Multi-label Classification with Principle Label Space Transformation

Farbound Tai B94901176@NTU.EDU.TW
Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW
Department of Computer Science and Information Engineering, National Taiwan University, Taipei 100, Taiwan

Abstract

We propose a novel hypercube view that perceives the label space of multi-label classification problems geometrically. The view allows us to not only unify many existing multi-label classification approaches, but also design a novel algorithm, Principle Label Space Transformation (PLST), which seeks important correlations between labels before learning. The simple and efficient PLST relies on only singular value decomposition as the key step. Experimental results demonstrate that PLST is faster than the traditional Binary Relevance approach and is superior to the modern Compressive Sensing approach in terms of both performance and efficiency.

1. Introduction

Multi-label classification problems naturally arise in domains such as text mining, vision, or bioinformatics. For instance, a document is usually associated with more than one category; a picture often includes many objects; a gene is usually multi-functional. The problem generalizes the traditional multi-class classification problem—the former allows *a set of labels* to be associated with an instance while the latter allows only one. Because of the wide range of potential applications such as scene classification (Boutell et al., 2004), video segmentation (Snoek et al., 2006), genomics (Barutcuoglu et al., 2006; Vens et al., 2008), music (Trohidis et al., 2008) and text (Schapire & Singer, 2000) categorization, multi-label classification is attracting more and more research attentions.

Existing multi-label classification approaches usually fall into one of the two categories (Tsoumakas et al., 2010a): *Algorithm Adaptation* (AA) or *Problem Transformation* (PT). As its name suggests, AA directly extends some specific algorithms to solve the multi-label classification problem. Typical members of AA include Adaboost.MH (Schapire & Singer, 2000), Rank-SVM (Elisseeff & Weston, 2001), Multi-label C4.5 (Clare & King, 2001) and ML-KNN (Zhang & Zhou, 2007). PT approaches, on the other hand, transform the multi-label classification problem to one or more reduced tasks. Typical members of PT include Label Power-set (LP), Binary Relevance (BR) and Label Ranking (LR; Fürnkranz et al., 2008). LP reduces multi-label classification to multi-class classification by treating each distinct label set as a unique multi-class label. BR, also known as one-versus-all, reduces multi-label classification to many different binary classification tasks, each for one of the labels. LR approaches transform the multi-label classification problem to the task of ranking all the labels on hand by relevance and the task of determining a threshold of relevance. As can be seen from above, an advantage of PT over AA is that any algorithm which deals with the reduced tasks can be easily extended to multi-label classification via the transformation.

In this paper, we propose a new way to perceive PT approaches: the hypercube view. The view describes all possible label sets in the multi-label classification problem as the vertices of a high-dimensional hypercube. The view not only unifies LP, BR and LR under the same framework, but also allows us to design better methods that make use of the geometric properties of those label-set vertices. We demonstrate the use of the hypercube view with a novel method, Principle Label Space Transformation (PLST), that captures the important correlations between labels using a flat in the high-dimensional space. The method only uses a simple linear encoding of the vertices and a simple linear decoding of the reduced predictions, both easily computed from the Singular Value Decomposition (SVD) of a matrix composed of the label-set vertices. Moreover, by keeping only the key correlations, our method can dramatically decrease the number of reduced tasks

Appearing in *Working Notes of the 2nd International Workshop on Learning from Multi-Label Data*, Haifa, Israel, 2010.

to be solved without loss of prediction accuracy.

A recent related work, multi-label prediction via Compressive Sensing (CS; Hsu et al., 2009), also seeks to perform multi-label classification with a linear encoding of the label-sets vertices. CS operates under the assumption of sparsity in the label sets and thus can describe the label-set vertices with a small number of linear random projections as its encoding. Although the encoding component of CS is linear, the decoding component is not. In particular, for each incoming test instance, CS needs to solve an optimization problem with respect to its sparsity assumption. That is, CS can be time consuming during prediction. In our experiments, we will demonstrate that PLST is not only computationally more efficient than CS, but also outperforms CS in terms of prediction accuracy.

The paper is organized as follows. In Section 2, we give a formal setup of the multi-label classification problem and introduce the hypercube view. Then, in Section 3, we describe our proposed method: PLST. Finally, we present the experimental results in Section 4 and conclude in Section 5.

