Noname manuscript No.
(will be inserted by the editor)

Dynamic Principal Projection for Cost-Sensitive Online Multi-Label Classification

Hong-Min Chu · Kuan-Hao Huang · Hsuan-Tien Lin

Received: date / Accepted: date

Abstract We study multi-label classification (MLC) with three important real-world issues: online updating, label space dimension 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 an online LSDR framework derived from a leading LSDR algorithm. In particular, CS-DPP is equipped with an efficient online dimension reducer motivated by matrix stochastic gradient, 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. 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.

Keywords Multi-label classification · Cost-sensitive · Label space dimension reduction

1 Introduction

The multi-label classification (MLC) problem allows each instance to be associated with a set of labels and reflects the nature of a wide spectrum of real-world applications [8,4,12]. Traditional MLC algorithms mainly tackle the batch MLC problem, where the input data are presented in a batch [24,28]. Nevertheless, in many MLC applications such as e-mail categorization [22], multi-label examples arrive as a stream. Online analysis is therefore required as batch MLC algorithms may not meet the needs to make a prediction and update the predictor on the fly. The needs 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, 13,17,26,14,25,33,6,2,5]. By viewing the label set of each instance as a high-dimensional

Hong-Min Chu, Kuan-Hao Huang, Hsuan-Tien Lin
CSIE Department, National Taiwan University, Taiwan
E-mail: {r04922031, r03922062, htlin}@csie.ntu.edu.tw

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) [26] 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 algorithms to exploit the key joint information between labels to be more robust to noise and be more effective on learning [26]. Nevertheless, to the best of our knowledge, all the LSDR algorithms mentioned above are designed only for the batch MLC problem.

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 [24] and makes Bayes-optimal predictions with respect to the given cost; CFT decomposes the cost into instance weights when training the classifiers in CC. Both algorithms, again, targets 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.

In Section 4, we further generalize DPP to cost-sensitive DPP (CS-DPP) to fully match the needs of CSOMLC with a theoretically-backed 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 to consider LSDR, basis drifting and cost-sensitivity. The results show that CS-DPP significantly outperforms other OMLC competitors across different CSOMLC problems, which validates the robustness and effectiveness of CS-DPP, as concluded in Section 6.

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], which is equivalent to another scheme, $\mathbf{y}[k] \in \{1, 0\}$, used in other MLC works [15, 24].

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 the 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 [28] 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 during evaluation. There are other criteria that jointly evaluate all labels, such as the F1 loss [28]

$$c_{\text{F}}(\mathbf{y}, \hat{\mathbf{y}}) = 1 - 2\frac{\sum_{k=1}^{K}[\![\mathbf{y}[k]=+1 \text{ and } \hat{\mathbf{y}}[k]=+1]\!]}{\sum_{k=1}^{K}([\![\mathbf{y}[k]=+1]\!] + [\![\hat{\mathbf{y}}[k]=+1]\!])}.$$

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 $c(\mathbf{y}, \hat{\mathbf{y}})$.We naturally assume that $c(\cdot,\cdot)$ satisfies $c(\mathbf{y}, \mathbf{y}) = 0$ and $\max_{\hat{\mathbf{y}}} c(\mathbf{y}, \hat{\mathbf{y}}) \leq 1$. 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 input cost function c such that the cumulative cost $\sum_{t=1}^{T} c(\mathbf{y}_t, \hat{\mathbf{y}}_t)$ with respect to the input c, where $\hat{\mathbf{y}}_t = g_t(\mathbf{x}_t)$, can be minimized.

Note that the cost function within the CSOMLC setting above corresponds to the *example-based* evaluation criteria for MLC, named because the prediction $\hat{\mathbf{y}}_t$ of each example is evaluated against the ground truth $\mathbf{y}_t$ independently. More sophisticated evaluation criteria such as micro-based and macro-based criteria [27,20] can also be found in the literature. The following equations highlight the difference between example-F1 (what our CSOMLC setting can handle), micro-F1 and macro-F1 when calculated on T predictions

$$\text{example-F1 loss} = 1 - \frac{2}{T}\sum_{t=1}^{T}\frac{\sum_{k=1}^{K}[\![\mathbf{y}_t[k]=+1 \text{ and } \hat{\mathbf{y}}_t[k]=+1]\!]}{\sum_{k=1}^{K}([\![\mathbf{y}_t[k]=+1]\!] + [\![\hat{\mathbf{y}}_t[k]=+1]\!])} \quad ;$$

$$\text{micro-F1 loss} = 1 - \frac{2}{K}\sum_{k=1}^{K}\frac{\sum_{t=1}^{T}[\![\mathbf{y}_t[k]=+1 \text{ and } \hat{\mathbf{y}}_t[k]=+1]\!]}{\sum_{t=1}^{T}([\![\mathbf{y}_t[k]=+1]\!] + [\![\hat{\mathbf{y}}_t[k]=+1]\!])} \quad ;$$

$$\text{macro-F1 loss} = 1 - 2\frac{\sum_{t=1}^{T}\sum_{k=1}^{K}[\![\mathbf{y}_t[k]=+1 \text{ and } \hat{\mathbf{y}}_t[k]=+1]\!]}{\sum_{t=1}^{T}\sum_{k=1}^{K}([\![\mathbf{y}_t[k]=+1]\!] + [\![\hat{\mathbf{y}}_t[k]=+1]\!])} \quad .$$

In particular, the three criteria differ by the averaging process. Average example-F1 computes the geometric mean of precision and recall (F1) *per example* and then computes the arithmetic mean over all examples; micro-F1 computes the geometic mean of precision and recall *per label* and then computes the arithmetic mean over all labels; macro-F1 computes the geometric mean of precision and recall *over the set of all example-label predictions*. The more sophisticated ones are known to be more difficult to optimize. Thus, similar to many existing cost-sensitive MLC algorithms for the batch setting [15], we consider only example-based criteria in this work, and leave the investigation of achieving cost-sensitivity for micro- and macro-based criteria to the future.

Several OMLC algorithms have been studied in the literature, including online binary relevance [23], Bayesian OMLC framework [34], and the multi-window approach using k

nearest neighbors [32]. 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 studied in the literature. Cost-sensitive RAkEL [19] and progressive RAkEL [31] are two algorithms that generalize a famous batch MLC algorithm called RAkEL [29] 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 condensed filter tree (CFT) [15] are two other algorithms that generalizes another famous batch MLC algorithm called classifier chain (CC) [24] 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 cost function and the estimation. PCC in principal achieves cost-sensitivity for any cost function, but the prediction can be time-consuming unless an efficient Bayes inference rule is designed for the cost function (*e.g.* the F1 loss [11]). CFT embeds the cost information into CC by an $O(K^2)$-time step that re-weights the training 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. CC-family algorithms typically suffer from the problem of ordering the labels properly to achieve decent performance. Some works start solving the ordering problem for the original CC algorithm, such as the easy-to-hard paradigm [18], but whether those works can be well-coupled with CFT or PCC has yet to be studied.

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) [26] encodes the label vectors $\{\mathbf{y}_n\}_{n=1}^N$ to their top principal components for the batch MLC problem. Some other LSDR algorithms, including conditional principal label space transformation (CPLST) [7], feature-aware implicit label space encoding (FaIE) [17], canonical-correlation-analysis method [25], and low-rank empirical risk minimization for multi-label learning [33], jointly take the feature and the label vectors into account during encoding [7,17,25,33] to further improve the performance.

The physical intuition behind LSDR algorithms is to capture the key joint information between labels before learning. By encoding to a more concise code space, LSDR algorithms enjoy the advantage of learning the predictor more effectively to improve the MLC performance. Moreover, compared with non-LSDR algorithms like RAkEL and CFT, LSDR algorithms are generally more efficient, which in turn makes them favorable candidates to be extended to online learning.

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 generalize it to the CSOMLC setting in Section 4.

Table 1: Summary of common notations

notation	meaning
d	number of features
K	number of labels
M	dimension of the code space
$\mathbf{x} \in \mathbb{R}^d$	feature vector
$\mathbf{y} \in \{+1, -1\}^K$	ground truth label vector
$\hat{\mathbf{y}} \in \{+1, -1\}^K$	predicted label vector
$\mathbf{c}(\mathbf{y}, \hat{\mathbf{y}})$	cost for predicting $\mathbf{y}$ as $\hat{\mathbf{y}}$
$\mathbf{z} \in \mathbb{R}^M$	code vector
$\mathbf{P} \in \mathbb{R}^{M \times K}$	encoding matrix from the label space to the code space
$\mathbf{W} \in \mathbb{R}^{d \times M}$	linear predictor matrix from the input space to the code space
$\mathbf{U} \in \mathbb{R}^{K \times K}$	(roughly) rank-M matrix within matrix stochastic gradient (MSG)
$(\mathbf{Q} \in \mathbb{R}^{(M+1) \times K}, \sigma \in \mathbb{R}^{M+1})$	decomposition of $\mathbf{U}$ such that $\mathbf{U} = \mathbf{Q}\mathrm{diag}(\sigma)\mathbf{Q}^\top$
$\Gamma \in [0, 1]^{M+1}$	discrete probability distribution for sampling the rows of $\mathbf{Q}$ to get $\mathbf{P}$
$\delta^{(k)} \in \mathbb{R}$	weight of the k-th label for representing the cost in CS-DPP
$\mathbf{C} \in \mathbb{R}^{K \times K}$	a diagonal matrix that stores $\{\sqrt{\delta^{(k)}}\}_{k=1}^K$ in CS-DPP

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 top principal components, and the rich literature of online PCA algorithms [1,21,16]. We shall first introduce the detail of PLST. Then, we discuss the potential difficulties along with our solutions to advance PLST to our proposed DPP. To facilitate reading, the common notations that will be used for the coming sections are summarized in Table 1.

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, as a batch LSDR algorithm, encodes each $\mathbf{y}_n \in \{+1, -1\}^K$ into a code vector $\mathbf{z}_n = \mathbf{P}^*(\mathbf{y}_n - \mathbf{o})$, where $\mathbf{o}$ is a fixed reference point for shifting $\mathbf{y}_n$, and $\mathbf{P}^*$ contains the top M eigenvectors of $\sum_{n=1}^N (\mathbf{y}_n - \mathbf{o})(\mathbf{y}_n - \mathbf{o})^\top$. While PLST works with any fixed $\mathbf{o}$, it is worth noting that when $\mathbf{o}$ is taken as $\frac{1}{N}\sum_{n=1}^N \mathbf{y}_n$, the code vector $\mathbf{z}_n$ contains the top M principal components of $\mathbf{y}_n$. 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

$$\hat{\mathbf{y}} = \mathrm{round}\left((\mathbf{P}^*)^\top \mathbf{r}(\mathbf{x}) + \mathbf{o}\right) \tag{1}$$

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

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:

[1] The naming of the $\mathrm{round}(\cdot)$ operator follows directly from the original paper of PLST [26], which represents $\mathbf{y} \in \{0, 1\}^K$ instead of $\{-1, +1\}^K$. Our use of sign is thus equivalent to the rounding steps used in the original PLST.

Theorem 1 *[26] When making a prediction* $\hat{\mathbf{y}}$ *from* $\mathbf{x}$ *by* $\hat{\mathbf{y}} = \operatorname{round}\left(\mathbf{P}^\top \mathbf{r}(\mathbf{x}) + \mathbf{o}\right)$ *with any left orthogonal matrix* $\mathbf{P}$*, the Hamming loss*

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

where $\mathbf{z} \equiv \mathbf{P}\mathbf{y}'$ *and* $\mathbf{y}' \equiv \mathbf{y} - \mathbf{o}$ *with respect to any fixed reference point* $\mathbf{o}$*.*

Theorem 1 bounds the Hamming loss by the prediction and reconstruction errors. Based on the results of singular value decomposition, $\mathbf{P}^*$ in PLST is the optimal solution for minimizing the total reconstruction error of the observed label vectors with respect to any fixed $\mathbf{o}$, and the particular reference point $\frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n$ minimizes the reconstruction error over all possible $\mathbf{o}$. Then, by minimizing the prediction error with regressor $\mathbf{r}$, PLST is able to minimize the Hamming loss approximately.

3.2 General Online LSDR Framework for DPP

The upper bound in Theorem 1 works for any regressor $\mathbf{r}$ and any left orthogonal encoding matrix $\mathbf{P}$. Based on the bound, we propose an online LSDR framework that approximately minimizes the Hamming loss with an online regressor $\mathbf{r}_t$ and an online encoding matrix $\mathbf{P}_t$ in each iteration t. Similar to PLST, the proposed framework works with any fixed referenced point $\mathbf{o}$. But for simplicity of illustration, we assume that $\mathbf{o} = \mathbf{0}$ to remove $\mathbf{o}$ from the derivations below. The steps of the framework are:

For $t = 1, \ldots, T$
  Receive $\mathbf{x}_t$ and predict $\hat{\mathbf{y}}_t = \operatorname{round}(\mathbf{P}_t^\top \mathbf{r}_t(\mathbf{x}_t))$
  Receive $\mathbf{y}_t$ and incur error $\ell^{(t)}(\mathbf{r}_t, \mathbf{P}_t)$
  Update $\mathbf{P}_t$ and $\mathbf{r}_t$

In each iteration t of the framework, an online prediction $\hat{\mathbf{y}}_t$ is made with the updated $\mathbf{r}_t$ and $\mathbf{P}_t$. We take the online error function $\ell^{(t)}(\mathbf{r}, \mathbf{P})$ to be $\|\mathbf{r}(\mathbf{x}_t) - \mathbf{P}\mathbf{y}_t\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})\mathbf{y}_t\|_2^2$, which upper bounds the Hamming loss $c_{\text{HAM}}(\mathbf{y}_t, \hat{\mathbf{y}}_t)$ of the online prediction. Then, by updating $\mathbf{r}_t$ and $\mathbf{P}_t$ with online learning algorithms that minimize the cumulative online error $\sum_{t=1}^{T} \ell^{(t)}(\mathbf{r}_t, \mathbf{P}_t)$, we can approximately minimize the cumulative Hamming loss.

The simple framework above transforms the OMLC problem to an online learning problem with an error function composed of two terms. Ideally, the online learning algorithm should update $\mathbf{P}_t$ and $\mathbf{r}_t$ to jointly minimize the total error from both terms. Optimizing the two terms jointly has been studied in batch LSDR algorithms like CPLST [7], which is a successor of PLST [26] that also operates with the upper bound in Theorem 1. Nevertheless, it is very challenging to extend CPLST to the online setting efficiently. In particular, a naıve online extension would require computing the hat matrix of the ridge regression part (from $\mathbf{x}$ to $\mathbf{z}$) within CPLST in order to obtain $\mathbf{P}_t$, and the hat matrix grows quadratically with the number of examples. That is, in an online setting, computing and storing the hat matrix needs at least $\Omega(T^2)$ complexity up to iteration T, which is practically infeasible.

Thus, we resort to PLST [26], the predecessor of CPLST, to make an initial attempt towards tackling OMLC problems. PLST minimizes the two terms separately in the batch setting, and our proposed extension of PLST similarly contains two online learning algorithms, one for minimizing each term. That is, we further decompose the online learning problem to two sub-problems, one for minimizing the cumulative reconstruction error (by updating $\mathbf{P}_t$),

and one for minimizing the cumulative prediction error (by updating $\mathbf{r}_t$). Designing efficient and effective algorithms for the two sub-problems turns out to be non-trivial, and will be discussed in Sections 3.3 and 3.4.

3.3 Online Minimization of Reconstruction Error

Next, we discuss the design of our first online learning algorithm to tackle the sub-problem of minimizing the cumulative reconstruction error $\sum_{t=1}^{T} \|(\mathbf{I}-\mathbf{P}_t^\top \mathbf{P}_t)\mathbf{y}_t\|_2^2$, which corresponds to the second term in (2). The goal is to generate a left-orthogonal matrix $\mathbf{P}_t \in \mathbb{R}^{M\times K}$ in each iteration that guarantees to minimize the cumulative reconstruction error theoretically.

Our design is motivated by a simple but promising online PCA algorithm, matrix stochastic gradient (MSG) [1]. MSG does not directly solve the sub-problem of our interest because the problem is non-convex over $\mathbf{P}_t$. Instead, MSG substitutes $\mathbf{P}_t^\top \mathbf{P}_t$ with a rank-M matrix $\mathbf{U}_t \in \mathbb{R}^{K\times K}$ and rewrites the cumulative reconstruction error as $\sum_{t=1}^{T} \mathbf{y}_t^\top(\mathbf{I}-\mathbf{U}_t)\mathbf{y}_t$. By further assuming that $\|\mathbf{y}_t\|_2 \leq 1$, MSG loosens the constraint of $rank(\mathbf{U}_t) = M$ to $tr(\mathbf{U}_t) = M$, and updates $\mathbf{U}_t$ with online projected gradient descent upon receiving a new $\mathbf{y}_t$ as

$$\mathbf{U}_{t+1} = \mathcal{P}_{tr}(\mathbf{U}_t + \eta \mathbf{y}_t \mathbf{y}_t^\top) \tag{3}$$

where η is the learning rate and $\mathcal{P}_{tr}(\cdot)$ is the projecting operator to a feasible $\mathbf{U}$. The less-constrained $\mathbf{U}_t$ in MSG carries the theoretical guarantee of minimizing the cumulative reconstruction error (subject to $\mathbf{U}_t$), but decomposing $\mathbf{U}_t$ to a left-orthogonal $\mathbf{P}_t \in \mathbb{R}^{M\times K}$ with theoretical guarantee on $\mathbf{P}_t$ is not only non-trivial but also time-consuming.

