Cost-sensitive Encoding for Label Space Dimension Reduction Algorithms on Multi-label Classification

Kuo-Hsuan Lo
Graduate Institute of Networking and Multimedia
National Taiwan University
r03944030@ntu.edu.tw

Hsuan-Tien Lin
Graduate Institute of Networking and Multimedia
National Taiwan University
htlin@csie.ntu.edu.tw

***Abstract*—Multi-label classification (MLC) extends multi-class classification by tagging each instance as multiple classes simultaneously. Different real-world MLC applications often demand different evaluation criteria (costs), which calls for cost-sensitive MLC (CSMLC) algorithms that can easily take the criterion of interest into account. Nevertheless, existing CSMLC algorithms generally suffer from high computational complexity. In this work, we study a family of MLC algorithms, called label-space dimension reduction (LSDR), which is known to be efficient for MLC but not cost-sensitive. We propose a general framework that directs LSDR algorithms to embed the cost information instead of the label information. The framework makes existing LSDR algorithms cost-sensitive while keeping their efficiency. Extensive experiments justify that the proposed framework is superior to both existing LSDR algorithms and CSMLC algorithms across different evaluation criteria.**

***Index Terms*—multi-label classification, cost-sensitive, cost-encoding, label space dimension reduction, label space dimension expansion**

INTRODUCTION

The multi-label classification (MLC) problem is an extension of the multi-class classification problem. The latter aims to classify an instance into one out of two or more classes, and the former aims to classify each instance into multiple target classes simultaneously. The multiple target classes that the MLC classifiers output are often represented as a binary vector, called the label vector, that indicates the existence of the class. Existing MLC algorithms could be categorized as algorithm adaptation or problem transformation [1]. Algorithm adaptation approaches extend a specific algorithm in order to tackle the MLC problem, and problem transformation approaches reduce the MLC problem to other machine learning tasks and solve them with tools in those tasks. Problem transformation approaches enjoy the benefit of being able to utilize the mature tools that have been developed, and will be the focus of this work.

Among those problem transformation approaches, classifier chain (CC) algorithms processes labels one by one and uses previous processed labels as additional features. Label space dimension expansion (LSDE) algorithms encode the original label vectors into a higher dimensional vector to increase the robustness of the original labels against the error made by the learning process. And label space dimension reduction (LSDR) algorithms transform the original label space to lower dimensional space and achieve better efficiency by learning and predicting on the lower-dimensional space.

The very first problem transformation we could consider is binary relevance (BR). It reduces the MLC problem into K different independent binary classification tasks and trains one single classifier for each label. Binary relevance has been criticized for ignoring the label dependency information [**?**], which could probably be further exploited.

Since we would like to handle the extensive information, storing the latent relation between labels in the expanded label space might be reasonable. Such transformation algorithms are known as label space dimension expansion (LSDE). In fact, in order to fully take advantage of the correlation between labels, an extreme algorithm in [1] called label powerset is proposed. Label powerset (LP), which considers each unique label permutation as one of the classes of a new single-label classification task, transforms MLC problem to several disjoint multi-class classification problems by treating each label permutation as a unique multi-class label.

However, such algorithm exhaustively listing the classes is criticized [**?**] for its number of labels grow exponentially, and the majority of labels are only associated with very few instances. An alternative called Random k-labelsets (RAkEL) [**?**] based on LP, takes only a subset of labels once by random selection and performs a majority vote in the end to reduce the growth of total classes of LP. RAkEL solves the computational issue by using a relatively small multi-class 2^k based on k elements randomly selected from the powerset of K labels.

In addition to LP and RAkEL, LSDE algorithm like ML-ECC [9] even takes advantage of the existing off-the-shelf error correcting code(ECC) developed for channel coding to improve the accuracy of MLC. ML-ECC treats the label space vectors as a block of binary code and encodes them as ECC does, trains the base learner in the expanded space and feed the decoder of the chosen ECC with predicted vectors. The framework shows that using ECC to encode the label space could improve the performance of RAkEL and BR on Hamming loss and 0/1 loss.

