Binary Linear Compression for Multi-label Classification*

Wen-Ji Zhou♣, Yang Yu♣†, Min-Ling Zhang♠
♣National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
♣Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
♠School of Computer Science and Engineering, Southeast University, Nanjing, China
♠Key Laboratory of Computer Network and Information Integration (SEU), Ministry of Education, China
†Corresponding email: yuy@nju.edu.cn

Abstract

In multi-label classification tasks, labels are commonly related with each other. It has been well recognized that utilizing label relationship is essential to multi-label learning. One way to utilizing label relationship is to map labels to a lower-dimensional space of uncorrelated labels, where the relationship could be encoded in the mapping. Previous linear mapping methods commonly result in regression subproblems in the lower-dimensional label space. In this paper, we disclose that mappings to a low-dimensional multi-label regression problem can be worse than mapping to a classification problem, since regression requires more complex model than classification. We then propose the binary linear compression (BILC) method that results in a binary label space, leading to classification subproblems. Experiments on several multi-label datasets show that, employing classification in the embedded space results in much simpler models than regression, leading to smaller structure risk. The proposed methods are also shown to be superior to some state-of-the-art approaches.

1 Introduction

The goal of multi-label classification is to learn a classifier that can predict a label vector for a given data point. Multi-label classification has been widely used in real-world applications, such as image/video annotation [Carneiro *et al.*, 2007] and query/keyword suggestions [Agrawal *et al.*, 2013]. While early multi-label classification methods, such as Binary Relevance (BR) [Boutell *et al.*, 2004], do not take label relationship into account, it has been well recognized that utilizing label relationship is essential in multi-label classification. Many studies have been done following this direction.

One way to utilize the relationship, is to map the original labels to a lower-dimensional embedding space, such that the relationship could be encoded in the mapping. There are already some approaches using mapping to reduce the dimension and correlation of labels, called embedding-based approaches, such as Compressed Sensing [Hsu *et al.*, 2009], PLST [Tai and Lin, 2012], landmarks labels [Balasubramanian and Lebanon, 2012], output codes [Zhang and Schneider, 2011], SLEEC [Bhatia *et al.*, 2015], etc. The common procedure of these approaches is that, firstly, they map the label vectors to a low-dimensional label space, then learn to predict the low-dimensional labels, and finally map the predicted low-dimensional labels back to the original label space. These approaches commonly employ a matrix as the mapping function from the original label space to the lower-dimensional space, which naturally results in a real-domain lower-dimensional space and thus the learning task is a regression problem in the new label space.

We notice that transforming a classification problem to a regression problem could increase the complexity of the learning task, since regression loss is much more sensitive than classification loss. Therefore, we hypothesize that it will be easier to learn if the mapping can lead to classification problems instead of regression.

In this paper, we validate the hypothesis by developing a binary linear compression method for multi-label classification, named BILC. BILC also employs a matrix as the mapping, however, the mapped labels in the lower-dimensional space are rounded to be binary, such that the learning in the new label space is classification. In this way, a random matrix (employed in compressed sensing) does not lead to a good performance. Therefore, we try to learn a good embedding matrix, so that it leads to a good reconstruction of the labels. Due to the nonlinear rounding process, optimizing the matrix is a non-convex and non-differentiable problem. Instead of transforming the problem to be a convex but unfaithful objective, we employ state-of-the-art derivative-free method [Yu *et al.*, 2016] to handle the optimization difficulty, which helps to validate our hypothesis directly. Experiments on several multi-label datasets show that BILC can have superior performance to some state-of-the-art approaches.

The rest of this paper is organized as follows. Section 2 introduces the background. Section 3 presents the proposed BILC method. Section 4 presents experiments. Section 5 concludes the paper.

*This research was supported by the NSFC (61573104, 61673202), Jiangsu Science Foundation (BK20160066), and Foundation for the Author of National Excellent Doctoral Dissertation of China (201451).

2 Background

2.1 Multi-Label Learning

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$ be the input feature vector, and $\boldsymbol{y}_i \in \{+1, -1\}^L$ be L-dimension label vector. The feature matrix is denoted as $X = [\boldsymbol{x}_1; \boldsymbol{x}_2; ...; \boldsymbol{x}_n] \in \mathbb{R}^{n \times d}$ and the label matrix is denoted as $Y = [\boldsymbol{y}_1; \boldsymbol{y}_2; ...; \boldsymbol{y}_n] \in \{+1, -1\}^{n \times L}$. Given $\mathcal{D}$, the goal is to learn a multi-label classifier $f : \mathbb{R}^d \to \{+1, -1\}^L$ that can predict the label vector for a given instance.