Capped MSG [1] is an extension of MSG with the hope of lightening the computational burden of decomposing $\mathbf{U}_t$. In particular, Capped MSG introduces an additional (non-convex) constraint of $rank(\mathbf{U}_t) \leq M+1$, and indirectly maintains the decomposition of $\mathbf{U}_t$ as $(\mathbf{Q}_t, \sigma_t)$, where the left-orthogonal matrix $\mathbf{Q}_t \in \mathbb{R}^{(M+1)\times K}$ and the vector of singular values $\sigma_t \in \mathbb{R}^{M+1}$ such that $\mathbf{U}_t = \mathbf{Q}_t \text{diag}(\sigma_t)\mathbf{Q}_t^\top$. The decomposed $(\mathbf{Q}_t, \sigma_t)$ in Capped MSG enjoys the same theoretical guarantee of minimizing the reconstruction error as the $\mathbf{U}_t$ in MSG, while the maintenance step of Capped MSG is more efficient than MSG. Nevertheless, because we want $\mathbf{P}_t$ to be M by K while $\mathbf{Q}_t$ is $(M+1)$ by K, the generated $\mathbf{Q}_t$ in Capped MSG cannot be directly used to solve our sub-problem. A naïve idea is to generate $\mathbf{P}_t$ by truncating the least important row of $\mathbf{Q}_t$, but the naïve idea is no longer backed by the theoretical guarantee of Capped MSG.

Aiming to address the above difficulties, we propose an efficient and effective algorithm to stochastically generate $\mathbf{P}_t$ from $(\mathbf{Q}_t, \sigma_t)$ maintained by Capped MSG in each iteration. To elaborate, let $\mathbf{Q}_t^{-i}$ be $\mathbf{Q}_t$ with its i-th row removed and $\sigma_t[i]$ be the eigenvalue corresponding to i-th row of $\mathbf{Q}_t$. We generate $\mathbf{P}_t$ by sampling from a discrete probability distribution Γ_t, which consists of $M+1$ events $\{\mathbf{Q}_t^{-i}\}_{i=1}^{M+1}$ with probability of $\mathbf{Q}_t^{-i}$ being $1-\sigma_t[i]$. As the projecting operator $\mathcal{P}_{tr}(\cdot)$ ensures $0 \leq \sigma_t[i] \leq 1$ for each $\sigma_t[i]$, one can easily verify Γ_t to be a valid distribution with the additional fact that $\sum_i \sigma_t[i] = tr(\mathbf{U}_t) = M$. The following lemma shows that the online encoding matrix generated by our simple stochastic algorithm is truly effective, and the proof can be found in the supplementary materials.

Lemma 2 *Suppose $(\mathbf{Q}_t, \sigma_t)$ is obtained after an updated of Capped MSG such that $\mathbf{U}_t = \mathbf{Q}_t diag(\sigma_t)\mathbf{Q}_t^\top$. If Γ_t is a discrete probability distribution over events $\{\mathbf{Q}_t^{-i}\}_{i=1}^{M+1}$ with probability of $\mathbf{Q}_t^{-i}$ being $1-\sigma_t[i]$, we have for any $\mathbf{y}$*

$$\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\mathbf{y}^\top(\mathbf{I}-\mathbf{P}_t^\top\mathbf{P}_t)\mathbf{y}] = \mathbf{y}^\top(\mathbf{I}-\mathbf{U}_t)\mathbf{y} \tag{4}$$

The proof of the lemma can be found in Appendix A.1. Moreover, our sampling algorithm is highly efficient regarding its $\mathcal{O}(M)$ time complexity. Note that there is an earlier work that contains another algorithm of similar spirit [21]. Somehow the algorithm's time complexity is $\mathcal{O}(K^2)$, which is less efficient than ours.

To sum up, our online learning algorithm that minimizes the cumulative reconstruction error for DPP takes Capped MSG as its building block to maintain $\mathbf{U}_t$ by $\mathbf{Q}_t$ and σ_t, and then samples the online encoding matrix $\mathbf{P}_t$ from Γ_t derived by $\mathbf{Q}_t$ in each iteration by our proposed sampling algorithm. Note that to fulfill the assumption of $\|\mathbf{y}_t\|_2 \leq 1$ required by Capped MSG, we apply a simple trick to scale each $\mathbf{y}_t \in \{+1, -1\}^K$ with a factor of $\frac{1}{\sqrt{K}}$. The predictions given by our online LSDR framework remain unchanged after the constant scaling due to the use of $\mathrm{round}(\cdot)$ operator.

3.4 Online Minimization of Prediction Error

Next, we discuss another proposed online learning algorithm to solve the second sub-problem of minimizing the cumulative prediction error $\sum_{t=1}^{T} \|\mathbf{r}_t(\mathbf{x}_t) - \mathbf{P}_t\mathbf{y}_t\|^2$, which corresponds to the first term in (2). The proposed online learning algorithm is based on the well-known online ridge regression, and incorporates two different carefully designed techniques to remedy the negative effect caused by the variation of $\mathbf{P}_t$ in each iteration.

The naïve online ridge regression parameterizes $\mathbf{r}_t(\mathbf{x})$ to be an online linear regressor $\mathbf{W}_t^\top \mathbf{x}$ with $\mathbf{W}_t \in \mathbb{R}^{d \times M}$, and update $\mathbf{W}_t$ by

$$\mathbf{W}_t = \arg\min_{\mathbf{W}} \quad \frac{\lambda}{2} tr(\mathbf{W}\mathbf{W}^\top) + \sum_{i=1}^{t-1} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{z}_i\|_2^2 \tag{5}$$

where $\mathbf{z}_i = \mathbf{P}_i\mathbf{y}_i$ is the code vector of $\mathbf{y}_i$ regarding $\mathbf{P}_i$, and λ is the regularization parameter. However, the naïve online ridge regression suffers from the drifting of projection basis caused by varying the online encoding matrix $\mathbf{P}_t$ as t advances. To elaborate, recall that the online regressor $\mathbf{W}_t$ aims to predict $\mathbf{z}_t = \mathbf{P}_t\mathbf{y}_t$ from $\mathbf{x}_t$, where the code vector $\mathbf{z}_t$ can essentially be viewed as the set of combination coefficients with reference projection basis formed by $\mathbf{P}_t$. However, $\mathbf{W}_t$ is learned from $\{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^{t-1}$, where the learning target $\{\mathbf{z}_i\}_{i=1}^{t-1}$ is mixed up with coefficients $\mathbf{z}_i$ induced from different projection basis $\mathbf{P}_i$. As a consequence, expecting $\mathbf{W}_t^\top \mathbf{x}_t$ to give accurate prediction of $\mathbf{z}_t$ for any specific $\mathbf{P}_t$ is unrealistic. For a very extreme case, if $\mathbf{P}_1 = \mathbf{P}_3 = \ldots = \mathbf{P}_{2\tau-1} = \mathbf{P}$ and $\mathbf{P}_2 = \mathbf{P}_4 = \ldots = \mathbf{P}_{2\tau} = -\mathbf{P}$, the $\mathbf{z}_i$'s in the odd and even iterations are of totally opposite meanings although the projection matrices $\mathbf{P}$ and $-\mathbf{P}$ are mathematically equivalent in quality. The totally opposite meanings make it impossible for $\mathbf{W}_t$ to predict $\mathbf{z}_t$ accurately.

To remedy the problem of basis drifting, we propose two different techniques, principal basis correction (PBC) and principal basis transform (PBT), to improve online regressor $\mathbf{W}_t$. Each of them enjoys different advantages.

3.4.1 Principal Basis Correction

The ideal solution to handle basis drifting is to "correct" the reference basis of each $\mathbf{z}_i$ to be the latest $\mathbf{P}_t$ used for prediction. More specifically, we want $\mathbf{W}_t$ to be the ridge regression solution obtained from $\{(\mathbf{x}_i, \mathbf{P}_t\mathbf{y}_i)\}_{i=1}^{t-1}$ instead of $\{(\mathbf{x}_i, \mathbf{P}_i\mathbf{y}_i)\}_{i=1}^{t-1}$. Such a correction step ensures that the reference basis for generating the previous $\mathbf{z}_i$'s is the same as the basis

that will be used for the predicting $\mathbf{z}_t$ and decoding $\hat{\mathbf{y}}_t$ from $\mathbf{z}_t$. Denote $\mathbf{W}_t^{\text{PBC}}$ as the ridge regression solution of $\{(\mathbf{x}_i, \mathbf{P}_t\mathbf{y}_i)\}_{i=1}^{t-1}$. The closed-form solution of $\mathbf{W}_t^{\text{PBC}}$ is

$$\mathbf{W}_t^{\text{PBC}} = \underbrace{(\lambda\mathbf{I} + \sum_{i=1}^{t-1}\mathbf{x}_i\mathbf{x}_i^\top)^{-1}}_{\mathbf{A}_t^{-1}}\underbrace{(\sum_{i=1}^{t-1}\mathbf{x}_i\mathbf{y}_i^\top)}_{\mathbf{B}_t}\mathbf{P}_t^\top \ . \tag{6}$$

The part $\mathbf{A}_t^{-1}\mathbf{B}_t$ is independent of the projection matrix $\mathbf{P}_t$. Thus, by maintaining another d by K matrix

$$\mathbf{H}_t = \mathbf{A}_t^{-1}\mathbf{B}_t$$

throughout the iterations, $\mathbf{W}_t^{\text{PBC}}$ can be easily obtained by $\mathbf{H}_t\mathbf{P}_t^\top$ for any $\mathbf{P}_t$. The update of $\mathbf{H}_t$ to $\mathbf{H}_{t+1}$, on the other hand, requires the calculation of $\mathbf{H}_{t+1} = (\mathbf{A}_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-1}(\mathbf{B}_t + \mathbf{x}_t\mathbf{y}_t^\top)$, which at a first glance has a time complexity of $\mathcal{O}(d^3 + Kd^2)$. Fortunately, we can speed up the calculation by applying the Sherman-Morrison formula, which states that

$$(\mathbf{A}_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-1} = \left(\mathbf{A}_t^{-1} - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{x}_t^\top\mathbf{A}_t^{-1}}{1+\gamma}\right)$$

with $\gamma = \mathbf{x}_t^\top\mathbf{A}_t^{-1}\mathbf{x}_t$. Then, the calculation can be rewritten as

$$\begin{aligned}
\mathbf{H}_{t+1} &= \left(\mathbf{A}_t^{-1} - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{x}_t^\top\mathbf{A}_t^{-1}}{1+\gamma}\right)\left(\mathbf{B}_t + \mathbf{x}_t\mathbf{y}_t^\top\right) \\
&= \mathbf{A}_t^{-1}\mathbf{B}_t - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{x}_t^\top\mathbf{A}_t^{-1}\mathbf{B}_t}{1+\gamma} + \mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{y}_t^\top - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{x}_t^\top\mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{y}_t^\top}{1+\gamma} \\
&= \mathbf{H}_t - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t\tilde{\mathbf{y}}_t^\top}{1+\gamma} + \mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{y}_t^\top - \frac{\gamma\mathbf{A}_t^{-1}\mathbf{x}_t\mathbf{y}_t^\top}{1+\gamma} \\
&= \mathbf{H}_t - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t(\tilde{\mathbf{y}}_t - \mathbf{y}_t)^\top}{1+\gamma},
\end{aligned}$$

where $\tilde{\mathbf{y}}_t = \mathbf{H}_t^\top\mathbf{x}_t$. The third line follows from the fact that $\mathbf{H}_t = \mathbf{A}_t^{-1}\mathbf{B}_t$. Thus, the d by K matrix $\mathbf{H}_t$ can be efficiently updated online by

$$\mathbf{H}_{t+1} = \mathbf{H}_t - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t(\tilde{\mathbf{y}}_t - \mathbf{y}_t)^\top}{1 + \mathbf{x}_t^\top\mathbf{A}_t^{-1}\mathbf{x}_t} \tag{7}$$

which requires only a time complexity of $\mathcal{O}(d^2 + Kd)$.

It is worth noting that $\mathbf{H}_t$ actually stores the online ridge regression solution from $\mathbf{x}$ to $\mathbf{y}$. Based on the definition of $\mathbf{H}_t$, we can then theoretically analyze the performance of our online ridge regression solution $\mathbf{W}_t^{\text{PBC}}$ from $\mathbf{x}$ to $\mathbf{z}$ with respect to the error $\ell^{(t)}(\cdot,\cdot)$ in our proposed online LSDR framework. Following the convention of online learning, we analyze the expected average regret $\frac{\mathcal{R}}{T}$, defined as

$$\frac{\mathcal{R}}{T} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\ell^{(t)}(\mathbf{W}_t^{\text{PBC}}, \mathbf{P}_t) - \ell^{(t)}(\mathbf{W}_\#, \mathbf{P}^*)], \tag{8}$$

for any given sequence of $\{(\mathbf{P}_t, \Gamma_t)\}_{t=1}^T$, where each $\mathbf{P}_t$ is sampled from the distribution Γ_t. $(\mathbf{W}_\#, \mathbf{P}^*)$ here denotes the offline reference algorithm that is allowed to peek the whole data stream $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$. As our algorithm aims to minimize the online error function by a similar decomposition of sub-problems as PLST , we particularly consider $(\mathbf{W}_\#, \mathbf{P}^*)$ to be the solution of PLST when treating $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$ as the input batch data. That is, $\mathbf{P}^*$ is the minimizer of $\sum_{t=1}^T \mathbf{y}_t^\top (\mathbf{I} - \mathbf{P}^\top \mathbf{P}) \mathbf{y}_t$, which corresponds to the second term of $\ell^{(t)}(\cdot, \cdot)$, and $\mathbf{W}_\#$ is the minimizer of $\sum_{t=1}^T \|\mathbf{W}^\top \mathbf{x}_t - \mathbf{P}^* \mathbf{y}_t\|_2^2$, which corresponds to the first term of $\ell^{(t)}(\cdot, \cdot)$ given $\mathbf{P}^*$. It can be easily proved that $\mathbf{W}_\# = \mathbf{H}^* (\mathbf{P}^*)^\top$ where $\mathbf{H}^*$ is the optimal linear regression solution of $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$. That is,

$$\mathbf{H}^* = \arg\min_{\mathbf{H}} \quad \sum_{t=1}^T \|\mathbf{H}^\top \mathbf{x}_t - \mathbf{y}_t\|_2^2 \,. \tag{9}$$

With the expected average regret defined, we can prove its convergence by assuming the convergence of the subspace spanned by $\mathbf{P}_t$ to the subspace spanned by $\mathbf{P}^*$. The assumption generally holds when the M-th and $(M+1)$-th eigenvalues of $\sum_{t=1}^T (\mathbf{y}_t - \mathbf{o})(\mathbf{y}_t - \mathbf{o})^\top$ are different, as the subspace spanned by $\mathbf{P}^*$ to reach the minimum reconstruction error is consequently unique. In particular, define the expected subspace difference

$$\Delta_t = \|\mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\mathbf{P}_t^\top \mathbf{P}_t] - \left(\mathbf{P}^*\right)^\top \mathbf{P}^*\|_2 \,. \tag{10}$$

Theorem 3 *With the definitions of* $\mathbf{H}_t$ *in* (7)*,* $\mathbf{H}^*$ *in* (9)*,* $\frac{\mathcal{R}}{T}$ *in* (8) *and* Δ_t *in* (10)*, assume that* $\|\mathbf{x}_t\| \leq 1$*,* $\|\mathbf{y}_t\| \leq 1$ *and* $\|\mathbf{H}_t \mathbf{x}_t - \mathbf{y}_t\|_2^2 \leq \epsilon$*.*

1. *For any given* T*, the expected cumulative regret* $\mathcal{R}$ *is upper-bounded by*

$$(1+\epsilon) \sum_{t=1}^T \Delta_t + \frac{M}{2} \|\mathbf{H}^*\|_F^2 + 2\epsilon M d \log\left(1 + \frac{T}{d}\right).$$

2. *If* $\lim_{T\to\infty} \Delta_T = 0$ *and* $\|\mathbf{H}^*\|_F \leq h^*$ *across all iterations,*[2] $\lim_{T\to\infty} \frac{\mathcal{R}}{T} = 0$*.*

The third assumption requires the residual errors of online ridge regression without projection to be bounded, which generally holds when there is some linear relationship between $\mathbf{x}_t$ and $\mathbf{y}_t$. The detailed of the proof of the theorem can be found in Appendix A.2. Theorem 3 guarantees the performance of PBC to be competitive with a reasonable offline baseline in the long run given the convergence of subspace spanned by $\mathbf{P}_t$. Such a guarantee makes online linear regressor with PBC a solid option for DPP to tackle the sub-problem of minimizing cumulative prediction error.

3.4.2 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 $\mathbf{P}_t$, the time and space complexity of PBC depends on $\Omega(Kd)$ at the cost of maintaining $\mathbf{H}_t \in \mathbb{R}^{d \times K}$. The $\Omega(Kd)$ dependency can make PBC computationally inefficient when both K and d are large.