On the other hand, despite that LSDE algorithms resolve the problem of neglecting latent information, the number of learning task has been increased. Hence, more algorithms based on the idea of reducing the label space dimensions and finding latent dependency at the same time are proposed. Label space

dimension reduction (LSDR) algorithms such as PLST [**?**], CPLST [**?**] and FaIE [**?**] first compress the original problem to a relatively small number of learning tasks and predict the compressed label vectors, then decompress them back to their original space. Such approaches [**?**], [**?**], [**?**] are effective because of the appropriate use of joint information within labels. In [**?**], principal label space transformation (PLST) uses singular value decomposition (SVD) to captures the correlation between labels, and the reduction transformation and its recovering operation are two linear functions. By doing so, the number of learning tasks decrease, and minimizing the squared loss of the recovered label space benefits the prediction accuracy on Hamming loss. In [**?**], conditional principal label space transformation (CPLST) combines the concepts of PLST and CCA. By adding the feature space into optimization, such feature-aware algorithm improves the performance by minimizing the upper bound of Hamming loss. The above-mentioned algorithms both are using explicitly defined reduction function. In [**?**], another feature-aware algorithm called feature-aware implicit label space encoding (FaIE) shows the feasibility of making no assumption concerning the reduction function.

Another category of problem transformation algorithms called classifier chain (CC) divides the MLC problem into multiple binary classification problems based on the number of labels. Each single-label classifier in the chain utilize the prediction of previous classifiers or ground truth as additional features and predicts its corresponding label.

COST-SENSITIVE ENCODING

On the other hand, the demands of various evaluation criteria on this algorithms grow in order to meet the real-world applications. Such different types of applications require different evaluation criteria, and the previous MLC problem could not meet the needs of a specifically defined loss function. Hence, these upcoming needs call for a new problem setting called cost-sensitive multi-label classification (CSMLC) problem.

In the MLC problem, we denote a feature vector by $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$, its corresponding label vector $\mathbf{y} \in \mathcal{Y} \subseteq \{0,1\}^K$, a given cost function $\mathbf{C}$ and a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, which contains N i.i.d examples drawn from an unknown distribution $\mathcal{P}$. For a prediction as $\tilde{\mathbf{y}} = h(\mathbf{x})$, in order to evaluate the difference between $h(\mathbf{x})$ and $\mathbf{y}$, the goal is to use $\mathcal{D}$ to find a classifier $h : \mathbb{R}^d \to \{0,1\}^K$ in training stage and hope that $h(\mathbf{x})$ predicts $\mathbf{y}$ of an unseen $\mathbf{x}$ in predicting stage. Such $h(\mathbf{x})$ should minimizes $E_{(\mathbf{x},\mathbf{y})\sim P}[\mathbf{C}(\mathbf{y}, h(\mathbf{x}))]$, which means minimizing $\mathbf{C}(\mathbf{y}, h(\mathbf{x}))$ when $(\mathbf{x},\mathbf{y})$ is drawn from $\mathcal{P}$. In the follow-up of our discussion, we use loss and score respectively for the cost function defined in Table I that should be minimized and maximized for the sake of brevity.

And in the CSMLC problem, we provide an extra cost function as a parameter for the algorithms to quantify the loss between the prediction and the truth. By doing so, the goal of the cost-sensitive multi-label classification (CSMLC) problem becomes using $\mathcal{D}$ and $\mathbf{C}$ to find a classifier $h : \mathbb{R}^d \to \{0,1\}^K$ in training stage and hope that $h(\mathbf{x})$ predicts $\mathbf{y}$ of an unseen $\mathbf{x}$ in predicting stage. Such $h(\mathbf{x})$ should minimize $E_{(\mathbf{x},\mathbf{y})\sim P}[C(\mathbf{y}, h(\mathbf{x}))]$. Furthermore, a cost function $\mathbf{C} : (\{0,1\}^K, \{0,1\}^K) \to \mathbb{R}$ could be further viewed as an implicit cost matrix of size $2^K \times 2^K$ with elements $\in \mathbb{R}$, in which every element stands for the cost. Notice that such representation is able to describe all possible example based cost functions, since the cost between every pair of two label permutation has been exhaustively listed in the cost matrix. Hence, algorithms that could properly utilize the function $\mathbf{C}$ properly during the training are considered to have cost generality.

