國立臺灣大學電機資訊學院資訊工程學系
碩士論文
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

動態特徵投影應用在成本導向線上多標籤分類問題
Dynamic Principal Projection for Cost-sensitive Online Multi-label Classification

朱鴻敏
Hong-Min Chu

指導教授：林軒田博士
Advisor: Hsuan-Tien Lin, Ph.D.

中華民國 106 年 6 月
June, 2017

doi:10.6342/NTU201701149

國立臺灣大學碩士學位論文
口試委員會審定書

動態特徵投影應用在成本導向線上多標籤分類問題

Dynamic Principal Projection for Cost-sensitive Multi-label Classification

本論文係朱鴻敏君（學號 R04922031）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 106 年 6 月 5 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

林軒田

（指導教授）

陳縕儂　　王鈺強

系主任　趙坤茂

誌謝

感謝林軒田老師這三年多來的指導。從我一無所知的加入 CLLab 開始，一直到現在完成這篇碩士論文，一路上來老師給了我許許多多的幫助。不管是研究上的討論，還是身為研究者的品格的言傳身教，我從老師身上學到了非常多。

感謝口試委員陳縕儂教授以及王鈺強教授撥冗參加我的口試，並給予我在論文寫作上非常有用的建議。

感謝 CLLab 的成員們，與他們的討論使我在研究上獲益良多，不管是新的想法方面還是新的研究趨勢方面。特別感謝一同與我口試的夥伴鄒侑霖，有了他的幫忙，讓這次口試能夠順利的完成。

感謝我的家人，在我做研究的期間能夠無條件的支持我，使我能夠順利得完成這篇論文。

最後，感謝阿拉，祐我順利的度過人生重要的一個階段，拿到碩士學位。

朱鴻敏　謹識

2017.06.27

doi:10.6342/NTU201701149

摘要

本論文研究三個重要且實際的議題：線上更新，標籤空間維度下降，以及成本導向性，在多標籤分類問題上。目前的多標籤分類問題演算法並未被設計來同時處理這三個議題。在本論文中，我們提出了一個創新的演算法，成本導向動態特徵投影，來同時解決這三個議題。本方法是基於一個將領先的標籤空間維度下降演算法利用線上主成份分析延伸到線上更新的框架。詳細的說，本方法使用矩陣隨機梯度下降法作為處理線上主成份分析問題的方法，並在與精心設計得線上回歸學習者結合時建立其理論骨幹。另外，本方法將成本資訊嵌入標籤權重之中以達有理論保證的成本導向性。我們也研究了本方法的實際改進以提高效率。實驗結果表明，本方法在不同的評估標準上達到比現有的多標籤分類演算法更好的實際表現，也證明了同時解決這三個問題的重要性。

doi:10.6342/NTU201701149

Abstract

We study multi-label classification (MLC) with three important real-world issues: online updating, label space dimensional reduction (LSDR), and cost-sensitivity. Current MLC algorithms have not been designed to address these three issues simultaneously. In this paper, we propose a novel algorithm, cost-sensitive dynamic principal projection (CS-DPP) that resolves all three issues. The foundation of CS-DPP is a framework that extends a leading LSDR algorithm to online updating with online principal component analysis (PCA). In-particular, CS-DPP investigates the use of matrix stochastic gradient as the online PCA solver, and establishes its theoretical backbone when coupled with a carefully-designed online regression learner. In addition, CS-DPP embeds the cost information into label weights to achieve cost-sensitivity along with theoretical guarantees. Practical enhancements of CS-DPP are also studied to improve its efficiency. Experimental results verify that CS-DPP achieves better practical performance than current MLC algorithms across different evaluation criteria, and demonstrate the importance of resolving the three issues simultaneously.

doi:10.6342/NTU201701149

Contents

誌謝 i

摘要 ii

Abstract iii

1 Introduction 1

2 Preliminaries and Related Work 4

3 Dynamic Principal Projection 7

3.1 Principal Label Space Transformation 7

3.2 Online PCA 8

3.3 Proposed Approach 9

3.4 Practical Variant and Implementation 12

4 Cost-Sensitive Extension 14

5 Experiments 17

5.1 Experiments Setup 17

5.2 Necessity of LSDR 18

5.3 Experiments on Basis Drifting 19

5.4 Experiments on Cost-Sensitivity 20

6 Conclusion 23

doi:10.6342/NTU201701149

A **24**

A.1 Proof of Theorem 2 24

A.2 Proof of Lemma 3 27

A.3 Proof of Lemma 4 28

A.4 Proof of Theorem 5 29

A.5 Details of Experiments 31

A.5.1 Datasets and Parameters 31

A.5.2 Necessity of LSDR 31

A.5.3 Experiments on Basis Drifting 32

A.5.4 Experiments on Cost-sensitivity 33

Bibliography **36**

doi:10.6342/NTU201701149

List of Figures

5.1 DPP vs. O-BR on noisy labels 18
5.2 PBC vs. PBT vs. None 20
5.3 CS-DPP vs. Others 21

doi:10.6342/NTU201701149

List of Tables

5.1 DPP vs. O-BR on Large Dataset 18

A.1 Statistics of datasets 31

A.2 DPP vs. O-BR on Noisy Data, Hamming loss 32

A.3 DPP vs. O-BR on Noisy Data, F1 loss 32

A.4 DPP vs. O-BR on Noisy Data, Accuracy loss 33

A.5 DPP vs. O-BR on Noisy Data, Normalized rank loss 33

A.6 CS-DPP with PBC vs. PBT vs. None, Hamming loss 34

A.7 CS-DPP with PBC vs. PBT vs. None, F1 loss 34

A.8 CS-DPP with PBC vs. PBT vs. None, Accuracy loss 34

A.9 CS-DPP with PBC vs. PBT vs. None, Normalized rank loss 34

A.10 CS-DPP vs. others, Hamming loss 34

A.11 CS-DPP vs. others, F1 loss 34

A.12 CS-DPP vs. others, Accuracy loss 35

A.13 CS-DPP vs. others, Normalized rank loss 35

doi:10.6342/NTU201701149

Chapter 1

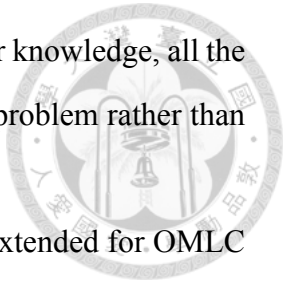
Introduction

The multi-label classification (MLC) problem allows each instance to be associated with a set of labels. The MLC problem reflects the nature of different real-world applications [8, 4, 12]. Traditional MLC algorithms mainly consider the batch MLC problem, where the input data are presented in a batch [22, 25]. Nevertheless, in many MLC applications such as e-mail categorization [20], multi-label examples arrive as a stream, which requires online analysis, as algorithms for batch MLC may not be suitable because of the potentially infinite amount of data. The need of such applications can be formalized as the online MLC (OMLC) problem.

The OMLC problem is generally more challenging than the batch one, and many mature algorithms for the batch problem have not yet been carefully extended to OMLC. Label space dimension reduction (LSDR) is a family of mature algorithms for the batch MLC problem [7, 13, 17, 24, 14, 23, 31, 6, 2, 5]. By viewing the label set of each instance as a high-dimensional label vector in a label space, LSDR encodes each label vector as a code vector in a lower-dimensional code space, and learns a predictor within the code space. An unseen instance is predicted by coupling the predictor with a decoder from the code space to the label space. For example, compressed sensing (CS) [13] encodes using random projections, and decodes with sparse vector reconstruction; principal label space transformation (PLST) [24] encodes by projecting to the key eigenvectors of the known label vectors obtained from principal component analysis (PCA), and decodes by reconstruction with the same eigenvectors. This low-dimensional encoding allows LSDR

doi:10.6342/NTU201701149

algorithms to exploit the key joint information between labels to be more robust to noise and be more effective on learning [24]. Nevertheless, to the best of our knowledge, all the LSDR algorithms mentioned above are designed for the batch MLC problem rather than the OMLC one.



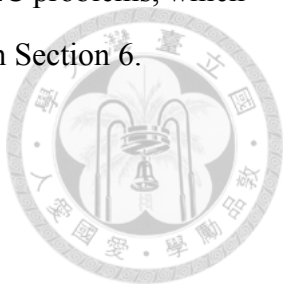
Another family of MLC algorithms that have not been carefully extended for OMLC contains the cost-sensitive MLC algorithms. In particular, different MLC applications usually come with different evaluation criteria (costs) that reflect their realistic needs. It is important to design MLC algorithms that are cost-sensitive to systematically cope with different costs, because an MLC algorithm that targets one specific cost may not always perform well under other costs [15]. Two representative cost-sensitive MLC algorithms are probabilistic classifier chain (PCC) [10] and condensed filter tree (CFT) [15]. PCC estimates the conditional probability with the classifier chain (CC) method [22] and makes Bayes-optimal predictions with respect to the given cost based on the estimations; CFT decomposes the cost into instance weights when training the classifiers in CC. Both algorithms, again, are designed for the batch MLC problem rather than the OMLC one.

From the discussions above, there is currently no algorithm that considers the three realistic needs of online updating, label space dimension reduction, and cost-sensitivity at the same time. The goal of this work is to study such algorithms. We first formalize the OMLC and cost-sensitive OMLC (CSOMLC) problems in Section 2 and discuss related work. We then extend LSDR for the OMLC problem and propose a novel online LSDR algorithm, dynamic principal projection (DPP), by connecting PLST with online PCA. In particular, we derive the DPP algorithm in Section 3 along with its theoretical guarantees, and resolve the issue of possible basis drifting caused by online PCA. Practical enhancements of DPP are also studied to improve its efficiency.

In Section 4, we further extend DPP to cost-sensitive DPP (CS-DPP) to fully match the needs of CSOMLC with a label-weighting scheme inspired by CFT. Extensive empirical studies demonstrate the strength of CS-DPP in addressing the three realistic needs in Section 5. In particular, we justify the necessity of considering LSDR, basis drifting and cost-sensitivity under the CSOMLC setting. The results show that CS-DPP signifi-

doi:10.6342/NTU201701149

cantly outperforms other OMLC competitors across different CSOMLC problems, which validates the robustness and effectiveness of CS-DPP, as concluded in Section 6.



doi:10.6342/NTU201701149

Chapter 2

Preliminaries and Related Work

For the MLC problem, we denote the feature vector of an instance as $\mathbf{x} \in \mathbb{R}^d$ and its corresponding label vector as $\mathbf{y} \in \mathcal{Y} \equiv \{+1, -1\}^K$, where $\mathbf{y}[k] = +1$ iff the instance is associated with the k-th label out of a total of K possible labels. We let $\mathbf{y}[k] \in \{+1, -1\}$ to conform with the common setting of online binary classification [9, 27], which is equivalent to another scheme, $\mathbf{y}[k] \in \{1, 0\}$, used in other MLC works [15, 22].

Traditional MLC methods consider the batch setting, where a training dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ is given at once, and the objective is to learn a classifier $g\colon \mathbb{R}^d \to \{+1, -1\}^K$ from $\mathcal{D}$ with the hope that $\hat{\mathbf{y}} = g(\mathbf{x})$ accurately predicts ground truth $\mathbf{y}$ with respect to an unseen $\mathbf{x}$. In this work, we focus on the OMLC setting, which assumes that instance $(\mathbf{x}_t, \mathbf{y}_t)$ arrives in sequence from a data stream. Whenever an $\mathbf{x}_t$ arrives at iteration t, the OMLC algorithm is required to make a prediction $\hat{\mathbf{y}}_t = g_t(\mathbf{x}_t)$ based on the current classifier g_t and feature vector $\mathbf{x}_t$. The ground truth $\mathbf{y}_t$ with respect to $\mathbf{x}_t$ is then revealed, and the penalty of $\hat{\mathbf{y}}_t$ is evaluated against $\mathbf{y}_t$.

Many evaluation criteria for comparing $\mathbf{y}$ and $\hat{\mathbf{y}}$ have been considered in the literature to satisfy different application needs. A simple criterion [25] is the Hamming loss $c_{\text{ham}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K} \sum_{k=1}^K [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!]$. The Hamming loss separately considers each label as equally important. In addition to the Hamming loss, there are other criteria that jointly evaluate all labels in $\hat{\mathbf{y}}$, such as

F1 loss $c_{\text{f}}(\mathbf{y}, \hat{\mathbf{y}}) = 1 - 2 \left(\sum_{k=1}^K [\![\mathbf{y}[k] = +1 \text{ and } \hat{\mathbf{y}}[k] = +1]\!] \right) / \left(\sum_{k=1}^K ([\![\mathbf{y}[k] = +1]\!] + [\![\hat{\mathbf{y}}[k] = +1]\!]) \right)$

doi:10.6342/NTU201701149

Accuracy loss $c_{\text{acc}}(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \left(\sum_{k=1}^{K} [\![\mathbf{y}[k] = +1 \text{ and } \hat{\mathbf{y}}[k] = +1]\!] \right) / \left(\sum_{k=1}^{K} [\![\mathbf{y}[k] = +1 \text{ or } \hat{\mathbf{y}}[k] = +1]\!] \right)$

Normalized Rank loss $c_{\text{nr}}(\mathbf{y}, \hat{\mathbf{y}}) = \underset{\mathbf{y}[i] > \mathbf{y}[j]}{\text{average}} \left([\![\hat{\mathbf{y}}[i] < \hat{\mathbf{y}}[j]]\!] + \frac{1}{2} [\![\hat{\mathbf{y}}[i] = \hat{\mathbf{y}}[j]]\!] \right)$

In this work, we follow existing cost-sensitive MLC approaches [15] to extend OMLC to the cost-sensitive OMLC (CSOMLC) setting, which further takes the evaluation criterion as an additional input to the learning algorithm. We call the criterion a *cost function* and overload $c\colon \{+1, -1\}^K \times \{+1, -1\}^K \to \mathbb{R}$ as its notation. The cost function evaluates the penalty of $\hat{\mathbf{y}}$ against $\mathbf{y}$ by $c(\mathbf{y}, \hat{\mathbf{y}})$, and includes the four loss functions discussed above. We naturally assume that $c(\cdot, \cdot)$ satisfies $c(\mathbf{y}, \mathbf{y}) = 0$ and $\max_{\hat{\mathbf{y}}} c(\mathbf{y}, \hat{\mathbf{y}}) \leq 1$. Given additional input, the CSOMLC algorithm shall behave differently when fed with different cost functions. In particular, the objective of a CSOMLC algorithm is to adaptively learn a classifier $g_t\colon \mathbb{R}^d \to \{+1, -1\}^K$ based on not only the data stream but also the given cost function c such that the cumulative cost $\sum_{t=1}^{T} c(\mathbf{y}_t, \hat{\mathbf{y}}_t)$ with $\hat{\mathbf{y}}_t = g_t(\mathbf{x}_t)$ over T iterations of $(\mathbf{x}_t, \mathbf{y}_t)$ can be minimized.

Several OMLC algorithms have been studied in the literature, including online binary relevance [21], Bayesian OMLC framework [32], and the multi-window approach using k nearest neighbors [30]. However, none of them are cost-sensitive. That is, they cannot take the cost function into account to improve learning performance.

Cost-sensitive MLC algorithms have also been investigated in the literature. Cost-sensitive RAkEL [18] and progressive RAkEL [29] are two algorithms that generalize a famous batch MLC algorithm called RAkEL [26] to cost-sensitive learning. The former achieves cost-sensitivity for any weighted Hamming loss, and the latter achieves this for any cost function. probabilistic classifier chain (PCC) [10] and conditional filter tree (CFT) [15] are two other algorithms that generalizes another famous batch MLC algorithm called classifier chain (CC) [22] to cost-sensitive learning. PCC estimates the conditional probability of the label vector via CC, and makes a Bayes-optimal prediction with respect to the estimation and cost function. While PCC in principle achieves cost-sensitivity for any cost function, the prediction step can be time-consuming unless an efficient Bayes inference rule can be specifically designed for the cost function (*e.g.* F1 loss [11]). CFT embeds the cost information into CC by an $O(K^2)$-time step that re-weights the training

doi:10.6342/NTU201701149

instances for each classifier. All four algorithms above are designed for the batch cost-sensitive MLC problem, and it is not clear how they can be modified for the CSOMLC problem.

Label space dimension reduction (LSDR) is another family of MLC algorithms. LSDR encodes each label vector as a code vector in the lower-dimensional code space, and learns a predictor from the feature vectors to the corresponding code vectors. The prediction of LSDR consists of the predictor followed by a decoder from the code space to the label space. For example, compressed sensing (CS) [13] uses random projection for encoding, takes a regressor as the predictor, and decodes by sparse vector reconstruction. Instead of random projection, principal label space transformation (PLST) [24] encodes the label vectors $\{\mathbf{y}_n\}_{n=1}^N$ to their top principal components for the batch MLC problem. Other LSDR algorithms consider the feature and label vectors jointly, including conditional principal label space transformation [7], feature-aware implicit label space encoding [17], canonical-correlation-analysis method [23], and low-rank empirical risk minimization for multi-label learning [31]. The code vectors produced by those LSDR algorithms capture the joint information between the labels to allow more robust and more effective learning. Nevertheless, the algorithms are all designed for the batch MLC problem rather than the OMLC one, and they are not cost-sensitive.

Motivated by the possible applications of online updating, the realistic needs of cost-sensitivity, and the potential effectiveness of label space dimension reduction, we take an initiative to study LSDR algorithms for the CSOMLC setting. In particular, we first adapt PLST to the OMLC setting in Section 3, and further extend it to the CSOMLC setting in Section 4.

doi:10.6342/NTU201701149

Chapter 3

Dynamic Principal Projection

In this section, we first propose an online LSDR algorithm, dynamic principal projection (DPP), that optimizes the Hamming loss. DPP is motivated by the connection between PLST, which encodes the label vectors to their principal components, and the rich literature of online PCA algorithms [1, 19, 16]. Before discussing our design to combine PLST with online PCA, we introduce their respective details first.