[2] The technicality of requiring $\|\mathbf{H}^*\|_F$ to be bounded is because we defined regret (up to the T-th iteration) with respect to the optimal offline solution upon receiving T examples, and hence $\mathbf{H}^*$ depends on T. Standard regret proof in online learning alternatively defines regret with respect to any *fixed* $\mathbf{H}$. Our proof could also go through with the alternative definition, which changes $\|\mathbf{H}^*\|_F$ to a constant $\|\mathbf{H}\|_F$ (that is trivially bounded).

Table 2: Time and space complexity for two DPP variants

	time complexity	space complexity
DPP-PBC	$\mathcal{O}(d^2 + MK + Kd + M^2K)$	$\mathcal{O}(d^2 + MK + Kd)$
DPP-PBT	$\mathcal{O}(d^2 + M^2d + M^2K)$	$\mathcal{O}(d^2 + MK + Md)$

To address the issue, we propose another technique, principal basis transform (PBT). Different from PBC, when a new online encoding matrix $\mathbf{P}_{t+1}$ is presented, PBT aims at a direct basis transform of the online linear regressor from $\mathbf{P}_t$ to $\mathbf{P}_{t+1}$. To be more specific, PBT assumes the regressor $\mathbf{W}_t^{\text{PBT}}$ to be the low-rank coefficients matrix of some *unknown* $\mathbf{H}'_t \in \mathbb{R}^{d\times K}$ with reference projection basis formed by $\mathbf{P}_t$, which can equivalently be described as $\mathbf{W}_t^{\text{PBT}} = \mathbf{H}'_t\mathbf{P}_t^\top$. The goal of PBT is to update $\mathbf{W}_t^{\text{PBT}}$ to $\mathbf{W}_{t+1}^{\text{PBT}}$ with $(\mathbf{x}_t, \mathbf{y}_t)$ such that the reference projection basis of $\mathbf{W}_{t+1}^{\text{PBT}}$ is now induced from $\mathbf{P}_{t+1}$. PBT achieves the goal by a two-step procedure. The first step is to find the low-rank coefficients matrix $\mathbf{W}'_t$ of $\mathbf{H}'_t$ based on the new reference basis formed by $\mathbf{P}_{t+1}$. However, as only the low rank coefficients matrix $\mathbf{W}_t^{\text{PBT}}$ rather than $\mathbf{H}'_t$ itself is known, we approximate $\mathbf{W}'_t$ by

$$\mathbf{W}'_t = \operatorname*{arg\,min}_{\mathbf{W}} \quad \|\mathbf{W}\mathbf{P}_{t+1} - \mathbf{W}_t^{\text{PBT}}\mathbf{P}_t\|_F^2 \,. \tag{11}$$

Solving (11) analytically gives

$$\mathbf{W}'_t = \mathbf{W}_t^{\text{PBT}}\mathbf{P}_t\mathbf{P}_{t+1}^\top \,. \tag{12}$$

The second step is to update $\mathbf{W}'_t$ with $(\mathbf{x}_t, \mathbf{y}_t)$ to obtain $\mathbf{W}_{t+1}^{\text{PBT}}$ by

$$\mathbf{W}_{t+1}^{\text{PBT}} = \mathbf{W}'_t - \frac{\mathbf{A}_t^{-1}\mathbf{x}_t(\tilde{\mathbf{z}}'_t - \mathbf{P}_{t+1}\mathbf{y}_t)^\top}{1 + \mathbf{x}_t^\top\mathbf{A}_t^{-1}\mathbf{x}_t} \tag{13}$$

where $\tilde{\mathbf{z}}'_t = \left(\mathbf{W}'_t\right)^\top \mathbf{x}_t$. Equation (13) can be derived with a similar use of the Sherman-Morrison formula as that for (7) by replacing $(\tilde{\mathbf{y}}_t, \mathbf{y}_t)$ with $(\tilde{\mathbf{z}}'_t, \mathbf{P}_t\mathbf{y}_t)$ respectively. One can easily verify that $\mathbf{W}_{t+1}^{\text{PBT}}$ obtained by (13) still keeps its reference basis as $\mathbf{P}_{t+1}$.

Comparing to PBC, PBT only has $\Omega(M^2(K + d))$ dependency, which is particularly useful when $M^2 \ll min(K, d)$. The appealing time complexity makes PBT a highly practical option for DPP to minimize the cumulative prediction error with. The time and space complexity of the two variants of DPP are listed in Table 2.

4 Generalization to Cost-Sensitive Learning

In this section, we generalize DPP to cost-sensitive DPP (CS-DPP), which meets the requirement of CSOMLC. The key ingredient to the generalization is a carefully designed label-weighting scheme that transforms cost $c(\mathbf{y}, \hat{\mathbf{y}})$ into the corresponding weighted Hamming loss. With the help of the label weighting scheme, we subsequently derive the optimization objective similar to Theorem 1 for general cost functions, which allows us to derive CS-DPP by reusing the building blocks of DPP.

We start from the detail of our label-weighting scheme based on the label-wise decomposition of $c(\mathbf{y}, \hat{\mathbf{y}})$. To represent the cost with the label weights, we propose a label-weighting

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

Parameters: λ, η, M

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: $\quad\hat{\mathbf{y}}_t \leftarrow \mathrm{round}(\mathbf{P}_{t-1}^\top \mathbf{W}_{t-1}^\top \mathbf{x}_t)$
4: $\quad$Obtain $\mathbf{C}_t$ by (15)
5: $\quad$Update $\mathbf{U}_{t-1}$ to $\mathbf{U}_t$ by Capped MSG (with $\mathbf{C}_t\mathbf{y}_t$) and sample $\mathbf{P}_t$ from Γ_t as defined in Lemma 2
6: $\quad\mathbf{W}'_{t-1} \leftarrow \mathbf{W}_{t-1}\mathbf{P}_{t-1}\mathbf{P}_t^\top$ (PBT)
7: $\quad$Update $\mathbf{W}'_{t-1}$, $\mathbf{A}_{t-1}^{-1}$ to $\mathbf{W}_t$, $\mathbf{A}_t^{-1}$ by (13) (with $\mathbf{C}_t\mathbf{y}_t$)
8: **end while**

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 $\hat{\mathbf{y}}_{\text{real}}^{(k)}$ and $\hat{\mathbf{y}}_{\text{pred}}^{(k)}$ as

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

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

$$\delta^{(k)} = |c(\mathbf{y}, \hat{\mathbf{y}}_{\text{pred}}^{(k)}) - c(\mathbf{y}, \hat{\mathbf{y}}_{\text{real}}^{(k)})| \,. \tag{14}$$

Recall that $\mathbf{y}$ is the ground truth vector and $\hat{\mathbf{y}}$ is the prediction vector from the algorithm. The two newly constructed vectors, $\hat{\mathbf{y}}_{\text{real}}^{(k)}$ and $\hat{\mathbf{y}}_{\text{pred}}^{(k)}$, can both be viewed as pseudo prediction vectors that are "better" than $\hat{\mathbf{y}}$, as they are both perfectly correct up to the $(k-1)$-th label. The two vectors only differ on the k-th prediction, which is correct for $\hat{\mathbf{y}}_{\text{real}}^{(k)}$ and incorrect for $\hat{\mathbf{y}}_{\text{pred}}^{(k)}$. The difference allows the term $\delta^{(k)}$ in (14) to quantify the price that the algorithm needs to pay if the k-th prediction is wrong. Then, the price $\delta^{(k)}$ can be viewed as an indicator of importance for predicting the k-th label correctly. Our label-weighting scheme follows such intuition by simply setting the weight of k-th label as $\delta^{(k)}$. The label-weighting scheme with (14) is not only intuitive, but also enjoys nice theoretical guarantee under 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 condition of the lemma, which generally holds for reasonable cost functions, simply says that for any label, a correct prediction should enjoy a lower cost than an incorrect prediction. The proof of the lemma can be found in Appendix A.3. Lemma 4 transforms $c(\mathbf{y}, \hat{\mathbf{y}})$ into the corresponding weighted Hamming loss, and thus enables the optimization over general cost functions. Note that condition implies that correcting a wrongly-predicted label leads to no higher cost, and is considered mild as general cost functions for MLC satisfy the condition.

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

$$\mathbf{C} = \mathrm{diag}(\sqrt{\delta^{(1)}}, ..., \sqrt{\delta^{(K)}}) \tag{15}$$

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

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 left orthogonal matrix* $\mathbf{P}$*, if* $c(\cdot,\cdot)$ *satisfies the condition of Lemma 4, the prediction cost*

$$c(\mathbf{y}, \hat{\mathbf{y}}) \leq \|\mathbf{r}(\mathbf{x}) - \mathbf{z_C}\|_2^2 + \|(\mathbf{I} - \mathbf{P}^\top \mathbf{P})(\mathbf{y'_C})\|_2^2$$

where $\mathbf{z_C} = \mathbf{P}(\mathbf{y'_C})$ *and* $\mathbf{y'_C} = \mathbf{C}\mathbf{y} - \mathbf{o}$ *with respect to any fixed reference point* $\mathbf{o}$*.*

Theorem 5 generalizes Theorem 1 to upper-bound the general cost $c(\mathbf{y}, \hat{\mathbf{y}})$ instead of the original Hamming loss $c_{\text{HAM}}(\mathbf{y}, \hat{\mathbf{y}})$. With Theorem 5, extending DPP to CS-DPP is a straightforward task by reusing the online updating algorithms 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, and we can easily write down similar steps for CS-DPP using PBC. Note that we simplify $\mathbf{W}_t^{\text{PBT}}$ to $\mathbf{W}_t$ in Algorithm 1 to make a cleaner presentation.

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 online LSDR, (2) experiments on basis drifting, and (3) experiments on cost-sensitivity. Furthermore, recall that the label weighting scheme of CS-DPP depends on the label order. We therefore conduct an additional set of experiments to study how different label orders affect the performance of CS-DPP. To assist the readers in understanding the experiments, we list the full names and acronyms of the algorithms to be compared along with their key differences in Table 3. The details of the algorithms will be illustrated as needed.

5.1 Experiments Setup

We conduct our experiments on eleven real-world datasets[3] downloaded from Mulan [30]. Statistics of datasets can be found in Table 4. In particular, datasets *eurlex-eurovec* and *delicious* are used only in the experiment to justify the necessity of online LSDR, and only 7500 sub-sampled instances are used on these two datasets to reduce the computational burden of the competitors in the experiment. In addition, only 50000 sub-sampled instances are used for *nuswide* because a competitor in the cost-sensitivity experiment is rather computationally inefficient. Data streams are generated by permuting datasets into different random orders. We perform sub-sampling on *eurlex-eurovec*, *delicious* and *nuswide* after computing the permutation so that each stream contains a diferent set of original instances for the three datasets.

All LSDR algorithms, except for competitors run on *delicious* and *eurlex-eurovec*, are coupled with online ridge regression and three different code space dimensions, $M = 10\%$, 25%, and 50% of K, are considered. For DPP we fix $\lambda = 1$ and follow [1] to use the time-decreasing learning rate $\eta = \frac{2}{\sqrt{t}}\frac{M}{K}$, and parameters of other algorithms will be elaborated along with their details in the corresponding section. For the two larger datasets *delicious* and *eurlex-eurovec*, we implement both DPP and O-BR using gradient descent instead of online ridge regression for calculating $\mathbf{W}_t$, where O-BR is the competitor that will be elaborated

[3] `CAL500`, `emotions`, `scene`, `yeast`, `enron`, `Corel5k`, `mediamill`, `nuswide`, `medical`, `delicious` and `eurlex − eurovec`

Table 3: Algorithms being compared in the experiments

acronym	full name	dimension reduction	encode	basis transform	decode	cost-sensitivity
O-BR	Online Binary Relevance	-	no	-	-	no
O-CS	Online Compressed Sensing	yes	random (static)	-	compressed sensing	no
O-RAND	Online Random Projection	yes	random (static)	-	pseudo inverse	no
DPP-PBC	Dynamic Principal Projection (DPP) with Principal Basis Correction	yes	online PCA (dynamic)	exact	PCA	no
DPP-PBT	Dynamic Principal Projection (DPP) with Principal Basis Transform (PBT)	yes	online PCA (dynamic)	approximate	PCA	no
CS-DPP	Cost-Sensitive DPP (with PBT)	yes	online PCA	approximate	PCA	yes

Table 4: 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

in Section 5.2. In particular, for PBC of DPP we replace the update of the online ridge regressor (6) with online gradient descent, while for PBT we replace (13), the update after basis transform, with a gradient descent update as well. Note that even with online ridge regression replaced with gradient descent, the ability of DPP with PBT or PBC to handle the basis drifting problem remains unchanged. 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*.

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: DPP vs. O-BR on large datasets

We consider four different cost functions, Hamming loss, Normalized rank loss, F1 loss and Accuracy loss.

$$c_{\text{HAM}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K}\left(\sum_{k=1}^{K} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!]\right)$$

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

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

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

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 remark that *lower* average cumulative cost imply better performance. We report the average results of each experiment after 15 repetitions.

5.2 Necessity of Online LSDR

In this experiment, we aim to justify the necessity to address LSDR for OMLC problems. We demonstrate that the ability of LSDR to preserve 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, which 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 [28] with online ridge regressor. The only difference between DPP and O-BR is whether the algorithm incorporates LSDR.

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 1, which contains error bars that represent the standard error of the average results. The standard errors are naturally larger when M is larger or when t (number of iterations) is small, but in general for $M \geq 25\% \cdot K$ and for $t \geq 400$ the standard errors are small enough to justify the difference. The complete results are listed in Appendix B.1.

The results from the first two rows of Figure 1 show that DPP with $M = 10\%$ of K performs competitively and even better than O-BR as p increases on dataset *emotions*. The

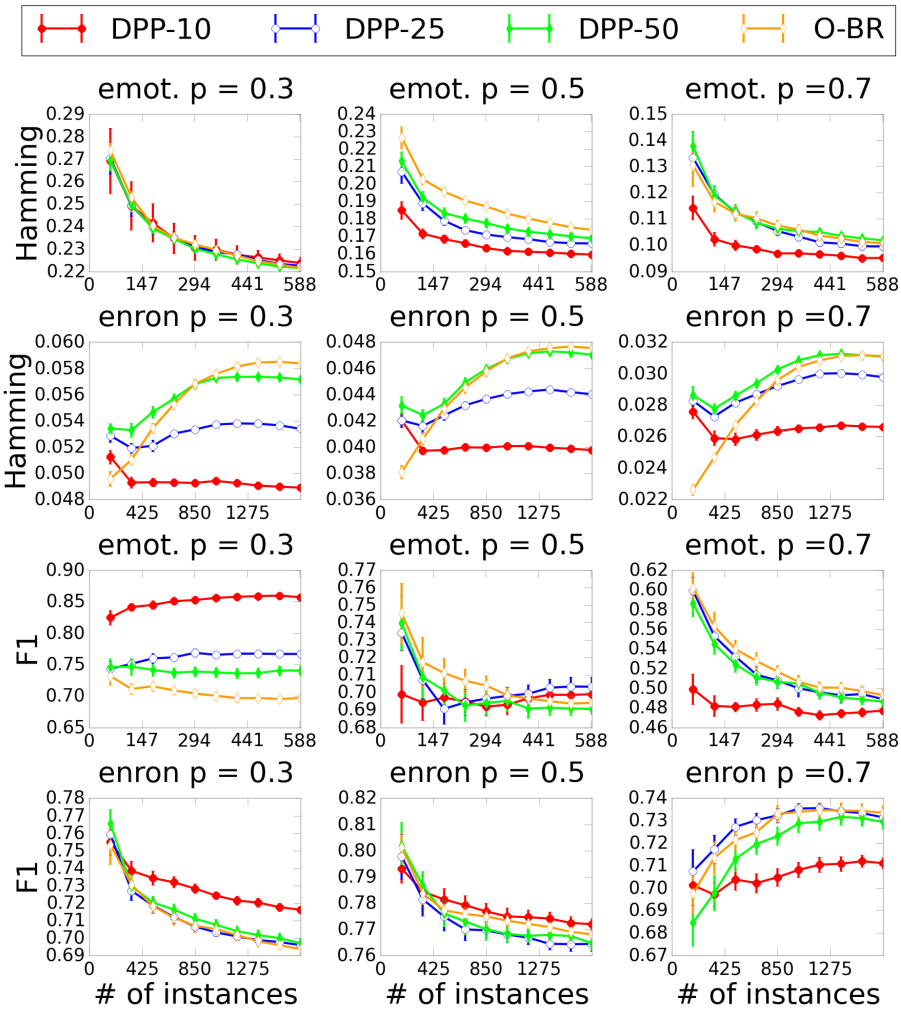

Fig. 1: DPP vs. O-BR on noisy labels

results from the last two rows of Figure 1 show that DPP always performs better on *enron*. We can also observe from Figure 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. 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*.[4] DPP uses $M = 10$ for *delicious* and $M = 25$ for *eurlex-eurovec*. We summarize the results and average run-time in Table 5. Table 5 indicates that DPP coupled

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

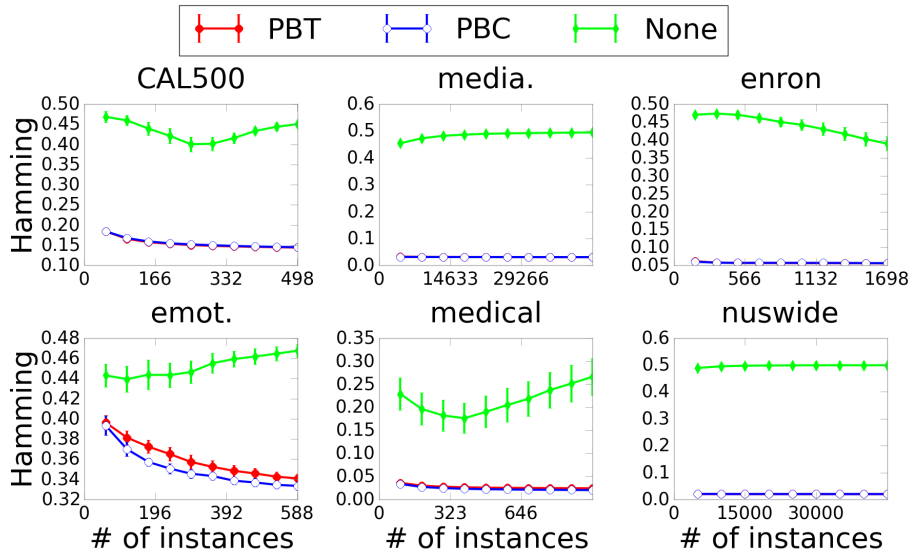

Fig. 2: PBC vs. PBT vs. None, $M = 10\%$ of K

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, as previously highlighted in Section 3.

