Prototypical Networks for Multi-Label Learning

Zhuo Yang$^1$, Yufei Han$^2$, Guoxian Yu$^3$, Xiangliang Zhang$^1$
$^1$King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
$^2$Norton Research Group, Sophia Antipolis, France
$^3$Southwest University, Chongqing, China

Abstract

We propose to address multi-label learning by jointly estimating the distribution of positive and negative instances for all labels. By a shared mapping function, each label’s positive and negative instances are mapped into a new space forming a mixture distribution of two components (positive and negative). Due to the dependency among labels, positive instances are mapped close if they share common labels, while positive and negative embeddings of the same label are pushed away. The distribution is learned in the new space, and thus well presents both the distance between instances in their original feature space and their common membership w.r.t. different categories. By measuring the density function values, new instances mapped to the new space can easily identify their membership to possible multiple categories. We use neural networks for learning the mapping function and use the expectations of the positive and negative embedding as prototypes of the positive and negative components for each label, respectively. Therefore, we name our proposed method PNML (prototypical networks for multi-label learning). Extensive experiments verify that PNML significantly outperforms the state-of-the-arts.

Introduction

Multi-label learning addresses the problem that one instance can be associated with multiple labels simultaneously. Formally, the goal is to learn a function $f$, which maps an instance $x \in \mathbb{R}^D$ to a label vector $y = [l_1, l_2, \ldots, l_K]$ ($l_i$ is 1 if $x$ is associated with the $i$-th label, and $l_i$ is 0 otherwise. Many real-world applications drive the study of this problem, such as image object recognition (Yang et al. 2016; Chen et al. 2019), text classification (Li, Ouyang, and Zhou 2015; Rubin et al. 2012), and bioinformatic problems (Diplaris et al. 2005; Yu et al. 2012).

It’s commonly known that exploiting label dependency is the key in building a multi-label classifier. Let $X \in \mathbb{R}^{N \times D}$ denote all training instances, and $Y \in \{0, 1\}^{N \times K}$ be the corresponding label matrix. The label dependency is mainly studied by 1) exploiting the label matrix $Y$ only (Huang et al. 2018; Haribaran et al. 2010; Feng, An, and He 2019; Huang et al. 2016; Tai and Lin 2012; Zhang and Schneider 2011; Zhang and Schneider 2012); and 2) jointly mapping $X$ and $Y$ into a same low rank label space (Guo 2017; Yu et al. 2014, Lin et al. 2014, Xu et al. 2014). Approaches using $Y$ alone to extract label dependency focus on only categories overlap on which instances (who they are), without the consideration of the instances features (what they are). We are therefore motivated to exploit both $X$ and $Y$ for capturing the label dependency, and break the following limits of existing approaches also working on $X$ and $Y$ at the same time. Mainly, the low-rank assumptions in these approaches cannot hold in most real world applications (Bhatia et al. 2015). Besides, the label dependency captured by them is linear.

We address the multi-label learning from a novel angle of distribution estimation. The intuition is that every label has a mixture distribution of two components, one is positive while the other is negative. This distribution can be estimated in a new embedding space that is a transformation of the original feature space through learning a mapping shared by all labels $f_o : \mathbb{R}^D \rightarrow \mathbb{R}^M$. To match the mixture distribution of two components for each label, its positive instances are mapped to be close, forming a compact cluster (positive component), while the negative instances are mapped to form another cluster (negative component), which is pushed away from the positive cluster. Due to the label dependency, positive instances of one label can also be positive for other labels. Negative instances of different labels also overlap. The mapping function $f_o$ is learned to push positive and negative instances away for each label, and pull positive instances close if they share common labels in the new embedding space. The distribution of the new representations of mapped instances (or called embedding) then well presents both the distance between instances in their original feature space (given in $X$, preserved by $f_o$) and their common membership w.r.t. different categories (given in $Y$, preserved by forming positive/negative components for each label).