2. Hypercube View

In the multi-label classification problem, we seek for a multi-label classifier that maps the input vector $\mathbf{x} \in \mathbb{R}^d$ to a set of label $\mathcal{Y}$, where $\mathcal{Y} \subseteq \mathcal{L} = \{1, 2, \ldots, K\}$ with K being the number of classes. Consider a training set $\mathcal{S}$ that contains N training examples of the form $(\mathbf{x}_n, \mathcal{Y}_n)$. Multi-label classification aims at using $\mathcal{S}$ to find a multi-label classifier $g\colon \mathbb{R}^d \to 2^{\mathcal{L}}$ such that $g(\mathbf{x})$ predicts $\mathcal{Y}$ well on any future test example $(\mathbf{x}, \mathcal{Y})$.

The key of the *hypercube view* is to represent the label set $\mathcal{Y}$ by a vector $\mathbf{y} \in \{0,1\}^K$, where the k-th component of $\mathbf{y}$ is 1 if and only if $k \in \mathcal{Y}$. Then, as shown in Figure 1, we can visualize each $\mathcal{Y}$ as a vertex of a K-dimensional hypercube. The k-th component of $\mathbf{y}$ corresponds to an axis of the hypercube, which represents the presence or absence of a label k in $\mathcal{Y}$. We will use $\mathcal{Y}$ and its corresponding $\mathbf{y}$ interchangeably in this paper. The hypercube view allows us to unify many existing PT approaches, as discussed below.

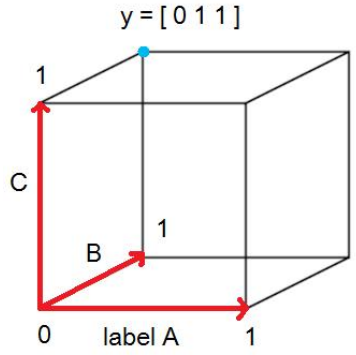

Figure 1. A Hypercube with $K = 3$

Hypercube View of Label Power-set. One of the simplest approaches to multi-label classification is Label Power-set (LP), as shown in Algorithm 1.

Algorithm 1 Label Power-set

1. pre-processing: map each vertex $\mathbf{y}_n$ (or each label-set $\mathcal{Y}_n$) to a hyper-label $y_n \in \{1, 2, \cdots, 2^K\}$ with a bijection function B.
2. training: learn a multi-class classifier $\tilde{g}(\mathbf{x})$ from $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$.
3. predicting: for each $\mathbf{x}$, return $B^{-1}\big(\tilde{g}(\mathbf{x})\big)$.

In particular, LP simply treats each vertex of the hypercube as a different hyper-label, and performs regular multi-class classification with the hyper-labels. The approach is often criticized for the large number of possible hyper-labels and the relatively few number of examples per hyper-label, which may degrade the learning performance.

Hypercube View of Binary Relevance. Another straight-forward approach to multi-label classification is Binary Relevance (BR). BR decomposes the original multi-label problem into K isolated relevance-learning sub-tasks, as shown in Algorithm 2.

Algorithm 2 Binary Relevance

1. training: for $k = 1$ to K, learn a relevance function $r_k(\mathbf{x})$ from $\Big\{\big(\mathbf{x}_n, \mathbf{y}_n[k]\big)\Big\}_{n=1}^N$.
2. predicting: for each input vector $\mathbf{x}$, return $\text{round}\Big(\big[r_1(\mathbf{x}), r_2(\mathbf{x}), \cdots, r_K(\mathbf{x})\big]\Big)$, where $\text{round}(\cdot)$ maps each component of the vector to the closest value in $\{0, 1\}$.

Using the hypercube view, the k-th iteration of BR can be thought as *projecting* the vertices to the k-th dimension (axis) before training. In addition, $\mathbf{r}(\mathbf{x}) = [r_1(\mathbf{x}), r_2(\mathbf{x}), \cdots, r_K(\mathbf{x})]$ can be thought as a point in the space $\mathbb{R}^K$, and the $\text{round}(\cdot)$ operation maps the point to the closest vertex of the hypercube in terms of the ℓ_1-distance.

Despite its effectiveness, BR is often criticized for neglecting the correlation between labels, which may carry useful information in multi-label classification tasks. Furthermore, the training complexity of BR is linear to the number of labels K, which can still be expensive if K is too large. Recently, Hsu et al. (2009) attempted to address this problem through Compressive Sensing. Their work will be explained later in this section.