$\mathbf{Hamming}(\mathbf{y}, \tilde{\mathbf{y}})$	$\sum_{k=1}^{K} [\![\mathbf{y}[k] \neq \tilde{\mathbf{y}}[k]]\!]$
$\mathbf{Rank}(\mathbf{y}, \tilde{\mathbf{y}})$	$\sum_{\mathbf{y}[i]>\mathbf{y}[j]} ([\![\tilde{\mathbf{y}}[i] < \tilde{\mathbf{y}}[j]]\!] + \frac{1}{2}[\![\tilde{\mathbf{y}}[i] = \tilde{\mathbf{y}}[j]]\!])$
$\mathbf{Accuracy}(\mathbf{y}, \tilde{\mathbf{y}})$	$\frac{\|\mathbf{y} \cap \tilde{\mathbf{y}}\|_1}{\|\mathbf{y} \cup \tilde{\mathbf{y}}\|_1}$
$\mathbf{Composite}(\mathbf{y}, \tilde{\mathbf{y}}) =$	$\mathbf{F1}(\mathbf{y}, \tilde{\mathbf{y}}) - 5 \cdot \mathbf{Hamming}(\mathbf{y}, \tilde{\mathbf{y}})$

TABLE I: Cost Function Definition

Among those algorithms solving the CSMLC problem, the original algorithms in the problem transformation category have been extended to fit the problem setting of the CSMLC as well, and each of them handles the cost information in different ways, which lead to their different cost generalities. For those methods utilizing label powerset to reduce the multi-label classification problem, in [7], the author proposes cost-sensitive RAkEL (CS-RAkEL) based on RAkEL optimizing on a certain cost function called weighted hamming loss, which transforms the cost of each label in a labelset to the total cost of the labelset. However, such cost function still treats each label independently and could not handle other types of cost functions. In [6], PRAkEL extends RAkEL by transforming the results of the cost function into the cost of each class generated in each labelset, which implicitly utilizes the general cost matrix. It transforms the RAkEL into a cost sensitive version and proposes a strategy for defining reference label vectors when dividing the label set into several subsets. Interesting, such reference rule could be applied seamlessly in our proposed framework. Another chain-based algorithm in [11] called Condensed filter tree (CFT) reduces the CSMLC problem into a cost-sensitive multi-class classification with the filter tree algorithm [**?**] via label powerset transformation.

In fact, Algorithms such as PRAkEL [6] and CFT [11] are able to deal with the general example-based cost function. PRAkEL provides an efficient and competitive when comparing itself to CFT. Despite that PRAkEL is able to deal with a general example-based cost function in time complexity $\propto K$, the algorithm reduces the problem to cost-sensitive multi-class classification, in which the base learner is restricted to have cost-sensitivity.

Given the fact the none of the existing CSMLC algorithms further take advantage of the previous work done in the paradigm of LSDR, we proposed an algorithm that could systematically make the LSDR algorithms take the general cost

matrix into account directly and tackle with general example-based cost function without more adaptation for each criterion and remain the efficiency originally brought by LSDR.

Among these MLC/CSMLC algorithms, another interesting aspect of how they digest the label should be mentioned in addition. Chain-based algorithms digest the label one-by-one thus suffer from the ordering problem. This progressive process of solving the problem could be further extended as the reference rule for the next round of learning. It could improve the performance by providing the cost function with prediction instead of ground truth. On the other hand, ensemble algorithm could utilize a certain amount of labels at once, avoiding the ordering problem. Algorithms such as ECC,EPCC [**?**], PRAKEL and CFT show the improvement brought by combining these two methodologies in both MLC and CSMLC problem settings. Before introducing our framework, we conclude these algorithms in Table II.

