Large Margin Metric Learning for Multi-label Prediction

Weiwei Liu and **Ivor W. Tsang***
Center for Quantum Computation and Intelligent Systems
University of Technology, Sydney, Australia
liuweiwei863@gmail.com, ivor.tsang@uts.edu.au

Abstract

Canonical correlation analysis (CCA) and maximum margin output coding (MMOC) methods have shown promising results for multi-label prediction, where each instance is associated with multiple labels. However, these methods require an expensive decoding procedure to recover the multiple labels of each testing instance. The testing complexity becomes unacceptable when there are many labels. To avoid decoding completely, we present a novel large margin metric learning paradigm for multi-label prediction. In particular, the proposed method learns a distance metric to discover label dependency such that instances with very different multiple labels will be moved far away. To handle many labels, we present an accelerated proximal gradient procedure to speed up the learning process. Comprehensive experiments demonstrate that our proposed method is significantly faster than CCA and MMOC in terms of both training and testing complexities. Moreover, our method achieves superior prediction performance compared with state-of-the-art methods.

Introduction

Multi-label prediction has been applied to various applications ranging from document classification and gene function prediction, to automatic image annotation. Each instance is associated with multiple labels that is represented by a sparse label vector. For instance, a document can be associated with a range of topics, such as *News*, *Finance* and *Sports* (Schapire and Singer 2000); a gene belongs to the functions of *protein synthesis*, *metabolism* and *transcription* (Barutcuoglu, Schapire, and Troyanskaya 2006); an image may have both *beach* and *urban* tags (Boutell et al. 2004).

A simple approach to multi-label learning is binary relevance (BR) (Tsoumakas, Katakis, and Vlahavas 2010), which trains a binary classifier for each label independently. To deal with many labels, Hsu et al. (2009) assume that label vectors have a little support. In other words, each label vector can be projected into a lower dimensional compressed label space, which can be deemed as **encoding**. A regression is then learned for each compressed label. Lastly, the compressed sensing (CS) is used to **decode** the labels from the regression outputs of each testing instance.

Many works have recently been developed in this **encoding-decoding** paradigm. These works mainly use different projection methods to transform the original label space into another effective label space as encoding. For example, principal label space transformation (PLST) (Tai and Lin 2012) uses principal component analysis (PCA) to project the output labels into a lower dimension. Zhang and Schneider propose the use of canonical correlation analysis (CCA) (2011) and maximum margin output coding (MMOC) (2012) to encode the original label space. Learning models are then trained in the transformed label space. To accurately recover the labels from the prediction of the learning models, expensive decoding schemes are usually required. Due to the combinatorial nature of the label space, exact decoding is usually intractable. Even when approximate inference (Zhang and Schneider 2011) is used, the decoding step is still very time-consuming when there are many labels.

Motivation Comprehensive experiments on large scale image databases show that the k nearest neighbor (kNN) algorithm achieves superior performance when handling many class problems (Deng et al. 2010). Moreover, Kwok and Tsang (2003) and Weinberger and Saul (2009) show that the single-label prediction performance of kNN can be improved by learning a distance metric to satisfy the following constraint: Two nearby instances from different classes will be pushed further apart with a large margin. However, in a multi-label setting, nearby instances may differ by only a few labels, or hundreds. It is non-trivial to define a proper scaling of the margin for any two nearby instances. To learn a distance metric for multi-label problems, we require two conditions: 1) The output labels of an instance can be determined by that instance. 2) The output labels of an instance can be predicted by the nearest neighbors of that instance. To be specific, for a testing instance $x^{(a)}$, an instance $x^{(b)}$ from the training set is very similar to $x^{(a)}$ based on their embeddings, and their corresponding labels are $y^{(a)}$ and $y^{(b)}$ respectively. In the embedding space, $y^{(b)}$ and $x^{(b)}$ are similar, $x^{(b)}$ and $x^{(a)}$ are similar, $x^{(a)}$ and $y^{(a)}$ are also similar, so $y^{(b)}$ is similar to $y^{(a)}$. Hence, we can use the labels of the k nearest neighbors of $x^{(a)}$ to predict $y^{(a)}$.

*Corresponding author

To achieve our goal, we present a novel multi-label learning paradigm to project both the input and output into the same embedding space. The input and output can then be compared in the same space, and it is evident that the embeddings of the input and the corresponding output should be similar. Moreover, we enforce the constraint that the distance between the embedded input of $x^{(a)}$ and its correct output $y^{(a)}$ should be smaller than the distance between the embedded of $x^{(a)}$ and the output $y^{(b)}$ of the nearest neighbors of $x^{(a)}$ with at least a margin measured by $\Delta(y^{(a)}, y^{(b)})$, the difference between $y^{(a)}$ and $y^{(b)}$. Thus, two nearby instances from different output labels will be pushed further apart by $\Delta(y^{(a)}, y^{(b)})$.

