Binary Linear Compression for Multi-label Classification

Wen-Ji Zhou* and Yang Yu† and 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 (Southeast University), Ministry of Education, China

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, which is referred to as 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 gives the specific process of BILC. Section 4 have several experiments and analysis of the experiment results. We make a summary discussion of this paper in Section 5.
2 Background

2.1 Multi-Label Learning

Given a dataset \( D = \{(x_i, y_i)_{i=1}^n\} \), \( x_i \in \mathbb{R}^d \) is the input feature vector, and \( y_i \in \{+1, -1\}^L \) is \( L \)-dimension label vector. The feature matrix is denoted as \( X = [x_1; x_2; \ldots; x_n] \in \mathbb{R}^{n \times d} \) and the label matrix is denoted as \( Y = [y_1; y_2; \ldots; y_n] \in \{+1, -1\}^{n \times L} \). Given \( D \), the goal is to learn a multi-label classifier \( f: \mathbb{R}^d \rightarrow \{+1, -1\}^L \) that can predict the label vector for a given instance.

Traditional single-label classification problems aim at learning a classifier that can tag a data with a single label from a label set. If we want to tag a data with the most relevant subset of a label set, then it is a multi-label classification problem. 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 [Tsoukas 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 [Fünkranz et al., 2008], Random \( k \)-labelsets [Tsoukas and Vlahavas, 2007], etc. The key idea of problem transformation methods is changing data to format 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 usually make changes to popular learning techniques, enabling them to deal with multi-label classification problem, such as ML-\( k \)NN [Zhang and Zhou, 2007], ML-DT [Li et al., 2010], BP-MLL [Zhang and Zhou, 2006], etc. For example, ML-\( k \)NN method adapts lazy learning technique \( k \)NN on multi-label classification. ML-DT adopts decision tree technique to solve multi-label classification tasks. BP-MLL adapts the popular backpropagation algorithm on multi-label learning.

2.2 Embedding-Based Method

Embedding-based method is a group of methods that can use label relationship to reduce the effective number of labels. They map label vectors to a lower-dimension space and do predictions in the new label space, then map the prediction results 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 \( L \)-dimensional linear subspace as \( z = U y \), which \( L < L \) and \( z \in \mathbb{R}^L \). \( U \) is called compression matrix. This process is usually called compression. And then train regressors to predict \( z \) from \( x \), which \( z = V(x) \). Finally, they find a reconstruction matrix \( \hat{U} \) to map embedded label vectors to the origin label space by \( y = \hat{U} V(x) \), which is called reconstruction or decompression.

Different embedding-based approaches mainly differ in the methods of compression and reconstruction. For example, Compressed Sensing 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:

\[
\min_{V \in \mathbb{R}^{L \times d}} ||P_{\Omega}(Y^T Y) - P_{\Omega}(X^T V^T V X)||_F^2 + \lambda ||V||_F^2 + \mu ||V X||_1
\]

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

\[
(P_{\Omega}(Y^T Y))_{ij} = \begin{cases} 
(y_i, y_j), & \text{if } (i, j) \in \Omega \\
0, & \text{otherwise}
\end{cases}
\]

SLEEC uses a set of regressors to get embedding and use \( k \)NN to predict label vector in high-dimensional label space.

These 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 propose a method that employs classification models.

There are a few previous studies that employed binary matrix decomposition to set the embedded space binary, e.g. [Wicker et al., 2012]. Since these methods operate on binary relationship operators directly, the optimal decomposition is quite hard to obtain. 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 \( z \) and learns from their feedbacks \( f(z) \) 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 support 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, 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 $M \in \mathbb{R}^{L \times L}$ and a reconstruction matrix $M = \mathbb{R}^{L \times L}$. We use compression matrix $M$ to map a label vector $y$ to $L$-dimension binary vector $z$.

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

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

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

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

$$\hat{y} = [M \hat{z}].$$

During the training and testing process above, the nonlinear $\text{sign}$ 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,

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

where $\text{Loss}$ 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.

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

$$\text{Loss}_{\text{Hamming}} = \frac{1}{n \cdot L} \sum_{i=1}^{n} \sum_{l=1}^{L} \mathbb{I}(\hat{Y}_l^i \neq Y_l^i),$$

where $\hat{Y}_l^i$ is the predicted $l$-th label on instance $i$, $Y_l^i$ 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,

$$\text{Loss}_{p@k} = \frac{1}{n \cdot k} \sum_{i=1}^{n} \sum_{r=1}^{k} \mathbb{I}(\text{TopLabel}(r; i) \in Y_l^i),$$

where $\text{TopLabel}(r; i)$ returns the predicted top-$r$-th label of the instance $i$, $Y_l^i$ 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,

$$\text{Loss}_{\text{AvgP}} = \sum_{k \in K} e^{-k} \cdot \text{Loss}_{p@k},$$

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 $\text{sign}$ 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.