5.3 Experiments on Basis Drifting

To empirically justify the necessity of handling basis drifting, we compare variants of DPP that (a) incorporates PBC by (6), (b) incorporates PBT by (13), and (c) neglects basis drifting as (5). We plot the results for Hamming loss with $M = 10\%$ of K in Figure 2 on six datasets, and report the complete results in Appendix B.2. The results on all datasets in Figure 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 2 reveals that PBC in general performs slightly better than PBC, reflecting its advantage of exact projection basis correction. 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

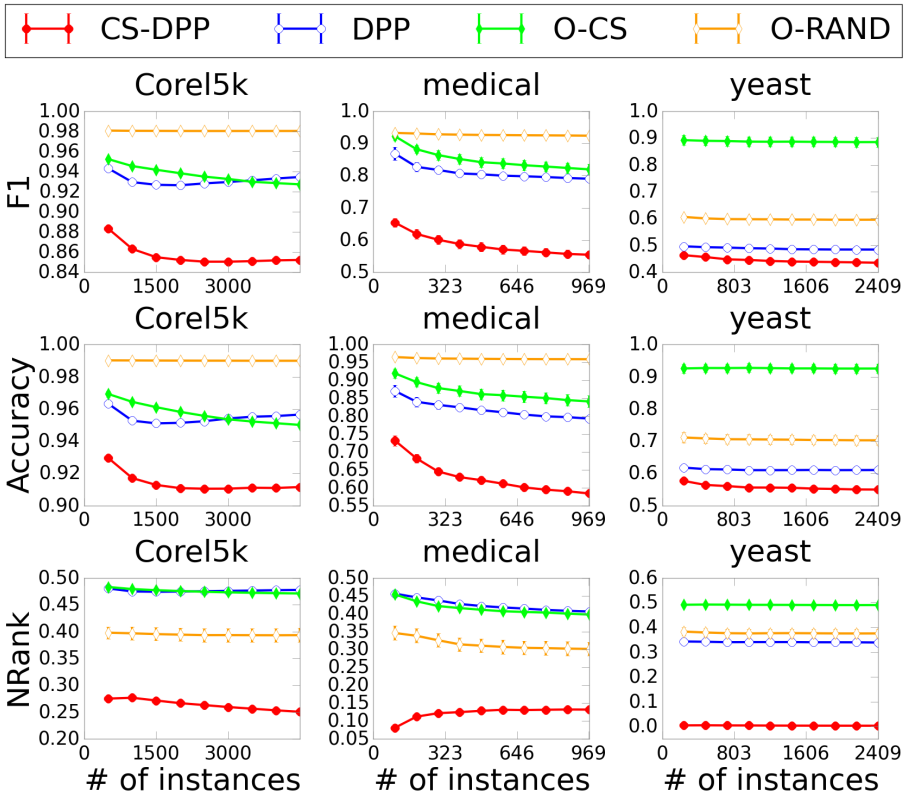

Fig. 3: CS-DPP vs. Others, $M = 10\%$ of K

LSDR algorithm has yet been proposed in the literature. We therefore design two simple online LSDR algorithms, online compressed sensing (O-CS) and online random projection (O-RAND), to compare with CS-DPP. O-CS is a straightforward extension of CS [13] with an online ridge regressor, and we follow [13] to determine the parameter of O-CS. 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 3 on three datasets, and report the complete results in Appendix B.3 Note that the results for CS-DPP here are obtained by using the original label order from the dataset.

5.4.1 CS-DPP versus DPP.

The results of Figure 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 counterpart, DPP.

	10% of K	25% of K	50 % of K
c_{HAM}	0.1458 ± 0.00019	0.1489 ± 0.00012	0.1503 ± 0.00008
c_{NR}	0.1247 ± 0.00224	0.1321 ± 0.00210	0.1371 ± 0.00222
c_{F1}	0.5914 ± 0.00108	0.5956 ± 0.00110	0.5949 ± 0.00101
c_{ACC}	0.7388 ± 0.00105	0.7428 ± 0.00131	0.7426 ± 0.00126

Table 6: Results of CS-DPP on *CAL500* with 50 random label orders

	10% of K	25% of K	50 % of K
c_{HAM}	0.2296 ± 0.00010	0.2162 ± 0.00009	0.2092 ± 0.00001
c_{NR}	0.0064 ± 0.00081	0.0170 ± 0.00242	0.0232 ± 0.00158
c_{F1}	0.4518 ± 0.00919	0.3841 ± 0.00199	0.3784 ± 0.00107
c_{ACC}	0.5448 ± 0.02252	0.4971 ± 0.00379	0.4901 ± 0.00124

Table 7: Results of CS-DPP on *yeast* with 50 random label orders

	10% of K	25% of K	50 % of K
c_{HAM}	0.0562 ± 0.00020	0.0600 ± 0.00011	0.0632 ± 0.00009
c_{NR}	0.1432 ± 0.00333	0.1364 ± 0.00244	0.1305 ± 0.00216
c_{F1}	0.5421 ± 0.00334	0.5392 ± 0.00291	0.5428 ± 0.00293
c_{ACC}	0.6573 ± 0.00360	0.6561 ± 0.00331	0.6627 ± 0.00315

Table 8: Results of CS-DPP on *enron* with 50 random label orders

5.4.2 CS-DPP versus Other Online LSDR Algorithms.

As shown in Figure 3, while DPP generally performs better than O-CS and O-RAND because of the advantage to preserve key label correlations 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 the third row of Figure 3. 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 3. The superiority of CS-DPP justifies the necessity to take cost-sensitivity into account.

5.5 Experiment on Effect of Label Order for CS-DPP

The goal of this experiment is to study how different label orders affect the performance of CS-DPP as our proposed label weighting scheme with (14) is label-order-dependent. To evaluate the impact of label orders, we run CS-DPP with 50 randomly generated label orders and $M = 10\%$, 25% and 50% of K on each dataset. The permutation of each dataset is fixed to the original one given in Mulan [30], which allows the variance of the performance to better indicate the effect of different orders.

We summarize the results of all four different cost functions with mean and standard deviation on datasets *CAL500*, *enron* and *yeast* in Table 6, 7 and 8 respectively, and report the complete results in Appendix B.4. Note that the results of Hamming loss are unaffected by the order of labels, and the reported deviation is due to the randomness from $\mathbf{P}_t$. From the results of Table 6, 7 and 8, we see that standard deviation is generally in a relatively small scale of 10^{-3}, indicating that the performance of CS-DPP is not that sensitive to the order of labels. Closer inspection of Table 7 reveals that the standard deviation of c_{ACC} on *yeast* with

$M = 10\%$ of K (which is only 2 in this case) is somewhat larger, but for sufficiently large M the label order does not seem to cause much variation.

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 with an online LSDR framework derived from PLST, 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 with theoretical guarantees based on our carefully designed label-weighting scheme. 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.

For possible future works, an interesting direction is to design an online LSDR algorithm capable of capturing the key joint information between features and labels. As discussed, the concept to capture such joint information has been investigated for batch MLC [7, 17, 33], but it remains to be challenging for online MLC. Another direction is to apply OMLC algorithms as a fast approximate solver for large-scale batch data, and see how they compete with traditional batch algorithms. Another interesting direction, as mentioned in Section 2, is to design online learning algorithms that achieve cost-sensitivity for the more sophisticated micro- and macro-based criteria.

References

1. Arora, R., Cotter, A., Srebro, N.: Stochastic optimization of PCA with capped MSG. In: NIPS 2013, pp. 1815–1823 (2013)
2. Balasubramanian, K., Lebanon, G.: The landmark selection method for multiple output prediction. In: ICML 2012 (2012)
3. Bartlett, P.: Online convex optimization: ridge regression, adaptivity (2008). URL `https://people.eecs.berkeley.edu/~bartlett/courses/281b-sp08/24.pdf`
4. Bello, J.P., Chew, E., Turnbull, D.: Multilabel classification of music into emotions. In: ICMIR 2008, pp. 325–330 (2008)
5. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: NIPS 2015, pp. 730–738 (2015)
6. Bi, W., Kwok, J.T.: Efficient multi-label classification with many labels. In: ICML 2013, pp. 405–413 (2013)
7. Chen, Y., Lin, H.: Feature-aware label space dimension reduction for multi-label classification. In: NIPS 2012, pp. 1538–1546 (2012)
8. Chua, T., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: NUS-WIDE: a real-world web image database from national university of singapore. In: CIVR 2009 (2009)
9. Crammer, K., Dekel, O., Keshet, J., S.-S., S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research **7**, 551–585 (2006)
10. Dembczynski, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML 2010, pp. 279–286 (2010)
11. Dembczynski, K., Waegeman, W., Cheng, W., Hüllermeier, E.: An exact algorithm for F-measure maximization. In: NIPS 2011, pp. 1404–1412 (2011)
12. Elisseeff, A., Weston, J.: A kernel method for multilabelled classification. In: NIPS 2001 (2001)
13. Hsu, D., Kakade, S., Langford, J., Zhang, T.: Multi-label prediction via compressed sensing. In: NIPS 2009, pp. 772–780 (2009)
14. Kapoor, A., Viswanathan, R., Jain, P.: Multilabel classification using bayesian compressed sensing. In: NIPS 2012, pp. 2654–2662 (2012)

15. Li, C., Lin, H.: Condensed filter tree for cost-sensitive multi-label classification. In: ICML 2014, pp. 423–431 (2014)
16. Li, C., Lin, H., Lu, C.: Rivalry of two families of algorithms for memory-restricted streaming pca. In: AISTATS 2016 (2016)
17. Lin, Z., Ding, G., Hu, M., Wang, J.: Multi-label classification via feature-aware implicit label space encoding. In: ICML 2014, pp. 325–333 (2014)
18. Liu, W., Tsang, I.W., Müller, K.R.: An easy-to-hard learning paradigm for multiple classes and multiple labels. Journal of Machine Learning Research (2017)
19. Lo, H., Wang, J., Wang, H., Lin, S.: Cost-sensitive multi-label learning for audio tag annotation and retrieval. IEEE Trans. Multimedia **13**(3), 518–529 (2011)
20. Mao, Q., Tsang, I.W.H., Gao, S.: Objective-guided image annotation. IEEE Transactions on Image Processing (2013)
21. Nie, J., Kotlowski, W., Warmuth, M.K.: Online PCA with optimal regrets. Journal of Machine Learning Research **17**, 194–200 (2016)
22. Osojnik, A., Panov, P., Deroski, S.: Multi-label classification via multi-target regression on data streams. Machine Learning (2017)
23. Read, J., Bifet, A., Holmes, G., Pfahringer, B.: Streaming multi-label classification. In: Proceedings of the Workshop on Applications of Pattern Analysis (WAPA) 2011, pp. 19–25 (2011)
24. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011)
25. Sun, L., Ji, S., Ye, J.: Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. IEEE TPAMI **33**(1), 194–200 (2011)
26. Tai, F., Lin, H.: Multilabel classification with principal label space transformation. Neural Computation **24**(9), 2508–2542 (2012)
27. Tang, L., Rajan, S., Narayanan, V.K.: Large scale multi-label classification via metalabeler. In: WWW 2009, pp. 211–220 (2009)
28. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, 2nd ed., pp. 667–685 (2010)
29. Tsoumakas, G., Vlahavas, I.P.: Random *k* -labelsets: An ensemble method for multilabel classification. In: ECML 2007, pp. 406–417 (2007)
30. Tsoumakas, G., Xioufis, E.S., Vilcek, J., Vlahavas, I.P.: MULAN: A java library for multi-label learning. Journal of Machine Learning Research **12**, 2411–2414 (2011)
31. Wu, Y., Lin, H.: Progressive k-labelsets for cost-sensitive multi-label classification. Machine Learning (2016). Accepted for Special Issue of ACML 2016
32. Xioufis, E.S., Spiliopoulou, M., Tsoumakas, G., Vlahavas, I.P.: Dealing with concept drift and class imbalance in multi-label stream classification. In: IJCAI 2011, pp. 1583–1588 (2011)
33. Yu, H., Jain, P., Kar, P., Dhillon, I.S.: Large-scale multi-label learning with missing labels. In: ICML 2014, pp. 593–601 (2014)
34. Zhang, X., Graepel, T., Herbrich, R.: Bayesian online learning for multi-label and multi-variate performance measures. In: AISTATS 2010 (2010)

Appendix A. Proof of Lemmas and Theorems

Appendix A.1 Proof of Lemma 2

Lemma 2 *Suppose* $(\mathbf{Q}_t, \sigma_t)$ *is obtained after an updated of Capped MSG such that* $\mathbf{U}_t = \mathbf{Q}_t diag(\sigma_t)\mathbf{Q}_t^\top$. *If* Γ_t *is a discrete probability distribution over events* $\{\mathbf{Q}_t^{-i}\}_{i=1}^{M+1}$ *with probability of* $\mathbf{Q}_t^{-i}$ *being* $1 - \sigma_t[i]$, *we have for any* $\mathbf{y}$

$$\mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\mathbf{y}^\top(\mathbf{I} - \mathbf{P}_t^\top \mathbf{P}_t)\mathbf{y}] = \mathbf{y}^\top(\mathbf{I} - \mathbf{U}_t)\mathbf{y} \tag{16}$$

Proof We first formally show that Γ_t is a well-defined probability distribution. By the definition of the projection operator of Capped MSG we have $0 \leq \sigma_t[i] \leq 1$ for each $\sigma_t[i]$ and $\sum_{i=1}^{M+1} 1 - \sigma_t[i] = M + 1 - \sum_{i=1}^{M+1} \sigma_t[i] = 1$ with $tr(\mathbf{U}_t) = M$. Γ_t is therefore a well-defined probability distribution.

Then it suffices to show that $\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\mathbf{P}_t^\top\mathbf{P}_t]=\mathbf{U}_t$ as

$$\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\mathbf{y}^\top(\mathbf{I}-\mathbf{P}_t^\top\mathbf{P}_t)\mathbf{y}]=\|\mathbf{y}\|_2^2-\mathbf{y}^\top\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\mathbf{P}_t^\top\mathbf{P}_t]\mathbf{y}$$

To see that $\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\mathbf{P}_t^\top\mathbf{P}_t]=\mathbf{U}_t$, first notice that by orthogonality of rows of $\mathbf{Q}_t$ we have $\mathbf{U}_t=\sum_{j=1}^{M+1}\sigma_t(j)\mathbf{e}_j\mathbf{e}_j^\top$ where $\mathbf{e}_j$ is the j-th row of $\mathbf{Q}_t$. We then have

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

where (a) is by $\sum_{i=1}^{M+1}\sigma_t[i]=M$

Appendix A.2 Proof of Theorem 3

Theorem 3 *With the definitions of* $\mathbf{H}_t$ *in* (7), $\mathbf{H}^*$ *in* (9), $\frac{\mathcal{R}}{T}$ *in* (8) *and* Δ_t *in* (10)*, assume that* $\|\mathbf{x}_t\|\leq 1$, $\|\mathbf{y}_t\|\leq 1$ *and* $\|\mathbf{H}_t\mathbf{x}_t-\mathbf{y}_t\|_2^2\leq\epsilon$.

1. *For any given* T*, the expected cumulative regret* $\mathcal{R}$ *is upper-bounded by*

$$(1+\epsilon)\sum_{t=1}^{T}\Delta_t+\frac{M}{2}\|\mathbf{H}^*\|_F^2+2\epsilon Md\log\left(1+\frac{T}{d}\right).$$

2. *If* $\lim_{T\to\infty}\Delta_T=0$ *and* $\|\mathbf{H}^*\|_F\leq h^*$ *across all iterations,* $\lim_{T\to\infty}\frac{\mathcal{R}}{T}=0$.