The main contributions in this work are:

1. To incorporate the feature and label correlations, we project both the input and output to the same embedding space, in which the input and output can be compared. A large margin formulation with k nearest neighbor constraints is proposed to learn the embedding space. Lastly, we transform the formulation to metric learning (Kulis 2013; Yang and Jin 2006) for multi-label problems.
2. After transformation, our optimization problem is reduced to a semidefinite programming problem. To handle many labels, the accelerated proximal gradient (APG) method (Beck and Teboulle 2009; Toh and Yun 2009) is adapted to solve the reduced problem.
3. To avoid the expensive decoding step, we select k nearest neighbors from the training set for each testing instance in the embedding space and make a rapid prediction based on the labels of those k nearest neighbors.
4. Experiments on a number of real-world multi-label data sets demonstrate that our method outperforms state-of-the-art approaches and is more efficient than methods that are based on the expensive decoding step.

Related Work

Tsoumakas and Katakis (2007) group existing multi-label classification methods into two major categories: *algorithm adaptation* (AA) or *problem transformation* (PT). AA extends specific learning algorithms to deal with multi-label classification problems, while PT transforms the learning task into one or more single-label classification problems. Several methods (Zhang and Zhou 2006; 2007; Brinker and Hullermeier 2007) fall into the AA category. Amongst them, multi-label k nearest neighbor (ML-kNN) is one of most popular methods due to its simplicity and good experimental results (Zhang and Zhou 2007). However, similar to kNN, it suffers from the curse of dimensionality. When there are many noisy features on high dimensional problems, generalization performance drops tremendously. Our proposed method learns a distance that embeds both the input and output in the same embedding space, which prunes away noisy features. Therefore, our method is capable of finding more discriminative k nearest neighbors for prediction.

From the PT perspective, Hsu et al. (2009) use random transformation to project the original label space into a low dimensional label space. A regression model is trained on each transformed label. The compressed sensing technique is used to recover multi-labels from the regression output. To capture label dependency, CCA is used by Zhang and Schneider (2011) to encode the label vectors. The work that is most relevant to our method is MMOC (Zhang and Schneider 2012), which embeds both the output and input in the same space by ensuring that the distance between the embedded input and the correct embedded output is less than the distance between the embedded input and any other embedded output with a large margin. Though MMOC has shown improved prediction performance, the training process involves an exponential number of constraints w.r.t. the number of labels, which is impractical for real-world applications, such as image annotation. Since both CCA and MMOC apply mean-field approximation (Zhang and Schneider 2011) in the decoding step, their prediction process is still expensive when there are many labels. PLST (Tai and Lin 2012) uses PCA to project the output to a lower dimension and applies the linear projection in the decoding procedure; its performance is inferior to CCA and MMOC in our experiments.

Some approaches attempt to exploit the different orders (first-order, second-order and high-order) of label correlations (Zhang and Zhang 2010). For example, Kang, Jin, and Sukthankar (2006) explicitly exploit high-order correlation between labels, but this involves an optimization problem with an exponential number of constraints. All these methods assume that the correlations are shared by all instances, thus Huang and Zhou (2012) try to exploit label correlations in the data locally and measure the similarity between instances in the label space rather than in the feature space. However, the label space is usually sparse when there are many labels, making it impossible to obtain accurate similarity between instances through the measurement in the label space.

A comprehensive review of multi-label learning algorithms can be found in Zhang and Zhou (2014) and references therein.

Large Margin Metric Learning

Notations and Preliminaries

Assume $x^{(i)} \in \mathbb{R}^{p\times 1}$ is a real vector representing an input (instance), $y^{(i)} \in \{0,1\}^{q\times 1}$ is a real vector representing the corresponding output ($i \in \{1 \ldots n\}$). n denotes the number of training samples. The input matrix is $X \in \mathbb{R}^{n\times p}$ and the output matrix is $Y \in \{0,1\}^{n\times q}$. $Nei(i)$ is the output set of k nearest neighbors of input instance $x^{(i)}$. For output encoding, $V = (v_1, v_2, ..., v_d) \in \mathbb{R}^{q\times d}$ ($d < q$) is the projection matrix that maps each output vector $y^{(i)}$ (q dimension) to $V^T y^{(i)}$ (d dimension). Let $P \in \mathbb{R}^{p\times q}$ also be the projection matrix. For input encoding, each input vector $x^{(i)}$ (p dimension) is projected to $V^T P^T x^{(i)}$ (d dimension). Then $x^{(i)}$ and $y^{(i)}$ can be compared in the projection space (d dimension).