3.1 Principal Label Space Transformation

Given the dimension $M \leq K$ of the code space and a batch training dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, PLST encodes each $\mathbf{y}_n \in \{+1, -1\}^K$ into a code vector $\mathbf{z}_n = \mathbf{P}^*(\mathbf{y}_n - \bar{\mathbf{y}})$, where $\bar{\mathbf{y}}$ is the empirical mean of $\{\mathbf{y}_n\}_{n=1}^N$, and the rows of $\mathbf{P}^*$ contain the projection directions to the top M principal components of $\{\mathbf{y}_n - \bar{\mathbf{y}}\}_{n=1}^N$. That is, $\mathbf{P}^*$ contains the top M eigenvectors of $\sum_{n=1}^N (\mathbf{y}_n - \bar{\mathbf{y}})(\mathbf{y}_n - \bar{\mathbf{y}})^\top$. A multi-target regressor $\mathbf{r}$ is then learned on $\{(\mathbf{x}_n, \mathbf{z}_n)\}_{n=1}^N$, and the prediction of an unseen instance $\mathbf{x}$ is made by

$$\hat{\mathbf{y}} = \text{round}\left((\mathbf{P}^*)^\top \mathbf{r}(\mathbf{x}) + \bar{\mathbf{y}}\right) \tag{3.1}$$

where $\text{round}(\mathbf{v}) = \Big(\text{sign}(\mathbf{v}[1]), \ldots, \text{sign}(\mathbf{v}[K])\Big)^\top$.

By projecting to the top principal components, PLST preserves the maximum amount of information within the observed label vectors. In addition, PLST is backed by the following theoretical guarantee:

doi:10.6342/NTU201701149

Theorem 1 ([24]). *When making a prediction* $\hat{\mathbf{y}}$ *from* $\mathbf{x}$ *by* $\hat{\mathbf{y}} = \text{round}\left(\mathbf{P}^\top \mathbf{r}(\mathbf{x}) + \mathbf{o}\right)$ *with any given reference vector* $\mathbf{o}$ *and any left orthogonal matrix* $\mathbf{P}$*, the Hamming loss*

$$c_{\text{ham}}(\mathbf{y}, \hat{\mathbf{y}}) \leq \frac{1}{K}\left(\underbrace{\|\mathbf{r}(\mathbf{x}) - \mathbf{z}\|_2^2}_{\textit{pred. error}} + \underbrace{\|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})(\mathbf{y} - \mathbf{o})\|_2^2}_{\textit{reconstruction error}}\right)$$

where $\mathbf{z} \equiv \mathbf{P}(\mathbf{y} - \mathbf{o})$.

Theorem 1 bounds the Hamming loss by the prediction and reconstruction errors. Based on the standard results of PCA, the pair $(\mathbf{P}^*, \bar{\mathbf{y}})$ in PLST is the optimal solution for minimizing the reconstruction error of the observed label vectors. Then, by minimizing the prediction error with regressor $\mathbf{r}$, PLST is able to minimize the Hamming loss approximately.

3.2 Online PCA

We start from the common setting considered in online PCA algorithms [1, 16, 19]. An online PCA algorithm is assumed to receive $\mathbf{y}_t \in \mathbb{R}^K$ at each iteration t with $\|\mathbf{y}_t\|_2 \leq 1$. Given the dimension $M \leq K$ of the lower-dimensional code space, the algorithm picks $\mathbf{P}_t \in \mathbb{R}^{M \times K}$ with orthogonal rows and suffers reconstruction error $\|(\mathbf{I} - \mathbf{P}_t^\top \mathbf{P}_t)\mathbf{y}_t\|_2^2$. The goal of the algorithm is to iteratively picks $\mathbf{P}_t$ such that $\sum_{t=1}^T \|(\mathbf{I} - \mathbf{P}_t^\top \mathbf{P}_t)\mathbf{y}_t\|_2^2$ can be close to the reconstruction error induced by the best offline matrix $\mathbf{P}^*$ over T iterations.

In this work, we consider a simple but promising algorithm, matrix stochastic gradient (MSG) [1, 19], as the foundation of DPP. MSG maintains an up-to-date projection matrix $\mathbf{U}_t \in \mathbb{R}^{K \times K}$ constrained by $tr(\mathbf{U}_t) = M$, which is the convex hull of $rank(\mathbf{U}_t) = M$. Upon receiving a new $\mathbf{y}_t$, MSG updates $\mathbf{U}_t$ to $\mathbf{U}_{t+1}$ as

$$\begin{aligned} \textit{Descent: } \mathbf{U}'_{t+1} &= \mathbf{U}_t + \eta \mathbf{y}_t \mathbf{y}_t^\top \\ \textit{Projection: } \mathbf{U}_{t+1} &= \underset{tr(\mathbf{U})=M}{\arg\min} \quad \|\mathbf{U} - \mathbf{U}'_{t+1}\|_F^2 \end{aligned} \tag{3.2}$$

where η is the learning rate.

To conform with the setting of online PCA algorithm, $\mathbf{P}_t$ needs to be produced (from $\mathbf{U}_t$) at each iteration. As shown in [28], any $\mathbf{U}_t$ is a convex combination of at most K rank M projection matrices. Letting Γ_t be the discrete distribution with these projection

doi:10.6342/NTU201701149

matrices as events and the corresponding combination coefficients as probabilities, we can easily sample a projection matrix that yields the same reconstruction error in expectation. The eigen decomposition of the sampled projection matrix then gives $\mathbf{P}_t$. A greedy algorithm to find such a convex combination with time complexity be $\mathcal{O}(K^2)$ is also given in [28].

3.3 Proposed Approach

Next, we proceed to the detail of our proposed online LSDR algorithm, dynamic principal projection (DPP), which focuses on the Hamming loss.

As neither $\mathbf{P}^*$ nor $\bar{\mathbf{y}}$ is known a priori, naïvely extending PLST to an OMLC algorithm by replacing $\mathbf{r}$ with an online regressor $\mathbf{r}_t$ cannot be carried out. The key idea of DPP is thus to additionally replace $\mathbf{P}^*$ with an adaptively updated $\mathbf{P}_t$ by incorporating MSG. Nevertheless, the problem of drifting of projection basis $\mathbf{P}_t$ arises, which can negatively affect the performance of $\mathbf{r}_t$ because $\mathbf{r}_t$ is learned on the low-dimensional components of $\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}$ composed of different sets of projection basis.

We first establish the framework of DPP using $\mathbf{P}_t$ from MSG instead of $\mathbf{P}^*$ and discuss our solutions to handle basis drifting.

General Framework. Theorem 1 bounds the Hamming loss by the prediction and reconstruction errors. Therefore, it is natural to take these two errors as the loss function for OMLC. Using the online linear predictor $\mathbf{r}_t(\mathbf{x}) = \mathbf{W}_t^\top \mathbf{x}$ and $\mathbf{P}_t$ from an online PCA algorithm, the framework of DPP is established as follows.

For $t = 1, \ldots, T$

Receive $\mathbf{x}_t$ and predict $\hat{\mathbf{y}}_t = \text{round}(\mathbf{P}_t^\top \mathbf{W}_t^\top \mathbf{x})$

Receive $\mathbf{y}_t$ and suffer loss $\ell^{(t)}(\mathbf{W}_t, \mathbf{P}_t)$

Update $\mathbf{P}_t$ and $\mathbf{W}_t$

where

$$\ell^{(t)}(\mathbf{W}, \mathbf{P}) = \|\mathbf{W}^\top \mathbf{x}_t - \mathbf{P}\mathbf{y}_t\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})\mathbf{y}_t\|_2^2$$

doi:10.6342/NTU201701149

The framework is established with $\mathbf{o} = \mathbf{0}$, which accommodates the setting of online PCA algorithms because it is assumed that *uncentered* $\mathbf{y}_t$ comes in stream.

Our goal is to optimize the cumulative loss $\sum_{t=1}^{T} \ell^{(t)}(\mathbf{W}_t, \mathbf{P}_t)$. To achieve so, we choose to employ the merits of PLST to exploit MSG for optimizing the cumulative reconstruction error $\sum_{t=1}^{T} \|(\mathbf{I} - \mathbf{P}_t^\top \mathbf{P}_t)\mathbf{y}_t\|^2$, and leave the optimization of prediction error to online ridge regression. To be more specific, we first derive a naïve updating procedure for $\mathbf{P}_t$ and $\mathbf{W}_t$ as follows:

$$\begin{aligned} &\textit{Update}\ \mathbf{U}\text{: } \mathbf{U}_{t+1} = \mathcal{P}_{\text{trace}}(\mathbf{U}_t + \eta \mathbf{y}_t \mathbf{y}_t^\top) \\ &\textit{Sample}\ \mathbf{P}\text{: } \mathbf{P}_{t+1} \sim \Gamma_{t+1}\ (\text{from } \mathbf{U}_{t+1}) \\ &\textit{Update}\ \mathbf{W}\text{: } \mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \frac{\lambda}{2}\|\mathbf{W}\|_F^2 + \sum_{i=1}^{t} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{P}_i \mathbf{y}_i\|_2^2 \end{aligned} \tag{3.3}$$

$\mathcal{P}_{\text{trace}}(\cdot)$ abbreviates the projection step in (3.2), and λ is the regularization parameter for online ridge regression. Additionally, in order to fully accommodate the constraint of $\|\mathbf{y}_t\|_2 \leq 1$ for online PCA, we apply a result-invariant trick (subject to a proper scaling of λ and η) that scales $\mathbf{y}_t$ by $\frac{1}{\sqrt{K}}$.

Drifting of Projection Basis. At a first glance, (3.3) suffices to extend PLST to an OMLC algorithm. Nevertheless, a closer look at the update of $\mathbf{W}_t$ reveals a vulnerability with respect to the drift of projection basis $\mathbf{P}_t$ as t advances. In particular, PLST, as a batch MLC algorithm, uses the same $\mathbf{P}^*$ to encode each label vector. In contrast, $\mathbf{W}_t$ is updated with code vectors $\{\mathbf{z}_i\}_{i=1}^{t-1}$ where $\mathbf{z}_i = \mathbf{P}_i \mathbf{y}_i$, and tries to predict $\mathbf{z}_t = \mathbf{P}_t \mathbf{y}_t$ from $\mathbf{x}_t$. However, each $\mathbf{z}_i$ is essentially the set of combination coefficients of *different sets of basis* formed by different $\mathbf{P}_i$. $\mathbf{W}_t$ may therefore fail to predict $\mathbf{z}_t$, i.e. the coefficients with respect to a potentially new and different $\mathbf{P}_t$, effectively.

To remedy the issue of basis drifting, we propose two different techniques, Principal Basis Correction (PBC) and Principal Basis Transform (PBT). Each of them enjoys different advantages.

Principal Basis Correction. The ideal solution for the problem of basis drifting is to "align" each $\mathbf{z}_i$ with the $\mathbf{P}_t$ that is used for prediction. More specifically, we want $\mathbf{W}_t$ to

doi:10.6342/NTU201701149

be learned from $\{(\mathbf{x}_i, \mathbf{P}_t\mathbf{y}_i)\}_{i=1}^{t-1}$ instead of $\{(\mathbf{x}_i, \mathbf{P}_i\mathbf{y}_i)\}_{i=1}^{t-1}$. This can be achieved if $\mathbf{W}_t^{\text{PBC}}$ is the ridge regression solution of $\{(\mathbf{x}_i, \mathbf{P}_t\mathbf{y}_i)\}_{i=1}^{t-1}$. It is straightforward to see that

$$\mathbf{W}_t^{\text{PBC}} = \underbrace{(\lambda\mathbf{I} + \sum_{i=1}^{t-1}\mathbf{x}_i\mathbf{x}_i^\top)^{-1}}_{\mathbf{A}_t^{-1}}\underbrace{(\sum_{i=1}^{t-1}\mathbf{x}_i\mathbf{y}_i^\top)}_{\mathbf{B}_t}\mathbf{P}_t^\top$$

By maintaining up-to-date $\mathbf{A}_t^{-1}$ and $\mathbf{B}_t$, which takes $\mathcal{O}(d^2)$ and $\mathcal{O}(Kd)$ space, respectively, $\mathbf{W}_t^{\text{PBC}}$ can be easily obtained by $\mathbf{A}_t^{-1}\mathbf{B}_t\mathbf{P}_t^\top$ for any $\mathbf{P}_t$.

Next, we analyze the performance of PBC with respect to its batch predecessor, PLST. For this comparison, it is natural to set up the offline cooperator as $(\mathbf{W}_\#, \mathbf{P}^*)$, where $\mathbf{P}^*$ minimizes $\sum_{t=1}^T \|(\mathbf{I} - (\mathbf{P}^\top\mathbf{P})\mathbf{y}_t\|_2^2$ and $\mathbf{W}_\#$ minimizes $\sum_{t=1}^T \|\mathbf{W}^\top\mathbf{x}_t - \mathbf{P}^*\mathbf{y}_t\|_2^2$. We show that, under the condition that the sequence $\{\mathbf{U}_t\}_{t=1}^T$ converges to $(\mathbf{P}^*)^\top\mathbf{P}^*$ as $T \to \infty$, the expected average regret

$$\frac{\mathcal{R}}{T} = \frac{1}{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\sum_{t=1}^T(\ell^{(t)}(\mathbf{W}_t^{\text{PBC}}, \mathbf{P}_t) - \ell^{(t)}(\mathbf{W}_\#, \mathbf{P}^*))]$$

has an upperbound that converges to 0 as $T \to \infty$, as formalized in the following theorem.

Theorem 2. *Assume that the sequence* $\{\|\mathbf{U}_t - (\mathbf{P}^*)^\top\mathbf{P}^*\|_2\}_{t=1}^T$ *converges to* 0 *as* $T \to \infty$. *Then there exists* $F(T) \geq \frac{\mathcal{R}}{T}$ *such that* $\lim_{T\to\infty} F(T) = 0$.

Principal Basis Transform. While PBC always gives the $\mathbf{W}_t^{\text{PBC}}$ learned on the correct code vectors with respect to the basis formed by $\mathbf{P}_t$, PBC has a dependency on $\Omega(Kd)$ because of the need to maintain $\mathbf{B}_t$. The $\Omega(Kd)$ dependency of the time complexity can make PBC computationally inefficient when both K and d are large.

To address the issue, we propose another solution, Principal Basis Transform (PBT), that does not require maintaining $\mathbf{B}_t$. Suppose we have a $\mathbf{W}'_{t-1}$ that predicts the combination coefficients of the basis formed by $\mathbf{P}_{t-1}$, and we aim for prediction with respect to the basis formed by $\mathbf{P}_t$. The key idea of PBT is to first reconstruct the prediction in label space by $\mathbf{P}_t^\top\mathbf{W}'_t$, and then project the prediction into low-dimensional space spanned by rows of $\mathbf{P}_t$ with minimal projection loss. Formally, PBT seeks $\mathbf{W}_t^{\text{PBT}}$ such that

$$\mathbf{W}_t^{\text{PBT}} = \arg\min_{\mathbf{W}} \quad \|\mathbf{W}\mathbf{P}_t - \mathbf{W}'_{t-1}\mathbf{P}_{t-1}\|_F^2 \tag{3.4}$$

doi:10.6342/NTU201701149

Solving (3.4) analytically gives

$$\mathbf{W}_t^{\text{PBT}} = \mathbf{W}'_{t-1}\mathbf{P}_{t-1}\mathbf{P}_t^\top \tag{3.5}$$

Finally, we update $\mathbf{W}_t^{\text{PBT}}$ with $(\mathbf{x}_t, \mathbf{P}_t\mathbf{y}_t)$ to obtain $\mathbf{W}'_t$ for the prediction of the next iteration. Note that $\mathbf{P}_t\mathbf{y}_t$ uses exactly the same basis as $\mathbf{W}_t^{\text{PBC}}$, and a direct update is therefore feasible.

One can see that PBT can be better than PBC because only dependency on $\Omega(M^2d)$ rather than $\Omega(Kd)$ is required as $\mathbf{P}_t\mathbf{P}_{t-1}^\top$ is first calculated in (3.5). In contrast, PBT can be worse than PBC because of the accumulated information loss every time (3.5) is applied. Therefore, we suggest PBT as a practical solution to remedy basis drifting when $\Omega(Kd)$ dependency of PBC is not acceptable. We shall also empirically demonstrate in Section 5 that PBT is generally competitive with PBC, while enjoying significant speedup for data with large K and d.

3.4 Practical Variant and Implementation

In this subsection, we first discuss the practical variant for updating $\mathbf{U}_t$ and the corresponding efficient sampling of $\mathbf{P}_t$. Then, we discuss an efficient implementation for updating $\mathbf{W}_t$.

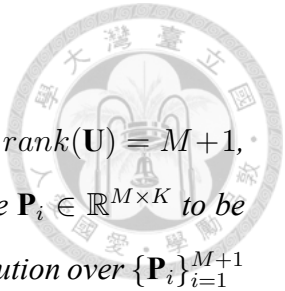
Efficient implementations of MSG have been studied in [1], which improved the time complexity from $O(K^2)$ of the naïve implementation to $\mathcal{O}(K \times rank^2(\mathbf{U}_t))$ at iteration t. Specifically, the descent step can be implemented by maintaining an up-to-date eigen decomposition of $\mathbf{U}_t = \mathbf{P}'diag(\sigma')(\mathbf{P}')^\top$, while the projection step is performed by clipping each value of σ' into $[0, 1]$ after a constant shift.