Hypercube View of Label Ranking. Label Ranking (LR) approaches learn two issues (jointly or separately) from the multi-label classification data set: the *order* of label relevance, and the *threshold* for label presence. Note that BR is a special case of LR when the ordering and the thresholding are taken from an underlying relevance-scoring function $\mathbf{r}(\mathbf{x})$.

Using the hypercube view, the ordering issue in LR can be thought as learning a K-step path from $[0, 0, \cdots, 0]$ to $[1, 1, \cdots, 1]$ using the hypercube edges. Each vertex $\mathbf{y}_n$ in the training examples then represent multiple K-step paths that goes through $\mathbf{y}_n$ with the ideal thresholding at $\|\mathbf{y}_n\|_1$.

Hypercube View of Compressive Sensing. Under the assumption that the label sets $\mathcal{Y}$ are sparse (i.e. containing only a few elements), it is possible to *compress* the label sets and learn to predict the compressed labels instead. Such a possibility allows Compressive Sensing (CS; Hsu et al., 2009) to reduce the number of sub-tasks in BR to be computationally feasible for data sets with a large K. In particular, each label set $\mathcal{Y}$ (vertex $\mathbf{y}$) can be taken as a K-dimensional signal. The theory of Compressive Sensing states that when the signals are sparse, one does not need to sample at the Nyquist rate in order to accurately recover the original signals. Thus, when all $\mathbf{y}$ contain only a few 1's, CS only needs to solve $M \ll K$ sub-tasks instead of K for multi-label classification, as shown in Algorithm 3.

Using the hypercube view, the m-th iteration of CS can be thought as *projecting* the vertices to a random direction before training. Because $M \ll K$, the subspace explored by CS is much smaller than the space that the hypercube resides in. CS is able to work on such a small subspace because of the label-set sparsity assumption, which implies that only a limited number of vertices in the hypercube are relevant for the multi-label classification task.

Although the random projection in the pre-processing step of CS is efficient, the prediction step requires solving an optimization problem for every coming input vector $\mathbf{x}$. Such a prediction step is very time consuming. In addition, the assumption on label-set sparsity puts a restriction on the practical use of the CS approach.

Algorithm 3 Compressive Sensing

1. pre-processing: compress $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{h}_n)\}$, where $\mathbf{h} = \mathbf{P}_s \cdot \mathbf{y}$ using an M by K random projection matrix $\mathbf{P}_s$ with M determined by the assumed sparsity level s.
2. training: for $m = 1$ to M, learn a function $r_m(\mathbf{x})$ from $\{(\mathbf{x}_n, \mathbf{h}_n[m])\}_{n=1}^N$.
3. prediction: for each input vector $\mathbf{x}$, compute $\mathbf{r}(\mathbf{x}) = \big[r_1(\mathbf{x}), r_2(\mathbf{x}), \cdots, r_M(\mathbf{x})\big]$. Then, obtain a sparse vector $\tilde{\mathbf{y}}$ such that $\mathbf{P}_s \cdot \tilde{\mathbf{y}}$ is "closest" to $\mathbf{r}(\mathbf{x})$ using an optimization algorithm. Finally, return $\tilde{\mathbf{y}}$.

3. Proposed Approach

As discussed, CS relies on label-set sparsity to consider a small number of vertices of the hypercube. Our proposed approach stems from the same consideration, but without requiring the label-set sparsity assumption. From the hypercube view, there are 2^K vertices of the hypercube, and each training example $(\mathbf{x}_n, \mathbf{y}_n)$ occupies only one vertex $\mathbf{y}_n$. In large multi-label classification data sets, it is typical for K to exceed hundreds or even thousands. Then, usually the number of training examples $N \ll 2^K$. In other words, very few vertices will be occupied by enough training examples. We call this phenomenon *hypercube sparsity* to distinguish it from the *label-set sparsity* that CS uses. Because of the hypercube sparsity, multi-label classification algorithms do not need to learn with the entire hypercube in $\mathbb{R}^K$ and can focus on a much smaller subspace of $\mathbb{R}^K$ instead.