A simple linear regression model for BR is to learn the matrix P through the following formulation:

$$argmin_{P\in R^{p\times q}} \frac{1}{2}||P^T X^T - Y^T||_F^2 \qquad (1)$$

where $||\cdot||_F$ is the Frobenius norm. However, this does not consider the relationships between labels. To reduce the noise in the data set, MMOC (Zhang and Schneider 2012) incorporates the feature and label correlations and proposes a maximum margin output coding formulation Eq. (2) to learn the projection matrix.

$$\begin{aligned} & argmin_{V\in R^{q\times d},\{\xi_i\geq 0\}_{i=1}^n} \frac{1}{2}||V||_F^2 + \frac{C}{n}\sum_{i=1}^n \xi_i \\ & s.t. \quad ||V^T P^T x^{(i)} - V^T y^{(i)}||_2^2 + \Delta(y^{(i)}, y) - \xi_i \\ & \qquad \leq ||V^T P^T x^{(i)} - V^T y||_2^2, \forall y \in \{0,1\}^q, \forall i \end{aligned} \tag{2}$$

Eq. (2) involves an exponentially large number of constraints. To address the combinatorial nature of the label space $\{0,1\}^q$, Zhang and Schneider (2012) use the overgenerating technique (Finley and Joachims 2008) with the cutting plane method. Training involves solving a box-constrained quadratic programming (QP) problem for each training sample i, which is time-consuming, and testing also involves solving a QP on $\{0,1\}^q$ space. Even if approximate inference (Zhang and Schneider 2011) is used to solve this QP problem, it is still computationally expensive.

Proposed Formulation

Inspired by kNN and MMOC, we propose the following large margin metric learning with k nearest neighbor constraints, or LM-kNN for short, to learn the projection matrix.

If the encoding scheme works well, the distance between the codeword of $x^{(i)}$ ($V^T P^T x^{(i)}$) and the codeword of $y^{(i)}$ ($V^T y^{(i)}$) should tend to 0 and be less than the distance between the codeword of $x^{(i)}$ and the codeword of any other output ($V^T y$). Then, the following large margin formulation is presented to learn projection matrix V:

$$\begin{aligned} & argmin_{V\in R^{q\times d},\{\xi_i\geq 0\}_{i=1}^n} \frac{1}{2}||V||_F^2 + \frac{C}{n}\sum_{i=1}^n \xi_i^2 \\ & s.t. \quad ||V^T P^T x^{(i)} - V^T y^{(i)}||_2^2 + \Delta(y^{(i)}, y) - \xi_i \\ & \qquad \leq ||V^T P^T x^{(i)} - V^T y||_2^2, \forall y \in Nei(i), \forall i \end{aligned} \tag{3}$$

where C is a positive constant that controls the trade-off between square-hinge loss function and regularizer. The constraints in Eq. (3) guarantee that the distance between the codeword of $x^{(i)}$ and the codeword of $y^{(i)}$ is less than the distance between the codeword of $x^{(i)}$ and the codeword of any other output. To give Eq. (3) more robustness, we add the loss function $\Delta(y^{(i)}, y)$ as the margin and minus the slack variable ξ_i in the left side of the constraints. Given the distance function $||V^T P^T x^{(i)} - V^T y^{(i)}||_2^2$ as the compatibility function, Eq. (3) can be viewed as an adapted form of structural SVMs (Tsochantaridis et al. 2005). The loss function is defined as $\Delta(y^{(i)}, y) = ||y^{(i)} - y||_1$ (Zhang and Schneider 2012), where $||\cdot||_1$ means the l_1 norm. Following (Weinberger and Saul 2009), we use Euclidean metric to measure the distances between instances $x^{(i)}$ and $x^{(j)}$ and then learn a new distance metric, which improves the performance of kNN.

Define a $q \times q$ symmetric positive semidefinite matrix (denoted by S_q^+) Q: $Q = VV^T \in S_q^+$ and $\phi_{x^{(i)},y^{(i)}} = P^T x^{(i)} - y^{(i)}$. We can transform Eq. (3) to the following metric learning (Kulis 2013; Yang and Jin 2006) problem:

$$\begin{aligned} & argmin_{Q\in S_q^+,\{\xi_i\geq 0\}_{i=1}^n} \frac{1}{2}trace(Q) + \frac{C}{n}\sum_{i=1}^n \xi_i^2 \\ & s.t. \quad \phi_{x^{(i)},y^{(i)}}^T Q \phi_{x^{(i)},y^{(i)}} + \Delta(y^{(i)}, y) - \xi_i \\ & \qquad \leq \phi_{x^{(i)},y}^T Q \phi_{x^{(i)},y}, \forall y \in Nei(i), \forall i \end{aligned} \tag{4}$$