Inspired by the prototypical networks for multi-class few-shot learning in (Snell, Swersky, and Zemel 2017), we define in the new space a positive prototype and a negative prototype for each label, which respectively are the expectations of the positive and negative embedding in the mixture distribution. The belongingness of one instance to a positive/negative component can then be simply measured as its distance to the prototype. Since the distributions of (embedding vectors belonging to) labels not only differ on the location of prototypes, and they also have different variance. Therefore, we learn a Mahalanobis distance metric for each
label to complete the classification task.

We highlight the advantages of our proposed approach and our contributions in this work as follows:

1. To the best of our knowledge, this approach is the first attempt to address multi-label problem by estimating distribution for each label as a mixture of positive and negative components.

2. The components are represented by prototypes, which encode the nonlinear label dependency measured from both \( X \) and \( Y \).

3. We conduct extensive experiments including ablation studies on fourteen benchmark data sets to verify the effectiveness of our approach. The results show that our PNML method ranks the first in 63% evaluation cases, and ranks the first or second in 80% cases. CAMEL (Feng, An, and He 2019) is the most recent and most competitive baseline. It ranks first in 28% cases only.

Related Work

There have been many designs of various types of multi-label learning models. We concentrate on the discussion of the most recent and relevant work regarding the ways of exploiting label dependency.

A group of approaches worked on label matrix \( Y \) to extract the label dependency. In (Huang et al. 2016; Huang et al. 2018) and (Cai et al. 2013), a pair-wise label correlation matrix \( C \) is calculated from \( Y \) and get involved into a term of the object function to regularize the classifier weights. Concretely, in (Huang et al. 2016), the label correlation term wants to make the classifier weight vectors of two labels have big inner product if the two labels have high dependency, while in (Cai et al. 2013) the term is used to make the classifier weight vectors have similar sparsity pattern if they are highly dependent. Except for the pair-wise label dependency, CAMEL (Feng, An, and He 2019) learns to represent any given label as a linear combination of all labels from \( Y \), which is used in the final classification. Among these approaches, IJSC (Huang et al. 2018) and CAMEL achieved state-of-the-art performance.

There are also approaches that extract a low-rank matrix from \( Y \) and then treat this low-rank matrix as the new label matrix (Tai and Lin 2012; Zhang and Schneider 2012). Concretely, these approaches firstly project \( Y \) into a low-dimensional space and classification is then done by learning the mapping from \( X \) to this new label matrix. Notably, these two steps are conducted independently in these approaches. In other words, \( X \) was not used for extracting the label dependency.

Inheriting the idea of low-rank label matrix, approaches (Guo 2017; Yu et al. 2014; Lin et al. 2014; Xu et al. 2014) have the intention to incorporate information from \( X \) into label dependency exploiting. They achieve it by optimizing the mapping from feature space to low-rank label space and the mapping from low-rank label space to original label space jointly. Nevertheless, these approaches have a main limitation of the low-rank assumption of label matrix, which is mostly violated in practical cases (Bhatia et al. 2015). Besides, the label dependency is often more complicated than linear. The linear mapping learned in these approaches thus cannot well capture the complex label dependency. From the angle of instance distribution, our approach learns positive and negative prototypes for each label by taking the expectation of instance distribution in the embedding space. The prototypes express the nonlinear label dependency captured from both \( X \) and \( Y \).

Another stream of multi-label models is based on metric learning. The general idea is to learn a distance metric under which the distance relation among the training data can be preserved. For example, MT-LMNN (Parameswaran and Weinberger 2010) treats the classification of each label as an individual task and a distance metric is learned for this label to keep an instance with this label stay closer to its neighbors also with this label. Especially, the distance metric in (Parameswaran and Weinberger 2010) has the form of \( M_0 + M_k \), in which \( M_0 \) is the part shared by all labels and \( M_k \) is the part tuned upon labels. Different from (Parameswaran and Weinberger 2010), LM-kNN (Liu and Tsang 2015) learns a mapping function \( V^T P \) for instance feature vectors and a mapping function \( V^T \) for instance label vectors to embed them into the same space. And in this new space, the Euclidean distance between \( V^T P \) and \( V^T y \) should trend to zero and be smaller than the distance between \( V^T P \) and \( V^T y \). Our approach can be interpreted by also doing metric learning, where the distance metric is an ensemble of the mapping function and the subsequently learned Mahalanobis distance metric. Under this metric, instance with (without) label \( k \) stay closer to \( k \)'s positive (negative) prototypes. Compared to MT-LMNN and LM-kNN, our approach optimizes directly on the distance from instance to label representations (prototypes), which matches the essence of label classification better. Besides, our metrics capture nonlinear pattern and thus are more expressive.