Traditional single-label classification problems aim at learning a classifier that can tag an instance with a single label from a label set. In multi-label classification problem, an instance is associated with multiple labels. Multi-label learning has been studied for decades, and many algorithms have been proposed. Most of them can be grouped into two types, problem transformation methods and algorithm adaptation methods [Tsoumakas and Katakis, 2007].

Problem transformation methods usually transform multi-label problems into other well studied problems, such as Binary Relevance [Boutell *et al.*, 2004], Calibrated Label Ranking [Fürnkranz *et al.*, 2008], Random k-labelsets [Tsoumakas and Vlahavas, 2007], etc. The key idea of problem transformation methods is to reduce the multi-label learning problem to another problem that is easy to deal with. For example, Binary Relevance method transforms a multi-label learning task into a group of binary classification tasks, and Random k-labelsets method transforms a multi-label learning task into an ensemble of multi-class classification tasks.

Algorithm adaptation methods, on the contrary, modify existing learning techniques to enable them to fit the multi-label classification problem, such as ML-kNN [Zhang and Zhou, 2007], ML-DT [Li *et al.*, 2010], BP-MLL [Zhang and Zhou, 2006], etc. For example, ML-kNN method adapts lazy learning technique k-NN on multi-label classification. ML-DT adapts decision tree technique to solve multi-label classification tasks. BP-MLL adapts the popular back-propagation algorithm for multi-label learning.

2.2 Embedding-Based Method

Embedding-based methods belong to a group of problem transformation methods, which focus on using label relationship to reduce the effective number of labels. They commonly map the label vectors to a new lower-dimension label space, and learning (often by Binary Relevance) in the new label space. The predictions in the new label space are finally mapped back to the original label space.

Several embedding-based methods that have been proposed. For example, Compressed Sensing [Hsu *et al.*, 2009], PLST [Tai and Lin, 2012], landmarks labels [Balasubramanian and Lebanon, 2012], output codes [Zhang and Schneider, 2011] and state-of-the-art SLEEC [Bhatia *et al.*, 2015].

These approaches often project the label vectors onto an $\hat{L}$-dimensional linear subspace as $\boldsymbol{z} = U\boldsymbol{y}$, which $\hat{L} < L$ and $\boldsymbol{z} \in \mathbb{R}^{\hat{L}}$. For this compression process, U is called as the compression matrix. After compression, they train a set of regressors to predict $\boldsymbol{z}$ from $\boldsymbol{x}$. Finally, they find a reconstruction matrix $\hat{U}$ to map embedded label vectors to the origin label space by $\boldsymbol{y} = \hat{U}V(\boldsymbol{x})$, which is called reconstruction or decompression.

Different embedding-based approaches mainly differ in the methods of compression and reconstruction. For example, Compressed Sensing (CS) uses a random matrix as compression matrix U. It uses greedy methods to reconstruct, such as Orthogonal Matching Pursuit [Pati *et al.*, 1993]. PLST gets compression matrix U and reconstruction matrix $\hat{U}$ by singular value decomposition. SLEEC aims at learning embeddings that preserving local distance, which can promote the accuracy of tail labels prediction, by optimizing the objective function:

$$\min_{V \in \mathbb{R}^{\hat{L} * d}} ||P_\Omega(Y^T Y) - P_\Omega(X^T V^T V X)||_F^2 + \lambda||V||_F^2 + \mu||VX||_1 \quad (1)$$

where V is a set of regressors and the index set Ω denotes the set of neighbors that we wish to preserve, i.e., $(i, j) \in \Omega$ iff $j \in \mathcal{N}_i$. $\mathcal{N}_i$ denotes a set of nearest neighbors of i. And P_Ω is defined as:

$$(P_\Omega(Y^T Y))_{ij} = \begin{cases} \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle, & if(i,j) \in \Omega \\ 0, & otherwise \end{cases}$$

SLEEC employs a set of regressors to predict the embedded labels, and reconstructs the labels in the original label space. Note that a reconstructed label is not binary, the nearest binary label is searched and used as the final prediction.

These embedding-based approaches naturally result in regression problems. All of them have to learn regressors in the embedded label space. In this paper, we argue that regression requires higher model complexity than classification, and thus classification models are more appealing.

There are a few previous studies that employed binary matrix decomposition to result in binary embedded space, e.g. [Wicker *et al.*, 2012]. Since these methods operate on binary relationship operators directly, the optimal decomposition is hard to approximate. Greedy algorithm, although commonly employed, is hard to perform well on large data.