Accelerated Proximal Gradient Update

Since our objective of Eq. (4) is smooth and the number of constrains is linear w.r.t n, we can apply the accelerated proximal gradient (APG) method (Beck and Teboulle 2009; Toh and Yun 2009) to efficiently solve the primal form of Eq. (4) with many labels. Let $p(Q) = \frac{1}{2}trace(Q)$ and $f(Q) = \frac{C}{n}\sum_{i=1}^n \xi_i^2$ where $\xi_i = \max\{0, \max_{y\in Nei(i)}(\Delta(y^{(i)}, y) - (\phi_{x^{(i)},y}^T Q\phi_{x^{(i)},y} - \phi_{x^{(i)},y^{(i)}}^T Q\phi_{x^{(i)},y^{(i)}}))\}$. We define

$$F(Q) = f(Q) + p(Q), Q \in S_q^+ \tag{5}$$

The derivative of f is denoted by ∇f. Yuan, Ho, and Lin (2012) show that ∇f is Lipschitz continuous on Q. For any $Z \in S_q^+$, consider the following QP problem of $F(Q)$ at Z:

$$\begin{aligned} A_\tau(Q,Z) &= f(Z) + < \nabla f(Z), Q - Z > \\ &\quad + \frac{\tau}{2}\|Q - Z\|_F^2 + p(Q) \\ &= \frac{\tau}{2}\|Q - G\|_F^2 + p(Q) + f(Z) - \frac{1}{2\tau}\|\nabla f(Z)\|_F^2 \end{aligned} \tag{6}$$

where $\tau > 0$ is a constant and $G = Z - \frac{1}{\tau}\nabla f(Z)$. To minimize $A_\tau(Q,Z)$ w.r.t. Q, it is reduced to solve Eq. (7):

$$argmin_{Q\in S_q^+} \frac{\tau}{2}||Q - G||_F^2 + p(Q) \tag{7}$$

To solve Eq. (7), we take the derivative of the objective function of Eq. (7) w.r.t. Q: $\tau(Q - G) + \frac{1}{2}I = 0$, then $Q = G - \frac{1}{2\tau}I$. We take the SVD of G as $G = U\overline{G}U^T$, and $Q = U\overline{G}U^T - \frac{1}{2\tau}UU^T$, then $Q = U(\overline{G} - \frac{1}{2\tau}I)U^T$. We use 0 to replace the negative entries in $\overline{G} - \frac{1}{2\tau}I$. Lastly, we obtain the symmetric positive semidefinite matrix solution of Eq. (7), denoted by $S_\tau(G)$. The detailed APG algorithm is shown in Algorithm 1:

In Algorithm 1, let L_f be the Lipschitz constant of ∇f and L_f estimated as $L_f = 0.01nC$. The optimality condition of Eq. (4) is $\nabla F(Q) = 0$. It is usually time-consuming to achieve this condition. In practice, we meet an ϵ-accurate solution instead. The stopping condition of Algorithm 1 is set as:

$$\frac{F(Q^\kappa) - F(Q^{\kappa+1})}{F(Q^\kappa)} \leq \epsilon \tag{8}$$

where ϵ is a small tolerance value. In practice, we set $\epsilon = 0.001$. Lastly, a sublinear convergence rate of algorithm 1 is guaranteed in the following theorem.

Algorithm 1 Accelerated Proximal Gradient Algorithm for Solving Eq. (4)

```
Input: $\eta \in (0,1)$ is a constant. Choose $Q^0 = Q^{-1} \in S_q^+$.
    $t^0 = t^{-1} = 1$ and $\kappa = 0$. Choose the Lipschitz constant
    $L_f$ and set $\tau_0 = L_f$
Output: The optimal solution to Eq. (4)
 1: Set $Z^\kappa = Q^\kappa + \frac{t^{\kappa-1}-1}{t^\kappa}(Q^\kappa - Q^{\kappa-1})$
 2: Set $\tau = \eta\tau_\kappa$
 3: for $j = 0, 1, 2, \ldots,$ do
 4:     Set $G = Z^\kappa - \frac{1}{\tau}\nabla f(Z^\kappa)$, compute $S_\tau(G)$
 5:     if $F(S_\tau(G)) \le A_\tau(S_\tau(G), Z^\kappa)$, then
 6:         set $\tau_\kappa = \tau$, stop
 7:     else
 8:         $\tau = \frac{1}{\eta}\tau$
 9:     end if
10: end for
11: Set $Q^{\kappa+1} = S_\tau(G)$
12: Compute $t^{\kappa+1} = \frac{1+\sqrt{1+4(t^\kappa)^2}}{2}$. Let $\kappa = \kappa + 1$
13: Quit if stopping condition is achieved. Otherwise, go to
    step 1
```

Theorem 1 *Let $\{Q^\kappa\}$ be the sequences generated by Algorithm 1 and L_f be the Lipschitz constant of ∇f. Then for any $\kappa \ge 1$, we have*

$$F(Q^\kappa) - F(Q^*) \le \frac{2L_f||Q^0 - Q^*||_F^2}{\eta(\kappa+1)^2} \tag{9}$$

where $Q^* = argmin_Q F(Q)$

The proof can be adapted from Beck and Teboulle (2009).