Algorithms category	none	some costs	general example-based costs
Chain-based	CC/ECC	PCC/EPCC reference rule	CFT instance weight
LSDE	LP,RAkEL ML-ECC	CS-RAkEL class weight	PRAkEL class weight
LSDR	PLST CPLST FaIE	none	none

TABLE II: Cost Function Utilization Capability of Existing Multi-label Classification Algorithms

PROPOSED METHOD

Our framework is constructed as followed. Given a dimension expansion codec composed of its encoder $enc(.) : \{0,1\}^K \to \{0,1\}^M$ and decoder $dec(.) : \{0,1\}^M \to \{0,1\}^K$. We use $enc(.)$ to expand the original label set $\mathbf{y}_n \in \{0,1\}^K$ to a codeword $\mathbf{b}_n \in \{0,1\}^M$. Then we apply our cost information embedding algorithm $\Phi : \{0,1\}^M \to \mathbb{R}^M$ on $\mathbf{b}_n$ to embed the cost information into the cost vector $\mathbf{c}$. Then a label space dimensional reduction algorithm $Re(.) : \mathbb{R}^M \to \mathbb{R}^{M_r}$ could be applied on $\mathbf{b}_n$ to reduce the tasks of learning. During the predicting stage, given a testing instance $(\mathbf{x}, \mathbf{y})$, we use h to predict the reduced vector $\tilde{\mathbf{z}}$ and then transform it back to $\tilde{\mathbf{y}}$ by sequentially applying the recovering function $Re^{-1}(.) : \mathbb{R}^{M_r} \to \mathbb{R}^M$, cost information decoding function $\Psi : \mathbb{R}^M \to \{0,1\}^M$ and the dimension expansion decoding function $dec(.) : \{0,1\}^M \to \{0,1\}^K$

Notice that cost information decoding function Ψ is actually a soft-to-hard bit function determined by the encoder $enc(.)$ we choose. For lazy codec $codec_{lazy}(.)$, Ψ is a relatively simple threshold function determining whether it is positive of negative prediction on every bit, denoted by $\Psi_{+-}(.)$.

And for $codec_{LP}(.)$ and $codec_{RAkEL}(.)$, Ψ only choose the most confident dimension and mark it as one, and leaves all the others as zeros, denoted by $\Psi_{max}(.)$. The steps of the framework are shown as follows :

- Parameter :
  1) Cost function $\mathbf{C}$
  2) Dimension expansion codec $enc(.)$ and $dec(.)$
  3) Cost information embedding function $\Phi(.)$, and its inverse function $\Psi(.)$.
  4) Dimension reduction algorithm $Re(.)$ and $Re^{-1}(.)$
  5) Regression based multi-label learner A_b
- Training : Given $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$
  1) Dimension expansion by $\mathbf{b}_n = enc(\mathbf{y}_n)$
  2) Cost information embedding by $\mathbf{c_n} = \Phi(\mathbf{b}_n, \mathbf{C})$;
  3) Applying reduction function by $\mathbf{z}_n = Re(\mathbf{c_n})$;
  4) Return a predictor $h = A_b(\{(\mathbf{x_n}, \mathbf{z}_n)\}_{n=1}^N)$.
- Prediction : Given any $\mathbf{x}$ drawn from $\mathcal{P}$
  1) Predicting a codeword $\tilde{\mathbf{z}} = h(\mathbf{x})$
  2) Applying recovering function by $\tilde{\mathbf{c}} = Re^{-1}(\tilde{\mathbf{z}})$
  3) Decoding the cost information by $\tilde{\mathbf{b}} = \Psi(\tilde{\mathbf{c}})$
  4) Decoding the expanded vector by $\tilde{\mathbf{y}} = dec(\tilde{\mathbf{b}})$

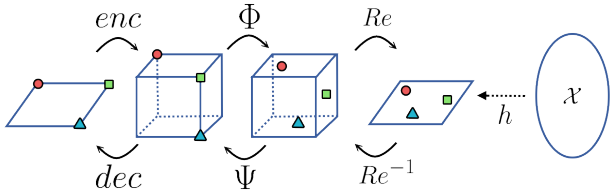


Fig. 1: Framework structure

Label Space Expansion Codec

In this section, we first proposed three codecs for our framework to encode the label vectors into expanded vectors prepared for the upcoming cost information embedding. We use the same subscript to indicate a codec and its encoder and decoder.