2.3 Derivative-Free Optimization

Previously optimization problems in learning tasks are mostly solved by gradient-based methods. However, the optimization may not always simple enough to fit the gradient-base methods. Often, a complex optimization has to be relaxed to a convex problem, sacrificing the faithfulness to the original problem.

Recently, the derivative-free optimization methods have made significant progress in both theoretical foundation and practical usage. A derivative-free optimization method considers the general optimization problem $\arg\max_{x \in X} f(x)$, where X is the domain and f can be quite complex. The methods, instead of calculating gradients of f, samples solutions x and learns from their feedbacks $f(x)$ for finding better solutions. Therefore, derivative-free optimization methods can be more suitable for problem with bad mathematical properties, including non-convexity, non-differentiability, and having many local optima.

Ancient derivative-free optimization methods includes representatives such as genetic algorithms [Goldberg, 1989], which are mostly consist of rule-of-thumbs heuristics. Recently, approaches with strong theoretical supports have emerged, including Bayesian optimization methods [Brochu *et al.*, 2010], optimistic optimization methods [Munos, 2014], and model-based optimization [Yu *et al.*, 2016]. These approaches have been shown successful in various applications including hyper parameter tuning, non-convex objective learning, etc. We thus employ state-of-the-art derivative-free optimization methods to solve our problem directly.

3 Binary Linear Compression

As mentioned above, most embedding-based methods have to use regression to train and predict on low-dimensional label space. Regression model has a larger model size and is more complex than classification, which is proved by some experiments in the experiment section later. So if we can use classification instead of regression, we may have a better generalized performance and a smaller model size.

We now present our method, BILC, an embedding-based method which learns a binary embedding and utilizes high-order relationship between labels. For a given dataset, we learn a compression matrix $M \in \mathbb{R}^{\hat{L}\times L}$ and a reconstruction matrix $\hat{M} \in \mathbb{R}^{L\times\hat{L}}$. We use compression matrix M to map a label vector $\boldsymbol{y}$ to $\hat{L}$-dimension binary vector $\boldsymbol{z}$,

$$\boldsymbol{z} = [M\boldsymbol{y}], \boldsymbol{z} \in \{+1, -1\}^{\hat{L}}, \hat{L} < L \tag{2}$$

where $[\cdot]$ denotes the `sign` operator that rounds the given value as $+1$ or -1 (for zeros its randomly outputs $+1$ or -1).

After obtaining $\boldsymbol{z}$, we learn a set of $\hat{L}$ number of base classifiers $\boldsymbol{C} = \{C_1, \ldots, C_{\hat{L}}\}$ with respect to the new label $\boldsymbol{z}$, i.e., to minimize some multi-label loss between $\boldsymbol{z}$ and $(C_1(\boldsymbol{x}), \ldots, C_{\hat{L}}(\boldsymbol{x}))$ over all instances $\boldsymbol{x}$.

During the test phase, for an unseen instance $\tilde{\boldsymbol{x}}$, we first use base classifiers $\boldsymbol{C}$ to predict the low-dimensional label vector $\tilde{\boldsymbol{z}} = \boldsymbol{C}(\tilde{\boldsymbol{x}})$, then use the reconstruction matrix $\hat{M}$ to map the predicted label vector to high-dimensional label space,

$$\tilde{\boldsymbol{y}} = [\hat{M}\tilde{\boldsymbol{z}}]. \tag{3}$$

During the training and testing process above, the nonlinear `sign` operator $[\cdot]$ is applied, which makes the system sophisticated and thus an arbitrary matrix M and $\hat{M}$ may not perform well. Therefore, it is crucial to learn a good embedding and reconstruction matrices.

3.1 Learning Embeddings

We wish to find a compression matrix M and a reconstruction matrix $\hat{M}$ with the criterion that the labels are kept as accurate as possible after the compression and decompression. This criterion is implemented in the following objective function,

$$\arg\min_{M,\hat{M}} Loss([\hat{M}[MY]], Y) \tag{4}$$

where $Loss$ is some multi-label classification loss function in the original label space. Frequently used multi-label classification functions include the Hamming loss function and the top-k average precision loss functions.