Note that label-set sparsity implies hypercube sparsity, but not vice versa. By definition, for a data set with label-set sparsity at s, all the hypercube vertices with more than s labels are unoccupied by training examples—the phenomenon of hypercube sparsity. For instance, if a data set is label-set sparse at $s = 2$, then such a data set is also hypercube sparse because the number of occupied vertices is at most $\binom{K}{2} + K + 1 \ll 2^K$.

On the other hand, hypercube sparsity does not necessarily imply label-set sparsity, because the few occupied label-set vertices may contain many labels. For instance, a data set with all label sets containing at

Algorithm 4 Linear Label Space Transformation

1. pre-processing: encodes $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{h}_n)\}$, where $\mathbf{h} = \mathbf{P} \cdot \mathbf{y}$ is a point on the M-flat using an M by K projection matrix $\mathbf{P}$.
2. training: for $m = 1$ to M, learn a function $r_m(\mathbf{x})$ from $\{(\mathbf{x}_n, \mathbf{h}_n[m])\}_{n=1}^N$.
3. prediction: for each input vector $\mathbf{x}$, compute $\mathbf{r}(\mathbf{x}) = \big[r_1(\mathbf{x}), r_2(\mathbf{x}), \cdots, r_M(\mathbf{x})\big]$. Then, return $D(\mathbf{r}(\mathbf{x}))$ where $D\colon \mathbb{R}^M \to \{0,1\}^K$ is a decoding function from the M-flat to the hypercube.

least $(K-1)$ labels is hypercube sparse with the number of occupied vertices being at most $K + 1 \ll 2^K$, but is by no means label-set sparse.

3.1. Linear Label Space Transformation

We now study a simple framework that focuses on a subspace instead of the whole hypercube in $\mathbb{R}^K$. The framework would take an M-flat as the subspace and encodes each vertex $\mathbf{y}$ of the hypercube to a point $\mathbf{h}$ in the flat by projection. Then, the original multi-label classification problem with $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ becomes a multi-dimensional regression problem with $\{(\mathbf{x}_n, \mathbf{h}_n)\}_{n=1}^N$. After obtaining a multi-dimensional regressor $\mathbf{r}(\mathbf{x})$ that predicts $\mathbf{h}$ well, the framework will then map $\mathbf{r}(\mathbf{x})$ back to a vertex of the hypercube in $\mathbb{R}^K$ using some decoder D. The framework will be named *Linear Label Space Transformation* (LLST), as shown in Algorithm 4.

Note that LLST takes BR and CS as special cases. For BR, we can simply take $\mathbf{P} = \mathbf{I}$ as the projection method, and D as the component-wise round-to-$\{0,1\}$ function. Because $M = K$, many regressors r_m are needed when K is large.

CS seeks to reduce the number of regressors by considering a flat with $M \ll K$. Its projection matrix $\mathbf{P}$ is chosen randomly from an appropriate distribution (such as Gaussian, Bernoulli, or Hadamard) and D, the reconstruction algorithm in the terminology of CS, requires solving an optimization problem for each different $\mathbf{x}$.

3.2. Principle Label Space Transformation

Because of the hypercube sparsity property in large multi-label classification data sets, LLST with $M \ll K$ could be advantageous in reducing computational cost. Our proposed approach, *Principle Label Space Transformation* (PLST), seeks to find the projection matrix $\mathbf{P}$ and the decoder D for such an M-flat through Singular Value Decomposition (SVD).

Algorithm 5 Principle Label Space Transformation

1. With a parameter M, perform SVD on $\mathbf{Y}$ and obtain $\mathbf{U}_M^T = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \ldots \quad \mathbf{u}_M]$.
2. Run LLST (Algorithm 4) using $\mathbf{P} = \mathbf{U}_M^T$ and $D(\mathbf{r}(\mathbf{x})) = \text{round}\Big(\mathbf{U}_M \cdot \mathbf{r}(\mathbf{x})\Big)$.

In particular, we form a matrix $\mathbf{Y}$ with each column being $\mathbf{y}_n$, the *occupied* vertices. Then, we perform SVD on the K by N matrix $\mathbf{Y}$ to obtain three matrices (Datta, 1995)

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{1}$$

Here $\mathbf{U}$ is a K by K unitary matrix, $\mathbf{\Sigma}$ is a K by N diagonal matrix, and $\mathbf{V}$ is a N by N unitary matrix. Through SVD, each $\mathbf{y}_n$ can be represented as a linear combination of the singular vectors $\mathbf{u}_m$ in $\mathbf{U}$. The vectors form a basis of a flat that passes through all the $\mathbf{y}_n$. The matrix $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values σ_m that corresponds to the singular vectors $\mathbf{u}_m$. We shall assume that the singular values are ordered such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_K$.