Nevertheless, the run-time of MSG, and also that of DPP, still critically depends on $rank(\mathbf{U}_t)$. Capped MSG, which is proposed in [1], is a practical variant of MSG that imposes a hard constraint of $rank(\mathbf{U}_t) \leq M'$ with $M \leq M'$ during the projection step. Capped MSG has been shown to enjoy significant speedup while still maintaining the quality of $\mathbf{U}_t$. We use $M' = M + 1$, as recommended in [1], for DPP.

As capped MSG guarantees the time complexity of updating $\mathbf{U}_t$ to be $\mathcal{O}(M^2K)$, the

doi:10.6342/NTU201701149

computational cost of sampling $\mathbf{P}_t$, which is $\mathcal{O}(K^2)$, becomes the main obstacle. We overcome this obstacle by presenting the following lemma.



Lemma 3. *Suppose* $\mathbf{U}$ *is obtained after an update of capped MSG with* $rank(\mathbf{U}) = M+1$*, and let* $\mathbf{P}'diag(\sigma')(\mathbf{P}')^\top$ *be the eigen decomposition of* $\mathbf{U}$*. Then define* $\mathbf{P}_i \in \mathbb{R}^{M\times K}$ *to be* $\mathbf{P}'$ with the i-th row excluded *and* Γ *to be a discrete probability distribution over* $\{\mathbf{P}_i\}_{i=1}^{M+1}$ *with probability of* $\mathbf{P}_i$ *being* $1-\sigma'_i$*, we have for any* $\mathbf{y}$

$$\mathbb{E}_{\mathbf{P}\sim\Gamma}[\mathbf{y}^\top\mathbf{P}^\top\mathbf{P}\mathbf{y}] = \mathbf{y}^\top\mathbf{U}\mathbf{y} \tag{3.6}$$

We refer our readers to the appendix for the proof. Because the up-to-date eigen decomposition of $\mathbf{U}_t$ is already maintained by (capped) MSG in each iteration, Lemma 3 directly gives an $\mathcal{O}(M)$ sampling procedure, which is significantly improved over the original $\mathcal{O}(K^2)$.

We now discuss the efficient implementation for updating $\mathbf{W}_t$. The optimal solution of $\mathbf{W}_{t+1}$ (without PBC or PBT) is known to be $\mathbf{W}_{t+1} = \mathbf{A}_{t+1}^{-1}(\sum_{i=1}^{t}\mathbf{x}_i\mathbf{z}_i^\top)$. Naïve calculation of $\mathbf{W}_{t+1}$ takes $\mathcal{O}(Md^2)$, even with the matrix inversion lemma due to the need for matrix multiplication. We eliminate the multiplication step by realizing that the updating of $\mathbf{W}_t$ has the following form:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \frac{\mathbf{A}_t^{-1}\mathbf{x}_{t+1}(\tilde{\mathbf{z}}_{t+1}-\mathbf{z}_{t+1})^\top}{1+\mathbf{x}_{t+1}^\top\mathbf{A}_t^{-1}\mathbf{x}_{t+1}} \tag{3.7}$$

where $\tilde{\mathbf{z}}_t = \mathbf{W}_t^\top\mathbf{x}_{t+1}$. (3.7) takes $\mathcal{O}(d^2+Md)$ by calculating $\mathbf{A}_t^{-1}\mathbf{x}_{t+1}$ first before the outer product.

(3.7) can be directly applied to obtain $\mathbf{W}'_{t+1}$ with PBT applied efficiently simply by replacing $\mathbf{W}_t$ with $\mathbf{W}_{t+1}^{\text{PBT}} = \mathbf{W}'_t\mathbf{P}_t\mathbf{P}_{t+1}^\top$. To efficiently implement PBC, one can instead maintain an alternative $\mathbf{H}_t$ by (3.7) with $\tilde{\mathbf{z}}_t, \mathbf{z}_t$ replaced by $\tilde{\mathbf{y}}_t = \mathbf{H}_t^\top\mathbf{x}_t, \mathbf{y}_t$, respectively, and calculate $\mathbf{W}_t^{\text{PBC}} = \mathbf{H}_t\mathbf{P}_t^\top$ afterward. We summarize the time complexity of updating $\mathbf{W}_t$ with PBC and PBT in the following table.

Time compl.	**W**-Update	**P**-Change
PBC	$\mathcal{O}(d^2+Kd)$	$\mathcal{O}(MKd)$
PBT	$\mathcal{O}(d^2+Md)$	$\mathcal{O}(M^2d+M^2K)$

doi:10.6342/NTU201701149

Chapter 4

Cost-Sensitive Extension

In this section, we extend DPP to cost-sensitive DPP (CS-DPP), which meets the requirement of CSOMLC. The key idea is based on a carefully designed label-weighting scheme that transforms cost $c(\mathbf{y}, \hat{\mathbf{y}})$ into the corresponding weighted Hamming loss. The optimization objective is then derived similarly to Theorem 1, which allows us to reuse the framework of DPP.

We start from the detail of our label-weighting scheme based on the label-wise decomposition of $c(\mathbf{y}, \hat{\mathbf{y}})$. The weight of each label arguably reflects its importance. However, many $c(\cdot, \cdot)$ (*e.g.* the F1 loss) do not evaluate each label independently. To allow the label weights to fully represent the cost, we propose a label-weighting scheme based on a label-wise and *order-dependent* decomposition of $c(\cdot, \cdot)$, which is motivated by a similar concept in [15]. The label-weighting scheme works as follows. Defining $\hat{\mathbf{y}}^{(k)}_{\text{real}}$ and $\hat{\mathbf{y}}^{(k)}_{\text{pred}}$ as

$$\hat{\mathbf{y}}^{(k)}_{\text{real}}[i] = \begin{cases} \mathbf{y}[i] & \text{if } i \leq k \\ \hat{\mathbf{y}}[i] & \text{if } i > k \end{cases} \textbf{ and } \hat{\mathbf{y}}^{(k)}_{\text{pred}}[i] = \begin{cases} \mathbf{y}[i] & \text{if } i < k \\ \hat{\mathbf{y}}[i] & \text{if } i \geq k \end{cases}$$

we decompose $c(\mathbf{y}, \hat{\mathbf{y}})$ into $\delta^{(1)}, \ldots, \delta^{(K)}$ such that

$$\delta^{(k)} = |c(\mathbf{y}, \hat{\mathbf{y}}^{(k)}_{\text{pred}}) - c(\mathbf{y}, \hat{\mathbf{y}}^{(k)}_{\text{real}})| \tag{4.1}$$

Our label-weighting scheme directly follows by simply setting the weight of k-th label as $\delta^{(k)}$.

The proposed label-weighting scheme with (4.1) enjoys nice theoretical guarantee un-

doi:10.6342/NTU201701149

Algorithm 1 Cost-Sensitive Dynamic Principal Projection with Principal Basis Transform

Parameters: λ, M, η

1: $\mathbf{P}_0 \leftarrow \mathbf{O}_{M\times K}$, $\mathbf{U}_0 \leftarrow \mathbf{O}_{K\times K}$, $\mathbf{A}_0^{-1} \leftarrow \frac{1}{\lambda}\mathbf{I}_{d\times d}$, $\mathbf{W}'_0 \leftarrow \mathbf{O}_{d\times M}$ ($\mathbf{O}$ is zero matrix)
2: **while** Receive $(\mathbf{x}_t, \mathbf{y}_t)$ **do**
3: $\quad\hat{\mathbf{y}}_t \leftarrow \text{round}(\mathbf{P}_{t-1}^\top \mathbf{W}_{t-1}'^\top \mathbf{x}_t)$
4: $\quad$Obtain $\mathbf{C}_t$ by (4.2)
5: $\quad$Update $\mathbf{U}_{t-1}$ to $\mathbf{U}_t$ by Capped MSG (using $\mathbf{C}_t\mathbf{y}_t$) and Sample $\mathbf{P}_t$ with Lemma 3
6: $\quad\mathbf{W}_t^{\text{PBT}} \leftarrow \mathbf{W}'_{t-1}\mathbf{P}_{t-1}\mathbf{P}_t^\top$ (PBT)
7: $\quad$Update $\mathbf{W}_t^{\text{PBT}}$, $\mathbf{A}_{t-1}^{-1}$ to $\mathbf{W}'_t$, $\mathbf{A}_t^{-1}$ by (3.7) (using $\mathbf{C}_t\mathbf{y}_t$ instead)
8: **end while**

der a mild condition of $c(\cdot,\cdot)$ as shown in the following lemma.

Lemma 4. *If* $c(\mathbf{y}, \mathbf{y}_{pred}^{(k)}) - c(\mathbf{y}, \mathbf{y}_{real}^{(k)}) \geq 0$ *holds for any* k, $\mathbf{y}$ *and* $\hat{\mathbf{y}}$, *then for any given* $\mathbf{y}$ *and* $\hat{\mathbf{y}}$, *we have*

$$c(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} \delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!]$$

The proof of the above lemma can be found in the appendix. Lemma 4 transforms $c(\mathbf{y}, \hat{\mathbf{y}})$ into the corresponding weighted Hamming loss, and thus enables the optimization over general cost functions.

Next, we propose CS-DPP, which extends DPP based on our proposed label-weighting scheme. Define $\mathbf{C}$ as

$$\mathbf{C} = diag(\sqrt{\delta^{(1)}}, ..., \sqrt{\delta^{(K)}}) \tag{4.2}$$

With $\mathbf{C}$, which carries the cost information, we establish a theorem similar to Theorem 1 to upperbound $c(\mathbf{y}, \hat{\mathbf{y}})$.

Theorem 5. *When making a prediction* $\hat{\mathbf{y}}$ *from* $\mathbf{x}$ *by* $\hat{\mathbf{y}} = \text{round}\left(\mathbf{P}^\top \mathbf{r}(\mathbf{x}) + \mathbf{o}\right)$ *with any given reference vector* $\mathbf{o}$ *and any left orthogonal matrix* $\mathbf{P}$, *if* $c(\cdot,\cdot)$ *satisfies the condition of Lemma 4, the prediction cost*

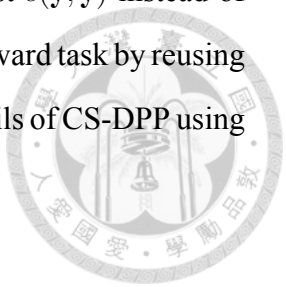
$$c(\mathbf{y}, \hat{\mathbf{y}}) \leq \|\mathbf{r}(\mathbf{x}) - \mathbf{z}^{\mathbf{C}}\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top\mathbf{P})(\mathbf{C}\mathbf{y} - \mathbf{o})\|_2^2$$

where $\mathbf{z}^{\mathbf{C}} = \mathbf{P}(\mathbf{C}\mathbf{y} - \mathbf{o})$.

The proof can be found in the appendix. This condition implies that correcting a wrongly-predicted label leads to no higher cost, and is considered mild as general cost functions satisfy the condition, including those mentioned in Section 2.

doi:10.6342/NTU201701149

Theorem 5 generalizes Theorem 1 to upperbound the general cost $c(\mathbf{y}, \hat{\mathbf{y}})$ instead of $c_{\text{ham}}(\mathbf{y}, \hat{\mathbf{y}})$. With Theorem 5, extending DPP to CS-DPP is a straightforward task by reusing the updating framework of DPP with $\mathbf{y}_t$ replaced by $\mathbf{C}_t\mathbf{y}_t$. The full details of CS-DPP using PBT is given in Algorithm 1.



doi:10.6342/NTU201701149

Chapter 5

Experiments

To empirically evaluate the performance, and also to study the effectiveness and necessity of design components of CS-DPP, we conduct three sets of experiments: (1) necessity justification of LSDR, (2) experiments on basis drifting, and (3) experiments on cost-sensitivity.

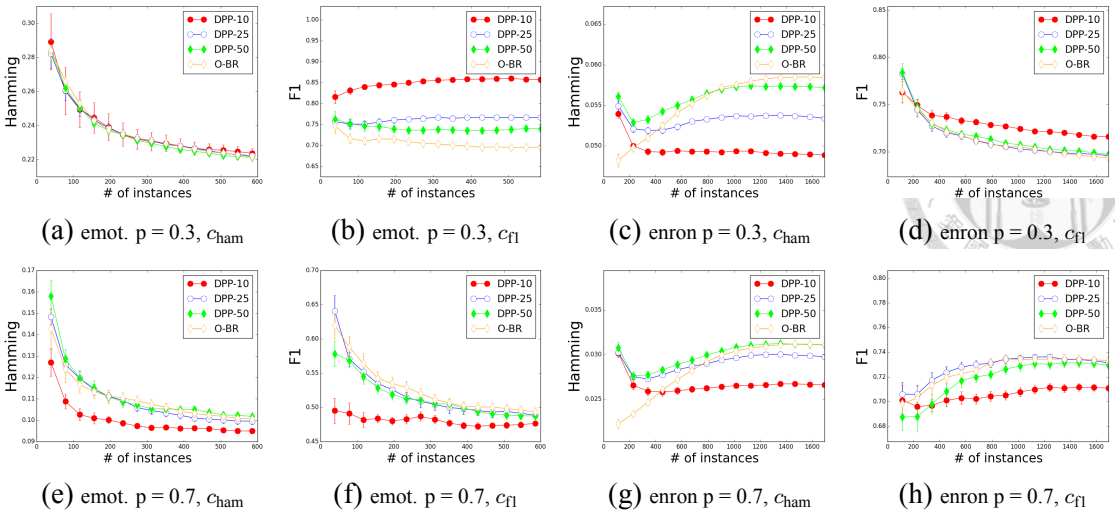
5.1 Experiments Setup

We conduct our experiments on nine real-world datasets[1] downloaded from Mulan[2]. Data streams are generated by permuting datasets into different random orders. All LSDR algorithms are coupled with online ridge regression and three different code space dimensions, $M = 10\%$, 25%, and 50% of K, are considered.

We consider four different cost functions: Hamming loss, Normalized rank loss, F1 loss, and Accuracy loss, as defined in Section 2 to justify the cost-sensitivity. The performances of different algorithms are compared using the average cumulative cost $\frac{1}{t}\sum_{i=1}^{t} c(\mathbf{y}_i, \hat{\mathbf{y}}_i)$ at each iteration t. We report the average results of each experiment after 15 repetitions.

[1] CAL500, emotions, scene, yeast, enron, Corel5k, mediamill, nuswide and medical
[2] http://mulan.sourceforge.net/datasets-mlc.html

doi:10.6342/NTU201701149



(a) emot. p = 0.3, c_{ham} (b) emot. p = 0.3, c_{f1} (c) enron p = 0.3, c_{ham} (d) enron p = 0.3, c_{f1}

(e) emot. p = 0.7, c_{ham} (f) emot. p = 0.7, c_{f1} (g) enron p = 0.7, c_{ham} (h) enron p = 0.7, c_{f1}

Figure 5.1: DPP vs. O-BR on noisy labels

Dataset	delicious			eurlex-eurovec		
Algorithms	PBT	PBC	O-BR	PBT	PBC	O-BR
c_{ham}	0.1136	0.1153	0.1245	0.4917	0.5011	0.4993
c_{NR}	0.5636	0.5641	0.5756	0.7435	0.7467	0.7433
c_{F1}	0.9143	0.9138	0.9076	0.9972	0.9928	0.9921
c_{Acc}	0.9512	0.9517	0.9494	0.9980	0.9964	0.9958
Avg. time (sec)	**21.49**	140.77	105.18	**60.81**	10522.25	4841.35

Table 5.1: DPP vs. O-BR on Large Dataset

5.2 Necessity of LSDR

In this experiment, we aim to justify the necessity to address LSDR for OMLC problems. We demonstrate that the ability of LSDR of preserving the key joint correlations between labels can be helpful when facing (1) data with noisy labels or (2) data with a large possible set of labels, where these types of data are often encountered in real-world OMLC problems. We compare DPP with online Binary Relevance (O-BR), which is a naïve extension from binary relevance [25] with online ridge regressor. The only difference between DPP and O-BR is whether the algorithm incorporates LSDR or not.

We first compare DPP and O-BR on data with noisy labels. We generate noisy data stream by randomly flipping each positive label $\mathbf{y}[i] = 1$ to negative with probability $p = \{0.3, 0.5, 0.7\}$, which simulates the real-world scenario in which human annotators fail to tag the existed labels. We plot the results of O-BR and DPP with $M = 10\%$, 25% and 50% of K on datasets *emotions* and *enron* with respect to Hamming loss and F1 loss in Figure 5.1. We report the complete results in the appendix.

doi:10.6342/NTU201701149

The results from the first two columns of Figure 5.1 show that DPP with $M = 10\%$ of K performs competitively and even better than O-BR as p increases. The results from the last two columns of Figure 5.1 show that DPP always performs better on *enron*. We can also observe from Figure 5.1 that DPP with smaller M tends to perform better as p increases. The above results clearly demonstrate that DPP better resists the effect of noisy labels with its incorporation of LSDR as the noise level (p) increases, while O-BR suffers more from the noise as it makes an independent prediction on each label. The observation that DPP with smaller M tends to perform better demonstrates that DPP is more robust to noise by preserving the key of the key joint correlations between labels with LSDR.

Next, we demonstrate that LSDR is also helpful for handling data with a large label set. We compare O-BR with DPP that is coupled with either PBC or PBT on datasets *delicious* and *eurlex-eurovec*[3]. DPP uses $M = 10$ for *delicious* and $M = 25$ for *eurlex-eurovec*. We summarize the results and average run-time in Table 5.1. The results from Table 5.1 indicate that DPP coupled with either PBT or PBC performs competitively with O-BR, while DPP with PBT enjoys significantly cheaper computational cost. The results demonstrate that DPP enjoys more effective and efficient learning for data with a large label set than O-BR, and also justifies the advantage of PBT over PBC in terms of efficiency when K and d are large while M is relatively small.