Algorithm 1 BILC

```
Input:
    Training Data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$
    A derivative-free optimization method $Opt$
    Multi-label classification loss $Loss$
    Embedding dimension $\hat{L}$
Procedure: Learning
 1: $M \leftarrow Opt.\mathrm{argmin}_M \left[\mathrm{Evaluation}(M; Loss, \{\boldsymbol{y}_i\})\right]$
 2: $z_i = [M\boldsymbol{y}_i]\ , \forall i$
 3: train base classifiers $\boldsymbol{C}$ over $\{(\boldsymbol{x}_i, \boldsymbol{z}_i)\}_{i=1}^n$ by Binary Relevant method
 4: return $M, \hat{M} = M^\top(MM^\top)^{-1}, \boldsymbol{C}$
Procedure: Evaluation($M; Loss, \{\boldsymbol{y}_i\}$)
 1: $\boldsymbol{z}_i = [M\boldsymbol{y}_i]\ , \forall i$
 2: $\hat{\boldsymbol{y}}_i = [M^\top(MM^\top)^{-1}\boldsymbol{z}_i]\ , \forall i$
 3: $\epsilon = Loss(\hat{\boldsymbol{y}}, \boldsymbol{y})$
 4: return $\epsilon$
Procedure: Prediction($\boldsymbol{x}$)
 1: for $l = 1$ to $\hat{L}$ do
 2:     $\tilde{\boldsymbol{z}}(l) = C_l(\boldsymbol{x})$
 3: end for
 4: $\tilde{\boldsymbol{y}} = [\hat{M} * \boldsymbol{z}]$
 5: return $\tilde{\boldsymbol{y}}$
```

The Hamming loss function evaluates the fraction of misclassified instance-label pairs,

$$Loss_{Hamming} = \frac{1}{n \cdot L}\sum_{i=1}^{n}\sum_{l=1}^{L}\mathbb{I}(\tilde{Y}_i^l \neq Y_i^l), \tag{5}$$

where $\tilde{Y}_i^l$ is the predicted l-th label on instance i, Y_i^l is that of the true label, and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if its inner expression is true and 0 otherwise. Hamming loss is suitable when there are a few dense labels, but may be not a good loss function when the labels are very sparse, since predicting a zero-vector can have very small loss in this case. For sparse labels, precision at top-k is often more preferred, described as the objective function,

$$Loss_{P@k} = \frac{1}{n \cdot k}\sum_{i=1}^{n}\sum_{r=1}^{k}\mathbb{I}(TopLabel(r;i) \in Y_i), \tag{6}$$

where $TopLabel(r;i)$ returns the predicted top r-th label of the instance i, Y_i is the true positive label set of the instance i. It can be noted that this loss has a parameter k. In many cases we want to learn models for general situations rather than some particular k. In this paper, we adopt the weighted average top-label precision as the loss function,

$$Loss_{AvgP} = \sum\nolimits_{k\in K} e^{-k} \cdot Loss_{P@k}, \tag{7}$$

where K is the set of selected k values.

3.2 Optimization

Note that Eq.(4) is hard to be optimized straightforwardly as it is non-differentiable and highly non-convex due to the two `sign` operators, particularly when the average top-label precision loss is incorporated. Fortunately, derivative-free optimization methods have made significant progress recently, which allow us to directly tackle the original objective instead of shifting the problem to be an unfaithful convex objective.

However, derivative-free optimization commonly converges slow if there are too many variables to optimize. Therefore, we simplify the optimization by letting the reconstruction matrix be the pseudo inverse of the compression matrix. The objective function now has only one matrix to be optimized,

$$\arg\min_M Loss\big([M^\top(MM^\top)^{-1}[MY]], Y\big). \quad (8)$$

Note that Eq.(8) enforces the reconstruction matrix and eliminates its degree of freedom, which may not result in the exact solution as Eq.(4), but can be more efficient to be solved.

Since derivative-free optimization methods in general can be applied to optimize arbitrary functions, in Algorithm 1, we let the optimization algorithm as an input, *Opt*, which can be directly invoked to solve the given objective function, represented as its 'argmin' method. The concrete algorithm will be disclosed in the experiment section.

3.3 Learning Base Classifiers

In the training phase, after generating labels in the low-dimensional space, we will train a set of base classifiers, which is still a multi-label classification task. We argue that as we map the original labels to a more compact lower-dimensional space, the label correlation has been absorbed in the embedding matrix, and the new labels are almost independent with each other. Therefore, Binary Relevance method is suitable to solve the task. That is, we train a classifier for each new label separately.

3.4 Prediction

In prediction phase, for an unseen instance $\tilde{\boldsymbol{x}}$, we use base classifiers $\boldsymbol{C}$ to get low-dimensional label vector $\tilde{\boldsymbol{z}} = \boldsymbol{C}(\tilde{\boldsymbol{x}})$. Then we use the reconstruction matrix $\hat{M} = M^\top(MM^\top)^{-1}$ and binary rounding operator to get original label vector $\tilde{\boldsymbol{y}} = [M^\top(MM^\top)^{-1}\tilde{\boldsymbol{z}}]$. The pseudo-code is described in the Prediction procedure of Algorithm 1.