We used prototypes to represent the estimated distribution components, inspired by prototypical networks (Snell, Swersky, and Zemel 2017). The prototypical networks model is designed to solve a multi-class problem with few-shot settings, with one prototype for each class. In our study, to solve a multi-label problem, we have a positive and a negative prototype for each label. The prototypes in the multi-class problem are independent. However, our prototypes in multi-label learning are correlated. Our model should be designed to make the correlation between prototypes be consistent with the correlation between labels. To the best of our knowledge, this is a new challenge. We are the first to adopt prototypes in multi-label learning.

Methodology

Overview of the Proposed Model

Figure 1 shows the overview of our proposed model, which consists of two modules. In the training stage, the embedding module is to learn the non-linear mapping function \( f_\phi \) that forms a mixture of two components for each label in the new embedding space, one positive and one negative. The expected value of embedding in each component is defined as a prototype. The output of this module is thus \( 2 \times K \)
prototypes. The classification module is to conduct distance metric learning for every label, based on the prototypes from the embedding module. Once the whole model is trained, a query vector is compared with every label’s two prototypes by going through the learned distance metric layer. For each label, softmax is used to determine if \( x_i \) belongs to this class or not. Next, we elaborate the two modules and the overall objective function for training.

### Embedding Module

The goal of this module is to learn \( f_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M \), which maps label-wise training instances to two compact clusters such that the distribution of each label in the new embedding space can be modeled as a mixture of two components (one positive and one negative). In Figure 1, \( f_\phi \) is the Embedding Layer network, e.g., a one-layer fully connected neural network with LeakyRelu (Maas, Hannun, and Ng 2013) as activation function.

Let \( e = f_\phi(x) \) be the embedding vector of \( x \) in the new space. In this space, embedding vectors belonging to each label \( k \) follow a two-component mixture distribution:

\[
p(e|\Omega) = \sum_{s \in \{k+, k-\}} \pi_s p_s(e|\theta_s)
\]

where \( p_s(e|\theta_s) \) is the distribution of one component with parameters \( \theta_s \) and cumulant function \( \psi_s \) and \( \pi \) is the mixing coefficient. Parameter set \( \Omega = \{\theta_{k+}, \theta_{k-}, \pi_{k+}, \pi_{k-}\} \) includes the mixing coefficients for both positive and negative component, as well as \( \theta \) for positive and negative component.

Each component can be described by a regular exponential family distribution \( p_s(e|\theta_s) \):

\[
p_s(e|\theta_s) = \exp(e^T \theta_s - \psi_s(\theta_s) - g_s(e))
\]

where \( g_s(e) \) is the carrier measure. Then, \( p(e|\Omega) \) becomes

\[
p(e|\Omega) = \sum_{s \in \{k+, k-\}} \pi_s \exp(e^T \theta_s - \psi_s(\theta_s) - g_s(e))
\]

(3)

When \( \Omega \) is known, like in the Gaussian mixture clustering model, the probability of \( e \) belonging to component \( k+ \) is:

\[
p(\hat{y} = k+|e) = \frac{\pi_{k+} \exp(e^T \theta_{k+} - \psi(\theta_{k+}) - g_{k+}(e))}{\sum_{s \in \{k+, k-\}} \pi_s \exp(e^T \theta_s - \psi(\theta_s) - g_s(e))}
\]

(4)

According to the Theorem 4 in (Banerjee et al. 2005), there is a unique Bregman divergence associated with every member of the exponential family. For example, spherical Gaussian distribution is associated with squared Euclidean distance, and multinomial distribution is associated with Kullback-Leibler divergence. Therefore, we can rewrite the regular exponential family distribution given in Eq.(2) by a regular Bregman divergence (Snell, Swersky, and Zemel 2017) as:

\[
p_s(e|\theta_s) = \exp(e^T \theta_s - \psi(\theta_s) - g_s(e)) = \exp(-d_\varphi(e, \mu(\theta_s)) - g_s(e))
\]