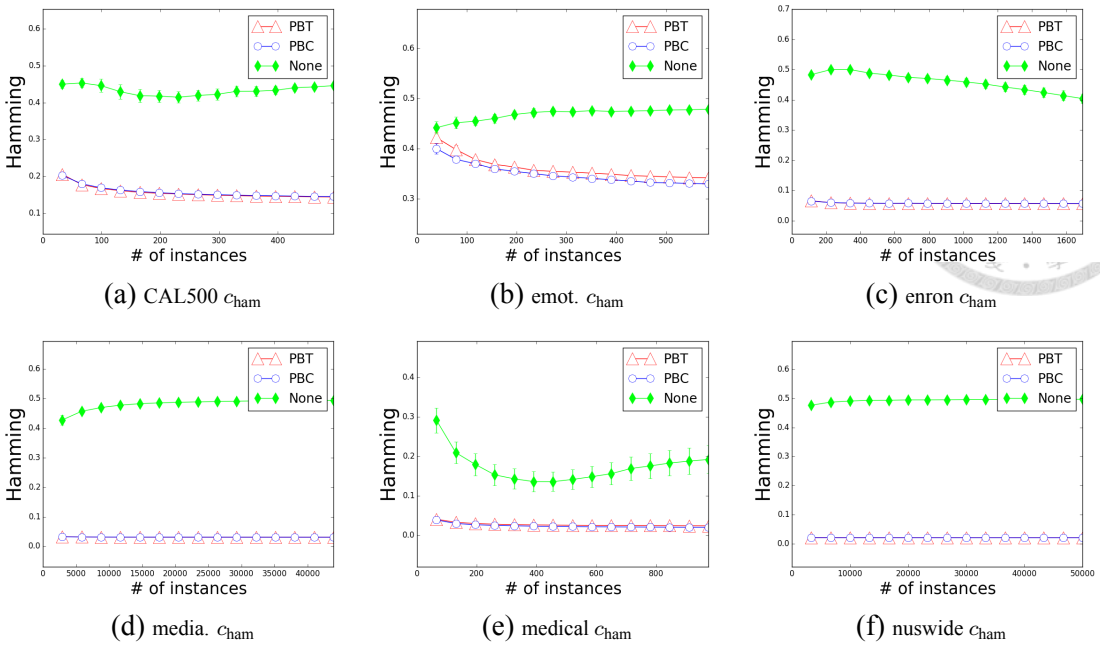
5.3 Experiments on Basis Drifting

To empirically justify the necessity of handling basis drifting, we compare variants of DPP that (1) performs PBC, (2) performs PBT, and (3) neglects basis drifting. We plot the results for Hamming loss with $M = 10\%$ of K in Figure 5.2 on datasets *CAL500*, *emotions*, *enron*, *mediamill*, *medical*, and *nuswide*, and report the complete results in the appendix. The results on all datasets in Figure 5.2 show that DPP with either PBC or PBT significantly improves the performance over its variant that neglects the basis drifting, which clearly demonstrates the necessity to handle the drifting of projection basis.

Further comparison of PBC and PBT based on Figure 5.2 reveals that PBT performs

[3] *delicious*: $d = 500$, $K = 983$, *eurlex-eurovec*: $d = 5000$, $K = 3993$.

doi:10.6342/NTU201701149



(a) CAL500 c_{ham} (b) emot. c_{ham} (c) enron c_{ham}

(d) media. c_{ham} (e) medical c_{ham} (f) nuswide c_{ham}

Figure 5.2: PBC vs. PBT vs. None

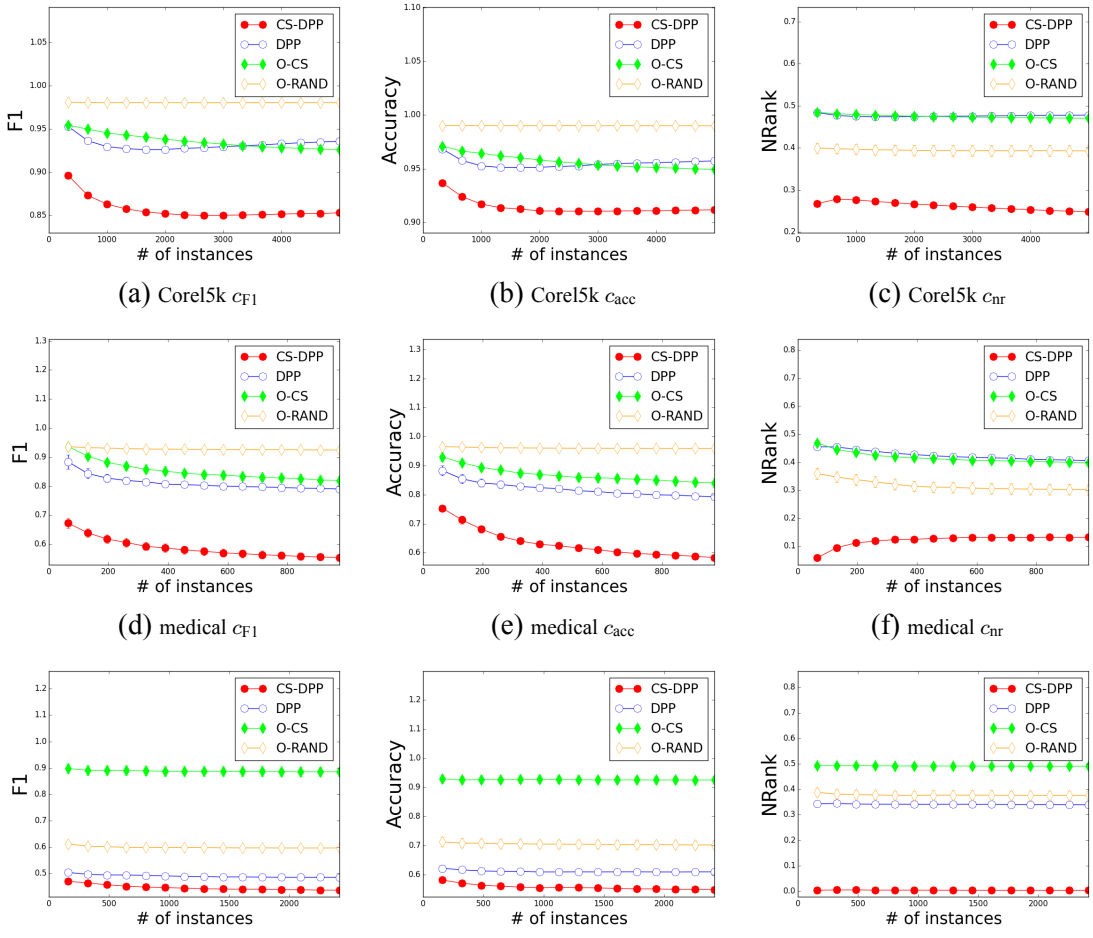
competitively with PBC. Nevertheless, as discussed in Section 5.2, PBT enjoys a nice computational speedup when K and d are large and M is relatively small, making PBT more suitable to handle data with a large label set.

5.4 Experiments on Cost-Sensitivity

To empirically justify the necessity of cost-sensitivity, we compare CS-DPP using PBT with DPP using PBT and other online LSDR algorithms. To the best of our knowledge, no online LSDR algorithm has yet been proposed in the literature. We therefore design two simple online LSDR algorithms, online Compressed Sensing (O-CS) and online Pseudo-inverse Decoding (O-RAND), to compare with CS-DPP. O-CS is a straightforward extension of CS [13] with an online ridge regressor. O-RAND encodes using random matrix $\mathbf{P}_R$ and simply decodes with the corresponding pseudo inverse $\mathbf{P}_R^\dagger$.

We plot the results with respect to all evaluation criteria except for the Hamming loss with $M = 10\%$ of K in Figure 5.3 on datasets *Corel5k*, *enron*, *medical*, and *yeast*. We report the complete results in the appendix.

doi:10.6342/NTU201701149



t

(a) Corel5k c_{F1} (b) Corel5k c_{acc} (c) Corel5k c_{nr}

(d) medical c_{F1} (e) medical c_{acc} (f) medical c_{nr}

(g) yeast c_{F1} (h) yeast c_{acc} (i) yeast c_{nr}

Figure 5.3: CS-DPP vs. Others

doi:10.6342/NTU201701149

CS-DPP versus DPP. The results of Figure 5.3 clearly indicate that CS-DPP performs significantly better than DPP on all evaluation criteria other than the Hamming loss, while CS-DPP reduces to DPP when $c_{\text{Ham}}(\cdot,\cdot)$ is used as the cost function. These observations demonstrate that CS-DPP, by optimizing the given cost function instead of Hamming loss, indeed achieves cost-sensitivity and is superior to its cost-insensitive predecessor, DPP.

CS-DPP versus Other Online LSDR Algorithms. As shown in Figure 5.3, while DPP generally performs better than O-CS and O-RAND because of the advantage to preserve key correlations between labels rather than random ones, it can nevertheless be inferior on some datasets with respect to specific cost functions due to its cost-insensitivity. For example, DPP loses to O-RAND on dataset *Corel5k* with respect to the Normalized rank loss, as shown in Figure 5.3(c). CS-DPP conquers the weakness of DPP with its cost-sensitivity, and significantly outperforms O-CS and O-RAND on all three datasets with respect to all three evaluation criteria, as demonstrated in Figure 5.3. The superiority of CS-DPP justifies the necessity to take cost-sensitivity into account.

doi:10.6342/NTU201701149

Chapter 6

Conclusion

We proposed a novel cost-sensitive online LSDR algorithm called cost-sensitive dynamic principal projection (CS-DPP). We established the foundation of CS-DPP via the connection of PLST and online PCA, and derived CS-DPP along with its theoretical guarantees on top of MSG. We successfully conquered the challenge of basis drifting using our carefully designed PBC and PBT. CS-DPP further achieves cost-sensitivity because of our label-weighting scheme with a nice theoretical guarantee. Practical enhancements of CS-DPP were also studied to improve its efficiency. The empirical results demonstrate that CS-DPP significantly outperforms other OMLC algorithms on all evaluation criteria, which validates the robustness and superiority of CS-DPP. The necessity for CS-DPP to address LSDR, basis drifting and cost-sensitivity was also empirically justified.

doi:10.6342/NTU201701149

Appendix A

A.1 Proof of Theorem 2

Theorem 2. *Assume that the sequence* $\{\|\mathbf{U}_t - (\mathbf{P}^*)^\top \mathbf{P}^*\|_2\}_{t=1}^T$ *converges to* 0 *as* $T \to \infty$. *Then there exists* $F(T) \geq \frac{\mathcal{R}}{T}$ *such that* $\lim_{T\to\infty} F(T) = 0$.

Recall that

$$\mathbf{W}_t^{\text{PBC}} = \underbrace{(\lambda \mathbf{I} + \sum_{i=1}^{t-1} \mathbf{x}_i \mathbf{x}_i^\top)^{-1}}_{\mathbf{A}_t^{-1}} \underbrace{(\sum_{i=1}^{t-1} \mathbf{x}_i \mathbf{y}_i^\top)}_{\mathbf{B}_t} \mathbf{P}_t^\top$$

For simplicity, we will overload $\mathbf{W}_t^{\text{PBC}}$ with $\mathbf{W}_t$, and denote $\mathbf{A}_t^{-1}\mathbf{B}_t$ as $\mathbf{H}_t$. Similarly, we have $\mathbf{W}_\# = \mathbf{H}^* (\mathbf{P}^*)^\top$, where

$$\mathbf{H}^* = \arg\min_{\mathbf{H}} \quad \sum_{t=1}^{T} \|\mathbf{H}^\top \mathbf{x}_t - \mathbf{y}_t\|_2^2$$

Before going into the details of the proof, we list several required (and minor) assumptions. We assume $\|\mathbf{H}_t \mathbf{x}_t - \mathbf{y}_t\|_2^2 \leq p^*$ for $t = 1, \ldots, T$ and $\|\mathbf{H}^*\|_F^2 \leq h^*$, which is similar to that assumed in [3]. We also assume that $\|\mathbf{x}_t\|_2 \leq 1$.

Proof. It is straight-forward to see that

$$\mathcal{R} = \underbrace{\mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\sum_{t=1}^{T} \|\mathbf{W}_t^\top \mathbf{x}_t - \mathbf{P}_t \mathbf{y}_t\|_2^2 - \|(\mathbf{W}^*)^\top \mathbf{x}_t - \mathbf{P}^* \mathbf{y}_t\|_2^2]}_{\mathcal{R}_{\text{Ridge}}} + \underbrace{\mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\sum_{t=1}^{T} \ell_{\text{MSG}}^{(t)}(\mathbf{P}_t) - \ell_{\text{MSG}}^{(t)}(\mathbf{P}^*)]}_{\mathcal{R}_{\text{MSG}}} \tag{A.1}$$

doi:10.6342/NTU201701149

where

$$\ell^{(t)}_{\text{MSG}}(\mathbf{P}) = \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})\mathbf{y}_t\|_2^2$$

as sampling of $\mathbf{P}_t$ from Γ_t is independent between each iteration t, which suggests that $\mathcal{R}_{\text{Ridge}}$ and $\mathcal{R}_{\text{MSG}}$ can be bounded separately.

We start bounding $\mathcal{R}_{\text{MSG}}$ by first observing that

$$\begin{aligned}
\mathcal{R}_{\text{MSG}} &= \sum_{t=1}^{T} \mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\ell^{(t)}_{\text{MSG}}(\mathbf{P}_t) - \ell^{(t)}_{\text{MSG}}(\mathbf{P}^*)] \\
&\leq \sum_{t=1}^{T} \mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\ell^{(t)}_{\text{MSG}}(\mathbf{P}_t)] - \ell^{(t)}_{\text{MSG}}(\mathbf{U}^*) \\
&= \sum_{t=1}^{T} \ell^{(t)}_{\text{MSG}}(\mathbf{U}_t) - \ell^{(t)}_{\text{MSG}}(\mathbf{U}^*) \qquad \text{(A.2)}
\end{aligned}$$

where $\mathbf{U}^*$ is the optimal projection matrix with respect to $\sum_{t=1}^{T} \ell^{(t)}_{\text{MSG}}(\mathbf{U})$ whose $tr(\mathbf{U}^*) = M$. The first equality follows from the observation that the sampling of $\mathbf{P}_t$ does not affect the update of $\mathbf{U}_t$, and therefore is independent between each iteration. The second inequality follows from the fact that $\sum_{t=1}^{T} \ell^{(t)}_{\text{MSG}}(\mathbf{U}^*) \leq \sum_{t=1}^{T} \ell^{(t)}_{\text{MSG}}(\mathbf{P}^*)$ as $tr((\mathbf{P}^*)^\top \mathbf{P}^*) = M$. The third equality follows by realizing that $(\mathbf{I} - \mathbf{P}_t^\top \mathbf{P}_t)$ is a projection matrix plus the fact that $\mathbb{E}[\mathbf{P}_t^\top \mathbf{P}_t] = \mathbf{U}_t$. Analysis of Eq. (4) follows standard analysis of online gradient descent and can be found in Appendix of [19], which gives an upperbound as $\frac{M(K-M)}{2\eta K} + \frac{\eta}{2}\sum_{t=1}^{\top} \|\mathbf{y}_t\|_2^2$. With $\|\mathbf{y}_t\|_2^2 \leq 1$ and minimization over η yields

$$\mathcal{R}_{\text{MSG}} \leq \sqrt{\frac{M(K-M)}{K}T} = F_1(T)$$

.

We next analyze $\mathcal{R}_{\text{Ridge}}$. We first rewrite $\mathcal{R}_{\text{Ridge}}$ as

$$\sum_{t=1}^{T} \mathbb{E}[\|\mathbf{P}_t(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 - \|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 + \|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 - \|\mathbf{P}^*((\mathbf{H}^*)^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2]$$

by $\mathbf{W}_t = \mathbf{H}_t \mathbf{P}_t^\top$ and $\mathbf{W}^* = \mathbf{H}^* (\mathbf{P}^*)^\top$. Note that we omit the subscript of expectation here. We first bound

$$\sum_{t=1}^{T} \mathbb{E}[\|\mathbf{P}_t(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 - \|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2]$$

doi:10.6342/NTU201701149

Let $\bar{\mathbf{y}}_t = \mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t$ and $\mathbf{U}^* = (\mathbf{P}^*)^\top \mathbf{P}^*$ (note its difference from that is used in analyzing $\mathcal{R}_{\text{MSG}}$), then we have

$$\begin{aligned}\mathbb{E}\sum_{t=1}^{T}[\bar{\mathbf{y}}_t^\top \mathbf{P}_t^T \mathbf{P}_t \bar{\mathbf{y}}_t^\top - \bar{\mathbf{y}}_t^\top \mathbf{U}^* \bar{\mathbf{y}}_t] &= \sum_{t=1}^{T} \bar{\mathbf{y}}_t^\top (\mathbf{U}_t - \mathbf{U}^*)\bar{\mathbf{y}}_t \\ &\leq \sum_{t=1}^{T} \|\bar{\mathbf{y}}_t\|_2^2 \frac{\|(\mathbf{U}_t - \mathbf{U}^*)\bar{\mathbf{y}}_t\|_2}{\|\bar{\mathbf{y}}_t\|_2} \\ &\leq p^* \sum_{t=1}^{T} \|\mathbf{U}_t - \mathbf{U}^*\|_2 = F_2(T)\end{aligned}$$

where the last inequality follows from $\|\bar{\mathbf{y}}_t\|_2^2 \leq p^*$ and the definition of matrix 2-norm.