3.5 Acceleration

For large datasets, an idea of accelerating embedding methods is to divide the feature space into multiple regions, and then train in each region independently, with fewer data and labels. For example, in [Bhatia *et al.*, 2015], a clustering method is employed to divide the feature space. Therefore, it is also possible to accelerate BILC by decomposition of the feature space through clustering or decision tree.

4 Experiments

4.1 Configuration

We employ 5 datasets in our experiments, including core5k [Duygulu *et al.*, 2002], bibtex [Katakis *et al.*, 2008], bookmarks [Katakis *et al.*, 2008], NUS-Wide [Chua *et al.*, 2009], Delicious [Tsoumakas *et al.*, 2008]. All the datasets are publicly available.

Table 1: Datasets properties

dataset	#features	#labels	#training instances	#test instances	density
core5k	499	374	4,050	450	3.52
bibtex	1,836	159	6,655	740	2.40
bookmarks	2,150	208	79,071	8,785	2.03
NUS-Wide	128	1,000	27,807	27,808	7.32
Delicious	500	983	14,494	1,611	19.02

We compare BILC with embedding-based methods, including Compressed Sensing [Hsu *et al.*, 2009], 1-Bit Compressed Sensing [Boufounos and Baraniuk, 2008], which can be regarded as a degeneration of BILC that does not optimize the compression matrix, PLST [Tai and Lin, 2012], SLEEC [Bhatia *et al.*, 2015] and BMaD [Wicker *et al.*, 2012], which is a greedy binary decomposition method. Adaboost [Freund and Schapire, 1997] is employed as the base classifier in BILC and 1-Bit Compressed Sensing. It is configured with pruned decision tree, 200 iterations. Correspondingly, LSBoost [Jerome *et al.*, 2001] is employed as the base regressor of Compressed Sensing, PLST and SLEEC. It is also configured with pruned decision tree, 200 iterations, and learning rate 0.1.

For BILC optimization, we use RACOS [Yu *et al.*, 2016] as the derivative-free optimization method through the ZOOpt package (`https://github.com/eyounx/ZOOpt`). As for RACOS, we use $0.4 * L * \hat{L}$ evaluation budget.

We use Precision@k to evaluate the multi-label classification performance in the experiments, which have been widely used in multi-label classification for sparse labels.

4.2 Classification vs Regression

We first evaluate our idea that classification results in simpler models than regression.

We use two embedding methods, Compressed Sensing (CS) and BILC. In the embedded space, we learn regression tree and classification tree (for rounded labels of CS) in the Binary Relevance method. The trees are grown to zero training error for ease of comparison. We then examine the model complexity by the tree depth and the number of tree nodes, we compare the reconstruction errors as well. To directly compare the difference between reconstruction results with ground truth labels, we employ Hamming loss. The four combinations of CS and BILC with regression and classification trees are denoted as CS+RegressionTree, CS+ClassificationTree, BILC+RegressionTree and BILC+ClassificationTree.

Figure 1 presents the results. It can be observed that the depth as well as the number of nodes of regression trees are significantly larger than those of classification trees. This implies that regression model needs higher complexity than classification model, which could result in higher structure risk in predicting labels. Meanwhile, regression models lead to comparable or slightly better Hamming loss (<0.01), which may not be able to compensate the structure risk.

4.3 Label Relationship in Embedded Space

To verify the relationship between the labels in the embedded space, we calculate the prediction accuracy of each la-