- Lazy codec, denoted as $codec_{lazy}$, means that it does not do any encoding and decoding but only maps $\{0,1\}^K \to \{-1,+1\}^K$.
- Extreme codec, denoted as $codec_{LP}$, uses the idea of label powerset. Such exhaustive algorithm uses dimension up to $M = min(N, 2^K)$. However, label powerset, either using all possible label permutations 2^K or unique permutation up to a number of N to encode, are infamous for the computational issue on such amount of dimensions.
- In terms of $codec_{RAkEL}$, its enc_{RAkEL} randomly partitions $\mathcal{Y} = \mathcal{L}_K = \{1,2,3,...,K\}$ into $G = \lceil \frac{K}{k} \rceil$ disjoint labelsets $\mathcal{S}_g = \{s_{g1}, ..., s_{gk}\}, g = 1, ..., G$, encodes each $\mathcal{S}$ like enc_{LP} does and concatenates them into one vector. While decoding, dec_{RAkEL} first splits them into g vectors with size 1×2^k, decodes each of them back to $\mathcal{S}_g = \{s_{g1}, ..., s_{gk}\}, g = 1, ..., G$ and aggregates them back to multi-label representation. The representation of each labelset is denoted as $\mathbf{y}[\mathcal{S}] \in \{0,1\}^k$.
  The above iteration would be done with c times including the learning and predicting process. In each iteration, a voting back is done. After c iterations, it decides the final prediction on each label by a majority voting.

Cost Information Embedding

We propose our cost information embedding algorithm for the above-mentioned codecs, and further discuss the shortcoming of the off-the-shelf error correcting code. We only discuss $codec_{RAkEL}$ later because $codec_{lazy}$ and $codec_{LP}$ could be regarded as the special cases as $k = 1, K$. For for $codec_{RAkEL}$, we sum from $j = 1$ until 2^k since we only consider the label permutations of k-labelsets. In fact, for $codec_{RAkEL}$, let $\hat{\mathbf{y}}_j[\mathcal{S}_g^m]$ denote one of all the possible permutation of $\mathcal{S}_g^m = \{s_{g1}^m, ..., s_{gk}^m\} \in \{0,1\}^k$, the cost information embedding algorithm in the labelset $\mathcal{S}_g^m$ should be

$$\begin{aligned}\mathbf{c}'_{n,g} &= \Theta\Big(\mathbf{c}_{n,g}\Big) = \Theta\Big(\Phi(\mathbf{b}_n[\mathcal{S}_g^m], \mathbf{C})\Big) \\ &= \Theta\Bigg(\sum_{j=1}^{2^k} \mathbf{C}\Big(\mathbf{y}_n[\mathcal{S}_g^m] \bigcup \tilde{\mathbf{y}}_n[\bar{\mathcal{S}}_g^m], \hat{\mathbf{y}}_j[\mathcal{S}_g^m] \bigcup \tilde{\mathbf{y}}_n[\bar{\mathcal{S}}_g^m]\Big) \\ &\quad \cdot \hat{\mathbf{b}}_j[\mathcal{S}_g^m]\Bigg)\end{aligned} \tag{1}$$

The reason we perform a subtraction

$$\Theta(\mathbf{c}_{n,g}) = \mathbf{c}_{n,g} - \mathbf{min}(\mathbf{c}_{n,g})$$

is to eliminate the shift on $\mathbf{c}_{n,g}$ brought by $\mathbf{y}_i[\mathcal{S}_g^m]$ and $\hat{\mathbf{y}}_j[\mathcal{S}_g^m]$ both referencing $\tilde{\mathbf{y}}_n[\bar{\mathcal{S}}_g^m]$.