Next we bound

$$\sum_{t=1}^{T} \mathbb{E}[\|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 - \|\mathbf{P}^*((\mathbf{H}^*)^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2]$$

We first define another game as

$$\mathcal{R}_z = \sum_{t=1}^{T} \|\bar{\mathbf{W}}_t^\top \mathbf{x}_t - \mathbf{z}_t\|_2^2 - \|\,(\bar{\mathbf{W}}^*)^\top \mathbf{x}_t - \mathbf{z}_t\|_2^2 \tag{A.3}$$

where $\mathbf{z}_t = \mathbf{P}^*\mathbf{y}_t$, $\bar{\mathbf{W}}_t = \arg\min_{\mathbf{W}} \ \frac{\lambda}{2}\sum_{i=1}^{t-1} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{z}_i\|_2^2$, and $\bar{\mathbf{W}}^*$ be the best offline cooperator of the game. It is not hard to notice that the game is exactly the same as the target we wish to bound by realizing that $\bar{\mathbf{W}}_t = \mathbf{H}_t\,(\mathbf{P}^*)\,\top$ and $\bar{\mathbf{W}}^* = \mathbf{H}^*\,(\mathbf{P}^*)\,\top$ and also the expectation can simply be removed. Furthermore, any $(\mathbf{x}_t, \mathbf{z}_t)$ has at least corresponding $(\mathbf{x}_t, \mathbf{y}_t)$ as $\mathbf{P}^*$ is a linear operator. Therefore, bounding (A.3) suffices to bound our target.

Now let

$$\ell_{\text{Ridge}}^{(t)}(\bar{\mathbf{W}}) = \|\bar{\mathbf{W}}^\top \mathbf{x}_t - \mathbf{z}_t\|_2^2$$

We next rewrite (A.3) as

$$\sum_{m=1}^{M}\sum_{t=1}^{T}(\ell_{\text{Ridge}}^{(t,m)}(\mathbf{w}_{t,m}) - \ell_{\text{Ridge}}^{(t,m)}(\mathbf{w}_m^*)) \tag{A.4}$$

where

$$\ell_{\text{Ridge}}^{(t,m)}(\mathbf{w}) = \|\mathbf{w}^\top \mathbf{x}_t - z_{t,m}\|_2^2$$

$z_{t,m}$ is the m-th element of $\mathbf{z}_t$, $\mathbf{w}_{t,m}$ is the m-th column of $\bar{\mathbf{W}}_t$, and $\mathbf{w}_m^*$ is the m-th column

doi:10.6342/NTU201701149

of $\bar{\mathbf{W}}^*$. Next, we have

$$\|\mathbf{w}_m^*\|_2 \leq \|\bar{\mathbf{W}}^*\|_F^2 \leq \|\mathbf{U}^*\|_F^2 \|\mathbf{H}^*\|_F^2 \leq Mh^*$$

where the last inequality follows $tr(\mathbf{U}^*) = M$, and

$$\ell_{\text{Ridge}}^{(t,m)}(\mathbf{w}_{t,m}) \leq \|\mathbf{W}_t^\top \mathbf{x}_t - \mathbf{z}_t\|_2^2 = \bar{\mathbf{y}}_t^T \mathbf{U}^* \bar{\mathbf{y}}_t \leq p^*$$

where the last inequality follows from the fact that $\mathbf{U}^*$ is a projection matrix. With the above, by plugging in $\lambda = 1$ and follow the analysis as shown in [3], we have

$$\sum_{t=1}^{T} (\ell_{\text{Ridge}}^{(t,m)}(\mathbf{w}_{t,m}) - \ell_{\text{Ridge}}^{(t,m)}(\mathbf{w}_m^*)) \leq \frac{M}{2} h^* + 2p^* d \log(1 + \frac{T}{d} \max_t((z_{t,m})^2)) \tag{A.5}$$

where d is the dimension of $\mathbf{x}_t$. Then by $\max_t((z_{t,m})^2) \leq 1$ which comes from

$$\max_t((z_{t,m})^2) \leq \max_t(\|\mathbf{z}_t\|_2^2) = \max_t(\|\mathbf{P}^* \mathbf{y}_t\|_2^2) \leq \max_t \|\mathbf{y}_t\|_2^2 \leq 1$$

and summing (A.5) over all $m = 1, \ldots, M$ we obtain the upperbound of (A.4) as

$$F_3(T) = \frac{M^2}{2} h^* + 2Mp^* d \log(1 + \frac{T}{d})$$

Now it is easy to see that $\frac{\mathcal{R}}{T} \leq F(T) = \frac{F_1(T)+F_2(T)+F_3(T)}{T}$. It is straight forward to see that $\lim_{T\to\infty} \frac{F_1(T)}{T} = 0$ and $\lim_{T\to\infty} \frac{F_3(T)}{T} = 0$. To see $\lim_{T\to\infty} \frac{F_2(T)}{T} = 0$, we have by assumption $\{\|\mathbf{U}_t - \mathbf{U}^*\|_2\}_{t=1}^T$ converges to 0 as $T \to \infty$, and the fact that the convergence of sequence implies the convergence of arithmetic mean of sequence. Combing the above we have $\lim_{T\to\infty} F(T) = 0$, which completes the proof.

□

A.2 Proof of Lemma 3

Lemma 3. *Suppose* $\mathbf{U}$ *is obtained after an update of Capped MSG with* $rank(\mathbf{U}) = M+1$*, and let* $\mathbf{P}' diag(\sigma')(\mathbf{P}')^\top$ *be the eigen decomposition of* $\mathbf{U}$*. Then define* $\mathbf{P}_i \in \mathbb{R}^{M\times K}$ *to be* $\mathbf{P}'$ with the i-th row excluded *and* Γ *to be a discrete probability distribution over* $\{\mathbf{P}_i\}_{i=1}^{M+1}$ *with probability of* $\mathbf{P}_i$ *being* $1 - \sigma_i'$*, we have for any* $\mathbf{y}$

$$\mathbb{E}_{\mathbf{P}\sim\Gamma}[\mathbf{y}^\top \mathbf{P}^\top \mathbf{P} \mathbf{y}] = \mathbf{y}^\top \mathbf{U} \mathbf{y} \tag{A.6}$$

doi:10.6342/NTU201701149

Proof. We first show that Γ is a well-defined probability distribution. By the definition of the projection step of MSG we have $0 \leq \sigma'_i \leq 1$ for each σ'_i and $\sum_{i=1}^{M+1} 1 - \sigma'_i = M + 1 - \sum_{i=1}^{M+1} \sigma'_i = 1$ with $tr(\mathbf{U}) = M$. Γ is therefore a well-defined probability distribution.

Then it suffices to show that $\mathbb{E}_{\mathbf{P}\sim\Gamma}[\mathbf{P}^\top\mathbf{P}] = \mathbf{U}$. To see this, first notice that by orthogonal rows of $\mathbf{P}'$ we have $\mathbf{U} = \sum_{j=1}^{M+1} \sigma'_j \mathbf{e}_j\mathbf{e}_j^\top$ where $\mathbf{e}_j$ is the j-th row of $\mathbf{P}'$. We then have

$$\begin{aligned}
\mathbb{E}_{\mathbf{P}\sim\Gamma}[\mathbf{P}^\top\mathbf{P}] &= \sum_{i=1}^{M+1}(1-\sigma'_i)\sum_{j=1}^{M+1}[\![i \neq j]\!]\mathbf{e}_j\mathbf{e}_j^\top \\
&= \sum_{j=1}^{M+1}(\mathbf{e}_j\mathbf{e}_j^\top \sum_{i=1}^{M+1}[\![i \neq j]\!](1-\sigma'_i)) \\
&= \sum_{j=1}^{M+1}(\sigma'_j\mathbf{e}_j\mathbf{e}_j^\top) \qquad (a) \\
&= \mathbf{U}
\end{aligned}$$

where (a) is by $\sum_{i=1}^{M+1} \sigma'_i = M$

□

A.3 Proof of Lemma 4

Lemma 4. *If* $c(\mathbf{y}, \mathbf{y}^{(k)}_{pred}) - c(\mathbf{y}, \mathbf{y}^{(k)}_{real}) \geq 0$ *holds for any* k, $\mathbf{y}$ *and* $\hat{\mathbf{y}}$, *then for any given* $\mathbf{y}$ *and* $\hat{\mathbf{y}}$ *we have*

$$c(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} \delta^{(k)}[\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!] \tag{A.7}$$

Proof. Recall the definition of $\mathbf{y}^{(k)}_{\text{real}}$ and $\mathbf{y}^{(k)_{\text{pred}}}$ to be

$$\hat{\mathbf{y}}^{(k)}_{\text{real}}[i] = \begin{cases} \mathbf{y}[i] & \text{if } i \leq k \\ \hat{\mathbf{y}}[i] & \text{if } i > k \end{cases} \textbf{ and } \hat{\mathbf{y}}^{(k)}_{\text{pred}}[i] = \begin{cases} \mathbf{y}[i] & \text{if } i < k \\ \hat{\mathbf{y}}[i] & \text{if } i \geq k \end{cases}$$

and the definition of $\delta^{(k)}$ to be

$$\delta^{(k)} = |c(\mathbf{y}, \hat{\mathbf{y}}^{(k)}_{\text{pred}}) - c(\mathbf{y}, \hat{\mathbf{y}}^{(k)}_{\text{real}})|$$

Now define $k_i, i = 1, \ldots, L$ be the sequence of indices such that $\mathbf{y}[k_i] \neq \hat{\mathbf{y}}[k_i]$ for every

doi:10.6342/NTU201701149

k_i and $k_i < k_{i+1}$. If such k_i does not exist than (A.7) holds trivially by $c(\mathbf{y}, \mathbf{y}) = 0$. Otherwise, by the condition of c we have

$$\begin{aligned}
&\sum_{k=1}^{K} \delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!] && (a)\\
&= \sum_{k=1}^{K} (c(\mathbf{y}, \hat{\mathbf{y}}^{(k)}_{\text{pred}}) - c(\mathbf{y}, \hat{\mathbf{y}}^{(k)}_{\text{real}})) [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!]\\
&= \sum_{i=1}^{L} c(\mathbf{y}, \hat{\mathbf{y}}^{(k_i)}_{\text{pred}}) - c(\mathbf{y}, \hat{\mathbf{y}}^{(k_i)}_{\text{real}})\\
&= c(\mathbf{y}, \hat{\mathbf{y}}^{(k_1)}_{\text{pred}}) - c(\mathbf{y}, \hat{\mathbf{y}}^{(k_L)}_{\text{real}}) && (b)\\
&= c(\mathbf{y}, \hat{\mathbf{y}}) && (c)
\end{aligned}$$

where (a) uses the condition of $c(\cdot, \cdot)$ to remove the absolute value function; (b) is from two possibilities of L: if $L = 1$ then the equation trivially holds; if $L > 1$ we use the observation that $\hat{\mathbf{y}}^{(k_i)}_{\text{real}} = \hat{\mathbf{y}}^{(k_{i+1})}_{\text{pred}}$ where the observation is by realizing $\mathbf{y}[j] = \hat{\mathbf{y}}[j]$ for any $k_i < j < k_{i+1}$; (c) follows from the observation that $\hat{\mathbf{y}}^{(k_1)}_{\text{pred}} = \hat{\mathbf{y}}$ and $\hat{\mathbf{y}}^{(k_L)}_{\text{real}} = \mathbf{y}$ and $c(\mathbf{y}, \mathbf{y}) = 0$. □

A.4 Proof of Theorem 5

Theorem 5. *When making a prediction* $\hat{\mathbf{y}}$ *from* $\mathbf{x}$ *by* $\hat{\mathbf{y}} = \text{round}\left(\mathbf{P}^\top \mathbf{r}(\mathbf{x}) + \mathbf{o}\right)$ *with any given reference vector* $\mathbf{o}$ *and any left orthogonal matrix* $\mathbf{P}$*, if* $c(\cdot, \cdot)$ *satisfies the condition of Lemma 4,*

$$c(\mathbf{y}, \hat{\mathbf{y}}) \leq \|\mathbf{r}(\mathbf{x}) - \mathbf{z}^{\mathbf{C}}\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})(\mathbf{C}\mathbf{y} - \mathbf{o})\|_2^2$$

where $\mathbf{z}^{\mathbf{C}} = \mathbf{P}(\mathbf{C}\mathbf{y} - \mathbf{o})$.

Recall the definition of $\mathbf{C}$ in the main context is

$$\mathbf{C} = diag(\sqrt{\delta^{(1)}}, ..., \sqrt{\delta^{(K)}}) \tag{A.8}$$

Next we show and prove the following lemma before we proceed to the complete proof.

Lemma 6. *Given the ground truth* $\mathbf{y}$*, if the binary-value prediction* $\hat{\mathbf{y}} \in \{+1, -1\}^K$ *is made by* round$(\tilde{\mathbf{y}})$ *where* $\tilde{\mathbf{y}}$ *is the real-value prediction* $\tilde{\mathbf{y}} \in \mathbb{R}^K$*. Then for any* $\mathbf{y}$*,* $\hat{\mathbf{y}}$*,* $\tilde{\mathbf{y}}$*, if* c

doi:10.6342/NTU201701149

satisfies the condition in Lemma 4, we have

$$c(\mathbf{y}, \hat{\mathbf{y}}) \leq \|\mathbf{C}\mathbf{y} - \tilde{\mathbf{y}}\|^2 \tag{A.9}$$

Proof. From Lemma 4 we have $c(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} \delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!]$. As $\|\mathbf{C}\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 = \sum_{k=1}^{K} (\sqrt{\delta^{(K)}}\mathbf{y}[k] - \tilde{\mathbf{y}}[k])^2$, it suffices to show that for all k we have

$$\delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!] \leq (\sqrt{\delta^{(k)}}\mathbf{y}[k] - \tilde{\mathbf{y}}[k])^2 \tag{A.10}$$

When $\delta^{(k)} = 0$, (A.10) holds trivially. When $\delta^{(k)} > 0$, we have

$$\begin{aligned}
&\delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!] \\
&= \delta^{(k)} ([\![\tilde{\mathbf{y}}[k] \geq 0]\!][\![\mathbf{y}[k] = -1]\!] + [\![\tilde{\mathbf{y}}[k] < 0]\!][\![\mathbf{y}[k] = +1]\!]) \\
&= \delta^{(k)} ([\![\frac{\tilde{\mathbf{y}}[k]}{\sqrt{\delta^{(k)}}} \geq 0]\!][\![\mathbf{y}[k] = -1]\!] + [\![\frac{\tilde{\mathbf{y}}[k]}{\sqrt{\delta^{(k)}}} < 0]\!][\![\mathbf{y}[k] = +1]\!]) \quad (\text{by } \delta^{(k)} > 0) \\
&\leq \delta^{(k)} ((\frac{\tilde{\mathbf{y}}[k]}{\sqrt{\delta^{(k)}}} - \mathbf{y}[k])^2 [\![\mathbf{y}[k] = -1]\!] + (\frac{\tilde{\mathbf{y}}[k]}{\sqrt{\delta^{(k)}}} - \mathbf{y}[k])^2 [\![\mathbf{y}[k] = +1]\!]) \\
&= \delta^{(k)} (\frac{\tilde{\mathbf{y}}[k]}{\sqrt{\delta^{(k)}}} - \mathbf{y}[k])^2 \\
&= (\sqrt{\delta^{(k)}}\mathbf{y}[k] - \tilde{\mathbf{y}}[k])^2
\end{aligned}$$

As $\delta^{(k)} \geq 0$ holds by its definition, (A.10) holds for every k. Summing (A.10) with respect to all k then completes the proof. □

With Lemma 6 established, we now prove Theorem 5.

Proof of Theorem 5. If the given c satisfies the condition in Lemma (4), and let $\tilde{\mathbf{y}} = \mathbf{P}^\top \mathbf{r}(\mathbf{x}) + \mathbf{o}$ and $\hat{\mathbf{y}} = \text{round}(\tilde{\mathbf{y}})$. Then for any $(\mathbf{x}, \mathbf{y})$ we have

$$\begin{aligned}
&c(\mathbf{y}, \hat{\mathbf{y}}) \\
&\leq \|\mathbf{C}\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 && (a) \\
&= \|((\tilde{\mathbf{y}} - \mathbf{o} - \mathbf{P}^\top \mathbf{P}\bar{\mathbf{y}}^{\mathbf{C}}) - (\bar{\mathbf{y}}^{\mathbf{C}} - \mathbf{P}^\top \mathbf{P}\bar{\mathbf{y}}^{\mathbf{C}}))\|_2^2 \\
&= \|(\mathbf{P}^\top(\mathbf{r}(\mathbf{x}) - \mathbf{z}^{\mathbf{C}}) - (\mathbf{I} - \mathbf{P}^\top \mathbf{P})\bar{\mathbf{y}}^{\mathbf{C}}\|_2^2 \\
&= \|(\mathbf{P}^\top(\mathbf{r}(\mathbf{x}) - \mathbf{z}^{\mathbf{C}})\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})\bar{\mathbf{y}}^{\mathbf{C}})\|_2^2 && (b) \\
&= \|\mathbf{r}(\mathbf{x}) - \mathbf{z}^{\mathbf{C}}\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})\bar{\mathbf{y}}^{\mathbf{C}}\|_2^2 && (c)
\end{aligned}$$

where $\bar{\mathbf{y}}^{\mathbf{C}} = \mathbf{C}\mathbf{y} - \mathbf{o}$ and $\mathbf{z}^{\mathbf{C}} = \mathbf{P}(\mathbf{C}\mathbf{y} - \mathbf{o})$. (a) is from Lemma A.10, while (b) and (c) follow from the orthogonal rows of $\mathbf{P}$. □

doi:10.6342/NTU201701149

Table A.1: Statistics of datasets

	# of features	# of labels	# of instances	cardinality
CAL500	68	174	502	26.044
Corel5k	499	374	5000	3.522
emotions	72	6	593	1.869
enron	1001	53	1702	3.378
mediamill	120	101	43907	4.376
medical	1449	45	978	1.245
scene	294	6	2407	1.074
yeast	103	14	2417	4.237
nuswide	128	81	50000*	1.869
delicious	500	983	7500*	19.020
eurlex-eurovec	5000	3993	7500*	5.310

We note that the proof above closely follows the proof of Theorem 1 in [24], while the key difference comes from Lemma 6 to handle the weighted Hamming loss.