Prediction

Traditionally, encoding-decoding methods involve the decoding process which usually requires solving QP problem on a combinatorial space. It is computationally expensive. Inspired by metric learning (Kulis 2013), we select k nearest neighbors from the training set for each testing instance in the embedding space, and conduct prediction based on the codeword distance with a testing instance and the labels of k nearest neighbors. For a new testing input variable x, we find k instances $\{x^{(1)}, \ldots, x^{(k)}\}$ in the training set which have a smaller codeword distance from x than other instances. Based on the codeword distance with x and output of $\{x^{(1)}, \ldots, x^{(k)}\}$, we compute the scores for each label for x. Lastly, we make the prediction for multi-label classification problems based on the scores. The equation of the distance between the codeword of x and the codeword of $x^{(i)}$ is $||\hat{M}(x) - \hat{M}(x^{(i)})||_2^2$. It can be computed as $(P^Tx - P^Tx^{(i)})^TQ(P^Tx - P^Tx^{(i)})$. Thus, the testing time is similar to kNN and much faster than the encoding-decoding methods. Following the setting in MMOC, we set 0.5 as the threshold without further optimization.

Complexity Analysis

Training time The formulation of MMOC involves an exponential number of constraints. The authors therefore use the overgenerating technique with the cutting plane method. Lastly, training involves solving a box-constrained QP problem for each training instance and then using CVX[1] to solve a semidefinite programming problem. The time complexity of QP is at least $\mathcal{O}(q^3)$, and from Wahba (1999), the training time complexity of MMOC is $\mathcal{O}(nq^3 + n^4)$ for each iteration at least. While the training time of our method (LM-kNN) is dominated by the APG algorithm. To achieve an ϵ-solution, the number of iterations needed by APG update is $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$. The time complexity for each iteration is $\mathcal{O}(q^3 + knpq^2)$.

Table 1: Time Complexity

Method	Training Time	Testing Time
BR	$\mathcal{O}(npq)$	$\mathcal{O}(pq)$
PLST	$\mathcal{O}(n^3 + q^2n + npq)$	$\mathcal{O}(q^2 + pq)$
CCA	$\mathcal{O}(\psi^3 + n(p^2 + q^2 + pq))$	$\mathcal{O}(q^3)$
MMOC	$\mathcal{O}(nq^3 + n^4)$	$\mathcal{O}(q^3)$
kNN	-	$\mathcal{O}(pn)$
ML-kNN	$\mathcal{O}(n^2p + nq)$	$\mathcal{O}(pn + q)$
LM-kNN	$\mathcal{O}(q^3 + knpq^2)$	$\mathcal{O}(qn + pq)$

Testing time We analyze the testing time for each testing instance. Both the testing time of CCA and MMOC involve solving QP on $\{0,1\}^q$ space. It is combinatorial in nature and intractable, thus mean-field approximation is used to obtain the approximated solution iteratively. The time complexity for each iteration is $\mathcal{O}(q^2)$, but it takes many times to converge. The time complexity is $\mathcal{O}(q^3)$ at least. The training and testing time complexity for the methods that are used in this paper is presented in Table 1. Let $\psi = \max\{n, p, q\}$.

Experiment

In this section, we evaluate the performance of our proposed Large Margin kNN (LM-kNN) for multi-label prediction. All the methods compared are implemented in MatLab. All experiments are conducted on a workstation with a 3.4GHZ Intel CPU and 32GB main memory running Linux platform.

Experimental Setup

Data Sets We conduct experiments on a variety of real-world data sets from different domains[2] (Table 2).

- scene (Boutell et al. 2004): Collects images of outdoor scenes.
- cal500 (Turnbull et al. 2008): Contains songs by different artists. Each song is labeled by 174 tags representing genres, instruments, emotions, and other related concepts.
- corel5k (Duygulu et al. 2002): Contains images from Stock Photo CDs. In total, there are 374 labels.
- delicious (Tsoumakas, Katakis, and Vlahavas 2008): Contains textual data of web pages along with 983 tags extracted from the del.icio.us social book marking site.
- Eur-Lex (Mencía and Fürnkranz 2008): Collects documents on European Union law. There are several EuroVoc

[1] http://cvxr.com/cvx/

[2] http://mulan.sourceforge.net

Table 2: Data Sets

Data Set	♯ Instances	♯ Features	♯ Labels
scene	2,407	294	6
cal500	502	68	174
corel5k	5,000	499	374
delicious	16,105	500	983
EUR-Lex (dc)	19,348	5,000	412
EUR-Lex (ed)	19,348	5,000	3,993

descriptors, directory codes and types of subject matter to describe the labels. Here, we use two of them which have more labels.