Note that (1) can be rewritten as

$$\mathbf{U}^T\mathbf{Y} = \mathbf{\Sigma}\mathbf{V}^T$$

where the orthogonal basis $\mathbf{U}^T$ can be seen as a projection matrix of $\mathbf{Y}$ that maps each $\mathbf{y}$ to a different coordinate system. Since the largest M singular values correspond to the principle directions of the original label space, we could discard the rest of the singular values and their associated basis vectors in $\mathbf{U}^T$ to obtain a smaller projection matrix $\mathbf{P} = \mathbf{U}_M^T = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_M]^T$ that maps the vertices $\mathbf{y}$ to the M-flat. Unlike CS, in which $\mathbf{P}$ is formed randomly, the projection matrix using the principle directions is guaranteed of the minimum encoding error from the (training) vertices to the M-flat. Note that the notion of principle directions is similar to the usual technique of Principle Component Analysis (PCA) for input pre-processing in machine learning (Hastie et al., 2001). In PCA, the principle components are obtained by decomposing the matrix formed from $\mathbf{x}_n$, and in our PLST, the principle directions are obtained by decomposing the matrix $\mathbf{Y}$ formed from $\mathbf{y}_n$.

We can now define an efficient decoder D for PLST. Because $\mathbf{P} = \mathbf{U}^T$ is an orthogonal matrix, $\mathbf{P}^{-1} = \mathbf{P}^T$. This means that $\mathbf{U}_M$ can be used to map any vector $\mathbf{r}$

on the flat back to a point $\mathbf{U}_M \cdot \mathbf{r}$ in $\mathbb{R}^K$. Then, a simple rounding to the nearest vertex in the ℓ_1-distance sense (like BR) finishes the decoding. The simple steps of PLST are listed in Algorithm 5.

4. Experiments

Next, we conduct experiments on six real-world data sets to compare the three algorithms under the LLST framework: BR, CS and our proposed PLST. The data sets are downloaded from Mulan (Tsoumakas et al., 2010b) and cover a variety of domains, sizes and characteristics, as shown in Table 1. We include data sets with a particularly large number of labels such as `delicious`, `corel5k`, `bibtex` and `mediamill` to test the effectiveness of CS and PLST in reducing the dimension of the label space.

The *cardinality* column of Table 1 is defined as the average number of labels per example. The *distinct* column of Table 1 shows the number of distinct label sets, or using the hypercube view, the number of vertices occupied by examples. Dividing the value of distinct by 2^K in Table 1, we see that hypercube sparsity indeed exists in every data set. On the other hand, the *nonzero* column of Table 1 shows the maximum number of non-zero entries in $\mathbf{y}_n$. Comparing the value of nonzero to K in Table 1, we see that most data sets come with a strong label-set sparsity except `yeast`.

In all experiments, we randomly partition each data set into 90% for training and 10% for testing. We record the mean and the standard error of the test Hamming Loss over 20 different random partitions. The Hamming Loss (HL) per example is defined as

$$\Delta(\tilde{\mathbf{y}}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \tilde{\mathbf{y}}[k] \oplus \mathbf{y}[k].$$

It is a popular error measure for multi-label classification and falls back to the usual binary classification error when $K = 1$. Evaluation with other measures will be included in the long version of this paper (Tai & Lin, 2010).

We couple BR, PLST and CS with Ridge Linear Regression (Hastie et al., 2001) with $\lambda = 0.01$ as the regression learner throughout the experiments. For CS, we follow the recommendation from Hsu et al. (2009) to use the Hadamard matrix as the projection matrix $\mathbf{P}$. From their experiments, there is no clear winner for the optimal reconstruction algorithm across different compression and sparsity levels. We use Orthogonal Matching Pursuit as the reconstruction algorithm that computes the decoding function D. The sparsity parameter for the reconstruction algorithm is set to the *nonzero* column in Table 1.