A.5 Details of Experiments

Here we report the details of each experiment, including details of cost functions, parameters, complete results and the characteristics of datasets we use.

A.5.1 Datasets and Parameters

We first provide the details of the datasets used in our experiments in Table A.1. Only 50000 instances are used for *nuswisde* because O-CS is particularly computationally complex. Only 7500 instances are used for *delicious* and *eurlex-eurovec* to reduce the computational burden from O-BR and DPP with PBC.

For DPP we fix $\lambda = 1$ and follow [1] to use the time-decreasing learning rate $\eta = \frac{2}{\sqrt{t}}\frac{M}{K}$. For O-CS we follow [13] to set the parameters. Specifically for experiments on *delicious* and *eurlex*, we implement both DPP and O-BR using gradient descent instead of online ridge regression. We use the time decreasing step-size $\frac{1}{\sqrt{t}}$ for gradient descent on *delicious*, and $\frac{0.001}{\sqrt{t}}$ on *eurlex-eurovec*.

A.5.2 Necessity of LSDR

We report the complete results of comparison between O-BR and DPP with $M = 10\%$, 25% and 50% of K from Table A.2 to Table A.5 with respect to all four evaluation criteria, where the best values (the lowest) are marked in bold.

doi:10.6342/NTU201701149

Table A.2: DPP vs. O-BR on Noisy Data, Hamming loss

$p = 0.3$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	0.1130 ± 0.0003	0.1143 ± 0.0001	0.1133 ± 0.0002	**0.1113 ± 0.0002**
Corel5k	**0.0070 ± 0.0000**	0.0072 ± 0.0000	0.0072 ± 0.0000	0.0071 ± 0.0000
emotions	**0.2213 ± 0.0011**	**0.2214 ± 0.0013**	**0.2226 ± 0.0013**	**0.2238 ± 0.0026**
enron	0.0584 ± 0.0002	0.0572 ± 0.0002	0.0534 ± 0.0002	**0.0489 ± 0.0001**
mediamill	**0.0271 ± 0.0000**	0.0272 ± 0.0000	0.0272 ± 0.0000	0.0272 ± 0.0000
medical	**0.0168 ± 0.0001**	0.0177 ± 0.0001	0.0183 ± 0.0001	0.0190 ± 0.0001
nuswide	**0.0151 ± 0.0000**	**0.0151 ± 0.0000**	**0.0151 ± 0.0000**	**0.0151 ± 0.0000**
scene	**0.1197 ± 0.0005**	0.1282 ± 0.0008	0.1273 ± 0.0005	0.1258 ± 0.0004
yeast	**0.2034 ± 0.0004**	**0.2032 ± 0.0004**	0.2045 ± 0.0004	0.2034 ± 0.0005

$p = 0.5$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.0815 ± 0.0003**	0.0834 ± 0.0001	0.0823 ± 0.0002	**0.0816 ± 0.0002**
Corel5k	**0.0049 ± 0.0000**	0.0051 ± 0.0000	0.0051 ± 0.0000	0.0051 ± 0.0000
emotions	0.1736 ± 0.0014	0.1689 ± 0.0017	0.1660 ± 0.0015	**0.1598 ± 0.0014**
enron	0.0475 ± 0.0002	0.0470 ± 0.0002	0.0440 ± 0.0002	**0.0398 ± 0.0002**
mediamill	**0.0217 ± 0.0000**	**0.0217 ± 0.0000**	**0.0217 ± 0.0000**	**0.0217 ± 0.0000**
medical	**0.0153 ± 0.0001**	0.0163 ± 0.0001	0.0160 ± 0.0001	0.0157 ± 0.0001
nuswide	**0.0109 ± 0.0000**	0.0110 ± 0.0000	0.0110 ± 0.0000	**0.0109 ± 0.0000**
scene	0.0965 ± 0.0006	0.0926 ± 0.0004	0.0915 ± 0.0004	**0.0902 ± 0.0004**
yeast	0.1581 ± 0.0005	0.1586 ± 0.0005	0.1573 ± 0.0004	**0.1543 ± 0.0004**

$p = 0.7$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.0483 ± 0.0003**	0.0499 ± 0.0002	0.0502 ± 0.0002	0.0490 ± 0.0002
Corel5k	**0.0029 ± 0.0000**	0.0031 ± 0.0000	0.0031 ± 0.0000	0.0031 ± 0.0000
emotions	0.1007 ± 0.0013	0.1017 ± 0.0011	0.0993 ± 0.0015	**0.0951 ± 0.0013**
enron	0.0311 ± 0.0002	0.0311 ± 0.0002	0.0298 ± 0.0002	**0.0266 ± 0.0002**
mediamill	**0.0130 ± 0.0000**	0.0130 ± 0.0000	0.0130 ± 0.0000	0.0130 ± 0.0000
medical	**0.0099 ± 0.0002**	0.0106 ± 0.0002	0.0105 ± 0.0001	**0.0097 ± 0.0001**
nuswide	**0.0066 ± 0.0000**	**0.0066 ± 0.0000**	**0.0066 ± 0.0000**	**0.0066 ± 0.0000**
scene	0.0562 ± 0.0004	**0.0544 ± 0.0003**	**0.0542 ± 0.0005**	**0.0538 ± 0.0005**
yeast	**0.0920 ± 0.0004**	**0.0918 ± 0.0003**	**0.0921 ± 0.0004**	**0.0915 ± 0.0004**

Table A.3: DPP vs. O-BR on Noisy Data, F1 loss

$p = 0.3$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.823 ± 0.002**	**0.823 ± 0.002**	0.830 ± 0.003	0.837 ± 0.002
Corel5k	0.949 ± 0.001	**0.945 ± 0.001**	0.949 ± 0.001	0.949 ± 0.001
emotions	**0.697 ± 0.005**	0.740 ± 0.008	0.767 ± 0.006	0.857 ± 0.003
enron	**0.694 ± 0.002**	**0.697 ± 0.003**	**0.696 ± 0.002**	0.716 ± 0.002
mediamill	**0.640 ± 0.001**	**0.640 ± 0.001**	**0.639 ± 0.001**	**0.639 ± 0.001**
medical	**0.550 ± 0.004**	**0.544 ± 0.006**	0.577 ± 0.004	0.645 ± 0.006
nuswide	**0.627 ± 0.001**	**0.627 ± 0.000**	**0.627 ± 0.000**	**0.626 ± 0.000**
scene	**0.626 ± 0.001**	0.695 ± 0.003	0.706 ± 0.003	0.717 ± 0.003
yeast	**0.669 ± 0.002**	0.678 ± 0.004	0.711 ± 0.004	0.733 ± 0.005

$p = 0.5$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.925 ± 0.002**	**0.925 ± 0.002**	0.932 ± 0.002	0.947 ± 0.002
Corel5k	0.898 ± 0.001	0.899 ± 0.001	**0.897 ± 0.001**	0.898 ± 0.001
emotions	**0.694 ± 0.004**	**0.691 ± 0.003**	**0.703 ± 0.006**	**0.699 ± 0.004**
enron	0.768 ± 0.001	**0.765 ± 0.003**	**0.764 ± 0.003**	0.772 ± 0.002
mediamill	**0.831 ± 0.001**	**0.830 ± 0.001**	**0.830 ± 0.001**	**0.830 ± 0.001**
medical	0.570 ± 0.002	0.563 ± 0.005	0.569 ± 0.004	**0.561 ± 0.003**
nuswide	0.537 ± 0.000	0.537 ± 0.000	**0.536 ± 0.000**	0.536 ± 0.000
scene	0.533 ± 0.003	0.525 ± 0.002	**0.519 ± 0.003**	0.524 ± 0.003
yeast	**0.853 ± 0.002**	**0.850 ± 0.002**	0.860 ± 0.002	0.876 ± 0.004

$p = 0.7$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.985 ± 0.001**	**0.983 ± 0.001**	**0.984 ± 0.001**	0.987 ± 0.000
Corel5k	**0.716 ± 0.002**	**0.713 ± 0.001**	**0.714 ± 0.001**	**0.712 ± 0.002**
emotions	0.493 ± 0.006	0.486 ± 0.005	0.489 ± 0.006	**0.477 ± 0.004**
enron	0.734 ± 0.003	0.729 ± 0.003	0.731 ± 0.002	**0.711 ± 0.003**
mediamill	**0.714 ± 0.001**	**0.715 ± 0.000**	**0.714 ± 0.000**	**0.715 ± 0.001**
medical	0.398 ± 0.007	0.401 ± 0.005	0.391 ± 0.004	**0.377 ± 0.004**
nuswide	**0.386 ± 0.001**	**0.386 ± 0.000**	**0.386 ± 0.000**	**0.386 ± 0.000**
scene	0.328 ± 0.003	0.323 ± 0.002	**0.316 ± 0.002**	**0.313 ± 0.002**
yeast	**0.746 ± 0.002**	**0.748 ± 0.002**	**0.747 ± 0.002**	**0.748 ± 0.002**

The results from Table A.2 to Table A.4 show that DPP outperforms O-BR as the value of p increases with respect ti Hamming loss, F1 loss and Accuracy loss, demonstrating the robustness of DPP. On the otter hand, the results in in Table A.5 show that, while DPP cannot outperform O-BR with respect to Normalized rank loss, DPP do start to perform competitively as the value of p increases. The observation again demonstrates that DPP indeed suffers less from noisy labels comparing to O-BR due to the incorporation with LSDR.

A.5.3 Experiments on Basis Drifting

We report the complete results of comparison between DPP using (1) PBC, (2) PBT, and (3) nothing in Table A.6 with respect to Hamming loss, where the best values (the lowest) are marked in bold. to further understand the behavior of basis drifting and the effectiveness of PBC and PBT for CS-DPP, we further compare CS-DPP coupled with PBC/PBT/none on F1 loss, Accuracy loss and Normalized rank loss. The results are reported in Table A.7 to Table A.9. From Table A.7 to Table A.9 we can draw the same conclusion as Table A.6. That is, CS-DPP with either PBT or PBC greatly outperforms CS-DPP that neglects the basis drifting. , and CS-DPP with PBT performs competitively with CS-DPP with PBC.

doi:10.6342/NTU201701149

Table A.4: DPP vs. O-BR on Noisy Data, Accuracy loss

$p = 0.3$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.898 ± 0.001**	**0.897 ± 0.001**	**0.900 ± 0.001**	0.911 ± 0.002
Corel5k	**0.957 ± 0.001**	**0.957 ± 0.001**	**0.958 ± 0.000**	0.960 ± 0.001
emotions	**0.719 ± 0.005**	0.764 ± 0.004	0.783 ± 0.002	0.858 ± 0.004
enron	**0.766 ± 0.002**	**0.770 ± 0.002**	**0.767 ± 0.002**	0.784 ± 0.002
mediamill	**0.721 ± 0.000**	**0.721 ± 0.000**	**0.721 ± 0.001**	**0.720 ± 0.001**
medical	**0.563 ± 0.005**	**0.556 ± 0.003**	0.589 ± 0.005	0.651 ± 0.004
nuswide	0.632 ± 0.000	0.633 ± 0.000	**0.632 ± 0.000**	0.632 ± 0.000
scene	**0.628 ± 0.003**	0.698 ± 0.003	0.710 ± 0.004	0.715 ± 0.002
yeast	**0.755 ± 0.002**	0.762 ± 0.003	0.783 ± 0.003	0.798 ± 0.009

$p = 0.5$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.961 ± 0.001**	**0.962 ± 0.001**	**0.961 ± 0.001**	0.970 ± 0.001
Corel5k	**0.902 ± 0.001**	**0.900 ± 0.001**	**0.900 ± 0.001**	**0.902 ± 0.001**
emotions	0.698 ± 0.004	0.706 ± 0.004	0.706 ± 0.004	**0.692 ± 0.004**
enron	**0.809 ± 0.002**	**0.809 ± 0.001**	**0.806 ± 0.002**	**0.810 ± 0.002**
mediamill	**0.840 ± 0.001**	**0.839 ± 0.001**	**0.840 ± 0.001**	**0.840 ± 0.001**
medical	0.568 ± 0.004	0.577 ± 0.004	0.570 ± 0.004	**0.565 ± 0.003**
nuswide	**0.537 ± 0.000**	**0.537 ± 0.000**	**0.536 ± 0.000**	**0.537 ± 0.000**
scene	0.533 ± 0.003	0.524 ± 0.002	0.522 ± 0.003	**0.515 ± 0.004**
yeast	**0.875 ± 0.001**	**0.873 ± 0.002**	**0.878 ± 0.002**	0.890 ± 0.002

$p = 0.7$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.990 ± 0.001**	0.991 ± 0.000	0.991 ± 0.000	0.992 ± 0.001
Corel5k	**0.716 ± 0.002**	**0.714 ± 0.002**	**0.715 ± 0.002**	**0.714 ± 0.002**
emotions	0.490 ± 0.006	0.493 ± 0.005	0.491 ± 0.005	**0.474 ± 0.004**
enron	0.753 ± 0.003	0.745 ± 0.002	0.742 ± 0.003	**0.726 ± 0.003**
mediamill	0.715 ± 0.000	**0.714 ± 0.000**	0.714 ± 0.001	0.715 ± 0.001
medical	0.404 ± 0.004	0.398 ± 0.005	0.399 ± 0.004	**0.377 ± 0.005**
nuswide	**0.386 ± 0.001**	**0.385 ± 0.000**	**0.386 ± 0.000**	**0.385 ± 0.001**
scene	0.328 ± 0.002	**0.321 ± 0.002**	**0.317 ± 0.002**	**0.318 ± 0.002**
yeast	**0.747 ± 0.002**	**0.747 ± 0.002**	**0.746 ± 0.002**	**0.746 ± 0.002**

Table A.5: DPP vs. O-BR on Noisy Data, Normalized rank loss

$p = 0.3$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.453 ± 0.001**	0.455 ± 0.001	0.454 ± 0.001	0.458 ± 0.001
Corel5k	**0.496 ± 0.000**	**0.496 ± 0.001**	**0.497 ± 0.000**	0.498 ± 0.000
emotions	**0.480 ± 0.004**	0.504 ± 0.005	0.527 ± 0.003	0.570 ± 0.002
enron	**0.386 ± 0.001**	**0.388 ± 0.001**	0.397 ± 0.001	0.414 ± 0.001
mediamill	**0.403 ± 0.000**	**0.402 ± 0.000**	**0.403 ± 0.000**	**0.402 ± 0.000**
medical	**0.448 ± 0.004**	**0.446 ± 0.002**	0.469 ± 0.005	0.538 ± 0.003
nuswide	**0.668 ± 0.000**	**0.667 ± 0.000**	**0.667 ± 0.000**	**0.668 ± 0.000**
scene	**0.560 ± 0.002**	0.622 ± 0.002	0.632 ± 0.002	0.643 ± 0.001
yeast	**0.406 ± 0.001**	0.413 ± 0.002	0.427 ± 0.002	0.443 ± 0.002

$p = 0.5$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	0.483 ± 0.001	**0.480 ± 0.001**	0.483 ± 0.000	0.485 ± 0.001
Corel5k	**0.543 ± 0.000**	**0.544 ± 0.001**	**0.544 ± 0.000**	**0.543 ± 0.001**
emotions	**0.633 ± 0.003**	0.640 ± 0.003	0.646 ± 0.002	0.650 ± 0.002
enron	0.491 ± 0.002	**0.488 ± 0.001**	0.491 ± 0.001	0.510 ± 0.002
mediamill	**0.548 ± 0.000**	0.550 ± 0.000	0.550 ± 0.001	0.549 ± 0.000
medical	**0.655 ± 0.005**	**0.661 ± 0.003**	**0.664 ± 0.005**	0.690 ± 0.003
nuswide	0.730 ± 0.000	0.730 ± 0.000	0.730 ± 0.000	**0.730 ± 0.000**
scene	**0.718 ± 0.002**	0.731 ± 0.002	0.739 ± 0.001	0.740 ± 0.001
yeast	**0.518 ± 0.001**	**0.520 ± 0.001**	0.524 ± 0.001	0.531 ± 0.002

$p = 0.7$

Dataset	O-BR	DPP-50	DPP-25	DPP-10
CAL500	**0.495 ± 0.000**	**0.495 ± 0.000**	0.495 ± 0.000	0.496 ± 0.000
Corel5k	**0.647 ± 0.001**	**0.646 ± 0.001**	**0.647 ± 0.001**	**0.646 ± 0.001**
emotions	**0.759 ± 0.002**	**0.758 ± 0.002**	**0.757 ± 0.002**	**0.763 ± 0.002**
enron	**0.634 ± 0.002**	**0.633 ± 0.002**	**0.635 ± 0.002**	0.644 ± 0.001
mediamill	**0.643 ± 0.000**	**0.643 ± 0.000**	**0.643 ± 0.000**	**0.643 ± 0.000**
medical	**0.814 ± 0.003**	**0.812 ± 0.003**	**0.815 ± 0.002**	**0.819 ± 0.003**
nuswide	0.808 ± 0.000	**0.807 ± 0.000**	0.807 ± 0.000	0.807 ± 0.000
scene	**0.841 ± 0.001**	**0.841 ± 0.001**	**0.842 ± 0.001**	**0.842 ± 0.001**
yeast	**0.625 ± 0.001**	**0.627 ± 0.001**	**0.627 ± 0.001**	**0.626 ± 0.001**

A.5.4 Experiments on Cost-sensitivity

We report the complete results of on all datasets with respect to all four cost functions in Table A.10 to Table A.13, where the best values (the lowest) are marked in bold. The conclusion can be drawn similarly to that is drawn from the discussion in the main paper.