(5)

where \( \varphi \) and \( \psi \) are conjugate Legendre functions, and \( d_\varphi \) is the Bregman divergence associated with \( \varphi \). And \( \mu(\theta) \) is the expectation:

\[
\mu(\theta) = E_{p_\psi}[e] = \int_{\mathbb{R}^M} e \exp(e|\theta) de
\]

(6)

For our case of \( e \in \mathbb{R}^M \), \( \mu(\theta) \) is the mean of \( e \). Since we have two components for each label, the \( \mu(\theta_{k+}) \) of the positive component will be the mean of \( E_{\text{pos},k} \) which denotes the embedded instance set of positive component. The \( \mu(\theta_{k-}) \)
will then be the mean of $E_{neg, k}$ which denotes the embedded instance set of negative component. We define the mean as prototype of a component. Formally,

$$P_{pos, k} = \mu(\theta_{k+}) = \frac{1}{|X_{pos, k}|} \sum_{x_i \in X_{pos, k}} f_\phi(x_i) \tag{7}$$

$$P_{neg, k} = \mu(\theta_{k-}) = \frac{1}{|X_{neg, k}|} \sum_{x_i \in X_{neg, k}} f_\phi(x_i) \tag{8}$$

where $X_{pos, k}$ ($X_{neg, k}$) is the set of positive (negative) instance for label $k$. With the new defined prototypes and the rewritten with Bregman divergence, the label prediction in Eq. (4) can be updated as Eq. (9) by treating positive component and negative component equally-weighted.

$$p(y_{i, k} = 1|x_i) = \frac{\exp(-d_\phi(f_\phi(x_i), P_{pos, k}))}{\sum_{S \in \{pos, neg\}} \exp(-d_\phi(f_\phi(x_i), P_S))} \tag{9}$$

Then our first part of the loss function is defined by cross-entropy, accounting for minimizing the prediction error,

$$L_c = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i, k} \log p(y_{i, k} = 1|x_i)$$

$$+ (1 - y_{i, k}) \log (1 - p(y_{i, k} = 1|x_i)) \tag{10}$$

We move next to the second module of distance metric learning for classification.

**Label-Wise Distance Metric Learning**

In Banerjee et al. 2005, various distance functions $d_\phi(\cdot, \cdot)$ are presented for popular distribution functions in exponential families. For example, multivariate spherical Gaussian distribution $p_\theta(e|$ is associated with squared Euclidean distance $\|e - \mu_\theta\|^2$. Although Euclidean distance is simple and showed its effectiveness in Snell, Swersky, and Zemel 2017, it is not appropriate to our multi-label learning, because components of our label distribution may not be spherical Gaussian distribution.

The non-spherical distribution can be firstly caused by the instances’ common membership among different categories. One component may be pulled to get close to several different components due to their common instances. Secondly, for multi-label learning, features may play different roles in different labels’ discriminant processes Zhang 2011, Huang et al. 2018, which implies that each label has its own specific distribution pattern.

These facts encourage us to learn distance metric $d_\phi(\cdot, \cdot)$ specifically for each label, rather than using Euclidean distance for all labels. As a popular choice, Mahalanobis distance metric has been widely used in metric learning Weinberger and Saul 2009, Parameswaran and Weinberger 2010.

Eq. (11) is the general form of Mahalanobis distance metric with matrix $U_k$ for label $k$.

$$d^k_{m}(e, \mu) = \sqrt{(e - \mu)^T U_k^T U_k (e - \mu)} \tag{11}$$

Eq. (11) can also be viewed that the feature vectors are firstly linearly transformed by $U_k$ and then the Euclidean distance is calculated on the transformed vectors. In this paper, we keep matrix $U_k$ as a square matrix with size $M \times M$ and expressed by a one-layer fully connected neural network with linear activation function, as shown in Distance Layer in Figure 1. Eq. (11) will use $d^k_{m}(\cdot, \cdot)$ for distance calculation, which consequently plays also in the loss $L_c$ given in Eq. (10). Moreover, we add regularizer $L_m$ to the overall loss function for preventing over-fitting.