- Reference Rule : While embedding the cost information, the algorithm $enc_{RAkEL}(.)$ splits the original problem into $g = 1, ..., G$ sub-problems, and such process iterates M times. Let m denote the index of current iteration and g denote the index of the current sub-problem, we only consider k-labelset $\mathbf{y}[\mathcal{S}_g^m]$ in each iteration. The rest of labelsets, denoted as $\mathbf{y}[\bar{\mathcal{S}}_g^m]$, are assumed to be perfectly predicted. However, since we could never reach the perfect prediction $\tilde{\mathbf{y}}[\bar{\mathcal{S}}_g^m] = \mathbf{y}[\bar{\mathcal{S}}_g^m]$, such over-optimistic assumption would make the cost information embedding process embed the unrealistic cost because of referencing $\mathbf{y}[\bar{\mathcal{S}}_g^m]$. In this paper, we use predicted reference label vector $\tilde{\mathbf{y}}[\bar{\mathcal{S}}_g^m]$ in the each iteration as the default setting of our experiments, and we only use perfect prediction on the first iteration $m = 1$. And we use $\bigcup$ to denote the operation of combining two disjoint sets of labels.
- For Off-the-shelf ECC Codec : Although previous study [9] has shown that the performance enhancement obtained from encoding mechanism such as Hamming code and BCH code on the MLC problem, the same coding techniques might not meet our framework. We discover the fact that if the cost function is a *reflective* function, such as weighted Hamming loss and rank loss, the aggregated costs, regarded as the confidence score, will be summed as zero by Theorem 1.

Theorem 1. *Let C be a cost function as well as a score function. A cost function $\mathbf{C}$ is called reflective if and only if* $\mathbf{C}(\mathbf{y}_i, \hat{\mathbf{y}}_j) - \frac{1}{2} \cdot \mathbf{C_{max}} = -\Big(\mathbf{C}(\mathbf{y}_i, Flip(\hat{\mathbf{y}}_j)) - \frac{1}{2} \cdot \mathbf{C_{max}}\Big)$.

If C is reflective, then the labels encoded with p-bits XOR operation from the data of K bits by $enc(.)$ in Equation 1 are always 0.

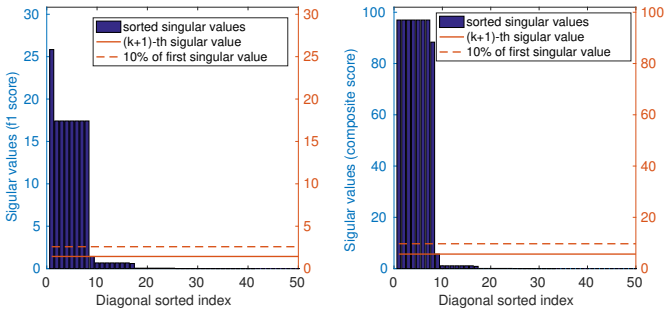
Sub-problem Dimension Reduction Trick

After the cost information is embedded in the encoded label space and forms the new codeword, with the dimension growing from K to $2^k \cdot \frac{K}{k}$, we are interested in reducing the number of learning tasks to our desired number M, which means performing a dimension reduction algorithm on the codeword of size $N \times (2^k \cdot \frac{K}{k})$ and reducing it to $N \times M$. If we consider the physical meaning of each dimension generated by the encoder enc_{RAkEL}, each axis represents a unique label permutation confidence level. After we compress the codeword to size $N \times M$ and perform training-predicting steps, we decode the predicted codeword with the recovering function of dimension reduction, then vote back to the label ballot box as RAkEL does. The A_R we choose to perform the dimension reduction and recovery on the codeword are done in the assembled encoded label space, it means that the dependency across subset $\mathbf{y}[S_i]$ and $\mathbf{y}[S_j]$ are potentially preserved. We could further exploit such mechanism and improve our framework by performing $\frac{K}{k}$ times of dimensions reduction in the encoded space of each subset from a number of dimensions 2^k to $\frac{M}{(\frac{K}{k})}$.

Feasibility of Cost Vector Space Dimension Reduction

As we perform dimension reduction on the cost matrix, we would like to further discuss the feasibility of dimension reduction from 2^k to k. As shown in Figure 3, although the cost matrix is a high dimensional data, the dependency of each adjacent dimensions are high as well, since each dimension in the cost matrix represents a label vector, and adjacent dimensions are generated by similar labels. The dependency within the cost matrix could be further regarded as information redundancy, which gives us a more promising result while performing dimension reduction on such cost matrix.