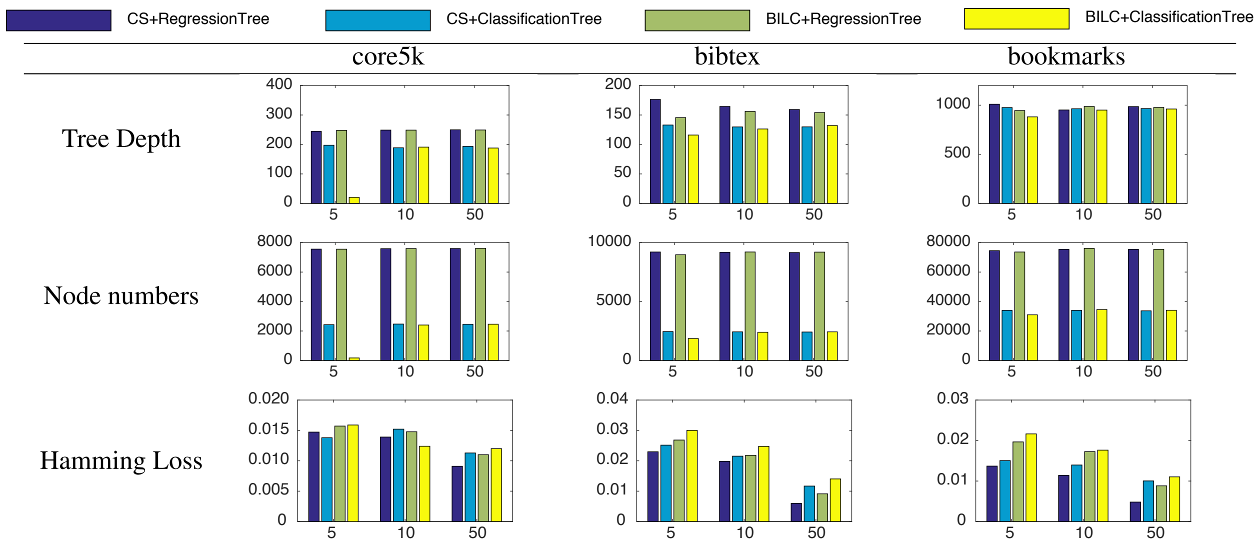

Figure 1: Comparison between classification and regression. The x-axis in each figure denotes the embedding dimension.

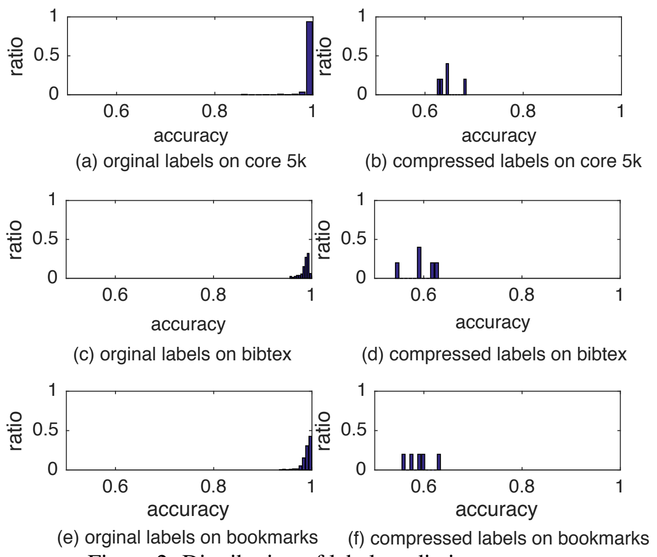

Figure 2: Distribution of label prediction accuracy

bel from all the rest of the labels. Specifically, for each label vector $Y_i(i \in \{1, 2, \ldots, n\})$, we use j-th label $Y_{ij}(j \in \{1, 2, \ldots, L\})$ as the target label, and use all other labels as the features, denoted as Y_i'. We then train a classifier in Y_i' to predict Y_i. Higher accuracy implies that the target label have a stronger relationship with the rest of the labels.

We calculate the prediction accuracies of every label, in both the origin label space and the low-dimensional label space. Figure 2 shows the distribution of these accuracies. It can be observed that, in the origin label dimension, most of the accuracies are close to 1, indicating high correlations among the original labels. After compression, classification accuracies are majorly distributed around 0.6 (note that 0.5 means random guess), which indicates that the labels are much less dependent with each other. This experiment confirms our intention of using Binary Relevance method in the embedded label space, as that the label relationship has mostly been captured in the mapping matrix.

4.4 Performance Comparison

We conduct experiments to compare the performance of BILC with other embedding-based approaches. Note that BILC does not employ the clustering here. We use the weighted average precision as the loss function of BILC, where the set of k includes 1, 3 and 5. We also calculate the Precision@1, Precision@3 and Precision@5 on each dataset, when embedding dimension $\hat{L} = 5, 10, 15, 20$. For each situation, we repeat experiments 10 times and calculate mean of 10 times experiments result.

As shown in Figure 3, BILC achieves the best weighted average precision (AvgP) in *bibtex* and *bookmarks*. BMaD employs a greedy method for decomposition, which fits for small data sets such as *core5k*. Breaking down into precisions of top-1, top-3 and top-5, it can be observed that BILC is better than SLEEC in most cases of the three metrics: when $\hat{L} = 20$, BILC improves SLEEC by 57.5% on *bookmarks* and 46% on *bibtex* in terms of Precision@1. Because the weighted average precision as our objective function gives the Precision@1 the largest weight, it is reasonable that the Precision@1 has the largest improvement over the other methods. We can also observed that the 1-Bit Compressed Sensing, which can be viewed as the degenerated BILC without optimizing the matrix, often has the worst performance. This implies the importance of the optimization. These results demonstrate that classification model can get a better performance than regression and BILC is superior to some state-of-the-art embedding-based methods.