$$L_m = \frac{1}{K} \sum_{k=1}^{K} \|U_k\|^2_F \tag{12}$$

**Label Correlation Regularizer**

In our approach, the prototypes can be viewed as labels’ representatives, especially the positive prototypes. In practice, the negative instances for a label are usually much more than the corresponding positive instances. Then for many labels, their negative instances are mostly in common. Consequently the negative prototypes of labels may be similar, and less interesting than positive prototypes.

We thus take the positive prototypes and enhance the correlation between their embedding. The motivation is, if label $j$ and label $k$ are strongly correlated, their representative positive prototypes should have large inner product, otherwise, the inner product will be small. Then we have another regularizer term to add to the loss function. It is defined as:

$$L_c = \frac{1}{2} \sum_{j=1}^{K} \sum_{k=1}^{K} (1 - c_{jk}) P_{pos, j}^T P_{pos, k} \tag{13}$$

where $c_{jk}$ indicates the correlation between label $j$ and $k$, and is calculated by cosine similarity between $j$-th column and $k$-th column of $Y$.

The overall loss function is

$$L_{loss} = L_c + \lambda_1 L_m + \lambda_2 L_{\text{ce}} \tag{14}$$

where $\lambda_1$ and $\lambda_2$ are the tradeoff parameters with non-negative values.

**Training procedure**

In the training process of our approach, for each label, we need to load the feature matrix $X$ for mapping and building prototypes. The computational cost significantly increases when data sets get larger. To mitigate this issue, we can sample instances for each label to reduce the amount of data involved for prototype computing. We denote $r_{pos, k}$ and $r_{neg, k}$ as the sampling rate for positive and negative instances, respectively. The influence of the rate on model performance will be evaluated in next section.

The training procedure will be first sampling positive and negative instances. Then update the network weights for embedding $f_\theta$ and for distance metric $d^k_{m}(\cdot, \cdot)$. Adam (Kingma and Ba 2015) is used as the optimizer.

**Experiments**

In this section, we organize the comparative study by including the state-of-the-art multi-label learning approaches evaluated on various benchmark data sets. Besides, we also conduct ablation study to further demonstrate the effectiveness of the proposed approach.
Experimental Setup

Datasets and Evaluation Metrics. Fourteen public benchmark datasets are used to evaluate all the involved approaches comprehensively. These datasets have different application contexts, including text processing, biology, music and image. They are widely used for benchmarking algorithms of multi-label learning (Feng, An, and He 2019, Huang et al. 2018, Zhang 2011). Table 1 summarizes the details of these data sets. We choose 5 popularly applied evaluation metrics to measure the performances (Sorower 2010). They are Accuracy, Macro-averaging $F_1$, Micro-averaging $F_1$, Average precision and Ranking Loss.

Comparing Approaches. The proposed PNML is compared to four state-of-the-art multi-label learning approaches, including BR (R. Boutell et al. 2004), ML-KNN (Zhang and Zhou 2007), JFSC (Huang et al. 2018) and CAMEL (Feng, An, and He 2019). BR is a first-order approach. ML-KNN derives from traditional KNN method and adapted to multi-label problem. CAMEL and JFSC are the most recent state-of-the-art approaches. Therefore, we select them for comparison, without comparing to other baselines that are inferior to CAMEL and JFSC.

BR and ML-KNN are implemented under the sk-mllearn package (Szymański and Kajdanowicz 2017) and a two-layer multi-layer perceptron (MLP) is used as the base classifier for BR. The unit number of hidden layer of MLP is determined by Eq.(15) in which $D$ is the original feature dimension. Besides, the number of nearest neighbors of ML-KNN is searched in \{3, 5, $\cdots$, 21\}. Codes and suggested parameters in the original papers are used for JFSC and CAMEL.

