chaining in multi-label classification. In *ECAI*, pp. 294–299, 2012.

Deng, Jia, Ding, Nan, Jia, Yangqing, Frome, Andrea, Murphy, Kevin, Bengio, Samy, Li, Yuan, Neven, Hartmut, and Adam, Hartwig. Large-scale object classification using label relation graphs. In *Computer Vision–ECCV 2014*, pp. 48–64. Springer, 2014.

Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Gasse, Maxime, Aussem, Alexandre, and Elghazel, Haytham. On the optimality of multi-label classification under subset zero-one loss for distributions satisfying the composition property. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2531–2539, 2015.

Ghamrawi, Nadia and McCallum, Andrew. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 195–200. ACM, 2005.

Guo, Yuhong and Gu, Suicheng. Multi-label classification using conditional dependency networks. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, pp. 1300, 2011.

Gupta, Maya R and Chen, Yihua. *Theory and use of the EM algorithm*. Now Publishers Inc, 2011.

Hsu, Daniel, Kakade, Sham, Langford, John, and Zhang, Tong. Multi-label prediction via compressed sensing. In *NIPS*, volume 22, pp. 772–780, 2009.

Jacobs, Robert A, Jordan, Michael I, Nowlan, Steven J, and Hinton, Geoffrey E. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jordan, Michael I and Jacobs, Robert A. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Kim, Dongwoo, Kim, Suin, and Oh, Alice. Dirichlet process with mixed random measures: a nonparametric topic model for labeled data. *arXiv preprint arXiv:1206.4658*, 2012.

Kumar, Abhishek, Vembu, Shankar, Menon, Aditya Krishna, and Elkan, Charles. Learning and inference in probabilistic classifier chains with beam search. In *Machine Learning and Knowledge Discovery in Databases*, pp. 665–680. Springer, 2012.

Kumar, Abhishek, Vembu, Shankar, Menon, Aditya Krishna, and Elkan, Charles. Beam search algorithms for multilabel learning. *Machine learning*, 92(1):65–89, 2013.

Lazarsfeld, Paul Felix, Henry, Neil W, and Anderson, Theodore Wilbur. *Latent structure analysis*. Houghton Mifflin Boston, 1968.

Liu, Liping and Dietterich, Thomas G. A conditional multinomial mixture model for superset label learning. In *Advances in neural information processing systems*, pp. 557–565, 2012.

Liu, Weiwei and Tsang, Ivor. On the optimality of classifier chain for multi-label classification. In *Advances in Neural Information Processing Systems*, pp. 712–720, 2015.

McAuliffe, Jon D and Blei, David M. Supervised topic models. In *Advances in neural information processing systems*, pp. 121–128, 2008.

McCallum, Andrew. Multi-label text classification with a mixture model trained by EM. In *AAAI99 workshop on text learning*, pp. 1–7, 1999.

McLachlan, Geoffrey and Peel, David. *Finite mixture models*. John Wiley & Sons, 2004.

Mena, Deiner, Montañés, Elena, Quevedo, José R, and Del Coz, Juan José. Using A* for inference in probabilistic classifier chains. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 3707–3713. AAAI Press, 2015.

Nocedal, Jorge and Wright, Stephen. *Numerical optimization*. Springer Science & Business Media, 2006.

Puurula, Antti. Mixture models for multi-label text classification. In *10th New Zealand Computer Science Research Student Conference*, 2011.

Ramage, Daniel, Hall, David, Nallapati, Ramesh, and Manning, Christopher D. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pp. 248–256. Association for Computational Linguistics, 2009.

Read, Jesse, Pfahringer, Bernhard, Holmes, Geoff, and Frank, Eibe. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Schapire, Robert E and Singer, Yoram. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2):135–168, 2000.

Sutton, Charles and McCallum, Andrew. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pp. 93–128, 2006.

Sutton, Charles and McCallum, Andrew. An introduction to conditional random fields. *Machine Learning*, 4(4): 267–373, 2011.

Tai, Farbound and Lin, Hsuan-Tien. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

Tsoumakas, Grigorios and Katakis, Ioannis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

Ueda, Naonori and Saito, Kazumi. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pp. 721–728, 2002.