In Figure 2, we choose k=8 and one of the dimension reduction tools – singular value decomposition to demonstrate the behavior of the cost matrix obtained from cost functions. We do not need reference rule here since we would like to observe the worst case of the approximated rank, so applying referenced labels would lead to a lower rank. And for the same reason, we do not consider data distribution as well. We highlight the $(k+1)-th$ singular value, which would be discarded during dimension reduction, to show the information loss done by the dimension reduction in a sense. We also highlight the 10% of the first singular value, which is regarded as a "negligible threshold", so the approximated rank would be the number of sigular values higher than this threshold. Interestingly, we could find out that the singular values drop while indices are larger than k, meaning that the criteria we use are *approximately low-rank*, and thus such insights grant us the feasibility of dimension reduction from 2^k into merely k in each sub-problem.



(a) f1 score (b) composite score

Fig. 2: Singular values of each cost function of k-labelset when k=8

high dimensional space

$$\begin{pmatrix} \mathbf{C}(\mathbf{y}_1,\mathbf{y}_1) & \dots & \mathbf{C}(\mathbf{y}_1,\mathbf{y}_j) & \dots & \mathbf{C}(\mathbf{y}_1,\mathbf{y}_{2^K}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}(\mathbf{y}_i,\mathbf{y}_1) & \dots & \mathbf{C}(\mathbf{y}_i,\mathbf{y}_j) & \dots & \mathbf{C}(\mathbf{y}_i,\mathbf{y}_{2^K}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}(\mathbf{y}_{2^K},\mathbf{y}_1) & \dots & \mathbf{C}(\mathbf{y}_{2^K},\mathbf{y}_j) & \dots & \mathbf{C}(\mathbf{y}_{2^K},\mathbf{y}_{2^K}) \end{pmatrix}$$

high label dependency

Fig. 3: Feasibility of cost vector space dimension reduction

EXPERIMENTS

The experiments are conducted on seven benchmark datasets from Mulan [**?**]. These datasets are composed of diverse domains and different scales of label dimensions. The basic properties and statistics are listed in table III.

Datasets	labels	features	instances	distinct labels
emotions	6	72	593	27
scene	6	294	2407	15
yeast	14	103	2417	198
birds	19	260	645	133
medical	45	1449	978	94
enron	53	1001	1702	753
cal500	174	68	502	502

TABLE III: Datasets

For statistical significance, we randomly split the data into 75% for training, 25% for testing. The parameter selection is conducted using 3-fold CV on the training data. After choosing the best parameter, we again train a model with all the training data with the chosen parameter and use that model for testing. Such random split is performed 20 times.

Algorithms and the Parameters

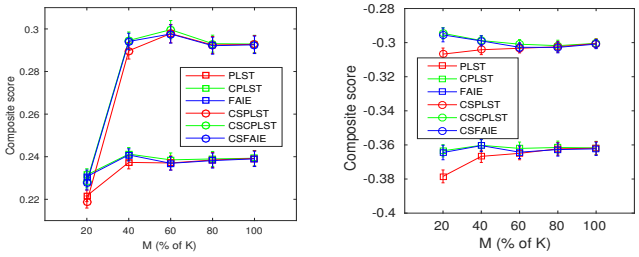
We first compare the results of existing LSDR algorithm [**?**], [**?**], [**?**] and the results of $codec_{RAkEL}$ to show the validity of the improvement for LSDR brought by CSEDR. In terms of reference rule, we use $\tilde{\mathbf{y}}[S_m^c]$ for both PRAkEL and CSEDR. In the second part of the experiments, we compare CSEDR with the state-of-the-art CSMLC algorithms CFT and PRAkEL. For both PRAkEL and CSEDR, the k of labelset is set to 8, and we slightly repeat the labels in order to make all $\lceil \frac{K}{k} \rceil$ subsets have the same cardinality, and the iteration number in both PRAkEL and CSEDR is set to 10. For CFT, the iteration number is restricted to 8. While comparing CSEDR with other algorithms, we only use $codec_{RAkEL}$ and CPLST as dimension reduction algorithm.