In our approach, the embedding dimension $M$, slope of LeakyRelu $\alpha$ and loss tradeoff parameters $\lambda_1$, $\lambda_2$ are hyperparameters to be determined. Empirically, $\alpha$ is set to 0.2 and $M$ is determined by Eq.(15). As for $\lambda_1$ and $\lambda_2$, they are searched in $\{10^{-7}, 5 \times 10^{-6}, 10^{-6}, 5 \times 10^{-6}, \cdots, 10^{-2}\}$. We tune the architecture of our proposed approach to verify the effectiveness of different parts of the algorithmic design.

Ablation Study. We tune the architecture of our proposed approach to verify the effectiveness of different parts of the algorithmic design.

- To confirm that our approach can learn well the label dependency and perform an effective mapping $f_\phi$, we assign different independently trained embedding layers to each label. Therefore $K$ single-label classifications are conducted independently without interaction. We name this variant of the proposed approach as PNML-I.

- To demonstrate the effectiveness of the distance metric learning component, we build another variant of our approach in which Euclidean distance is used for each label and we name it PNML-D.

Results

For each comparing approach, 5-fold cross-validation is performed on the training data of each data set. Tables 2 and 3 report the average results (mean±std) of each comparing algorithm over 14 data sets in terms of each evaluation metric. The best performance among all the approaches is shown in boldface. Based on these experimental results, the following observations can be made.

- Our approach PNML outperforms baseline approaches in most cases. Concretely, if we treat one evaluation metric for one dataset as one case, there are 70 cases in total and PNML ranks the first in 63% (44/70) evaluation cases, and ranks the first or second in 80% cases. CAMEL (Feng, An, and He 2019) is the most recent and most competitive baseline. It ranks only first in 28% cases. This observation verifies the effectiveness of our approach.

- PNML achieves statistically superior performance against PNML-I and PNML-D in terms of nearly each evaluation metric, which indicates that the learned embedding function $f_\phi$ does transfer meaningful knowledge across labels.
Table 2: Experimental results of evaluated algorithms on the first seven data sets on various evaluation metrics. ↑ (↓) indicates the larger (smaller) the value, the better the performance. The best and second best results are in bold and with *, respectively.

| Algorithm | Accuracy ↑ | Macro-averaging $F_1$ | Average precision ↑ | Ranking loss ↓ |
|-----------|------------|------------------------|---------------------|----------------|
|           | emotions   | scene                  | image               | arts           | science       | education     | enron         |
| BR        | 0.478±0.030 | 0.672±0.031            | 0.530±0.001         | 0.341±0.010    | 0.350±0.008   | 0.365±0.007   | 0.419±0.008   |
| ML-KNN    | 0.418±0.007 | 0.692±0.034            | 0.522±0.026         | 0.189±0.040    | 0.280±0.010   | 0.324±0.014   | 0.273±0.074   |
| JFSC      | 0.255±0.019 | 0.603±0.017            | 0.451±0.026         | 0.377±0.008*   | 0.365±0.008*  | 0.346±0.012   | 0.418±0.011   |
| CAMEL     | 0.505±0.030 | 0.702±0.018*           | 0.589±0.015         | 0.342±0.012    | 0.331±0.017   | 0.360±0.010   | 0.440±0.016*  |
| PNML-I    | 0.489±0.022*| 0.691±0.029            | 0.562±0.006         | 0.363±0.004    | 0.347±0.009   | 0.379±0.011   | 0.420±0.006   |
| PNML-D    | 0.405±0.021 | 0.661±0.015            | 0.543±0.015         | 0.358±0.005    | 0.365±0.012*  | 0.386±0.011*  | 0.431±0.006   |
| PNML      | 0.489±0.008*| 0.708±0.025            | 0.583±0.005*        | 0.381±0.013    | 0.394±0.006   | 0.394±0.013   | 0.472±0.018   |

and the learned label-specific distance metric achieves better distance measurement than Euclidean.

• In terms of evaluation metric, PNML performs best on Macro-averaging $F_1$, which indicates PNML is more friendly to rarely encountered labels comparing to other approaches. This observation matches the results of (Snell, Swersky, and Zemel 2017), in which prototypical networks are used to solve few-shot learning problem.