Yang, Shuang-Hong, Zha, Hongyuan, and Hu, Bao-Gang. Dirichlet-bernoulli alignment: A generative model for multi-class multi-label multi-instance corpora. In *Advances in neural information processing systems*, pp. 2143–2150, 2009.

Yuksel, Seniha Esen, Wilson, Joseph N, and Gader, Paul D. Twenty years of mixture of experts. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(8): 1177–1193, 2012.

Zhang, Min-Ling and Zhang, Kun. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 999–1008. ACM, 2010.

Zhang, Min-Ling and Zhou, Zhi-Hua. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

Zhou, Tianyi, Tao, Dacheng, and Wu, Xindong. Compressed labeling on distilled labelsets for multi-label learning. *Machine Learning*, 88(1-2):69–126, 2012.

# Supplementary Material for Conditional Bernoulli Mixtures for Multi-label Classification

## A Several Implementations and Experiments Details

### A.1 Dealing with Empty Predictions

Predicting empty label subsets could be undesirable when we know a priori that each instance matches at least one label. The dynamic programming prediction algorithm in the paper can be easily modified to output the most probably non-empty subsets. In our experiments, we allow CBM to predict empty sets only when the training set contains empty sets. This strategy is shown to improve the test performance slightly. Sometimes, this could make CBM with 1 component perform slightly differently from BinRel (see Fig.3 in the paper).

### A.2 Hyper Parameter Tuning

In our experiments, we do cross-validation on the training set to tune the following hyper parameters

- LR Gaussian prior variance $V_{LR}$, on grids $\{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5, 10^6\}$;
- CRF Gaussian prior variance $V_{CRF}$, on grids $\{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5, 10^6\}$;
- CBM+LR number of components $K$, on grids $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60\}$;
- CBM+GB number of components $K$, on grids $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60\}$.

Tuning results are shown in Table 1.

Table 1: Tuned Hyper Parameters

| dataset | $V_{LR}$ | $V_{CRF}$ | $K$ (CBM+LR) | $K$ (CBM+GB) |
|---|---|---|---|---|
| SCENE | 1.0 | 1.0 | 20 | 25 |
| RCV1 | $10^6$ | 1.0 | 45 | 45 |
| TMC2007 | $10^{-1}$ | $10^{-1}$ | 40 | 20 |
| MEDIAMILL | $10^3$ | 1 | 50 | 5 |
| NUS-WIDE | 1.0 | 1.0 | 50 | 10 |

Our gradient boosting implementation uses regression trees of 5 leaves and shrinkage rate 0.1 as default values. For PCC, the beam search width is set to be 15, as suggested in [3].

## B Results for Jaccard Index and Hamming Loss

Let $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ be a multi-label dataset with ground truth labels, and $\{\hat{\mathbf{y}}_n\}_{n=1}^N$ be the predictions made by a classifier. The Jaccard Index is defined as

$$\frac{1}{N}\sum_{n=1}^{N}\frac{|\mathcal{Y}_n \cap \hat{\mathcal{Y}}_n|}{|\mathcal{Y}_n \cup \hat{\mathcal{Y}}_n|}, \tag{1}$$

where $\mathcal{Y}_n$ and $\hat{\mathcal{Y}}_n$ are the subsets corresponding to $\mathbf{y}_n$ and $\hat{\mathbf{y}}_n$ respectively. Jaccard Index is a well-behaved evaluation measure, often more practical than Subset Accuracy, because it assigns partial credit to "almost correct" answers and handles label imbalance well. For Jaccard Index, the analysis in [2] shows that an optimal classifier with respect to the

EUM utility can be decomposed into binary classifiers $h^*_\ell(\mathbf{x}) = \mathbb{1}[p(y_\ell = 1|\mathbf{x}) > \eta]$, where $\eta$ is a common threshold shared by all labels $\ell \in \mathcal{L}$.