4.1. Analysis of Results

The performance of BR, PLST, and CS at different sizes of the reduced sub-tasks are shown in Table 2. We see that PLST is capable to achieve reasonable performance by reducing the label space to a lower dimensional flat. We have also plotted these results on `delicious` and `corel5k` in Figures 2 and 3. The figures have zoomed-in sections to better differentiate between BR and PLST.

As can be seen in Table 2, PLST significantly and universally outperforms CS in every data set. This is even more obvious in Figures 2 and 3 where the HL curve for PLST is always below that of CS's at every dimension. The only exception to this trend is when the label dimension is reduced to 20% in `emotions`. Nevertheless, since this corresponds to a reducing from a label set of size 6 to a single regression label, it is an extreme case and is less relevant to the approach comparison in general.

Table 3 records the running time of BR, PLST and CS at the optimal reduced sub-task size M^* and Table 4 records their respective HL. Here M^* is defined as the minimum dimension at which the HL difference between BR and PLST is within their respective standard errors. In other words, this can be seen as the reduced dimension at which no performance loss is incurred. All the timing experiments were performed on the AMD Opteron Quad Core 2378 2.4 GHz Processor with 512 KB of cache. The programming environment was in MATLAB version 7.5.0.338 (R2007b). For most of the data sets with large amount of labels, PLST is able to drastically reduce the learning and inference time compared to BR. This is less obvious in small data sets like `yeast` and `emotion` since their number of labels is already small before the transformation.

From Table 2-4 and Figures 2-3, it is clear that PLST is highly effective at reducing the number of sub-tasks solved for multi-label classification. Large data sets like `delicious`, `corel5k` and `mediamill` can be reduced to only 13%, 5%, and 11% of their original computational effort respectively with no sacrifice in performance. Note that we can further reduce the computational effort by tolerating a slight increase in HL, as can be seen from Table 2. These results demonstrate that PLST can take advantage of the hypercube sparsity to efficiently solve multi-label classification problems.

Table 1. Data Set Statistics

data set	domain	N	K	cardinality	distinct	hypercube sparsity	nonzero
`delicious`	text	16105	983	19.02	15806	1.93×10^{-292}	25
`corel5k`	text	5000	374	3.52	3175	8.25×10^{-110}	5
`bibtex`	text	7395	159	2.40	2856	3.90×10^{-45}	28
`mediamill`	video	43507	101	4.38	6555	2.59×10^{-27}	18
`yeast`	biology	2417	14	4.24	198	1.21×10^{-2}	11
`emotions`	music	593	6	1.87	27	4.22×10^{-1}	3