### Algorithm 1 BILC

**Input:**
- Training Data $D = \{(x_i, y_i)\}_{i=1}^{n}$
- A derivative-free optimization method $\text{Opt}$
- Multi-label classification loss $\text{Loss}$
- Embedding dimension $L$

**Procedure:** Learning

1. $M \leftarrow \text{Opt. argmin}_M \left\{ \text{Evaluation}(M; \text{Loss}, \{y_i\}) \right\}$
2. $z_i = [M y_i], \forall i$
3. train base classifiers $C$ over $\{(x_i, z_i)\}_{i=1}^{n}$ by Binary Relevant method
4. return $M, \hat{M} = M^T (MM^T)^{-1}, C$

**Procedure:** Evaluation($M; \text{Loss}, \{y_i\}$)

1. $z_i = [M y_i], \forall i$
2. $\hat{y}_i = [M^T (MM^T)^{-1} z_i], \forall i$
3. $\epsilon = \text{Loss}(\hat{y}, y)$
4. return $\epsilon$

**Procedure:** Prediction($x$)

1. for $l = 1$ to $\hat{L}$ do
2. $\hat{z}(l) = C_l(x)$
3. end for
4. $\hat{y} = [M \ast \hat{z}]$
5. return $\hat{y}$

where $\hat{Y}_l^i$ is the predicted $l$-th label on instance $i$, $Y_l^i$ 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,
Therefore, we simplify the optimization by letting the recon-
struction matrix be the pseudo inverse of the compression ma-
trix. The objective function now has only one matrix to be
optimized,
\[
\arg \min_M \text{Loss} \left( \left[ MM^T \right]^{-1} [MY]^2, Y \right).
\]
Note that Eq. (8) enforces the reconstruction matrix and elimi-
nates 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, repre-
sented 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 indepen-
dent 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 \(\hat{x}\), we use base
classifiers \(\hat{C}\) to get low-dimensional label vector \(\hat{z} = \hat{C}(\hat{x})\).
Then we use the reconstruction matrix \(\hat{M} = M^T \left( M M^T \right)^{-1} \) and
binary rounding operator to get original label vector \(\hat{y} = \left[ M^T \left( M M^T \right)^{-1} \right] \hat{z}\). The pseudo-code is described in
the Prediction procedure of Algorithm 1.

### 3.5 Acceleration

For large datasets, an idea of accelerating embedding meth-
ods is to divide the feature space into multiple regions, and
then train in each region independently, with fewer data and
labels. For example, in [Bhatta 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], book-
marks [Katakis et al., 2008], NUS-Wide [Chua et al., 2009],
Delicious [Tsoumakas et al., 2008]. All the datasets are pub-
lcily available.

We compare BILC with embedding-based methods, in-
cluding Compressed Sensing [Hsu et al., 2009], 1-Bit Com-
pressed Sensing [Boufounos and Baraniuk, 2008], which can
be regarded as a degeneration of BILC that does not opti-
mize the compression matrix, PLST [Tai and Lin, 2012],
SLEEC [Bhatta et al., 2015] and BMaD [Wicker et al.,
[2012], which is a greedy binary decomposition method. Ad-
daboost [Freund and Schapire, 1997] is employed as the base
classifier in BILC and 1-Bit Compressed Sensing. It is con-
figured with pruned decision tree, 200 iterations. Correspond-
ingly, 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 * L
number of iterations, and set samplesize 50, positivesize 2, uncertainhits 1 and probability
0.99, which are the default parameters.

We use Precision@k to evaluate the multi-label classifica-
tion 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 sim-
pler models for embedded multi-label classification prob-
lems, 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 ex-
amine the model complexity by the depth of the trees and
the number of tree nodes, we compare the reconstruction
errors as well. To directly compare the difference be-
tween reconstruction results with ground truth labels, we
use Hamming loss as the metric. The four combinations
of CS and BILC with regression and classification trees
are denoted as CS+RegressionTree, CS+ClassificationTree,
BILC+RegressionTree and BILC+ClassificationTree, corre-
spindingly.

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 embed-
ded space, we calculate the prediction accuracy of each la-
bel from all the rest of the labels. Specifically, for each la-
bel vector \(Y_i(i \in \{1, 2, \ldots, n\})\), we use \(j\)-th label \(Y_{ij}(j \in \)
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 routine. In these experiments, 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 separately on each dataset, when embedding dimension $L = 5, 10, 15, 20$. 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 many cases. BMaD employs a greedy method for decomposition, which might fit for small data sets such as core5k dataset. Breaking down into precision 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 $L = 20$, BILC improves SLEEC by 57.5% on bookmarks dataset and 46% on bibtex dataset 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 mapping 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.
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 3, 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. The result shows that BILC can be used in subspace that divided by tree or clustering.

### 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