Proof We start by separating the definition of $\mathcal{R}$ to two terms: one for how $\mathbf{P}_t$ in MSG converges to $\mathbf{P}^*$, and the other for how $\mathbf{W}_t^{\text{PBC}}$ for $\mathbf{P}_t$ in ridge regression differs to $\mathbf{W}_\#$ for $\mathbf{P}^*$. For simplicity, we will denote $\mathbf{W}_t^{\text{PBC}}$ by $\mathbf{W}_t$. Then,

$$\begin{aligned}\mathcal{R}=&\sum_{t=1}^{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\ell^{(t)}(\mathbf{W}_t,\mathbf{P}_t)-\ell^{(t)}(\mathbf{W}_\#,\mathbf{P}^*)]\\=&+\sum_{t=1}^{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\|\mathbf{W}_t^\top\mathbf{x}_t-\mathbf{P}_t\mathbf{y}_t\|_2^2+\|(\mathbf{I}-\mathbf{P}_t^\top\mathbf{P}_t)\mathbf{y}_t\|_2^2]\\&-\sum_{t=1}^{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\|\mathbf{W}_\#^\top\mathbf{x}_t-\mathbf{P}^*\mathbf{y}_t\|_2^2+\|(\mathbf{I}-(\mathbf{P}^*)^\top\mathbf{P}^*)\mathbf{y}_t\|_2^2]\\=&\underbrace{+\sum_{t=1}^{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\|(\mathbf{I}-\mathbf{P}_t^\top\mathbf{P}_t)\mathbf{y}_t\|_2^2]-\|(\mathbf{I}-(\mathbf{P}^*)^\top\mathbf{P}^*)\mathbf{y}_t\|_2^2]}_{\mathcal{R}_{\text{MSG}}}\\&\underbrace{+\sum_{t=1}^{T}\mathbb{E}_{\mathbf{P}_t\sim\Gamma_t}[\|\mathbf{W}_t^\top\mathbf{x}_t-\mathbf{P}_t\mathbf{y}_t\|_2^2]-\|\mathbf{W}_\#^\top\mathbf{x}_t-\mathbf{P}^*\mathbf{y}_t\|_2^2}_{\mathcal{R}_{\text{ridge}}}\end{aligned}$$

We can bound $\mathcal{R}_{\text{MSG}}$ first. Let $\mathbf{U}_t = \mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\mathbf{P}_t^\top \mathbf{P}_t]$ and $\mathbf{U}^* = (\mathbf{P}^*)^\top \mathbf{P}^*$, by linearity of expectation,

$$\begin{aligned}\mathcal{R}_{\text{MSG}} &= \sum_{t=1}^{T} \mathbf{y}_t^\top (\mathbf{U}_t - \mathbf{U}^*)\mathbf{y}_t \\ &\leq \sum_{t=1}^{T} \|\mathbf{U}_t - \mathbf{U}^*\|_2 \qquad (17) \\ &\leq \sum_{t=1}^{T} \Delta_t\end{aligned}$$

where (17) from the assumption of $\|\mathbf{y}_t\|_2 \leq 1$ and the definition of the matrix 2-norm.

Next, we bound $\mathcal{R}_{\text{ridge}}$. With the definitions of $\mathbf{H}_t$ in (7) and $\mathbf{H}^*$ in (9), $\mathcal{R}_{\text{ridge}}$ can be further decomposed to

$$\begin{aligned}\mathcal{R}_{\text{ridge}} = &+ \underbrace{\sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{P}_t \sim \Gamma_t}[\|\mathbf{P}_t(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2] - \|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 \right)}_{\mathcal{R}_1} \\ &+ \underbrace{\sum_{t=1}^{T} \left(\|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 - \|\mathbf{P}^*((\mathbf{H}^*)^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 \right)}_{\mathcal{R}_2}.\end{aligned}$$

Bounding $\mathcal{R}_1$ is very similar to bounding $\mathcal{R}_{\text{MSG}}$. In particular,

$$\begin{aligned}\mathcal{R}_1 &= \sum_{t=1}^{T} (\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)^\top (\mathbf{U}_t - \mathbf{U}^*)(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t) \\ &\leq \sum_{t=1}^{T} \epsilon \|\mathbf{U}_t - \mathbf{U}^*\|_2 \qquad (18) \\ &\leq \sum_{t=1}^{T} \epsilon \Delta_t\end{aligned}$$

where (18) follows from the assumption of $\|\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t\|_2^2 \leq \epsilon$.

The term $\mathcal{R}_2$ can be viewed as an online ridge regression process from $\mathbf{x}$ to $\mathbf{P}^*\mathbf{y}$, because it can be easily proved that $\mathbf{H}_t(\mathbf{P}^*)^\top$ is the ridge regression solution after receiving $(\mathbf{x}_1, \mathbf{P}^*\mathbf{y}_1)$, $(\mathbf{x}_2, \mathbf{P}^*\mathbf{y}_2)$, …, $(\mathbf{x}_{t-1}, \mathbf{P}^*\mathbf{y}_{t-1})$. Also, as discussed in Section 3.4, $\mathbf{W}_{\#} = \mathbf{H}^*(\mathbf{P}^*)^\top$ is the optimal linear regression solution of $\{(\mathbf{x}_t, \mathbf{P}^*\mathbf{y}_t)\}_{t=1}^T$. The assumption of $\|\mathbf{H}_t\mathbf{x}_t - \mathbf{y}_t\|_2^2 \leq \epsilon$ implies that

$$\|\mathbf{P}^*\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{P}^*\mathbf{y}_t\|_2^2 = (\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)^\top \mathbf{U}^* (\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t) \leq \epsilon$$

as well. Similarly, he assumption of $\|\mathbf{y}_t\|_2 \leq 1$ implies that $\|\mathbf{P}^* y_t\|_2 \leq 1$. Then, a standard ridge regression analysis (see, e.g. [3]) by provng that $\mathbf{A}_t = \lambda\mathbf{I} + \sum_{i=1}^{t-1} \mathbf{x}_i \mathbf{x}_i^\top$ grows linearly

with t leads to

$$\begin{aligned}\mathcal{R}_2 &= \sum_{t=1}^{T}\left(\|\mathbf{P}^*(\mathbf{H}_t^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2 - \|\mathbf{P}^*((\mathbf{H}^*)^\top \mathbf{x}_t - \mathbf{y}_t)\|_2^2\right) \\ &\leq \frac{1}{2}\|\mathbf{P}^*\mathbf{H}^*\|_F^2 + 2\epsilon M d \log\left(1+\frac{T}{d}\right) \\ &\leq \frac{M}{2}\|\mathbf{H}^*\|_F^2 + 2\epsilon M d \log\left(1+\frac{T}{d}\right) \qquad (19)\end{aligned}$$

where (19) is because $\|\mathbf{P}^*\|_F^2 = tr(\mathbf{U}^*) = M$.

Summing $\mathcal{R}_{\text{MSG}}$, $\mathcal{R}_1$ and $\mathcal{R}_2$ results in

$$\mathcal{R} \leq (1+\epsilon)\sum_{t=1}^{T}\Delta_t + \frac{M}{2}\|\mathbf{H}^*\|_F^2 + 2\epsilon M d \log(1+\frac{T}{d}), \qquad (20)$$

which proves the first part of the theorem. The second part easily follows because the convergence of a sequence implies the convergence of the mean.

Appendix A.3 Proof of Lemma 4

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*

$$c(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} \delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!] \qquad (21)$$

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

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

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

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

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 k_i and $k_i < k_{i+1}$. If such k_i does not exist than (21) holds trivially by $c(\mathbf{y}, \mathbf{y}) = 0$. Otherwise, by the condition of c we have

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

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 $\hat{\mathbf{y}}_{\text{real}}^{(k_i)} = \hat{\mathbf{y}}_{\text{pred}}^{(k_{i+1})}$ 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 $\hat{\mathbf{y}}_{\text{pred}}^{(k_1)} = \hat{\mathbf{y}}$ and $\hat{\mathbf{y}}_{\text{real}}^{(k_L)} = \mathbf{y}$ and $c(\mathbf{y},\mathbf{y}) = 0$.

Appendix 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 left orthogonal matrix $\mathbf{P}$, if $c(\cdot,\cdot)$ satisfies the condition of Lemma 4, the prediction cost*

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

where $\mathbf{z}_\mathbf{C} = \mathbf{P}(\mathbf{y}'_\mathbf{C})$ and $\mathbf{y}'_\mathbf{C} = \mathbf{C}\mathbf{y} - \mathbf{o}$ with respect to any fixed reference point $\mathbf{o}$.

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

$$\mathbf{C} = \text{diag}(\sqrt{\delta^{(1)}}, ..., \sqrt{\delta^{(K)}}) \tag{22}$$

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 $\text{round}(\tilde{\mathbf{y}})$ 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 satisfies the condition in Lemma 4, we have*

$$c(\mathbf{y},\hat{\mathbf{y}}) \leq \|\mathbf{C}\mathbf{y} - \tilde{\mathbf{y}}\|^2 \tag{23}$$

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

$$\delta^{(k)} [\![\mathbf{y}[k] \neq \hat{\mathbf{y}}[k]]\!] \leq (\sqrt{\delta^{(k)}}\mathbf{y}[k] - \tilde{\mathbf{y}}[k])^2 \tag{24}$$

When $\delta^{(k)} = 0$, (24) holds trivially. When $\delta^{(k)} > 0$, we have

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

where the second equality uses the fact that $\delta^{(k)} > 0$. As $\delta^{(k)} \geq 0$ holds by its definition, (24) holds for every k. Summing (24) with respect to all k then completes the proof.

With Lemma 6 established, we now prove Theorem 5.

Proof (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 $\hat{\mathbf{y}} = \mathrm{round}(\tilde{\mathbf{y}})$. Then for any $(\mathbf{x}, \mathbf{y})$ we have

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

where we recall that $\bar{\mathbf{y}}'_{\mathbf{C}} = \mathbf{C}\mathbf{y} - \mathbf{o}$ and $\mathbf{z}_{\mathbf{C}} = \mathbf{P}(\mathbf{y}'_{\mathbf{c}})$. (a) is from Lemma 24, while (b) and (c) follow from the orthogonal rows of $\mathbf{P}$.

We note that the proof above closely follows the proof of Theorem 1 in [26], while the key difference comes from Lemma 6 to handle the weighted Hamming loss.

Appendix B. Complete Results of Experiments

Here we report the complete results of each experiment.

Appendix B.1 Necessity of Online LSDR

We report the complete results of comparison between O-BR and DPP with $M = 10\%, 25\%$ and 50% of K from Table 9 to Table 11 with respect to all four evaluation criteria, where the best values (the lowest) are marked in bold.

The results show that DPP outperforms O-BR as the value of p increases with respect to Hamming loss, F1 loss and Accuracy loss, demonstrating the robustness of DPP. On the otter hand, the results related to Normalized rank loss from Table 9 to Table 11 show that, while DPP cannot outperform O-BR regarding this specific criterion, DPP does 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.

Appendix B.2 Experiments on Basis Drifting

The complete results of comparison between DPP using (1) PBC, (2) PBT, and (3) nothing regarding Hamming loss can be found in Table 12, Table 13 and Table 14, 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 also compare CS-DPP coupled with PBC/PBT/none on F1 loss, Accuracy loss and Normalized rank loss, and summarize the results in the same tables. From these results we can again draw the same conclusion as that in Section 5.3. 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.

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	O-BR	0.1130 ± 0.0003	$\mathbf{0.823 \pm 0.002}$	$\mathbf{0.453 \pm 0.001}$	$\mathbf{0.898 \pm 0.001}$
	DPP-50	0.1143 ± 0.0001	$\mathbf{0.823 \pm 0.002}$	0.455 ± 0.001	$\mathbf{0.897 \pm 0.001}$
	DPP-25	0.1133 ± 0.0002	0.830 ± 0.003	0.454 ± 0.001	$\mathbf{0.900 \pm 0.001}$
	DPP-10	$\mathbf{0.1113 \pm 0.0002}$	0.837 ± 0.002	0.458 ± 0.001	0.911 ± 0.002
Corel5k	O-BR	$\mathbf{0.0070 \pm 0.0000}$	0.949 ± 0.001	$\mathbf{0.496 \pm 0.000}$	$\mathbf{0.957 \pm 0.001}$
	DPP-50	0.0072 ± 0.0000	$\mathbf{0.945 \pm 0.001}$	$\mathbf{0.496 \pm 0.001}$	$\mathbf{0.957 \pm 0.001}$
	DPP-25	0.0072 ± 0.0000	0.949 ± 0.001	$\mathbf{0.497 \pm 0.000}$	$\mathbf{0.958 \pm 0.000}$
	DPP-10	0.0071 ± 0.0000	0.949 ± 0.001	0.498 ± 0.000	0.960 ± 0.001
emotions	O-BR	$\mathbf{0.2213 \pm 0.0011}$	$\mathbf{0.697 \pm 0.005}$	$\mathbf{0.480 \pm 0.004}$	$\mathbf{0.719 \pm 0.005}$
	DPP-50	$\mathbf{0.2214 \pm 0.0013}$	0.740 ± 0.008	0.504 ± 0.005	0.764 ± 0.004
	DPP-25	$\mathbf{0.2226 \pm 0.0013}$	0.767 ± 0.006	0.527 ± 0.003	0.783 ± 0.002
	DPP-10	$\mathbf{0.2238 \pm 0.0026}$	0.857 ± 0.003	0.570 ± 0.002	0.858 ± 0.004
enron	O-BR	0.0584 ± 0.0002	$\mathbf{0.694 \pm 0.002}$	$\mathbf{0.386 \pm 0.001}$	$\mathbf{0.766 \pm 0.002}$
	DPP-50	0.0572 ± 0.0002	$\mathbf{0.697 \pm 0.003}$	$\mathbf{0.388 \pm 0.001}$	$\mathbf{0.770 \pm 0.002}$
	DPP-25	0.0534 ± 0.0002	$\mathbf{0.696 \pm 0.002}$	0.397 ± 0.001	$\mathbf{0.767 \pm 0.002}$
	DPP-10	$\mathbf{0.0489 \pm 0.0001}$	0.716 ± 0.002	0.414 ± 0.001	0.784 ± 0.002
mediamill	O-BR	$\mathbf{0.0271 \pm 0.0000}$	$\mathbf{0.640 \pm 0.001}$	$\mathbf{0.403 \pm 0.000}$	$\mathbf{0.721 \pm 0.000}$
	DPP-50	0.0272 ± 0.0000	$\mathbf{0.640 \pm 0.001}$	$\mathbf{0.402 \pm 0.000}$	$\mathbf{0.721 \pm 0.000}$
	DPP-25	0.0272 ± 0.0000	$\mathbf{0.639 \pm 0.001}$	$\mathbf{0.403 \pm 0.000}$	$\mathbf{0.721 \pm 0.001}$
	DPP-10	0.0272 ± 0.0000	$\mathbf{0.639 \pm 0.001}$	$\mathbf{0.402 \pm 0.000}$	$\mathbf{0.720 \pm 0.001}$
medical	O-BR	$\mathbf{0.0168 \pm 0.0001}$	$\mathbf{0.550 \pm 0.004}$	$\mathbf{0.448 \pm 0.004}$	$\mathbf{0.563 \pm 0.005}$
	DPP-50	0.0177 ± 0.0001	$\mathbf{0.544 \pm 0.006}$	$\mathbf{0.446 \pm 0.002}$	$\mathbf{0.556 \pm 0.003}$
	DPP-25	0.0183 ± 0.0001	0.577 ± 0.004	0.469 ± 0.005	0.589 ± 0.005
	DPP-10	0.0190 ± 0.0001	0.645 ± 0.006	0.538 ± 0.003	0.651 ± 0.004
nuswide	O-BR	$\mathbf{0.0151 \pm 0.0000}$	$\mathbf{0.627 \pm 0.001}$	$\mathbf{0.668 \pm 0.000}$	0.632 ± 0.000
	DPP-50	0.0151 ± 0.0000	$\mathbf{0.627 \pm 0.000}$	$\mathbf{0.667 \pm 0.000}$	0.633 ± 0.000
	DPP-25	0.0151 ± 0.0000	$\mathbf{0.627 \pm 0.000}$	$\mathbf{0.667 \pm 0.000}$	$\mathbf{0.632 \pm 0.000}$
	DPP-10	0.0151 ± 0.0000	$\mathbf{0.626 \pm 0.000}$	$\mathbf{0.668 \pm 0.000}$	0.632 ± 0.000
scene	O-BR	$\mathbf{0.1197 \pm 0.0005}$	$\mathbf{0.626 \pm 0.001}$	$\mathbf{0.560 \pm 0.002}$	$\mathbf{0.628 \pm 0.003}$
	DPP-50	0.1282 ± 0.0008	0.695 ± 0.003	0.622 ± 0.002	0.698 ± 0.003
	DPP-25	0.1273 ± 0.0005	0.706 ± 0.003	0.632 ± 0.002	0.710 ± 0.004
	DPP-10	0.1258 ± 0.0004	0.717 ± 0.003	0.643 ± 0.001	0.715 ± 0.002
yeast	O-BR	$\mathbf{0.2034 \pm 0.0004}$	$\mathbf{0.669 \pm 0.002}$	$\mathbf{0.406 \pm 0.001}$	$\mathbf{0.755 \pm 0.002}$
	DPP-50	$\mathbf{0.2032 \pm 0.0004}$	0.678 ± 0.004	0.413 ± 0.002	0.762 ± 0.003
	DPP-25	0.2045 ± 0.0004	0.711 ± 0.004	0.427 ± 0.002	0.783 ± 0.003
	DPP-10	0.2034 ± 0.0005	0.733 ± 0.005	0.443 ± 0.002	0.798 ± 0.009

Table 9: DPP vs. O-BR on Noisy Data, $p = 0.3$

Appendix B.3 Experiments on Cost-sensitivity

We report the complete results of on all datasets with respect to all four cost functions in Table 15 to Table 17, where the best values (the lowest) are marked in bold. These complete results validate the conclusion in Section 5.4.