• In terms of data type, PNML performs best on text dataset, which is also the place where PNML is obviously superior to PNML-I. This observation implies that the labels in text dataset are more correlated than in other tasks. And the way PNML employs to leverage information across labels matches the relationship structure of these text labels.

**Influence of Instance Sampling Rate.** In our approach, to reduce the computational cost, we sample instances at rate $r_{pos,k}$ and $r_{neg,k}$ in the training process. Here, we show the influence of sampling rates on model predictive performance and efficiency. We choose dataset education for the experimental study and record the change of Macro-averaging $F_1$, Micro-averaging $F_1$ and run-time of five folds under different sampling rates in Figure 2. It can be observed that the performance keeps nearly the same under different sampling rates, while run-time drops down with the decreasing of sampling rate. Therefore, a small sampling rate is acceptable.
Table 3: Experimental results of evaluated algorithms on the last seven data sets on various evaluation metrics. ↑ (↓) indicates the larger (smaller) the value, the better the performance. The best and second best results are in bold and with *, respectively.

| Algorithm | yeast | genbase | medical | rcv1-s1 | rcv1-s3 | rcv1-s5 | bibtex |
|-----------|-------|---------|---------|---------|---------|---------|--------|
| BR        | 0.487 ± 0.008 | 0.988 ± 0.003 | 0.701 ± 0.039 | 0.324 ± 0.005 | 0.378 ± 0.011 | 0.381 ± 0.006 | 0.321 ± 0.005 |
| ML-KNN    | 0.519 ± 0.006* | 0.973 ± 0.006 | 0.612 ± 0.034 | 0.338 ± 0.005 | 0.342 ± 0.011 | 0.349 ± 0.006 | 0.313 ± 0.004 |
| JFSC      | 0.503 ± 0.012 | 0.994 ± 0.005 | **0.755 ± 0.017** | 0.354 ± 0.004 | 0.391 ± 0.005* | 0.397 ± 0.006* | 0.352 ± 0.000* |
| CAMEL     | **0.524 ± 0.014** | 0.992 ± 0.005* | 0.739 ± 0.021* | 0.300 ± 0.006 | 0.354 ± 0.012 | 0.358 ± 0.011 | 0.312 ± 0.006 |
| PNML-I    | 0.477 ± 0.015 | 0.986 ± 0.005 | 0.685 ± 0.015 | 0.352 ± 0.005 | 0.383 ± 0.006 | 0.394 ± 0.007 | 0.352 ± 0.004* |
| PNML-D    | 0.420 ± 0.012 | 0.972 ± 0.002 | 0.705 ± 0.020 | 0.367 ± 0.002* | 0.384 ± 0.003 | 0.389 ± 0.005 | 0.305 ± 0.002 |
| PNML      | 0.451 ± 0.012 | 0.987 ± 0.006 | 0.731 ± 0.018 | **0.395 ± 0.005** | **0.422 ± 0.006** | **0.428 ± 0.005** | **0.387 ± 0.004** |

Conclusion and Future Work

In this paper, we propose PNML to cope with challenges in multi-label learning. PNML addresses multi-label learning by density estimation in a new embedding space. While assuming each label has a mixture distribution of two components, a mapping function is learned to map label-wise training instances to two compact clusters, one for each component. Then a positive and negative prototype is defined for each label by taking the mean of embedding in positive and negative component, respectively. After that, label-wise classification is performed based on the distance from instances to positive and negative prototypes, measured by label-specific distance metric learned by neural networks.

Extensive experiments including ablation study clearly verify the effectiveness of our approach. Our approach can be extended to weak label problem and positive unlabeled problem in the future.

References

[Banerjee et al. 2005] Banerjee, A.; Merugu, S.; Dhillon, I. S.; and Ghosh, J. 2005. Clustering with bregman divergences. Journal of machine learning research 6:17051749.

[Bhatia et al. 2015] Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse local embeddings for extreme multi-label classification. In Proceedings of the 28th Inter-
knn: A lazy learning approach to multi-label learning. *Pattern Recognition* 40:2038–2048.

[Zhang 2011] Zhang, M. 2011. Lift: Multi-label learning with label-specific features. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 1609–1614.