doi:10.6342/NTU201701149

Table A.6: CS-DPP with PBC vs. PBT vs. None, Hamming loss

$M = 10\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.4464 ± 0.0074	$\mathbf{0.1443 \pm 0.0001}$	0.1454 ± 0.0002
Corel5k	0.4814 ± 0.0063	$\mathbf{0.0099 \pm 0.0000}$	0.0100 ± 0.0000
emotions	0.4787 ± 0.0039	0.3419 ± 0.0033	$\mathbf{0.3301 \pm 0.0012}$
enron	0.4030 ± 0.0160	$\mathbf{0.0560 \pm 0.0001}$	0.0565 ± 0.0001
mediamill	0.4936 ± 0.0016	0.0309 ± 0.0000	$\mathbf{0.0308 \pm 0.0000}$
medical	0.1923 ± 0.0352	0.0242 ± 0.0001	$\mathbf{0.0204 \pm 0.0002}$
nuswide	0.4975 ± 0.0006	0.0201 ± 0.0000	$\mathbf{0.0201 \pm 0.0000}$
scene	0.4609 ± 0.0080	$\mathbf{0.1796 \pm 0.0001}$	$\mathbf{0.1797 \pm 0.0001}$
yeast	0.4979 ± 0.0015	$\mathbf{0.2294 \pm 0.0010}$	$\mathbf{0.2307 \pm 0.0011}$

$M = 25\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.4374 ± 0.0100	$\mathbf{0.1471 \pm 0.0002}$	0.1476 ± 0.0001
Corel5k	0.4997 ± 0.0018	$\mathbf{0.0100 \pm 0.0000}$	0.0101 ± 0.0000
emotions	0.4988 ± 0.0022	$\mathbf{0.2768 \pm 0.0051}$	$\mathbf{0.2819 \pm 0.0036}$
enron	0.4844 ± 0.0050	$\mathbf{0.0581 \pm 0.0002}$	0.0601 ± 0.0002
mediamill	0.4917 ± 0.0015	$\mathbf{0.0307 \pm 0.0000}$	$\mathbf{0.0307 \pm 0.0000}$
medical	0.4493 ± 0.0161	0.0171 ± 0.0002	$\mathbf{0.0152 \pm 0.0001}$
nuswide	0.4978 ± 0.0007	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.0201 \pm 0.0000}$
scene	0.5002 ± 0.0012	$\mathbf{0.1787 \pm 0.0014}$	$\mathbf{0.1797 \pm 0.0014}$
yeast	0.4992 ± 0.0014	$\mathbf{0.2139 \pm 0.0006}$	$\mathbf{0.2144 \pm 0.0005}$

$M = 50\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.4141 ± 0.0176	$\mathbf{0.1487 \pm 0.0002}$	$\mathbf{0.1490 \pm 0.0002}$
Corel5k	0.5014 ± 0.0017	$\mathbf{0.0101 \pm 0.0000}$	$\mathbf{0.0101 \pm 0.0000}$
emotions	0.4941 ± 0.0029	$\mathbf{0.2308 \pm 0.0014}$	$\mathbf{0.2306 \pm 0.0012}$
enron	0.4953 ± 0.0016	$\mathbf{0.0626 \pm 0.0002}$	0.0643 ± 0.0001
mediamill	0.4907 ± 0.0018	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.0307 \pm 0.0000}$
medical	0.4177 ± 0.0370	0.0136 ± 0.0001	$\mathbf{0.0130 \pm 0.0001}$
nuswide	0.4972 ± 0.0007	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.0201 \pm 0.0000}$
scene	0.5015 ± 0.0012	$\mathbf{0.1731 \pm 0.0010}$	$\mathbf{0.1720 \pm 0.0015}$
yeast	0.4982 ± 0.0011	$\mathbf{0.2077 \pm 0.0003}$	$\mathbf{0.2079 \pm 0.0003}$

Table A.7: CS-DPP with PBC vs. PBT vs. None, F1 loss

$M = 10\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.733 ± 0.001	$\mathbf{0.601 \pm 0.001}$	$\mathbf{0.603 \pm 0.001}$
Corel5k	0.957 ± 0.000	0.853 ± 0.001	$\mathbf{0.850 \pm 0.001}$
emotions	0.618 ± 0.004	$\mathbf{0.445 \pm 0.003}$	$\mathbf{0.450 \pm 0.007}$
enron	0.802 ± 0.002	0.534 ± 0.002	$\mathbf{0.528 \pm 0.002}$
mediamill	0.692 ± 0.016	$\mathbf{0.460 \pm 0.000}$	$\mathbf{0.460 \pm 0.000}$
medical	0.896 ± 0.002	0.554 ± 0.012	$\mathbf{0.508 \pm 0.006}$
nuswide	0.933 ± 0.001	$\mathbf{0.649 \pm 0.000}$	$\mathbf{0.648 \pm 0.000}$
scene	0.761 ± 0.003	$\mathbf{0.723 \pm 0.002}$	$\mathbf{0.724 \pm 0.002}$
yeast	0.616 ± 0.002	$\mathbf{0.435 \pm 0.004}$	$\mathbf{0.433 \pm 0.003}$

$M = 25\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.732 ± 0.002	$\mathbf{0.604 \pm 0.001}$	$\mathbf{0.602 \pm 0.001}$
Corel5k	0.965 ± 0.000	$\mathbf{0.845 \pm 0.000}$	$\mathbf{0.844 \pm 0.000}$
emotions	0.631 ± 0.004	$\mathbf{0.401 \pm 0.003}$	$\mathbf{0.398 \pm 0.004}$
enron	0.812 ± 0.002	$\mathbf{0.517 \pm 0.001}$	$\mathbf{0.519 \pm 0.001}$
mediamill	0.842 ± 0.009	$\mathbf{0.458 \pm 0.000}$	$\mathbf{0.457 \pm 0.000}$
medical	0.902 ± 0.002	0.338 ± 0.005	$\mathbf{0.316 \pm 0.004}$
nuswide	0.930 ± 0.003	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.648 \pm 0.000}$
scene	0.747 ± 0.002	$\mathbf{0.632 \pm 0.003}$	$\mathbf{0.631 \pm 0.004}$
yeast	0.622 ± 0.001	0.389 ± 0.001	$\mathbf{0.385 \pm 0.001}$

$M = 50\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.735 ± 0.002	$\mathbf{0.602 \pm 0.001}$	$\mathbf{0.602 \pm 0.001}$
Corel5k	0.969 ± 0.000	$\mathbf{0.843 \pm 0.000}$	$\mathbf{0.842 \pm 0.001}$
emotions	0.631 ± 0.003	$\mathbf{0.381 \pm 0.002}$	$\mathbf{0.377 \pm 0.002}$
enron	0.821 ± 0.003	$\mathbf{0.523 \pm 0.001}$	$\mathbf{0.522 \pm 0.001}$
mediamill	0.895 ± 0.008	$\mathbf{0.457 \pm 0.000}$	$\mathbf{0.457 \pm 0.000}$
medical	0.907 ± 0.002	$\mathbf{0.252 \pm 0.002}$	$\mathbf{0.250 \pm 0.002}$
nuswide	0.940 ± 0.004	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.648 \pm 0.000}$
scene	0.745 ± 0.001	$\mathbf{0.554 \pm 0.003}$	$\mathbf{0.558 \pm 0.003}$
yeast	0.630 ± 0.001	$\mathbf{0.382 \pm 0.001}$	$\mathbf{0.382 \pm 0.001}$

Table A.8: CS-DPP with PBC vs. PBT vs. None, Accuracy loss

$M = 10\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.843 ± 0.001	$\mathbf{0.749 \pm 0.001}$	$\mathbf{0.748 \pm 0.001}$
Corel5k	0.980 ± 0.000	0.912 ± 0.001	$\mathbf{0.910 \pm 0.000}$
emotions	0.696 ± 0.005	$\mathbf{0.563 \pm 0.007}$	$\mathbf{0.560 \pm 0.009}$
enron	0.875 ± 0.001	$\mathbf{0.642 \pm 0.002}$	$\mathbf{0.638 \pm 0.001}$
mediamill	0.728 ± 0.001	0.583 ± 0.000	$\mathbf{0.582 \pm 0.000}$
medical	0.932 ± 0.003	0.583 ± 0.008	$\mathbf{0.549 \pm 0.007}$
nuswide	0.959 ± 0.001	$\mathbf{0.675 \pm 0.000}$	$\mathbf{0.675 \pm 0.000}$
scene	0.825 ± 0.002	$\mathbf{0.798 \pm 0.003}$	$\mathbf{0.796 \pm 0.002}$
yeast	0.727 ± 0.001	$\mathbf{0.549 \pm 0.003}$	$\mathbf{0.541 \pm 0.003}$

$M = 25\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.846 ± 0.002	$\mathbf{0.750 \pm 0.001}$	$\mathbf{0.751 \pm 0.001}$
Corel5k	0.983 ± 0.000	$\mathbf{0.905 \pm 0.000}$	$\mathbf{0.904 \pm 0.000}$
emotions	0.722 ± 0.003	$\mathbf{0.513 \pm 0.003}$	$\mathbf{0.509 \pm 0.003}$
enron	0.884 ± 0.001	$\mathbf{0.633 \pm 0.001}$	$\mathbf{0.633 \pm 0.001}$
mediamill	0.759 ± 0.009	$\mathbf{0.581 \pm 0.000}$	$\mathbf{0.580 \pm 0.000}$
medical	0.931 ± 0.004	0.374 ± 0.004	$\mathbf{0.360 \pm 0.004}$
nuswide	0.964 ± 0.001	$\mathbf{0.675 \pm 0.000}$	$\mathbf{0.675 \pm 0.000}$
scene	0.830 ± 0.002	$\mathbf{0.692 \pm 0.003}$	$\mathbf{0.697 \pm 0.004}$
yeast	0.737 ± 0.001	$\mathbf{0.495 \pm 0.001}$	$\mathbf{0.497 \pm 0.001}$

$M = 50\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.844 ± 0.002	$\mathbf{0.752 \pm 0.001}$	$\mathbf{0.751 \pm 0.001}$
Corel5k	0.986 ± 0.000	$\mathbf{0.901 \pm 0.000}$	0.903 ± 0.001
emotions	0.729 ± 0.002	$\mathbf{0.481 \pm 0.002}$	$\mathbf{0.481 \pm 0.002}$
enron	0.889 ± 0.002	$\mathbf{0.636 \pm 0.001}$	$\mathbf{0.636 \pm 0.001}$
mediamill	0.838 ± 0.019	0.581 ± 0.000	$\mathbf{0.581 \pm 0.000}$
medical	0.944 ± 0.002	$\mathbf{0.303 \pm 0.002}$	$\mathbf{0.299 \pm 0.002}$
nuswide	0.964 ± 0.002	$\mathbf{0.674 \pm 0.000}$	$\mathbf{0.675 \pm 0.000}$
scene	0.832 ± 0.001	$\mathbf{0.626 \pm 0.004}$	$\mathbf{0.623 \pm 0.004}$
yeast	0.745 ± 0.001	$\mathbf{0.493 \pm 0.001}$	$\mathbf{0.492 \pm 0.001}$

Table A.9: CS-DPP with PBC vs. PBT vs. None, Normalized rank loss

$M = 10\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.393 ± 0.002	$\mathbf{0.137 \pm 0.001}$	0.144 ± 0.002
Corel5k	0.357 ± 0.001	0.248 ± 0.001	$\mathbf{0.237 \pm 0.001}$
emotions	0.376 ± 0.008	$\mathbf{0.159 \pm 0.021}$	$\mathbf{0.133 \pm 0.023}$
enron	0.385 ± 0.002	$\mathbf{0.124 \pm 0.003}$	0.132 ± 0.001
mediamill	0.416 ± 0.004	$\mathbf{0.066 \pm 0.002}$	0.072 ± 0.002
medical	0.346 ± 0.003	0.132 ± 0.005	$\mathbf{0.096 \pm 0.003}$
nuswide	0.520 ± 0.001	$\mathbf{0.356 \pm 0.001}$	$\mathbf{0.358 \pm 0.001}$
scene	0.362 ± 0.007	$\mathbf{0.264 \pm 0.012}$	$\mathbf{0.231 \pm 0.016}$
yeast	0.422 ± 0.003	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.003 \pm 0.000}$

$M = 25\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.392 ± 0.002	$\mathbf{0.151 \pm 0.002}$	$\mathbf{0.150 \pm 0.002}$
Corel5k	0.366 ± 0.001	0.223 ± 0.001	$\mathbf{0.220 \pm 0.001}$
emotions	0.420 ± 0.005	$\mathbf{0.078 \pm 0.016}$	$\mathbf{0.046 \pm 0.015}$
enron	0.386 ± 0.002	$\mathbf{0.136 \pm 0.002}$	$\mathbf{0.135 \pm 0.001}$
mediamill	0.429 ± 0.001	$\mathbf{0.070 \pm 0.000}$	$\mathbf{0.068 \pm 0.000}$
medical	0.361 ± 0.004	0.043 ± 0.003	$\mathbf{0.036 \pm 0.002}$
nuswide	0.523 ± 0.000	0.334 ± 0.001	$\mathbf{0.329 \pm 0.001}$
scene	0.373 ± 0.004	0.185 ± 0.013	$\mathbf{0.142 \pm 0.011}$
yeast	0.424 ± 0.002	$\mathbf{0.017 \pm 0.001}$	$\mathbf{0.016 \pm 0.001}$

$M = 50\%$ of K

Dataset	CS-DPP-None	CS-DPP-PBT	CS-DPP-PBC
CAL500	0.398 ± 0.002	$\mathbf{0.154 \pm 0.001}$	$\mathbf{0.151 \pm 0.001}$
Corel5k	0.369 ± 0.001	$\mathbf{0.214 \pm 0.001}$	$\mathbf{0.213 \pm 0.000}$
emotions	0.386 ± 0.004	$\mathbf{0.034 \pm 0.003}$	$\mathbf{0.033 \pm 0.003}$
enron	0.385 ± 0.002	$\mathbf{0.130 \pm 0.001}$	$\mathbf{0.129 \pm 0.001}$
mediamill	0.426 ± 0.001	0.062 ± 0.000	$\mathbf{0.059 \pm 0.000}$
medical	0.368 ± 0.002	$\mathbf{0.021 \pm 0.001}$	$\mathbf{0.019 \pm 0.001}$
nuswide	0.528 ± 0.001	0.307 ± 0.000	$\mathbf{0.304 \pm 0.000}$
scene	0.385 ± 0.002	0.125 ± 0.005	$\mathbf{0.104 \pm 0.009}$
yeast	0.413 ± 0.001	$\mathbf{0.024 \pm 0.001}$	$\mathbf{0.026 \pm 0.001}$

Table A.10: CS-DPP vs. others, Hamming loss

$M = 10\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.1610 ± 0.0006	0.4042 ± 0.0052	$\mathbf{0.1453 \pm 0.0001}$	$\mathbf{0.1454 \pm 0.0002}$
Corel5k	0.0117 ± 0.0000	0.3734 ± 0.0044	$\mathbf{0.0100 \pm 0.0000}$	$\mathbf{0.0100 \pm 0.0000}$
emotions	$\mathbf{0.3338 \pm 0.0073}$	0.3847 ± 0.0099	$\mathbf{0.3335 \pm 0.0042}$	$\mathbf{0.3301 \pm 0.0012}$
enron	0.0739 ± 0.0006	0.3907 ± 0.0090	$\mathbf{0.0563 \pm 0.0001}$	$\mathbf{0.0565 \pm 0.0001}$
mediamill	0.0485 ± 0.0011	0.3737 ± 0.0070	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.0308 \pm 0.0000}$
medical	0.0272 ± 0.0006	0.3674 ± 0.0093	$\mathbf{0.0204 \pm 0.0002}$	$\mathbf{0.0204 \pm 0.0002}$
nuswide	0.0239 ± 0.0004	0.3707 ± 0.0107	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.0201 \pm 0.0000}$
scene	0.2168 ± 0.0047	0.3711 ± 0.0172	$\mathbf{0.1797 \pm 0.0001}$	$\mathbf{0.1797 \pm 0.0001}$
yeast	0.3077 ± 0.0021	0.4162 ± 0.0096	$\mathbf{0.2314 \pm 0.0014}$	$\mathbf{0.2307 \pm 0.0011}$

$M = 25\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.8221 ± 0.0018	0.3374 ± 0.0057	$\mathbf{0.1479 \pm 0.0002}$	$\mathbf{0.1476 \pm 0.0001}$
Corel5k	0.9853 ± 0.0002	0.2857 ± 0.0042	$\mathbf{0.0101 \pm 0.0000}$	$\mathbf{0.0101 \pm 0.0000}$
emotions	0.5810 ± 0.0085	0.3586 ± 0.0096	$\mathbf{0.2807 \pm 0.0038}$	$\mathbf{0.2819 \pm 0.0036}$
enron	0.8025 ± 0.0023	0.3150 ± 0.0109	$\mathbf{0.0599 \pm 0.0001}$	$\mathbf{0.0601 \pm 0.0002}$
mediamill	0.8832 ± 0.0020	0.2854 ± 0.0076	$\mathbf{0.0307 \pm 0.0000}$	$\mathbf{0.0307 \pm 0.0000}$
medical	0.8173 ± 0.0018	0.2921 ± 0.0134	$\mathbf{0.0150 \pm 0.0001}$	$\mathbf{0.0152 \pm 0.0001}$
nuswide	0.8721 ± 0.0031	0.2843 ± 0.0119	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.0201 \pm 0.0000}$
scene	0.6286 ± 0.0076	0.2985 ± 0.0205	$\mathbf{0.1788 \pm 0.0016}$	$\mathbf{0.1797 \pm 0.0014}$
yeast	0.5961 ± 0.0060	0.3464 ± 0.0111	$\mathbf{0.2136 \pm 0.0004}$	$\mathbf{0.2144 \pm 0.0005}$