Appendix B.4 Experiments on Effect of Label Orders

The complete average results and the corresponding standard deviations of CS-DPP run on 50 random label orders are reported in Table 18. The results indicate that the standard deviation over the average results of 50 random orders are of 10^{-3} scale generally, indicating that our CS-DPP is relatively not sensitive to the change of label order. On the other hand, the results of CS-DPP have comparatively large deviation on several datasets for some cost functions, such as the Normalized rank loss on dataset *emotions* with $M = 10\%$ of K. We attribute the reason to the instability of interaction between the randomness of $\mathbf{P}_t$ and different label orders based on the fact that larger deviations are observed only when $M = 10\%$ of K.

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	O-BR	$\mathbf{0.0815 \pm 0.0003}$	$\mathbf{0.925 \pm 0.002}$	0.483 ± 0.001	$\mathbf{0.961 \pm 0.001}$
	DPP-50	0.0834 ± 0.0001	$\mathbf{0.925 \pm 0.002}$	$\mathbf{0.480 \pm 0.001}$	$\mathbf{0.962 \pm 0.001}$
	DPP-25	0.0823 ± 0.0002	0.932 ± 0.002	0.483 ± 0.000	$\mathbf{0.961 \pm 0.001}$
	DPP-10	$\mathbf{0.0816 \pm 0.0002}$	0.947 ± 0.002	0.485 ± 0.001	0.970 ± 0.001
Corel5k	O-BR	$\mathbf{0.0049 \pm 0.0000}$	0.898 ± 0.001	$\mathbf{0.543 \pm 0.000}$	$\mathbf{0.902 \pm 0.001}$
	DPP-50	0.0051 ± 0.0000	0.899 ± 0.001	$\mathbf{0.544 \pm 0.001}$	$\mathbf{0.900 \pm 0.001}$
	DPP-25	0.0051 ± 0.0000	$\mathbf{0.897 \pm 0.001}$	$\mathbf{0.544 \pm 0.000}$	$\mathbf{0.900 \pm 0.001}$
	DPP-10	0.0051 ± 0.0000	0.898 ± 0.001	$\mathbf{0.543 \pm 0.001}$	$\mathbf{0.902 \pm 0.001}$
emotions	O-BR	0.1736 ± 0.0014	$\mathbf{0.694 \pm 0.004}$	$\mathbf{0.633 \pm 0.003}$	0.698 ± 0.004
	DPP-50	0.1689 ± 0.0017	$\mathbf{0.691 \pm 0.003}$	0.640 ± 0.003	0.706 ± 0.004
	DPP-25	0.1660 ± 0.0015	$\mathbf{0.703 \pm 0.006}$	0.646 ± 0.002	0.706 ± 0.004
	DPP-10	$\mathbf{0.1598 \pm 0.0014}$	$\mathbf{0.699 \pm 0.004}$	0.650 ± 0.002	$\mathbf{0.692 \pm 0.004}$
enron	O-BR	0.0475 ± 0.0002	0.768 ± 0.001	0.491 ± 0.002	$\mathbf{0.809 \pm 0.002}$
	DPP-50	0.0470 ± 0.0002	$\mathbf{0.765 \pm 0.003}$	$\mathbf{0.488 \pm 0.001}$	$\mathbf{0.809 \pm 0.001}$
	DPP-25	0.0440 ± 0.0002	$\mathbf{0.764 \pm 0.003}$	0.491 ± 0.001	$\mathbf{0.806 \pm 0.002}$
	DPP-10	$\mathbf{0.0398 \pm 0.0002}$	0.772 ± 0.002	0.510 ± 0.002	$\mathbf{0.810 \pm 0.002}$
mediamill	O-BR	$\mathbf{0.0217 \pm 0.0000}$	$\mathbf{0.831 \pm 0.001}$	$\mathbf{0.548 \pm 0.000}$	$\mathbf{0.840 \pm 0.001}$
	DPP-50	$\mathbf{0.0217 \pm 0.0000}$	$\mathbf{0.830 \pm 0.001}$	0.550 ± 0.000	$\mathbf{0.839 \pm 0.001}$
	DPP-25	$\mathbf{0.0217 \pm 0.0000}$	$\mathbf{0.830 \pm 0.001}$	0.550 ± 0.001	$\mathbf{0.840 \pm 0.001}$
	DPP-10	$\mathbf{0.0217 \pm 0.0000}$	$\mathbf{0.830 \pm 0.001}$	0.549 ± 0.000	$\mathbf{0.840 \pm 0.001}$
medical	O-BR	$\mathbf{0.0153 \pm 0.0001}$	0.570 ± 0.002	$\mathbf{0.655 \pm 0.005}$	0.568 ± 0.004
	DPP-50	0.0163 ± 0.0001	0.563 ± 0.005	$\mathbf{0.661 \pm 0.003}$	0.577 ± 0.004
	DPP-25	0.0160 ± 0.0001	0.569 ± 0.004	$\mathbf{0.664 \pm 0.005}$	0.570 ± 0.004
	DPP-10	0.0157 ± 0.0001	$\mathbf{0.561 \pm 0.003}$	0.690 ± 0.003	$\mathbf{0.565 \pm 0.003}$
nuswide	O-BR	$\mathbf{0.0109 \pm 0.0000}$	0.537 ± 0.000	0.730 ± 0.000	$\mathbf{0.537 \pm 0.000}$
	DPP-50	0.0110 ± 0.0000	0.537 ± 0.000	0.730 ± 0.000	$\mathbf{0.537 \pm 0.000}$
	DPP-25	0.0110 ± 0.0000	$\mathbf{0.536 \pm 0.000}$	0.730 ± 0.000	$\mathbf{0.536 \pm 0.000}$
	DPP-10	0.0109 ± 0.0000	0.536 ± 0.000	$\mathbf{0.730 \pm 0.000}$	$\mathbf{0.537 \pm 0.000}$
scene	O-BR	0.0965 ± 0.0006	0.533 ± 0.003	$\mathbf{0.718 \pm 0.002}$	0.533 ± 0.003
	DPP-50	0.0926 ± 0.0004	0.525 ± 0.002	0.731 ± 0.002	0.524 ± 0.002
	DPP-25	0.0915 ± 0.0004	$\mathbf{0.519 \pm 0.003}$	0.739 ± 0.001	0.522 ± 0.003
	DPP-10	$\mathbf{0.0902 \pm 0.0004}$	0.524 ± 0.003	0.740 ± 0.001	$\mathbf{0.515 \pm 0.004}$
yeast	O-BR	0.1581 ± 0.0005	$\mathbf{0.853 \pm 0.002}$	$\mathbf{0.518 \pm 0.001}$	$\mathbf{0.875 \pm 0.001}$
	DPP-50	0.1586 ± 0.0005	$\mathbf{0.850 \pm 0.002}$	$\mathbf{0.520 \pm 0.001}$	$\mathbf{0.873 \pm 0.002}$
	DPP-25	0.1573 ± 0.0004	0.860 ± 0.002	0.524 ± 0.001	$\mathbf{0.878 \pm 0.002}$
	DPP-10	$\mathbf{0.1543 \pm 0.0004}$	0.876 ± 0.004	0.531 ± 0.002	0.890 ± 0.002

Table 10: DPP vs. O-BR on Noisy Data, $p = 0.5$

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	O-BR	$\mathbf{0.0483 \pm 0.0003}$	$\mathbf{0.985 \pm 0.001}$	$\mathbf{0.495 \pm 0.000}$	$\mathbf{0.990 \pm 0.001}$
	DPP-50	0.0499 ± 0.0002	$\mathbf{0.983 \pm 0.001}$	$\mathbf{0.495 \pm 0.000}$	0.991 ± 0.000
	DPP-25	0.0502 ± 0.0002	$\mathbf{0.984 \pm 0.001}$	0.495 ± 0.000	0.991 ± 0.000
	DPP-10	0.0490 ± 0.0002	0.987 ± 0.000	0.496 ± 0.000	0.992 ± 0.001
Corel5k	O-BR	$\mathbf{0.0029 \pm 0.0000}$	$\mathbf{0.716 \pm 0.002}$	$\mathbf{0.647 \pm 0.001}$	$\mathbf{0.716 \pm 0.002}$
	DPP-50	0.0031 ± 0.0000	$\mathbf{0.713 \pm 0.001}$	$\mathbf{0.646 \pm 0.001}$	$\mathbf{0.714 \pm 0.002}$
	DPP-25	0.0031 ± 0.0000	$\mathbf{0.714 \pm 0.001}$	$\mathbf{0.647 \pm 0.001}$	$\mathbf{0.715 \pm 0.002}$
	DPP-10	0.0031 ± 0.0000	$\mathbf{0.712 \pm 0.002}$	$\mathbf{0.646 \pm 0.001}$	$\mathbf{0.714 \pm 0.002}$
emotions	O-BR	0.1007 ± 0.0013	0.493 ± 0.006	$\mathbf{0.759 \pm 0.002}$	0.490 ± 0.006
	DPP-50	0.1017 ± 0.0011	0.486 ± 0.005	$\mathbf{0.758 \pm 0.002}$	0.493 ± 0.005
	DPP-25	0.0993 ± 0.0015	0.489 ± 0.006	$\mathbf{0.757 \pm 0.002}$	0.491 ± 0.005
	DPP-10	$\mathbf{0.0951 \pm 0.0013}$	$\mathbf{0.477 \pm 0.004}$	$\mathbf{0.763 \pm 0.002}$	$\mathbf{0.474 \pm 0.004}$
enron	O-BR	0.0311 ± 0.0002	0.734 ± 0.003	$\mathbf{0.634 \pm 0.002}$	0.753 ± 0.003
	DPP-50	0.0311 ± 0.0002	0.729 ± 0.003	$\mathbf{0.633 \pm 0.002}$	0.745 ± 0.002
	DPP-25	0.0298 ± 0.0002	0.731 ± 0.002	$\mathbf{0.635 \pm 0.002}$	0.742 ± 0.003
	DPP-10	$\mathbf{0.0266 \pm 0.0002}$	$\mathbf{0.711 \pm 0.003}$	0.644 ± 0.001	$\mathbf{0.726 \pm 0.003}$
mediamill	O-BR	$\mathbf{0.0130 \pm 0.0000}$	$\mathbf{0.714 \pm 0.001}$	$\mathbf{0.643 \pm 0.000}$	0.715 ± 0.000
	DPP-50	0.0130 ± 0.0000	$\mathbf{0.715 \pm 0.000}$	$\mathbf{0.643 \pm 0.000}$	$\mathbf{0.714 \pm 0.000}$
	DPP-25	0.0130 ± 0.0000	$\mathbf{0.714 \pm 0.000}$	$\mathbf{0.643 \pm 0.000}$	0.714 ± 0.001
	DPP-10	0.0130 ± 0.0000	$\mathbf{0.715 \pm 0.001}$	$\mathbf{0.643 \pm 0.000}$	0.715 ± 0.001
medical	O-BR	$\mathbf{0.0099 \pm 0.0002}$	0.398 ± 0.007	$\mathbf{0.814 \pm 0.003}$	0.404 ± 0.004
	DPP-50	0.0106 ± 0.0002	0.401 ± 0.005	$\mathbf{0.812 \pm 0.003}$	0.398 ± 0.005
	DPP-25	0.0105 ± 0.0001	0.391 ± 0.004	$\mathbf{0.815 \pm 0.002}$	0.399 ± 0.004
	DPP-10	$\mathbf{0.0097 \pm 0.0001}$	$\mathbf{0.377 \pm 0.004}$	$\mathbf{0.819 \pm 0.003}$	$\mathbf{0.377 \pm 0.005}$
nuswide	O-BR	$\mathbf{0.0066 \pm 0.0000}$	$\mathbf{0.386 \pm 0.001}$	0.808 ± 0.000	$\mathbf{0.386 \pm 0.001}$
	DPP-50	0.0066 ± 0.0000	$\mathbf{0.386 \pm 0.000}$	$\mathbf{0.807 \pm 0.000}$	$\mathbf{0.385 \pm 0.000}$
	DPP-25	0.0066 ± 0.0000	$\mathbf{0.386 \pm 0.000}$	0.807 ± 0.000	$\mathbf{0.386 \pm 0.000}$
	DPP-10	0.0066 ± 0.0000	$\mathbf{0.386 \pm 0.000}$	0.807 ± 0.000	$\mathbf{0.385 \pm 0.001}$
scene	O-BR	0.0562 ± 0.0004	0.328 ± 0.003	$\mathbf{0.841 \pm 0.001}$	0.328 ± 0.002
	DPP-50	$\mathbf{0.0544 \pm 0.0003}$	0.323 ± 0.002	$\mathbf{0.841 \pm 0.001}$	$\mathbf{0.321 \pm 0.002}$
	DPP-25	$\mathbf{0.0542 \pm 0.0005}$	$\mathbf{0.316 \pm 0.002}$	$\mathbf{0.842 \pm 0.001}$	$\mathbf{0.317 \pm 0.002}$
	DPP-10	$\mathbf{0.0538 \pm 0.0005}$	$\mathbf{0.313 \pm 0.002}$	$\mathbf{0.842 \pm 0.001}$	$\mathbf{0.318 \pm 0.002}$
yeast	O-BR	$\mathbf{0.0920 \pm 0.0004}$	$\mathbf{0.746 \pm 0.002}$	$\mathbf{0.625 \pm 0.001}$	$\mathbf{0.747 \pm 0.002}$
	DPP-50	$\mathbf{0.0918 \pm 0.0003}$	$\mathbf{0.748 \pm 0.002}$	$\mathbf{0.627 \pm 0.001}$	$\mathbf{0.747 \pm 0.002}$
	DPP-25	$\mathbf{0.0921 \pm 0.0004}$	$\mathbf{0.747 \pm 0.002}$	$\mathbf{0.627 \pm 0.001}$	$\mathbf{0.746 \pm 0.002}$
	DPP-10	$\mathbf{0.0915 \pm 0.0004}$	$\mathbf{0.748 \pm 0.002}$	$\mathbf{0.626 \pm 0.001}$	$\mathbf{0.746 \pm 0.002}$

Table 11: DPP vs. O-BR on Noisy Data, $p = 0.7$

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	CS-DPP-None	0.4464 ± 0.0074	0.733 ± 0.001	0.393 ± 0.002	0.843 ± 0.001
	CS-DPP-PBT	$\mathbf{0.1443 \pm 0.0001}$	$\mathbf{0.601 \pm 0.001}$	$\mathbf{0.137 \pm 0.001}$	$\mathbf{0.749 \pm 0.001}$
	CS-DPP-PBC	0.1454 ± 0.0002	$\mathbf{0.603 \pm 0.001}$	0.144 ± 0.002	$\mathbf{0.748 \pm 0.001}$
Corel5k	CS-DPP-None	0.4814 ± 0.0063	0.957 ± 0.000	0.357 ± 0.001	0.980 ± 0.000
	CS-DPP-PBT	$\mathbf{0.0099 \pm 0.0000}$	0.853 ± 0.001	0.248 ± 0.001	0.912 ± 0.001
	CS-DPP-PBC	0.0100 ± 0.0000	$\mathbf{0.850 \pm 0.001}$	$\mathbf{0.237 \pm 0.001}$	$\mathbf{0.910 \pm 0.000}$
emotions	CS-DPP-None	0.4787 ± 0.0039	0.618 ± 0.004	0.376 ± 0.008	0.696 ± 0.005
	CS-DPP-PBT	0.3419 ± 0.0033	$\mathbf{0.445 \pm 0.003}$	$\mathbf{0.159 \pm 0.021}$	$\mathbf{0.563 \pm 0.007}$
	CS-DPP-PBC	$\mathbf{0.3301 \pm 0.0012}$	$\mathbf{0.450 \pm 0.007}$	$\mathbf{0.133 \pm 0.023}$	$\mathbf{0.560 \pm 0.009}$
enron	CS-DPP-None	0.4030 ± 0.0160	0.802 ± 0.002	0.385 ± 0.002	0.875 ± 0.001
	CS-DPP-PBT	$\mathbf{0.0560 \pm 0.0001}$	0.534 ± 0.002	$\mathbf{0.124 \pm 0.003}$	$\mathbf{0.642 \pm 0.002}$
	CS-DPP-PBC	0.0565 ± 0.0001	$\mathbf{0.528 \pm 0.002}$	0.132 ± 0.001	$\mathbf{0.638 \pm 0.001}$
mediamill	CS-DPP-None	0.4936 ± 0.0016	0.692 ± 0.016	0.416 ± 0.004	0.728 ± 0.001
	CS-DPP-PBT	0.0309 ± 0.0000	$\mathbf{0.460 \pm 0.000}$	$\mathbf{0.066 \pm 0.002}$	0.583 ± 0.000
	CS-DPP-PBC	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.460 \pm 0.000}$	0.072 ± 0.002	$\mathbf{0.582 \pm 0.000}$
medical	CS-DPP-None	0.1923 ± 0.0352	0.896 ± 0.002	0.346 ± 0.003	0.932 ± 0.003
	CS-DPP-PBT	0.0242 ± 0.0001	0.554 ± 0.012	0.132 ± 0.005	0.583 ± 0.008
	CS-DPP-PBC	$\mathbf{0.0204 \pm 0.0002}$	$\mathbf{0.508 \pm 0.006}$	$\mathbf{0.096 \pm 0.003}$	$\mathbf{0.549 \pm 0.007}$
nuswide	CS-DPP-None	0.4975 ± 0.0006	0.933 ± 0.001	0.520 ± 0.001	0.959 ± 0.001
	CS-DPP-PBT	0.0201 ± 0.0000	$\mathbf{0.649 \pm 0.000}$	$\mathbf{0.356 \pm 0.001}$	$\mathbf{0.675 \pm 0.000}$
	CS-DPP-PBC	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.358 \pm 0.001}$	$\mathbf{0.675 \pm 0.000}$
scene	CS-DPP-None	0.4609 ± 0.0080	0.761 ± 0.003	0.362 ± 0.007	0.825 ± 0.002
	CS-DPP-PBT	$\mathbf{0.1796 \pm 0.0001}$	$\mathbf{0.723 \pm 0.002}$	$\mathbf{0.264 \pm 0.012}$	$\mathbf{0.798 \pm 0.003}$
	CS-DPP-PBC	$\mathbf{0.1797 \pm 0.0001}$	$\mathbf{0.724 \pm 0.002}$	$\mathbf{0.231 \pm 0.016}$	$\mathbf{0.796 \pm 0.002}$
yeast	CS-DPP-None	0.4979 ± 0.0015	0.616 ± 0.002	0.422 ± 0.003	0.727 ± 0.001
	CS-DPP-PBT	$\mathbf{0.2294 \pm 0.0010}$	$\mathbf{0.435 \pm 0.004}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.549 \pm 0.003}$
	CS-DPP-PBC	$\mathbf{0.2307 \pm 0.0011}$	$\mathbf{0.433 \pm 0.003}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.541 \pm 0.003}$