4.5 Acceleration

We have several experiments on Delicious dataset and NUS-Wide dataset to show the performance of acceleration method

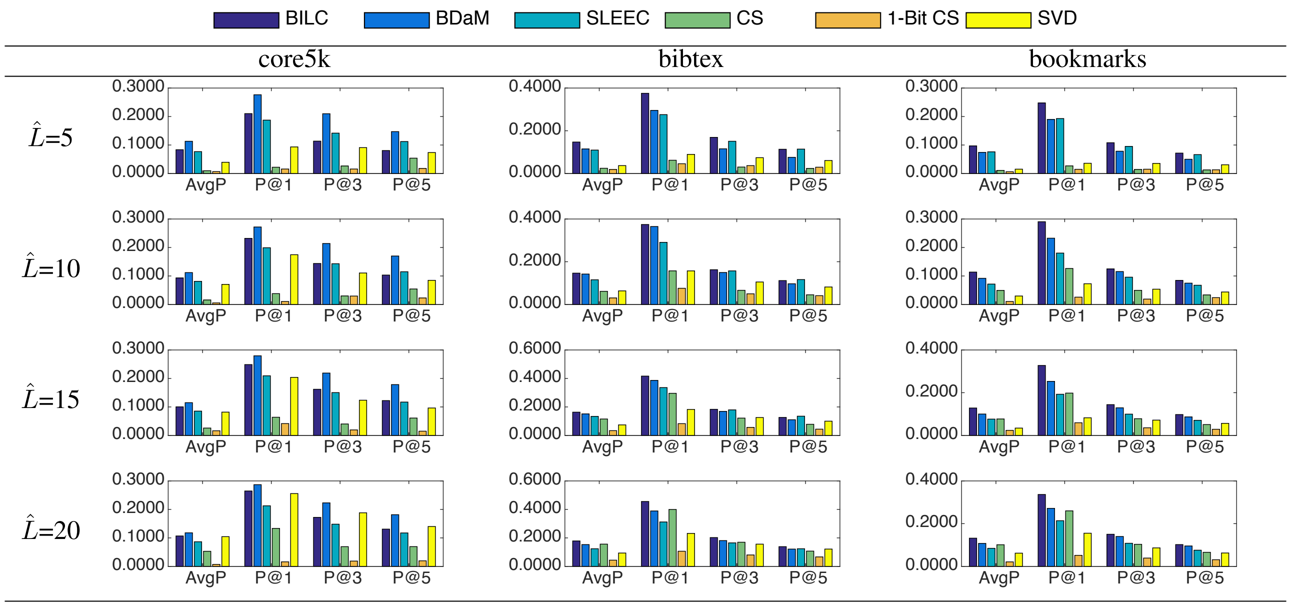

Figure 3: Comparison among different embedding-based multi-label classification methods on three datasets and four embedding dimensions.

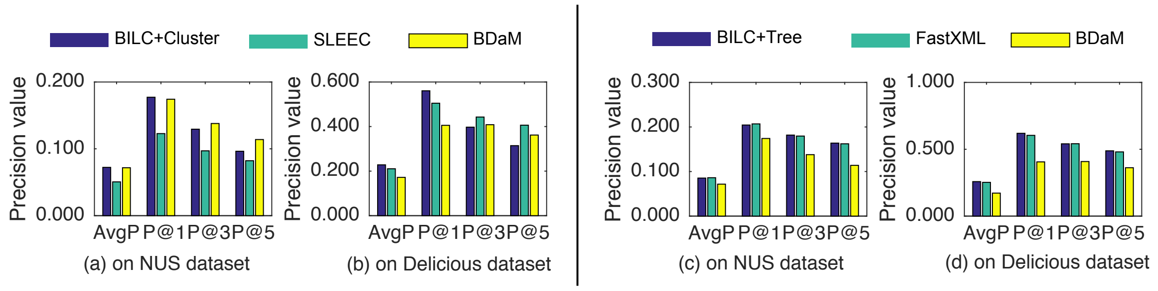

Figure 4: Comparsion of two acceleration methods of BILC. Figures (a) and (b) show comparisons among BILC+Cluster and SLEEC with the same clusters, while figures (c) and (d) show comparisons between BILC+Tree and FastXML with the same decision tree growing method. To judge the performance of acceleration methods of BILC, we also compared them with BDaM.

of BILC. In BILC+Cluster experiment, we set embedding dimension $\hat{L} = 5$ set numCluster = 3 for both BILC+Cluster and SLEEC. In BILC+Tree experiment, we set $\hat{L} = 20$. We use the same split-node method as FastXML [Prabhu and Varma, 2014]. Then learn M and $\hat{M}$ in every leaf node which have effective labels more than embedding dimension. Otherwise we use predict method of FastXML directly on this leaf node.