$M = 50\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.8315 ± 0.0011	0.2620 ± 0.0045	$\mathbf{0.1487 \pm 0.0002}$	$\mathbf{0.1490 \pm 0.0002}$
Corel5k	0.9860 ± 0.0002	0.1687 ± 0.0033	$\mathbf{0.0101 \pm 0.0000}$	$\mathbf{0.0101 \pm 0.0000}$
emotions	0.5079 ± 0.0096	0.3357 ± 0.0086	$\mathbf{0.2276 \pm 0.0010}$	$\mathbf{0.2306 \pm 0.0012}$
enron	0.7172 ± 0.0041	0.2205 ± 0.0079	$\mathbf{0.0642 \pm 0.0002}$	$\mathbf{0.0643 \pm 0.0001}$
mediamill	0.8998 ± 0.0049	0.1775 ± 0.0107	$\mathbf{0.0307 \pm 0.0000}$	$\mathbf{0.0307 \pm 0.0000}$
medical	0.6863 ± 0.0047	0.1673 ± 0.0116	$\mathbf{0.0132 \pm 0.0001}$	$\mathbf{0.0130 \pm 0.0001}$
nuswide	0.8200 ± 0.0053	0.1767 ± 0.0097	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.0201 \pm 0.0000}$
scene	0.5023 ± 0.0184	0.2678 ± 0.0116	$\mathbf{0.1711 \pm 0.0020}$	$\mathbf{0.1720 \pm 0.0015}$
yeast	0.4504 ± 0.0104	0.2926 ± 0.0060	$\mathbf{0.2080 \pm 0.0003}$	$\mathbf{0.2079 \pm 0.0003}$

Table A.11: CS-DPP vs. others, F1 loss

$M = 10\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.953 ± 0.003	0.750 ± 0.004	0.654 ± 0.002	$\mathbf{0.603 \pm 0.001}$
Corel5k	0.926 ± 0.002	0.980 ± 0.001	0.918 ± 0.001	$\mathbf{0.850 \pm 0.001}$
emotions	0.900 ± 0.014	0.621 ± 0.033	$\mathbf{0.428 \pm 0.003}$	0.450 ± 0.007
enron	0.885 ± 0.010	0.867 ± 0.007	0.552 ± 0.003	$\mathbf{0.528 \pm 0.002}$
mediamill	0.821 ± 0.025	0.899 ± 0.007	0.474 ± 0.000	$\mathbf{0.460 \pm 0.000}$
medical	0.819 ± 0.016	0.924 ± 0.005	0.602 ± 0.008	$\mathbf{0.508 \pm 0.006}$
nuswide	0.746 ± 0.005	0.956 ± 0.002	0.673 ± 0.000	$\mathbf{0.648 \pm 0.000}$
scene	0.920 ± 0.010	$\mathbf{0.743 \pm 0.030}$	0.999 ± 0.000	$\mathbf{0.724 \pm 0.002}$
yeast	0.885 ± 0.018	0.596 ± 0.008	0.463 ± 0.005	$\mathbf{0.433 \pm 0.003}$

$M = 25\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.739 ± 0.000	0.734 ± 0.006	0.653 ± 0.001	$\mathbf{0.602 \pm 0.001}$
Corel5k	0.981 ± 0.000	0.978 ± 0.001	0.912 ± 0.001	$\mathbf{0.844 \pm 0.000}$
emotions	0.535 ± 0.006	0.591 ± 0.029	0.446 ± 0.011	$\mathbf{0.398 \pm 0.004}$
enron	0.878 ± 0.001	0.830 ± 0.007	0.528 ± 0.002	$\mathbf{0.519 \pm 0.001}$
mediamill	0.915 ± 0.000	0.879 ± 0.007	0.472 ± 0.000	$\mathbf{0.457 \pm 0.000}$
medical	0.943 ± 0.001	0.894 ± 0.006	0.377 ± 0.004	$\mathbf{0.316 \pm 0.004}$
nuswide	0.957 ± 0.000	0.951 ± 0.003	0.672 ± 0.000	$\mathbf{0.648 \pm 0.000}$
scene	0.676 ± 0.004	0.709 ± 0.025	0.911 ± 0.005	$\mathbf{0.631 \pm 0.004}$
yeast	0.564 ± 0.006	0.583 ± 0.012	0.423 ± 0.002	$\mathbf{0.385 \pm 0.001}$

$M = 50\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.740 ± 0.000	0.703 ± 0.006	0.654 ± 0.001	$\mathbf{0.602 \pm 0.001}$
Corel5k	0.981 ± 0.000	0.971 ± 0.002	0.911 ± 0.001	$\mathbf{0.842 \pm 0.001}$
emotions	0.538 ± 0.007	0.545 ± 0.020	0.470 ± 0.007	$\mathbf{0.377 \pm 0.002}$
enron	0.872 ± 0.001	0.779 ± 0.009	0.527 ± 0.001	$\mathbf{0.522 \pm 0.001}$
mediamill	0.916 ± 0.001	0.825 ± 0.014	0.472 ± 0.000	$\mathbf{0.457 \pm 0.000}$
medical	0.937 ± 0.001	0.836 ± 0.012	0.280 ± 0.003	$\mathbf{0.250 \pm 0.002}$
nuswide	0.957 ± 0.000	0.942 ± 0.005	0.672 ± 0.000	$\mathbf{0.648 \pm 0.000}$
scene	0.664 ± 0.008	0.633 ± 0.017	0.809 ± 0.008	$\mathbf{0.558 \pm 0.003}$
yeast	0.524 ± 0.009	0.500 ± 0.013	0.406 ± 0.001	$\mathbf{0.382 \pm 0.001}$

doi:10.6342/NTU201701149

Table A.12: CS-DPP vs. others, Accuracy loss

$M = 10\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.971 ± 0.002	0.858 ± 0.004	0.787 ± 0.001	$\mathbf{0.748 \pm 0.001}$
Corel5k	0.949 ± 0.001	0.990 ± 0.000	0.943 ± 0.000	$\mathbf{0.910 \pm 0.000}$
emotions	0.924 ± 0.009	0.683 ± 0.021	$\mathbf{0.558 \pm 0.004}$	$\mathbf{0.560 \pm 0.009}$
enron	0.927 ± 0.009	0.923 ± 0.004	0.646 ± 0.002	$\mathbf{0.638 \pm 0.001}$
mediamill	0.868 ± 0.014	0.950 ± 0.003	0.594 ± 0.000	$\mathbf{0.582 \pm 0.000}$
medical	0.840 ± 0.017	0.959 ± 0.003	0.628 ± 0.008	$\mathbf{0.549 \pm 0.007}$
nuswide	0.741 ± 0.003	0.973 ± 0.002	0.691 ± 0.000	$\mathbf{0.675 \pm 0.000}$
scene	0.902 ± 0.009	$\mathbf{0.782 \pm 0.009}$	0.999 ± 0.000	$\mathbf{0.796 \pm 0.002}$
yeast	0.926 ± 0.014	0.702 ± 0.014	0.597 ± 0.005	$\mathbf{0.541 \pm 0.003}$

$M = 25\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.849 ± 0.000	0.843 ± 0.004	0.784 ± 0.001	$\mathbf{0.751 \pm 0.001}$
Corel5k	0.991 ± 0.000	0.989 ± 0.000	0.939 ± 0.000	$\mathbf{0.904 \pm 0.000}$
emotions	0.682 ± 0.006	0.654 ± 0.013	0.531 ± 0.005	$\mathbf{0.509 \pm 0.003}$
enron	0.934 ± 0.000	0.912 ± 0.005	$\mathbf{0.634 \pm 0.002}$	$\mathbf{0.633 \pm 0.001}$
mediamill	0.956 ± 0.000	0.940 ± 0.003	0.592 ± 0.000	$\mathbf{0.580 \pm 0.000}$
medical	0.970 ± 0.000	0.952 ± 0.003	0.402 ± 0.005	$\mathbf{0.360 \pm 0.004}$
nuswide	0.978 ± 0.000	0.968 ± 0.002	0.690 ± 0.000	$\mathbf{0.675 \pm 0.000}$
scene	0.803 ± 0.002	0.746 ± 0.008	0.927 ± 0.003	$\mathbf{0.697 \pm 0.004}$
yeast	0.702 ± 0.003	0.696 ± 0.005	0.539 ± 0.003	$\mathbf{0.497 \pm 0.001}$

$M = 50\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.849 ± 0.000	0.824 ± 0.004	0.785 ± 0.001	$\mathbf{0.751 \pm 0.001}$
Corel5k	0.991 ± 0.000	0.986 ± 0.001	0.939 ± 0.000	$\mathbf{0.903 \pm 0.001}$
emotions	0.671 ± 0.007	0.679 ± 0.021	0.546 ± 0.005	$\mathbf{0.481 \pm 0.002}$
enron	0.930 ± 0.000	0.879 ± 0.005	$\mathbf{0.632 \pm 0.001}$	0.636 ± 0.001
mediamill	0.955 ± 0.000	0.900 ± 0.007	0.592 ± 0.000	$\mathbf{0.581 \pm 0.000}$
medical	0.966 ± 0.000	0.911 ± 0.005	0.312 ± 0.002	$\mathbf{0.299 \pm 0.002}$
nuswide	0.977 ± 0.000	0.957 ± 0.002	0.690 ± 0.000	$\mathbf{0.675 \pm 0.000}$
scene	0.771 ± 0.005	0.699 ± 0.018	0.827 ± 0.009	$\mathbf{0.623 \pm 0.004}$
yeast	0.666 ± 0.008	0.649 ± 0.013	0.520 ± 0.001	$\mathbf{0.492 \pm 0.001}$

Table A.13: CS-DPP vs. others, Normalized rank loss

$M = 10\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	0.497 ± 0.001	0.397 ± 0.006	0.399 ± 0.001	$\mathbf{0.144 \pm 0.002}$
Corel5k	0.470 ± 0.001	0.393 ± 0.013	0.470 ± 0.000	$\mathbf{0.237 \pm 0.001}$
emotions	0.508 ± 0.006	0.363 ± 0.020	0.223 ± 0.009	$\mathbf{0.133 \pm 0.023}$
enron	0.463 ± 0.004	0.320 ± 0.015	0.304 ± 0.001	$\mathbf{0.132 \pm 0.001}$
mediamill	0.454 ± 0.009	0.391 ± 0.020	0.307 ± 0.000	$\mathbf{0.072 \pm 0.002}$
medical	0.397 ± 0.004	0.301 ± 0.019	0.311 ± 0.005	$\mathbf{0.096 \pm 0.003}$
nuswide	0.600 ± 0.003	0.532 ± 0.013	0.580 ± 0.000	$\mathbf{0.358 \pm 0.001}$
scene	0.491 ± 0.003	$\mathbf{0.295 \pm 0.029}$	0.500 ± 0.000	$\mathbf{0.231 \pm 0.016}$
yeast	0.490 ± 0.002	0.376 ± 0.018	0.340 ± 0.003	$\mathbf{0.003 \pm 0.000}$

$M = 25\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	$\mathbf{0.025 \pm 0.002}$	0.393 ± 0.006	0.397 ± 0.001	0.150 ± 0.002
Corel5k	$\mathbf{0.003 \pm 0.001}$	0.424 ± 0.008	0.467 ± 0.000	0.220 ± 0.001
emotions	0.174 ± 0.023	0.319 ± 0.023	0.235 ± 0.009	$\mathbf{0.046 \pm 0.015}$
enron	$\mathbf{0.106 \pm 0.008}$	0.329 ± 0.020	0.282 ± 0.001	0.135 ± 0.001
mediamill	0.092 ± 0.007	0.371 ± 0.023	0.306 ± 0.000	$\mathbf{0.068 \pm 0.000}$
medical	0.090 ± 0.005	0.261 ± 0.023	0.184 ± 0.003	$\mathbf{0.036 \pm 0.002}$
nuswide	$\mathbf{0.319 \pm 0.010}$	0.593 ± 0.014	0.580 ± 0.000	$\mathbf{0.329 \pm 0.001}$
scene	$\mathbf{0.155 \pm 0.017}$	0.304 ± 0.028	0.470 ± 0.002	$\mathbf{0.142 \pm 0.011}$
yeast	0.205 ± 0.018	0.356 ± 0.021	0.301 ± 0.002	$\mathbf{0.016 \pm 0.001}$

$M = 50\%$ of K

Dataset	O-CS	O-RAND	DPP	CS-DPP
CAL500	$\mathbf{0.014 \pm 0.001}$	0.389 ± 0.006	0.396 ± 0.001	0.151 ± 0.001
Corel5k	$\mathbf{0.004 \pm 0.001}$	0.447 ± 0.009	0.467 ± 0.000	0.213 ± 0.000
emotions	0.225 ± 0.014	0.360 ± 0.015	0.277 ± 0.004	$\mathbf{0.033 \pm 0.003}$
enron	$\mathbf{0.138 \pm 0.009}$	0.314 ± 0.009	0.274 ± 0.001	$\mathbf{0.129 \pm 0.001}$
mediamill	$\mathbf{0.060 \pm 0.003}$	0.331 ± 0.015	0.306 ± 0.000	$\mathbf{0.059 \pm 0.000}$
medical	0.111 ± 0.010	0.256 ± 0.019	0.136 ± 0.001	$\mathbf{0.019 \pm 0.001}$
nuswide	0.321 ± 0.008	0.556 ± 0.011	0.580 ± 0.000	$\mathbf{0.304 \pm 0.000}$
scene	0.191 ± 0.021	0.299 ± 0.020	0.423 ± 0.003	$\mathbf{0.104 \pm 0.009}$
yeast	0.238 ± 0.012	0.343 ± 0.012	0.291 ± 0.001	$\mathbf{0.026 \pm 0.001}$

doi:10.6342/NTU201701149

Bibliography

[1] R. Arora, A. Cotter, and N. Srebro. Stochastic optimization of PCA with capped MSG. In *NIPS*, pages 1815–1823, 2013.

[2] K. Balasubramanian and G. Lebanon. The landmark selection method for multiple output prediction. In *ICML*, 2012.

[3] P. Bartlett. Online convex optimization: ridge regression, adaptivity, 2008.

[4] J. P. Bello, E. Chew, and D. Turnbull. Multilabel classification of music into emotions. In *ICMIR*, pages 325–330, 2008.

[5] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, pages 730–738, 2015.

[6] W. Bi and J. T. Kwok. Efficient multi-label classification with many labels. In *ICML*, pages 405–413, 2013.

[7] Y. Chen and H. Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1538–1546, 2012.

[8] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. NUS-WIDE: a real-world web image database from national university of singapore. In *CIVR*, 2009.

[9] K. Crammer, O. Dekel, J. Keshet, S. S.-S., and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[10] K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

doi:10.6342/NTU201701149

[11] K. Dembczynski, W. Waegeman, W. Cheng, and E. Hüllermeier. An exact algorithm for F-measure maximization. In *NIPS*, pages 1404–1412, 2011.

[12] A. Elisseeff and J. Weston. A kernel method for multilabelled classification. In *NIPS*, 2001.

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

[14] A. Kapoor, R. Viswanathan, and P. Jain. Multilabel classification using bayesian compressed sensing. In *NIPS*, pages 2654–2662, 2012.

[15] C. Li and H. Lin. Condensed filter tree for cost-sensitive multi-label classification. In *ICML*, pages 423–431, 2014.

[16] C. Li, H. Lin, and C. Lu. Rivalry of two families of algorithms for memory-restricted streaming pca. In *AISTATS*, 2016.

[17] Z. Lin, G. Ding, M. Hu, and J. Wang. Multi-label classification via feature-aware implicit label space encoding. In *ICML*, pages 325–333, 2014.

[18] H. Lo, J. Wang, H. Wang, and S. Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Trans. Multimedia*, 13(3):518–529, 2011.

[19] J. Nie, W. Kotlowski, and M. K. Warmuth. Online PCA with optimal regrets. *Journal of Machine Learning Research*, 17:194–200, 2016.

[20] A. P. P. Osojnik and D. S. Multi-label classification via multi-target regression on data streams. *Machine Learning*, 2017.

[21] J. Read, A. Bifet, G. Holmes, and B. Pfahringer. Streaming multi-label classification. In *Proceedings of the Workshop on Applications of Pattern Analysis (WAPA)*, pages 19–25, 2011.

[22] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

doi:10.6342/NTU201701149

[23] L. Sun, S. Ji, and J. Ye. Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *IEEE TPAMI*, 33(1):194–200, 2011.

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

[25] G. Tsoumakas, I. Katakis, and I. P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 667–685. 2010.

[26] G. Tsoumakas and I. P. Vlahavas. Random k -labelsets: An ensemble method for multilabel classification. In *ECML*, pages 406–417, 2007.

[27] J. Wang, P. Zhao, and S. C. H. Hoi. Cost-sensitive online classification. *IEEE Trans. Knowl. Data Eng.*, 26(10):2425–2438, 2014.

[28] M. K. Warmuth and D. Kuzmin. Randomized online pca algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.

[29] Y. Wu and H. Lin. Progressive k-labelsets for cost-sensitive multi-label classification. *Machine Learning*, 2016. Accepted for Special Issue of ACML 2016.

[30] E. S. Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. P. Vlahavas. Dealing with concept drift and class imbalance in multi-label stream classification. In *IJCAI*, pages 1583–1588, 2011.

[31] H. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, pages 593–601, 2014.

[32] X. Zhang, T. Graepel, and R. Herbrich. Bayesian online learning for multi-label and multi-variate performance measures. In *AISTATS*, 2010.

doi:10.6342/NTU201701149