Table 12: CS-DPP with PBC vs. PBT vs. None, M = 10% of K

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	CS-DPP-None	0.4374 ± 0.0100	0.732 ± 0.002	0.392 ± 0.002	0.846 ± 0.002
	CS-DPP-PBT	$\mathbf{0.1471 \pm 0.0002}$	$\mathbf{0.604 \pm 0.001}$	$\mathbf{0.151 \pm 0.002}$	$\mathbf{0.750 \pm 0.001}$
	CS-DPP-PBC	0.1476 ± 0.0001	$\mathbf{0.602 \pm 0.001}$	$\mathbf{0.150 \pm 0.002}$	$\mathbf{0.751 \pm 0.001}$
Corel5k	CS-DPP-None	0.4997 ± 0.0018	0.965 ± 0.000	0.366 ± 0.001	0.983 ± 0.000
	CS-DPP-PBT	$\mathbf{0.0100 \pm 0.0000}$	$\mathbf{0.845 \pm 0.000}$	0.223 ± 0.001	$\mathbf{0.905 \pm 0.000}$
	CS-DPP-PBC	0.0101 ± 0.0000	$\mathbf{0.844 \pm 0.000}$	$\mathbf{0.220 \pm 0.001}$	$\mathbf{0.904 \pm 0.000}$
emotions	CS-DPP-None	0.4988 ± 0.0022	0.631 ± 0.004	0.420 ± 0.005	0.722 ± 0.003
	CS-DPP-PBT	$\mathbf{0.2768 \pm 0.0051}$	$\mathbf{0.401 \pm 0.003}$	$\mathbf{0.078 \pm 0.016}$	$\mathbf{0.513 \pm 0.003}$
	CS-DPP-PBC	$\mathbf{0.2819 \pm 0.0036}$	$\mathbf{0.398 \pm 0.004}$	$\mathbf{0.046 \pm 0.015}$	$\mathbf{0.509 \pm 0.003}$
enron	CS-DPP-None	0.4844 ± 0.0050	0.812 ± 0.002	0.386 ± 0.002	0.884 ± 0.001
	CS-DPP-PBT	$\mathbf{0.0581 \pm 0.0002}$	$\mathbf{0.517 \pm 0.001}$	$\mathbf{0.136 \pm 0.002}$	$\mathbf{0.633 \pm 0.001}$
	CS-DPP-PBC	0.0601 ± 0.0002	$\mathbf{0.519 \pm 0.001}$	$\mathbf{0.135 \pm 0.001}$	$\mathbf{0.633 \pm 0.001}$
mediamill	CS-DPP-None	0.4917 ± 0.0015	0.842 ± 0.009	0.429 ± 0.001	0.759 ± 0.009
	CS-DPP-PBT	$\mathbf{0.0307 \pm 0.0000}$	$\mathbf{0.458 \pm 0.000}$	$\mathbf{0.070 \pm 0.000}$	$\mathbf{0.581 \pm 0.000}$
	CS-DPP-PBC	$\mathbf{0.0307 \pm 0.0000}$	$\mathbf{0.457 \pm 0.000}$	$\mathbf{0.068 \pm 0.000}$	$\mathbf{0.580 \pm 0.000}$
medical	CS-DPP-None	0.4493 ± 0.0161	0.902 ± 0.002	0.361 ± 0.004	0.931 ± 0.004
	CS-DPP-PBT	0.0171 ± 0.0002	0.338 ± 0.005	0.043 ± 0.003	0.374 ± 0.004
	CS-DPP-PBC	$\mathbf{0.0152 \pm 0.0001}$	$\mathbf{0.316 \pm 0.004}$	$\mathbf{0.036 \pm 0.002}$	$\mathbf{0.360 \pm 0.004}$
nuswide	CS-DPP-None	0.4978 ± 0.0007	0.930 ± 0.003	0.523 ± 0.000	0.964 ± 0.001
	CS-DPP-PBT	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	0.334 ± 0.001	$\mathbf{0.675 \pm 0.000}$
	CS-DPP-PBC	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.329 \pm 0.001}$	$\mathbf{0.675 \pm 0.000}$
scene	CS-DPP-None	0.5002 ± 0.0012	0.747 ± 0.002	0.373 ± 0.004	0.830 ± 0.002
	CS-DPP-PBT	$\mathbf{0.1787 \pm 0.0014}$	$\mathbf{0.632 \pm 0.003}$	0.185 ± 0.013	$\mathbf{0.692 \pm 0.003}$
	CS-DPP-PBC	$\mathbf{0.1797 \pm 0.0014}$	$\mathbf{0.631 \pm 0.004}$	$\mathbf{0.142 \pm 0.011}$	$\mathbf{0.697 \pm 0.004}$
yeast	CS-DPP-None	0.4992 ± 0.0014	0.622 ± 0.001	0.424 ± 0.002	0.737 ± 0.001
	CS-DPP-PBT	$\mathbf{0.2139 \pm 0.0006}$	0.389 ± 0.001	$\mathbf{0.017 \pm 0.001}$	$\mathbf{0.495 \pm 0.001}$
	CS-DPP-PBC	$\mathbf{0.2144 \pm 0.0005}$	$\mathbf{0.385 \pm 0.001}$	$\mathbf{0.016 \pm 0.001}$	$\mathbf{0.497 \pm 0.001}$

Table 13: CS-DPP with PBC vs. PBT vs. None, M = 25% of K

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	CS-DPP-None	0.4141 ± 0.0176	0.735 ± 0.002	0.398 ± 0.002	0.844 ± 0.002
	CS-DPP-PBT	$\mathbf{0.1487 \pm 0.0002}$	$\mathbf{0.602 \pm 0.001}$	$\mathbf{0.154 \pm 0.001}$	$\mathbf{0.752 \pm 0.001}$
	CS-DPP-PBC	$\mathbf{0.1490 \pm 0.0002}$	$\mathbf{0.602 \pm 0.001}$	$\mathbf{0.151 \pm 0.001}$	$\mathbf{0.751 \pm 0.001}$
Corel5k	CS-DPP-None	0.5014 ± 0.0017	0.969 ± 0.000	0.369 ± 0.001	0.986 ± 0.000
	CS-DPP-PBT	$\mathbf{0.0101 \pm 0.0000}$	$\mathbf{0.843 \pm 0.000}$	$\mathbf{0.214 \pm 0.001}$	$\mathbf{0.901 \pm 0.000}$
	CS-DPP-PBC	$\mathbf{0.0101 \pm 0.0000}$	$\mathbf{0.842 \pm 0.001}$	$\mathbf{0.213 \pm 0.000}$	0.903 ± 0.001
emotions	CS-DPP-None	0.4941 ± 0.0029	0.631 ± 0.003	0.386 ± 0.004	0.729 ± 0.002
	CS-DPP-PBT	$\mathbf{0.2308 \pm 0.0014}$	$\mathbf{0.381 \pm 0.002}$	$\mathbf{0.034 \pm 0.003}$	$\mathbf{0.481 \pm 0.002}$
	CS-DPP-PBC	$\mathbf{0.2306 \pm 0.0012}$	$\mathbf{0.377 \pm 0.002}$	$\mathbf{0.033 \pm 0.003}$	$\mathbf{0.481 \pm 0.002}$
enron	CS-DPP-None	0.4953 ± 0.0016	0.821 ± 0.003	0.385 ± 0.002	0.889 ± 0.002
	CS-DPP-PBT	$\mathbf{0.0626 \pm 0.0002}$	$\mathbf{0.523 \pm 0.001}$	$\mathbf{0.130 \pm 0.001}$	$\mathbf{0.636 \pm 0.001}$
	CS-DPP-PBC	0.0643 ± 0.0001	$\mathbf{0.522 \pm 0.001}$	$\mathbf{0.129 \pm 0.001}$	$\mathbf{0.636 \pm 0.001}$
mediamill	CS-DPP-None	0.4907 ± 0.0018	0.895 ± 0.008	0.426 ± 0.001	0.838 ± 0.019
	CS-DPP-PBT	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.457 \pm 0.000}$	0.062 ± 0.000	0.581 ± 0.000
	CS-DPP-PBC	$\mathbf{0.0307 \pm 0.0000}$	$\mathbf{0.457 \pm 0.000}$	$\mathbf{0.059 \pm 0.000}$	$\mathbf{0.581 \pm 0.000}$
medical	CS-DPP-None	0.4177 ± 0.0370	0.907 ± 0.002	0.368 ± 0.002	0.944 ± 0.002
	CS-DPP-PBT	0.0136 ± 0.0001	$\mathbf{0.252 \pm 0.002}$	$\mathbf{0.021 \pm 0.001}$	$\mathbf{0.303 \pm 0.002}$
	CS-DPP-PBC	$\mathbf{0.0130 \pm 0.0001}$	$\mathbf{0.250 \pm 0.002}$	$\mathbf{0.019 \pm 0.001}$	$\mathbf{0.299 \pm 0.002}$
nuswide	CS-DPP-None	0.4972 ± 0.0007	0.940 ± 0.004	0.528 ± 0.001	0.964 ± 0.002
	CS-DPP-PBT	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	0.307 ± 0.000	$\mathbf{0.674 \pm 0.000}$
	CS-DPP-PBC	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.304 \pm 0.000}$	$\mathbf{0.675 \pm 0.000}$
scene	CS-DPP-None	0.5015 ± 0.0012	0.745 ± 0.001	0.385 ± 0.002	0.832 ± 0.001
	CS-DPP-PBT	$\mathbf{0.1731 \pm 0.0010}$	$\mathbf{0.554 \pm 0.003}$	0.125 ± 0.005	$\mathbf{0.626 \pm 0.004}$
	CS-DPP-PBC	$\mathbf{0.1720 \pm 0.0015}$	$\mathbf{0.558 \pm 0.003}$	$\mathbf{0.104 \pm 0.009}$	$\mathbf{0.623 \pm 0.004}$
yeast	CS-DPP-None	0.4982 ± 0.0011	0.630 ± 0.001	0.413 ± 0.001	0.745 ± 0.001
	CS-DPP-PBT	$\mathbf{0.2077 \pm 0.0003}$	$\mathbf{0.382 \pm 0.001}$	$\mathbf{0.024 \pm 0.001}$	$\mathbf{0.493 \pm 0.001}$
	CS-DPP-PBC	$\mathbf{0.2079 \pm 0.0003}$	$\mathbf{0.382 \pm 0.001}$	$\mathbf{0.026 \pm 0.001}$	$\mathbf{0.492 \pm 0.001}$

Table 14: CS-DPP with PBC vs. PBT vs. None, M = 50% of K

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	O-CS	0.1610 ± 0.0006	0.953 ± 0.003	0.497 ± 0.001	0.971 ± 0.002
	O-RAND	0.4042 ± 0.0052	0.750 ± 0.004	0.397 ± 0.006	0.858 ± 0.004
	DPP	$\mathbf{0.1453 \pm 0.0001}$	0.654 ± 0.002	0.399 ± 0.001	0.787 ± 0.001
	CS-DPP	$\mathbf{0.1454 \pm 0.0002}$	$\mathbf{0.603 \pm 0.001}$	$\mathbf{0.144 \pm 0.002}$	$\mathbf{0.748 \pm 0.001}$
Corel5k	O-CS	0.0117 ± 0.0000	0.926 ± 0.002	0.470 ± 0.001	0.949 ± 0.001
	O-RAND	0.3734 ± 0.0044	0.980 ± 0.001	0.393 ± 0.013	0.990 ± 0.000
	DPP	$\mathbf{0.0100 \pm 0.0000}$	0.918 ± 0.001	0.470 ± 0.000	0.943 ± 0.000
	CS-DPP	$\mathbf{0.0100 \pm 0.0000}$	$\mathbf{0.850 \pm 0.001}$	$\mathbf{0.237 \pm 0.001}$	$\mathbf{0.910 \pm 0.000}$
emotions	O-CS	$\mathbf{0.3338 \pm 0.0073}$	0.900 ± 0.014	0.508 ± 0.006	0.924 ± 0.009
	O-RAND	0.3847 ± 0.0099	0.621 ± 0.033	0.363 ± 0.020	0.683 ± 0.021
	DPP	$\mathbf{0.3335 \pm 0.0042}$	$\mathbf{0.428 \pm 0.003}$	0.223 ± 0.009	$\mathbf{0.558 \pm 0.004}$
	CS-DPP	$\mathbf{0.3301 \pm 0.0012}$	0.450 ± 0.007	$\mathbf{0.133 \pm 0.023}$	$\mathbf{0.560 \pm 0.009}$
enron	O-CS	0.0739 ± 0.0006	0.885 ± 0.010	0.463 ± 0.004	0.927 ± 0.009
	O-RAND	0.3907 ± 0.0090	0.867 ± 0.007	0.320 ± 0.015	0.923 ± 0.004
	DPP	$\mathbf{0.0563 \pm 0.0001}$	0.552 ± 0.003	0.304 ± 0.001	0.646 ± 0.002
	CS-DPP	$\mathbf{0.0565 \pm 0.0001}$	$\mathbf{0.528 \pm 0.002}$	$\mathbf{0.132 \pm 0.001}$	$\mathbf{0.638 \pm 0.001}$
mediamill	O-CS	0.0485 ± 0.0011	0.821 ± 0.025	0.454 ± 0.009	0.868 ± 0.014
	O-RAND	0.3737 ± 0.0070	0.899 ± 0.007	0.391 ± 0.020	0.950 ± 0.003
	DPP	$\mathbf{0.0308 \pm 0.0000}$	0.474 ± 0.000	0.307 ± 0.000	0.594 ± 0.000
	CS-DPP	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.460 \pm 0.000}$	$\mathbf{0.072 \pm 0.002}$	$\mathbf{0.582 \pm 0.000}$
medical	O-CS	0.0272 ± 0.0006	0.819 ± 0.016	0.397 ± 0.004	0.840 ± 0.017
	O-RAND	0.3674 ± 0.0093	0.924 ± 0.005	0.301 ± 0.019	0.959 ± 0.003
	DPP	$\mathbf{0.0204 \pm 0.0002}$	0.602 ± 0.008	0.311 ± 0.005	0.628 ± 0.008
	CS-DPP	$\mathbf{0.0204 \pm 0.0002}$	$\mathbf{0.508 \pm 0.006}$	$\mathbf{0.096 \pm 0.003}$	$\mathbf{0.549 \pm 0.007}$
nuswide	O-CS	0.0239 ± 0.0004	0.746 ± 0.005	0.600 ± 0.003	0.741 ± 0.003
	O-RAND	0.3707 ± 0.0107	0.956 ± 0.002	0.532 ± 0.013	0.973 ± 0.002
	DPP	$\mathbf{0.0201 \pm 0.0000}$	0.673 ± 0.000	0.580 ± 0.000	0.691 ± 0.000
	CS-DPP	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.358 \pm 0.001}$	$\mathbf{0.675 \pm 0.000}$
scene	O-CS	0.2168 ± 0.0047	0.920 ± 0.010	0.491 ± 0.003	0.902 ± 0.009
	O-RAND	0.3711 ± 0.0172	$\mathbf{0.743 \pm 0.030}$	$\mathbf{0.295 \pm 0.029}$	$\mathbf{0.782 \pm 0.009}$
	DPP	$\mathbf{0.1797 \pm 0.0001}$	0.999 ± 0.000	0.500 ± 0.000	0.999 ± 0.000
	CS-DPP	$\mathbf{0.1797 \pm 0.0001}$	$\mathbf{0.724 \pm 0.002}$	$\mathbf{0.231 \pm 0.016}$	$\mathbf{0.796 \pm 0.002}$
yeast	O-CS	0.3077 ± 0.0021	0.885 ± 0.018	0.490 ± 0.002	0.926 ± 0.014
	O-RAND	0.4162 ± 0.0096	0.596 ± 0.008	0.376 ± 0.018	0.702 ± 0.014
	DPP	$\mathbf{0.2314 \pm 0.0014}$	0.463 ± 0.005	0.340 ± 0.003	0.597 ± 0.005
	CS-DPP	$\mathbf{0.2307 \pm 0.0011}$	$\mathbf{0.433 \pm 0.003}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.541 \pm 0.003}$