Hamming Loss measures bit-wise errors and is defined as

$$\frac{1}{NL}\sum_{n=1}^{N}\sum_{\ell=1}^{L}\mathbb{1}[y_{n\ell} \neq \hat{y}_{n\ell}]. \tag{2}$$

Hamming Loss is less discriminative than subset accuracy and Jaccard Index because it ignores label imbalance. In practice, each instance usually matches only a few labels out of many candidates. A trivial classifier predicting empty set could also get a decent Hamming Loss. According to [1], the optimal classifier for Hamming Loss simply predicts each label independently based on its marginal probability

$$h^*_\ell(\mathbf{x}) = \arg\max_{y_\ell} p(y_\ell|\mathbf{x}). \tag{3}$$

The theoretical analysis in [1] also shows that a multi-label classifier designed for optimizing one measure may perform poorly under other measures. Test performance based on Jaccard Index and Hamming Loss is shown in Table 2. BinRel achieves the best Hamming Loss on all datasets, as expected. CBM achieve highest Jaccard Index on three out of five datasets, which is reasonable given the fact that CBM only targets Subset Accuracy and multi-label optimality is measure related.

Table 2: The testing performance of different methods on five datasets. All numbers are in percentages. Best performances are bolded.

| | dataset | SCENE (image) | | RCV1 (text) | | TMC2007 (text) | | MEDIAMILL (video) | | NUS-WIDE (image) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #labels / #label subsets | 6 / | 15 | 103 / | 799 | 22 / | 1341 | 101 / | 6555 | 81 / | 18K |
| | #features / #datapoints | 294 / | 2407 | 47K / | 6000 | 49K / | 29K | 120 / | 44K | 128 / | 270K |
| Method | Learner | Jaccard | Hamming | Jaccard | Hamming | Jaccard | Hamming | Jaccard | Hamming | Jaccard | Hamming |
| BinRel | LR | 58.4 | 10.8 | 64.9 | **1.4** | 52.2 | **6.5** | 42.3 | 3.1 | 32.3 | **2.1** |
| PowSet | LR | 71.9 | 9.5 | 67.2 | 1.9 | 51.9 | 6.8 | 35.2 | 3.6 | 32.3 | **2.1** |
| CC | LR | 67.5 | 10.9 | 63.4 | 1.6 | 52.4 | **6.5** | 40.5 | 3.2 | 32.2 | **2.1** |
| PCC | LR | 69.4 | 10.3 | 63.5 | 1.6 | 52.8 | **6.5** | 38.0 | 3.5 | 32.1 | 2.2 |
| ECC-label | LR | 65.6 | 10.1 | 64.8 | **1.4** | 52.1 | **6.5** | 41.1 | 3.1 | 32.3 | **2.1** |
| ECC-subset | LR | 68.0 | 10.3 | 67.4 | **1.4** | 52.2 | **6.5** | 41.1 | 3.2 | 32.3 | **2.1** |
| CDN | LR | 66.4 | 10.9 | 50.9 | 2.8 | 44.0 | 8.5 | 38.6 | 4.2 | 32.8 | 2.7 |
| pairCRF | linear | 72.8 | 9.2 | 64.6 | 1.7 | **53.4** | **6.5** | 35.9 | 3.5 | 32.7 | 2.2 |
| CBM | LR | 73.6 | 8.9 | **69.5** | **1.4** | 53.1 | 6.7 | 41.2 | 3.4 | **34.2** | **2.1** |
| BinRel | GB | 63.9 | **8.3** | 55.8 | 1.7 | 52.2 | 6.7 | **44.2** | **3.0** | 30.6 | **2.1** |
| PowSet | GB | 74.9 | 8.6 | 51.3 | 2.9 | 42.7 | 9.0 | 38.5 | 3.9 | 24.3 | 2.4 |
| CBM | GB | **75.2** | 8.5 | 62.5 | 1.8 | 52.8 | 6.8 | 43.0 | 3.2 | 31.6 | 2.2 |

## References

[1] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

[2] Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent multilabel classification. In *Advances in Neural Information Processing Systems*, pages 3303–3311, 2015.

[3] Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Beam search algorithms for multilabel learning. *Machine learning*, 92(1):65–89, 2013.