As shown in Figure 4, BILC+Cluster have higher P@k than SLEEC on both two datasets in many case. BILC+Tree has comparable performance with respect to FastXML method. BILC+Cluster and BILC+Tree have higher AvgP than BDaM on both two data sets.

5 Conclusion

This paper aims at using classification model instead of regression model in embedding-based method and utilizing high-order relationship between labels to serve multi-label classification, since classification can be simpler than regression. We present BILC, a binary compression embedding-based multi-label classification method, which uses derivative-free optimization method to learn a binary embedded space instead of real-valued. The embedding learned by BILC reduces label correlations significantly, which shows that BILC can utilize label relationship effectively. The experiments against other embedding-based method show that BILC has better performance than most embedding-based methods, such as Compressed Sensing, PLST, SLEEC, BDaM, etc. Furthermore, to accelerate BILC, we develop algorithms that combine BILC with cluster technique and tree technique, which are called BILC+Cluster and BILC+Tree. In future work, we will try to improve the running speed of BILC and apply BILC to large-scale multi-label classification task.

References

[Agrawal *et al.*, 2013] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: recommending advertiser bid phrases for web pages. In *WWW*, pages 13–24, 2013.

[Balasubramanian and Lebanon, 2012] Krishnakumar Balasubramanian and Guy Lebanon. The landmark selection method for multiple output prediction. In *ICML*, 2012.

[Bhatia *et al.*, 2015] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, pages 730–738, 2015.

[Boufounos and Baraniuk, 2008] Petros Boufounos and Richard G. Baraniuk. 1-bit compressive sensing. In *Proceedings of the 42nd Annual Conference on Information Sciences and Systems, Princeton, NJ*, pages 16–21, 2008.

[Boutell *et al.*, 2004] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[Brochu *et al.*, 2010] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Computing Research Repository*, abs/1012.2599, 2010.

[Carneiro *et al.*, 2007] Gustavo Carneiro, Antoni B. Chan, Pedro J. Moreno, and Nuno Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):394–410, 2007.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: a real-world web image database from national university of singapore. In *Proceedings of the 8th ACM International Conference on Image and Video Retrieval, Santorini Island, Greece*, 2009.

[Duygulu *et al.*, 2002] Pinar Duygulu, Kobus Barnard, João F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, pages 97–112, 2002.

[Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Fürnkranz *et al.*, 2008] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.

[Goldberg, 1989] David E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[Hsu *et al.*, 2009] Daniel J. Hsu, Sham Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *NIPS*, pages 772–780, 2009.

[Jerome *et al.*, 2001] Friedman Jerome, Hastie Trevor, and Tibshirani Robert. *The Elements of Statistical Learning*. Springer, Berlin, 2001.

[Katakis *et al.*, 2008] Ioannis Katakis, Tsoumakas Grigorios, and Vlahavas Ioannis. Multilabel text classification for automated tag suggestion. *ECML/PKDD*, 75, 2008.

[Li *et al.*, 2010] Hong Li, Yuejian Guo, Min Wu, Ping Li, and Yao Xiang. Combine multi-valued attribute decomposition with multi-label learning. *Expert Systems with Applications*, 37(12):8721–8728, 2010.

[Munos, 2014] Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.

[Pati *et al.*, 1993] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[Prabhu and Varma, 2014] Yashoteja Prabhu and Manik Varma. Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *SIGKDD*, pages 263–272, 2014.

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

[Tsoumakas and Katakis, 2007] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 2007.

[Tsoumakas and Vlahavas, 2007] Grigorios Tsoumakas and Ioannis P. Vlahavas. Random *k* -labelsets: An ensemble method for multilabel classification. In *ECML/PKDD*, pages 406–417, 2007.

[Tsoumakas *et al.*, 2008] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD*, pages 30–44, 2008.

[Wicker *et al.*, 2012] Jörg Wicker, Bernhard Pfahringer, and Stefan Kramer. Multi-label classification using boolean matrix decomposition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 179–186. ACM, 2012.

[Yu *et al.*, 2016] Yang Yu, Hong Qian, and Yi-Qi Hu. Derivative-free optimization via classification. In *AAAI*, pages 2286–2292, 2016.

[Zhang and Schneider, 2011] Yi Zhang and Jeff G. Schneider. Multi-label output codes using canonical correlation analysis. In *AISTAT*, pages 873–882, 2011.

[Zhang and Zhou, 2006] Min-Ling Zhang and Zhi-Hua Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[Zhang and Zhou, 2007] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.