Table 15: CS-DPP vs Others, M = 10% of K

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	O-CS	0.1610 ± 0.0006	0.953 ± 0.003	0.497 ± 0.001	0.971 ± 0.002
	O-RAND	0.4042 ± 0.0052	0.750 ± 0.004	0.397 ± 0.006	0.858 ± 0.004
	DPP	$\mathbf{0.1453 \pm 0.0001}$	0.654 ± 0.002	0.399 ± 0.001	0.787 ± 0.001
	CS-DPP	$\mathbf{0.1454 \pm 0.0002}$	$\mathbf{0.603 \pm 0.001}$	$\mathbf{0.144 \pm 0.002}$	$\mathbf{0.748 \pm 0.001}$
Corel5k	O-CS	0.0117 ± 0.0000	0.926 ± 0.002	0.470 ± 0.001	0.949 ± 0.001
	O-RAND	0.3734 ± 0.0044	0.980 ± 0.001	0.393 ± 0.013	0.990 ± 0.000
	DPP	$\mathbf{0.0100 \pm 0.0000}$	0.918 ± 0.001	0.470 ± 0.000	0.943 ± 0.000
	CS-DPP	$\mathbf{0.0100 \pm 0.0000}$	$\mathbf{0.850 \pm 0.001}$	$\mathbf{0.237 \pm 0.001}$	$\mathbf{0.910 \pm 0.000}$
emotions	O-CS	$\mathbf{0.3338 \pm 0.0073}$	0.900 ± 0.014	0.508 ± 0.006	0.924 ± 0.009
	O-RAND	0.3847 ± 0.0099	0.621 ± 0.033	0.363 ± 0.020	0.683 ± 0.021
	DPP	$\mathbf{0.3335 \pm 0.0042}$	$\mathbf{0.428 \pm 0.003}$	0.223 ± 0.009	$\mathbf{0.558 \pm 0.004}$
	CS-DPP	$\mathbf{0.3301 \pm 0.0012}$	0.450 ± 0.007	$\mathbf{0.133 \pm 0.023}$	$\mathbf{0.560 \pm 0.009}$
enron	O-CS	0.0739 ± 0.0006	0.885 ± 0.010	0.463 ± 0.004	0.927 ± 0.009
	O-RAND	0.3907 ± 0.0090	0.867 ± 0.007	0.320 ± 0.015	0.923 ± 0.004
	DPP	$\mathbf{0.0563 \pm 0.0001}$	0.552 ± 0.003	0.304 ± 0.001	0.646 ± 0.002
	CS-DPP	$\mathbf{0.0565 \pm 0.0001}$	$\mathbf{0.528 \pm 0.002}$	$\mathbf{0.132 \pm 0.001}$	$\mathbf{0.638 \pm 0.001}$
mediamill	O-CS	0.0485 ± 0.0011	0.821 ± 0.025	0.454 ± 0.009	0.868 ± 0.014
	O-RAND	0.3737 ± 0.0070	0.899 ± 0.007	0.391 ± 0.020	0.950 ± 0.003
	DPP	$\mathbf{0.0308 \pm 0.0000}$	0.474 ± 0.000	0.307 ± 0.000	0.594 ± 0.000
	CS-DPP	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.460 \pm 0.000}$	$\mathbf{0.072 \pm 0.002}$	$\mathbf{0.582 \pm 0.000}$
medical	O-CS	0.0272 ± 0.0006	0.819 ± 0.016	0.397 ± 0.004	0.840 ± 0.017
	O-RAND	0.3674 ± 0.0093	0.924 ± 0.005	0.301 ± 0.019	0.959 ± 0.003
	DPP	$\mathbf{0.0204 \pm 0.0002}$	0.602 ± 0.008	0.311 ± 0.005	0.628 ± 0.008
	CS-DPP	$\mathbf{0.0204 \pm 0.0002}$	$\mathbf{0.508 \pm 0.006}$	$\mathbf{0.096 \pm 0.003}$	$\mathbf{0.549 \pm 0.007}$
nuswide	O-CS	0.0239 ± 0.0004	0.746 ± 0.005	0.600 ± 0.003	0.741 ± 0.003
	O-RAND	0.3707 ± 0.0107	0.956 ± 0.002	0.532 ± 0.013	0.973 ± 0.002
	DPP	$\mathbf{0.0201 \pm 0.0000}$	0.673 ± 0.000	0.580 ± 0.000	0.691 ± 0.000
	CS-DPP	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.358 \pm 0.001}$	$\mathbf{0.675 \pm 0.000}$
scene	O-CS	0.2168 ± 0.0047	0.920 ± 0.010	0.491 ± 0.003	0.902 ± 0.009
	O-RAND	0.3711 ± 0.0172	$\mathbf{0.743 \pm 0.030}$	$\mathbf{0.295 \pm 0.029}$	$\mathbf{0.782 \pm 0.009}$
	DPP	$\mathbf{0.1797 \pm 0.0001}$	0.999 ± 0.000	0.500 ± 0.000	0.999 ± 0.000
	CS-DPP	$\mathbf{0.1797 \pm 0.0001}$	$\mathbf{0.724 \pm 0.002}$	$\mathbf{0.231 \pm 0.016}$	$\mathbf{0.796 \pm 0.002}$
yeast	O-CS	0.3077 ± 0.0021	0.885 ± 0.018	0.490 ± 0.002	0.926 ± 0.014
	O-RAND	0.4162 ± 0.0096	0.596 ± 0.008	0.376 ± 0.018	0.702 ± 0.014
	DPP	$\mathbf{0.2314 \pm 0.0014}$	0.463 ± 0.005	0.340 ± 0.003	0.597 ± 0.005
	CS-DPP	$\mathbf{0.2307 \pm 0.0011}$	$\mathbf{0.433 \pm 0.003}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.541 \pm 0.003}$

Table 16: CS-DPP vs Others, M = 25% of K

Dataset	Alg.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	O-CS	0.1610 ± 0.0006	0.953 ± 0.003	0.497 ± 0.001	0.971 ± 0.002
	O-RAND	0.4042 ± 0.0052	0.750 ± 0.004	0.397 ± 0.006	0.858 ± 0.004
	DPP	$\mathbf{0.1453 \pm 0.0001}$	0.654 ± 0.002	0.399 ± 0.001	0.787 ± 0.001
	CS-DPP	$\mathbf{0.1454 \pm 0.0002}$	$\mathbf{0.603 \pm 0.001}$	$\mathbf{0.144 \pm 0.002}$	$\mathbf{0.748 \pm 0.001}$
Corel5k	O-CS	0.0117 ± 0.0000	0.926 ± 0.002	0.470 ± 0.001	0.949 ± 0.001
	O-RAND	0.3734 ± 0.0044	0.980 ± 0.001	0.393 ± 0.013	0.990 ± 0.000
	DPP	$\mathbf{0.0100 \pm 0.0000}$	0.918 ± 0.001	0.470 ± 0.000	0.943 ± 0.000
	CS-DPP	$\mathbf{0.0100 \pm 0.0000}$	$\mathbf{0.850 \pm 0.001}$	$\mathbf{0.237 \pm 0.001}$	$\mathbf{0.910 \pm 0.000}$
emotions	O-CS	$\mathbf{0.3338 \pm 0.0073}$	0.900 ± 0.014	0.508 ± 0.006	0.924 ± 0.009
	O-RAND	0.3847 ± 0.0099	0.621 ± 0.033	0.363 ± 0.020	0.683 ± 0.021
	DPP	$\mathbf{0.3335 \pm 0.0042}$	$\mathbf{0.428 \pm 0.003}$	0.223 ± 0.009	$\mathbf{0.558 \pm 0.004}$
	CS-DPP	$\mathbf{0.3301 \pm 0.0012}$	0.450 ± 0.007	$\mathbf{0.133 \pm 0.023}$	$\mathbf{0.560 \pm 0.009}$
enron	O-CS	0.0739 ± 0.0006	0.885 ± 0.010	0.463 ± 0.004	0.927 ± 0.009
	O-RAND	0.3907 ± 0.0090	0.867 ± 0.007	0.320 ± 0.015	0.923 ± 0.004
	DPP	$\mathbf{0.0563 \pm 0.0001}$	0.552 ± 0.003	0.304 ± 0.001	0.646 ± 0.002
	CS-DPP	$\mathbf{0.0565 \pm 0.0001}$	$\mathbf{0.528 \pm 0.002}$	$\mathbf{0.132 \pm 0.001}$	$\mathbf{0.638 \pm 0.001}$
mediamill	O-CS	0.0485 ± 0.0011	0.821 ± 0.025	0.454 ± 0.009	0.868 ± 0.014
	O-RAND	0.3737 ± 0.0070	0.899 ± 0.007	0.391 ± 0.020	0.950 ± 0.003
	DPP	$\mathbf{0.0308 \pm 0.0000}$	0.474 ± 0.000	0.307 ± 0.000	0.594 ± 0.000
	CS-DPP	$\mathbf{0.0308 \pm 0.0000}$	$\mathbf{0.460 \pm 0.000}$	$\mathbf{0.072 \pm 0.002}$	$\mathbf{0.582 \pm 0.000}$
medical	O-CS	0.0272 ± 0.0006	0.819 ± 0.016	0.397 ± 0.004	0.840 ± 0.017
	O-RAND	0.3674 ± 0.0093	0.924 ± 0.005	0.301 ± 0.019	0.959 ± 0.003
	DPP	$\mathbf{0.0204 \pm 0.0002}$	0.602 ± 0.008	0.311 ± 0.005	0.628 ± 0.008
	CS-DPP	$\mathbf{0.0204 \pm 0.0002}$	$\mathbf{0.508 \pm 0.006}$	$\mathbf{0.096 \pm 0.003}$	$\mathbf{0.549 \pm 0.007}$
nuswide	O-CS	0.0239 ± 0.0004	0.746 ± 0.005	0.600 ± 0.003	0.741 ± 0.003
	O-RAND	0.3707 ± 0.0107	0.956 ± 0.002	0.532 ± 0.013	0.973 ± 0.002
	DPP	$\mathbf{0.0201 \pm 0.0000}$	0.673 ± 0.000	0.580 ± 0.000	0.691 ± 0.000
	CS-DPP	$\mathbf{0.0201 \pm 0.0000}$	$\mathbf{0.648 \pm 0.000}$	$\mathbf{0.358 \pm 0.001}$	$\mathbf{0.675 \pm 0.000}$
scene	O-CS	0.2168 ± 0.0047	0.920 ± 0.010	0.491 ± 0.003	0.902 ± 0.009
	O-RAND	0.3711 ± 0.0172	$\mathbf{0.743 \pm 0.030}$	$\mathbf{0.295 \pm 0.029}$	$\mathbf{0.782 \pm 0.009}$
	DPP	$\mathbf{0.1797 \pm 0.0001}$	0.999 ± 0.000	0.500 ± 0.000	0.999 ± 0.000
	CS-DPP	$\mathbf{0.1797 \pm 0.0001}$	$\mathbf{0.724 \pm 0.002}$	$\mathbf{0.231 \pm 0.016}$	$\mathbf{0.796 \pm 0.002}$
yeast	O-CS	0.3077 ± 0.0021	0.885 ± 0.018	0.490 ± 0.002	0.926 ± 0.014
	O-RAND	0.4162 ± 0.0096	0.596 ± 0.008	0.376 ± 0.018	0.702 ± 0.014
	DPP	$\mathbf{0.2314 \pm 0.0014}$	0.463 ± 0.005	0.340 ± 0.003	0.597 ± 0.005
	CS-DPP	$\mathbf{0.2307 \pm 0.0011}$	$\mathbf{0.433 \pm 0.003}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.541 \pm 0.003}$

Table 17: CS-DPP vs Others, M = 50% of K

Dataset	Reduced Dim.	Hamm. loss	F1 loss	Acc. loss	Norm. rank loss
CAL500	$M = 10\%$ of K	0.1458 ± 0.00019	0.5914 ± 0.00108	0.1247 ± 0.00224	0.7388 ± 0.00105
	$M = 25\%$ of K	0.1489 ± 0.00012	0.5956 ± 0.00110	0.1321 ± 0.00210	0.7428 ± 0.00131
	$M = 50\%$ of K	0.1503 ± 0.00009	0.5949 ± 0.00101	0.1371 ± 0.00222	0.7426 ± 0.00127
Corel5k	$M = 10\%$ of K	0.0102 ± 0.00000	0.8379 ± 0.00175	0.2382 ± 0.00193	0.9026 ± 0.00138
	$M = 25\%$ of K	0.0103 ± 0.00000	0.8248 ± 0.00174	0.2102 ± 0.00102	0.8936 ± 0.00161
	$M = 50\%$ of K	0.0102 ± 0.00000	0.8186 ± 0.00138	0.1991 ± 0.00123	0.8914 ± 0.00152
emotions	$M = 10\%$ of K	0.3421 ± 0.00167	0.4511 ± 0.00525	0.0745 ± 0.08548	0.5881 ± 0.02669
	$M = 25\%$ of K	0.2743 ± 0.00000	0.3964 ± 0.00476	0.0235 ± 0.00597	0.5068 ± 0.00653
	$M = 50\%$ of K	0.2324 ± 0.00000	0.3809 ± 0.00450	0.0237 ± 0.00244	0.4858 ± 0.00463
enron	$M = 10\%$ of K	0.0562 ± 0.00020	0.5421 ± 0.00335	0.1432 ± 0.00333	0.6573 ± 0.00360
	$M = 25\%$ of K	0.0600 ± 0.00011	0.5392 ± 0.00291	0.1364 ± 0.00244	0.6561 ± 0.00332
	$M = 50\%$ of K	0.0632 ± 0.00009	0.5428 ± 0.00293	0.1305 ± 0.00216	0.6627 ± 0.00316
mediamill	$M = 10\%$ of K	0.0309 ± 0.00001	0.4564 ± 0.00037	0.0617 ± 0.00108	0.5790 ± 0.00049
	$M = 25\%$ of K	0.0308 ± 0.00000	0.4535 ± 0.00030	0.0597 ± 0.00062	0.5756 ± 0.00022
	$M = 50\%$ of K	0.0308 ± 0.00000	0.4534 ± 0.00027	0.0565 ± 0.00026	0.5755 ± 0.00027
medical	$M = 10\%$ of K	0.0202 ± 0.00014	0.5246 ± 0.01649	0.0949 ± 0.00836	0.5764 ± 0.02145
	$M = 25\%$ of K	0.0150 ± 0.00010	0.3416 ± 0.00815	0.0337 ± 0.00431	0.4026 ± 0.00942
	$M = 50\%$ of K	0.0130 ± 0.00003	0.2783 ± 0.00618	0.0201 ± 0.00276	0.3361 ± 0.00979
nuswide	$M = 10\%$ of K	0.0201 ± 0.00000	0.6338 ± 0.00064	0.3394 ± 0.00294	0.6627 ± 0.00063
	$M = 25\%$ of K	0.0201 ± 0.00000	0.6305 ± 0.00057	0.3124 ± 0.00189	0.6600 ± 0.00045
	$M = 50\%$ of K	0.0201 ± 0.00000	0.6290 ± 0.00035	0.2945 ± 0.00076	0.6588 ± 0.00042
scene	$M = 10\%$ of K	0.2837 ± 0.00000	0.7433 ± 0.00184	0.1732 ± 0.00593	0.7917 ± 0.00111
	$M = 25\%$ of K	0.1873 ± 0.00000	0.6387 ± 0.00199	0.2265 ± 0.02788	0.6882 ± 0.00196
	$M = 50\%$ of K	0.1723 ± 0.00005	0.5571 ± 0.00206	0.1708 ± 0.02179	0.6138 ± 0.00221
yeast	$M = 10\%$ of K	0.2296 ± 0.00010	0.4518 ± 0.00920	0.0064 ± 0.00081	0.5448 ± 0.02253
	$M = 25\%$ of K	0.2162 ± 0.00009	0.3841 ± 0.00200	0.0170 ± 0.00242	0.4971 ± 0.00379
	$M = 50\%$ of K	0.2092 ± 0.00001	0.3784 ± 0.00107	0.0232 ± 0.00158	0.4901 ± 0.00124

Table 18: Results of CS-DPP on 50 random label orders