Baseline Methods We compare our LM-kNN with several state-of-the-art multi-label prediction methods:

- BR (Tsoumakas, Katakis, and Vlahavas 2010).
- PLST (Tai and Lin 2012): Uses principal component analysis (PCA) for encoding, and rounding for decoding.
- CCA (Zhang and Schneider 2011): Uses canonical correlation analysis to encode the label vectors, and mean-field approximation is applied in the decoding step.
- MMOC (Zhang and Schneider 2012): Adapts a maximum margin criterion to learn output coding for multi-labels. Its decoding scheme is the same as CCA.
- kNN: We adapt the k nearest neighbor (kNN) algorithm to solve multi-label classification problems. Euclidean metric is used to measure the distances between instances.
- ML-kNN (Zhang and Zhou 2007): Based on kNN, the maximum posteriori principle is used by this method to determine the labels of the testing instance.

For BR, we use linear classification/regression package LIBLINEAR (Fan et al. 2008) with L2-regularized logistic regression (primal) to train the classifier. As with the experimental settings in Zhang and Schneider (2012), the number of output projections d is set to the number of original labels q ($d = q$) for PLST, CCA and MMOC and the decoding parameter is set as $\lambda = 1$ for CCA and MMOC. In our experiment, we find that the performance of kNN or ML-kNN have no significant difference on most datasets with varying k. Following the setting in (Zhang and Zhou 2007), we set $k = 10$ for kNN, ML-kNN and our method. $\eta = 0.4$ is set for the APG algorithm and $C = 10$ is set for MMOC and our method. We set the stopping condition threshold as $\epsilon = 0.01$ for EUR-Lex (ed) data set. Besides PLST (Tai and Lin 2012), CCA (Zhang and Schneider 2011) and MMOC (Zhang and Schneider 2012) have also been shown to outperform the original compressed-sensing-based method (Hsu et al. 2009), thus we make no comparison with Hsu et al. (2009) in this paper.

Performance Measurements To fairly measure the performance of our method and baseline methods, we consider the following evaluation measurements (Mao, Tsang, and Gao 2013):

- Micro-F1: computes true positives, true negatives, false positives and false negatives over labels, and then calculates an overall F-1 score.

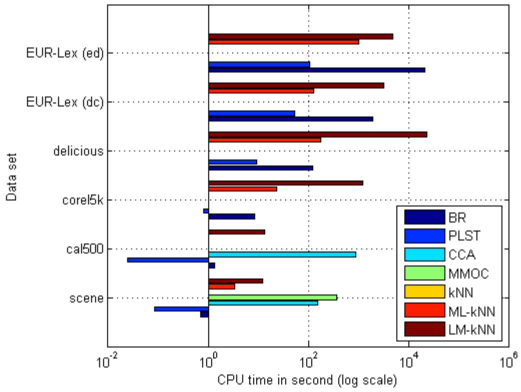

Figure 1: The training time

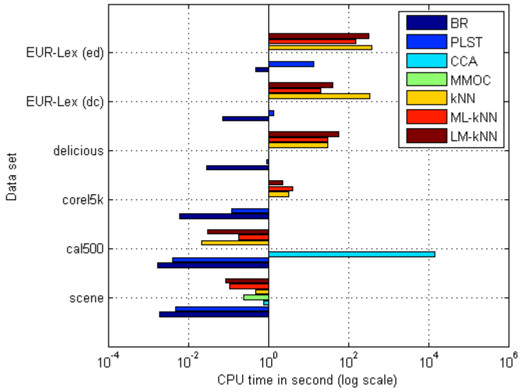

Figure 2: The testing time

- Example-F1: computes the F-1 score for all the labels of each testing sample and takes the average over those samples.

Since Mao, Tsang, and Gao (2013) have shown that Macro-F1 is sensitive to the rareness of labels and hamming loss is not a proper measure for multi-label problems with many labels, we do not consider those two measurements here. We perform 10-fold cross-validation on each data set and report the mean and standard error of each evaluation measurement.

Prediction Performance

Tables 3 and 4 list the two measurement results for our method and baseline approaches in respect of the different data sets. Recall that CCA and MMOC are very computationally expensive, especially for solving the QP problem on $\{0, 1\}^q$ space which is combinatorial in nature and intractable during the decoding step, so they cannot be run on most of the larger data sets. In our experiment, the decoding procedure on corel5k data set already takes more than two days for CCA. Because MMOC has to solve a box-constrained QP for each training sample and use CVX to tackle optimization problems during the training step, it runs

Table 3: The results of Micro-F1 on the various data sets (mean ± standard deviation). The best ones are in bold.