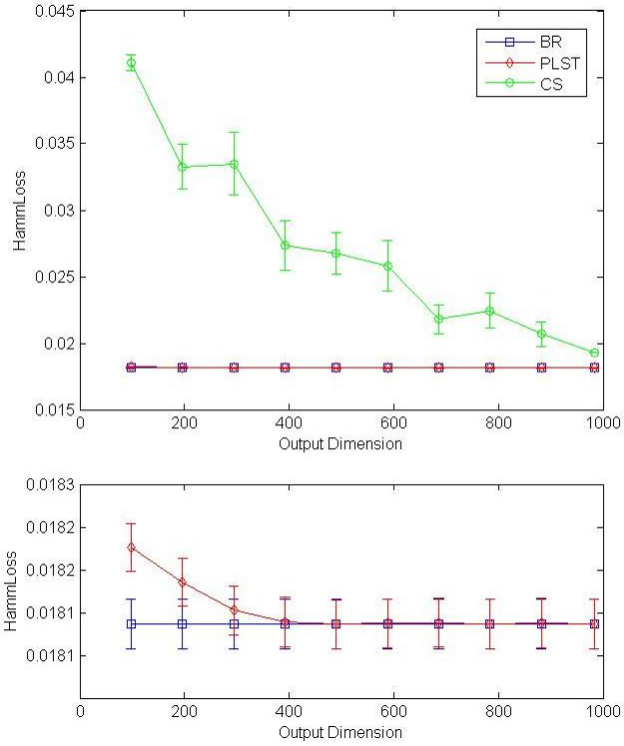

Figure 2. `delicious`: Test HL for BR, PLST and CS

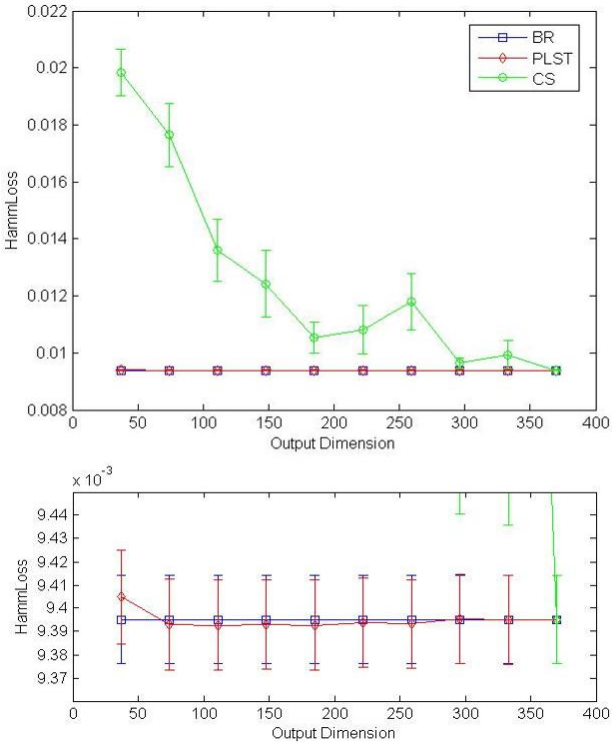

Figure 3. `corel5k`: Test HL for BR, PLST and CS

5. Conclusion

We presented the novel hypercube view for problem transformation approaches to multi-label classification. The view offers geometric interpretations to many existing algorithms including Binary Relevance, Label Power-set, Label Ranking and Compressive Sensing. Inspired by this view, we introduced the notion of hypercube sparsity and took it into account by Principle Linear Space Transformation (PLST). Experimental results verified that PLST is successful in reducing the computational effort for multi-label classification, especially for data sets with large numbers of labels. We also compared our approach to Compressive Sensing and demonstrated that PLST is faster as well as more accurate by a significant amount.

As demonstrated through experiments, PLST was able to achieve similar performance with substantially less dimensions compared to the original label-space. An immediate future work is to conclude how to automatically and efficiently determine a reasonable parameter M for PLST.

To further validate the advantages of PLST, future work includes extensive performance comparison of PLST with related multi-label classification algorithms, including hypergraph spectral learning (Sun et al., 2008). It would also be interesting to consider (linear or nonlinear) encoding to binary classification instead of regression tasks.

Acknowledgements

We thank the reviewers for their many valuable suggestions. This research was supported by the National Science Council of Taiwan via NSC 98-2221-E-002-192.

References

Barutcuoglu, Z., Schapire, R. E., and Troyanskaya, O. G. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

Clare, A. and King, R. D. Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 42–53, 2001.

Datta, B. N. *Numerical Linear Algebra and Applications*. Brooks/Cole Publishing, 1995.

Elisseeff, A. and Weston, J. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14*, pp. 681–688, 2001.

Fürnkranz, J., Hüllermeier, E., Lozamencía, E., and Brinker, K. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

Hsu, D., Kakade, S. M., Langford, J., and Zhang, T. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems 22*, pp. 772–780, 2009.

Schapire, R. E. and Singer, Y. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

Snoek, C. G. M., Worring, M., van Gemert, J. C., Geusebroek, J. M., and Smeulders, A. W. M. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pp. 421–430, 2006.

Sun, L., Ji, S., and Ye, J. Hypergraph spectral learning for multi-label classification. In *KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pp. 668–676, 2008.

Tai, F. and Lin, H.-T. Multi-label classification with principle label space transformation. Technical report, National Taiwan University, 2010.

Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. Multilabel classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pp. 325–330, 2008.

Tsoumakas, G., Katakis, I., and Vlahavas, I. *Mining Multi-label Data*. Data Mining and Knowledge Discovery Handbook, O. Maimon, L. Rokach (Ed.). Springer, 2nd edition, 2010a.