Data Set	BR	PLST	CCA	MMOC	kNN	ML-kNN	LM-kNN
scene	0.6911 ± 0.0259	0.5924 ± 0.0326	0.7281 ± 0.0286	0.7297 ± 0.0287	0.7221 ± 0.0193	**0.7387** ± 0.0278	0.7300 ± 0.0301
cal500	0.3448 ± 0.0161	0.3032 ± 0.0095	0.3533 ± 0.0207	-	**0.3593** ± 0.0127	0.3184 ± 0.0162	0.3542 ± 0.0157
corel5k	0.0956 ± 0.0260	0.0801 ± 0.0081	-	-	0.0321 ± 0.0075	0.0278 ± 0.0117	**0.1670** ± 0.0137
delicious	0.2598 ± 0.0052	0.1423 ± 0.0058	-	-	0.2154 ± 0.0047	0.1738 ± 0.0047	**0.3104** ± 0.0058
EUR-Lex (dc)	0.2673 ± 0.0709	0.2304 ± 0.0269	-	-	0.6540 ± 0.0088	0.6252 ± 0.0071	**0.7207** ± 0.0063
EUR-Lex (ed)	0.0374 ± 0.0015	0.1059 ± 0.0115	-	-	0.4011 ± 0.0061	0.3489 ± 0.0055	**0.4344** ± 0.0051

Table 4: The results of Example-F1 on the various data sets (mean ± standard deviation). The best ones are in bold.

Data Set	BR	PLST	CCA	MMOC	kNN	ML-kNN	LM-kNN
scene	0.6169 ± 0.0329	0.4588 ± 0.0346	0.7320 ± 0.0287	**0.7338** ± 0.0294	0.6815 ± 0.0174	0.6874 ± 0.0311	0.7101 ± 0.0306
cal500	0.3446 ± 0.0150	0.3062 ± 0.0108	0.3516 ± 0.0197	-	**0.3561** ± 0.0131	0.3216 ± 0.0180	0.3511 ± 0.0161
corel5k	0.0781 ± 0.0166	0.0587 ± 0.0054	-	-	0.0223 ± 0.0056	0.0178 ± 0.0069	**0.1295** ± 0.0092
delicious	0.2174 ± 0.0047	0.1131 ± 0.0039	-	-	0.1878 ± 0.0046	0.1518 ± 0.0044	**0.2553** ± 0.0043
EUR-Lex (dc)	0.2556 ± 0.0669	0.1530 ± 0.0201	-	-	0.5741 ± 0.0106	0.5496 ± 0.0084	**0.6812** ± 0.0072
EUR-Lex (ed)	0.0370 ± 0.0015	0.0725 ± 0.0083	-	-	0.3409 ± 0.0057	0.3005 ± 0.0049	**0.3818** ± 0.0045

out of memory even for the cal500 data set with 68 features and 174 labels. From the results, we can see that:

- CCA or MMOC's performance is comparable to the best results on scene and cal500. This means that CCA and MMOC are most successful in small data sets with few labels, but cannot deal with larger data sets with many labels.
- In our experiment, PLST generally underperforms. Thus, PLST with simple linear projection in encoding and decoding procedure is not effective.
- BR is much inferior to kNN, ML-kNN and LM-kNN on much larger data sets EUR-Lex (dc) and EUR-Lex (ed). BR does not consider the distributions and relationships between labels, so it achieves much lower accuracy on many labels data sets.
- Zhang and Zhou (2007) do not compare ML-kNN with kNN. Our empirical results show that kNN usually outperforms ML-kNN. ML-kNN estimates the prior and posterior probabilities from k nearest neighbors in the training set based on frequency counting. However, k is usually small, which leads to large estimation error.
- kNN and LM-kNN are most successful on all data sets. However, LM-kNN outperforms kNN because kNN is sensitive to noisy data.

Training Time and Testing Time

In this section, we compare the time performance of the various methods used in the experiment. Figures 1 and 2 report the once training and testing times of 10-fold cross-validation for our method and baseline approaches in terms of different data sets respectively. The results illustrate that PLST is fast in terms of training and testing time, which is consistent with the empirical results in Tai and Lin (2012), however, it is not effective in general. BR is fast in terms of testing time, however, it has slow training time and underperforms on the much larger data sets EUR-Lex (dc) and EUR-Lex (ed). Our method LM-kNN is almost 10 times faster than CCA and MMOC in terms of both the training and testing times even on smaller data sets, and is comparable to kNN and ML-kNN in terms of testing time.

Conclusion

To achieve the better performance than BR, Zhang and Schneider (2012) proposed MMOC to incorporate the feature or label correlations. However, MMOC has to deal with the optimization problem with an exponentially large number of constraints. Even if the overgenerating technique with the cutting plane method is used to solve this problem, it is still computationally very expensive. Inspired by kNN and MMOC, we propose the large margin formulation with k nearest neighbors constraints to solve the projection matrix which reduces the number of constraints from $\mathcal{O}(n2^q)$ to $\mathcal{O}(nk)$ and is more robust than kNN. Our problem is then transformed to a metric learning problem. To handle large scale applications, the accelerated proximal gradient (APG) method is adapted to solve the reduced semidefinite programming problem. Lastly, instead of performing expensive combinatorial optimization or approximate inference for the decoding procedure, we simply adopt the kNN strategy, which selects k nearest neighbors from the training set for each testing instance in the projection space and makes rapid prediction based on the labels of those k nearest neighbors. Overall, extensive experiments on a number of real-world multi-label data sets demonstrate that our method outperforms state-of-the-art approaches for accuracy. For scalability, our method is almost 10 times faster than CCA and MMOC in terms of both the training and testing times, and is comparable to kNN and ML-kNN in terms of testing time. For faster multi-label prediction, we will study the incorporation of the cover tree technique (Beygelzimer, Kakade, and Langford 2006) into our framework to speed up the batch-model kNN search.

Acknowledgments

This research was supported by the Australian Research Council Future Fellowship FT130100746.

References

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

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences* 2(1):183–202.

Beygelzimer, A.; Kakade, S.; and Langford, J. 2006. Cover trees for nearest neighbor. In *ICML*.

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

Brinker, K., and Hullermeier, E. 2007. Case-based multilabel ranking. In *IJCAI*, 702–707.

Deng, J.; Berg, A. C.; Li, K.; and Fei-Fei, L. 2010. What does classifying more than 10,000 image categories tell us? In *ECCV*.

Duygulu, P.; Barnard, K.; Freitas, J. F. G. d.; and Forsyth, D. A. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, 97–112.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Finley, T., and Joachims, T. 2008. Training structural svms when exact inference is intractable. In *ICML*, 304–311.

Hsu, D.; Kakade, S.; Langford, J.; and Zhang, T. 2009. Multi-label prediction via compressed sensing. In *NIPS*, 772–780.

Huang, S.-J., and Zhou, Z.-H. 2012. Multi-label learning by exploiting label correlations locally. In *AAAI*.

Kang, F.; Jin, R.; and Sukthankar, R. 2006. Correlated label propagation with application to multi-label learning. In *CVPR*, 1719–1726.

Kulis, B. 2013. Metric learning: A survey. *Foundations and Trends in Machine Learning* 5(4):287–364.

Kwok, J., and Tsang, I. W. 2003. Learning with idealized kernels. In *ICML*, 400–407.

Mao, Q.; Tsang, I. W.-H.; and Gao, S. 2013. Objective-guided image annotation. *IEEE Transactions on Image Processing* 22(4):1585–1597.

Mencía, E. L., and Fürnkranz, J. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML/PKDD*, 50–65.

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

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

Toh, K.-c., and Yun, S. 2009. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Technical report.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6:1453–1484.

Tsoumakas, G., and Katakis, I. 2007. Multi label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3):1–13.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. P. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2010. Mining multi-label data. In Maimon, O., and Rokach, L., eds., *Data Mining and Knowledge Discovery Handbook*. Springer US. 667–685.

Turnbull, D.; Barrington, L.; Torres, D.; and Lanckriet, G. 2008. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing* 16(2):467–476.

Wahba, G. 1999. Advances in kernel methods. Cambridge, MA, USA: MIT Press. chapter Support Vector Machines, Reproducing Kernel Hilbert Spaces, and Randomized GACV, 69–88.

Weinberger, K. Q., and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine* 10:207–244.

Yang, L., and Jin, R. 2006. Distance metric learning: A comprehensive survey. *Michigan State Universiy* 2.

Yuan, G.-X.; Ho, C.-H.; and Lin, C.-J. 2012. An improved glmnet for l1-regularized logistic regression. *Journal of Machine Learning Research* 13:1999–2030.

Zhang, Y., and Schneider, J. G. 2011. Multi-label output codes using canonical correlation analysis. In *AISTATS*, 873–882.

Zhang, Y., and Schneider, J. G. 2012. Maximum margin output coding. In *ICML*.

Zhang, M.-L., and Zhang, K. 2010. Multi-label learning by exploiting label dependency. In *KDD*, 999–1008.

Zhang, M.-L., and Zhou, Z.-H. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering* 18(10):1338–1351.

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

Zhang, M.-L., and Zhou, Z.-H. 2014. A review on multi-label learning algorithms. *IEEE Trans. Knowledge and Data Engineering* 26(8):1819–1837.