Tsoumakas, G., Vilcek, J., and Xioufis, E. S. Mulan: A java library for multi-label learning, 2010b. `http://mulan.sourceforge.net/datasets.html`.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

Zhang, M. and Zhou, Z. ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

Table 2. Test Hamming Loss of BR, PLST and CS

		100%	80%	60%	40%	20%
delicious	BR	0.0181 ± 0.0000	-	-	-	-
	PLST	0.0181 ± 0.0000	0.0181 ± 0.0000	0.0181 ± 0.0000	0.0181 ± 0.0000	0.0182 ± 0.0000
	CS	0.0193 ± 0.0000	0.0225 ± 0.0013	0.0258 ± 0.0019	0.0273 ± 0.0019	0.0333 ± 0.0017
corel5k	BR	0.0094 ± 0.0000	-	-	-	-
	PLST	0.0094 ± 0.0000	0.0094 ± 0.0000	0.0094 ± 0.0000	0.0094 ± 0.0000	0.0094 ± 0.0000
	CS	0.0094 ± 0.0000	0.0096 ± 0.0002	0.0108 ± 0.0009	0.0124 ± 0.0012	0.0176 ± 0.0011
bibtex	BR	0.0121 ± 0.0000	-	-	-	-
	PLST	0.0121 ± 0.0001	0.0122 ± 0.0001	0.0123 ± 0.0001	0.0124 ± 0.0001	0.0128 ± 0.0001
	CS	0.0297 ± 0.0055	0.0623 ± 0.0059	0.1048 ± 0.0076	0.1417 ± 0.0066	0.0913 ± 0.0029
mediamill	BR	0.0300 ± 0.0001	-	-	-	-
	PLST	0.0300 ± 0.0001	0.0300 ± 0.0001	0.0300 ± 0.0001	0.0300 ± 0.0001	0.0301 ± 0.0001
	CS	0.0569 ± 0.0125	0.0822 ± 0.0128	0.0987 ± 0.0068	0.1491 ± 0.0071	0.1062 ± 0.0033
yeast	BR	0.1992 ± 0.0021	-	-	-	-
	PLST	0.1992 ± 0.0021	0.1991 ± 0.0020	0.1993 ± 0.0020	0.2045 ± 0.0021	0.2278 ± 0.0015
	CS	0.2319 ± 0.0043	0.2854 ± 0.0040	0.3010 ± 0.0035	0.3194 ± 0.0026	0.3100 ± 0.0031
emotions	BR	0.3007 ± 0.0105	-	-	-	-
	PLST	0.3007 ± 0.0105	0.2999 ± 0.0124	0.2946 ± 0.0129	0.3235 ± 0.0136	0.3563 ± 0.0140
	CS	0.3047 ± 0.0093	0.3196 ± 0.0088	0.3385 ± 0.0071	0.3324 ± 0.0077	0.3421 ± 0.0086

Table 3. Computational Time of BR, PLST and CS at Optimal Reduction Size

data set	K	M^*	M^*/K (%)	BR (K) regression (sec)	PLST (M^*) regression (sec)	PLST (M^*) encode + decode (sec)	CS (M^*) regression (sec)	CS (M^*) encode + decode (sec)
delicious	983	131	13.3	3267.45	434.38	54.11	436.78	421.97
corel5k	373	19	5.1	405.55	21.38	2.02	20.60	1.16
bibtex	159	104	65.4	3597.10	2341.37	0.63	2349.80	257.57
mediamill	101	11	10.9	74.12	7.94	2.20	8.04	19.18
yeast	14	6	42.9	0.42	0.18	0.01	0.18	0.21
emotions	6	2	33.3	0.12	0.00	0.00	0.00	0.02

Table 4. Test Hamming Loss of BR, PLST and CS at Optimal Reduction Size

data set	K	M^*	M^*/K (%)	BR (K)	PLST (M^*)	CS (M^*)
delicious	983	131	13.3	0.01813 ± 0.00003	0.01819 ± 0.00003	0.03979 ± 0.00054
corel5k	373	19	5.1	0.00940 ± 0.00002	0.00943 ± 0.00002	0.02121 ± 0.00062
bibtex	159	104	65.4	0.01211 ± 0.00007	0.01226 ± 0.00007	0.10198 ± 0.00844
mediamill	101	11	10.9	0.03003 ± 0.00006	0.03015 ± 0.00006	0.08309 ± 0.00153
yeast	14	6	42.9	0.19916 ± 0.00211	0.20338 ± 0.00196	0.31173 ± 0.00280
emotions	6	2	33.3	0.30069 ± 0.01053	0.32347 ± 0.01356	0.33486 ± 0.01024