Base learner and the Parameter Selection

While comparing to CFT in the linear case, we use L2-regularized L2-loss Support Vector Classification in LIBLINEAR [**?**] for CFT and L2-regularized L2-loss Support Vector Regression in LIBLINEAR for CSEDR. In non-linear case, CFT and CSEDR both use Random Forest [**?**] implemented in MATLAB. While comparing to PRAkEL, we use the RED-OSSVR [**?**] implemented in the cost-sensitive with weighted instance extensions of LIBSVM [**?**] for PRAkEL and L2-regularized L2-loss Support Vector Regression in LIBLINEAR for CSEDR.

Comparison on Existing LSDR Algorithms

Because all of the cost-insensitive LSDR algorithms are optimizing Hamming loss, so we compare their composite score in Figure 4 with CSEDR. For all PLST, CPLST and FaIE, we could see that after applying CSEDR, their performance on different evaluating criteria has been improved in different dimension used during the reduction.



(a) enron (b) cal500

Fig. 4: Performance of LSDR algorithms and the their CSEDR versions on composite score (↑)

Comparison with State-of-the-art Algorithms

We only compare CSEDR with the algorithms capable of dealing with the general example-based cost function. EPCC indeed has commendable performance while comparing on F1 and Hamming. However, in [6] and [11], they both reported that under composite score, which is a linear combination of F1 score and Hamming loss, EPCC could not compete with CFT and PRAkEL because of lacking inference rule and using approximate rules instead. In order to demonstrate CSEDR's capability of optimizing general example-based cost function, we compare CSEDR with CFT and PRAkEL on composite score additionally. We show the one of the linear case results in Table IV, and two of the non-linear cases in Table V.

Tabel VI shows that CSEDR is competitive with PRAkEL under different criteria. Imaginably, both algorithms have the same ensemble design originated from RAkEL. And thus they share the same time complexity $\propto K$ as we discuss in Chapter 3.6. However, CSEDR still has two advantages over PRAkEL. First, though both algorithms associate k labels with one or more sub-problems, PRAkEL reduced each k-labelset problem into one cost-sensitive classification sub-problem of the number of classes 2^k, so each k-labelset sub-problem need to be trained at once. However, CSEDR preprocesses each cost vector of dimension 2^k into a vector of dimension K/k with dimension reduction, and those reduced soft-bit labels could be trained disjointly while still preserving cost information of one k-labelset. Second, CSEDR does not require its base learner to be cost-sensitive, making itself possess more freedom on choosing the base leaner.

	PRAkEL	CFT	CSEDR
scene	0.209±0.012	0.203±0.016	**0.207±0.016**
emotion	-0.511±0.041	-0.522±0.050	**-0.481±0.037**
yeast	-0.403±0.014	-0.412±0.017	**-0.397±0.015**
cal500	-0.302±0.009	-0.304±0.009	**-0.299±0.008**
birds	0.247±0.019	0.242±0.021	**0.267±0.019**
enron	0.329±0.012	**0.348±0.010**	0.289±0.011
medical	0.739±0.010	0.734±0.011	**0.745±0.012**

TABLE IV: Comparison on Composite score (↑), linear case

	Accu. score ↑		Comp. score ↑	
	CFT	CSEDR	CFT	CSEDR
scene	0.652±0.004	**0.757±0.005**	0.211±0.007	**0.353±0.007**
emotion	0.560±0.003	**0.589±0.004**	-0.364±0.020	**-0.339±0.018**
yeast	0.519±0.004	**0.5493±0.003**	-0.376±0.005	**-0.331±0.006**
cal500	0.259±0.005	**0.303±0.006**	-0.302±0.012	**-0.301±0.015**
birds	0.574±0.006	**0.583±0.004**	0.398±0.010	**0.402±0.008**
enron	0.450±0.003	**0.481±0.003**	0.341±0.004	**0.375±0.003**
medical	0.691±0.005	**0.770±0.003**	0.712±0.004	**0.741±0.003**

TABLE V: Comparing with CFT on Accuracy score (↑) and Composite score (↑), non-linear case

	CPLST	PRAkEL	CFT(linear)	CFT(non-linear)
F1 score	7/0/0	1/1/5	2/1/4	7/0/0
Rank loss	7/0/0	3/1/3	2/2/3	7/0/0
Accuracy score	7/0/0	3/1/3	3/2/2	7/0/0
Composite score	7/0/0	4/2/1	5/1/1	5/2/0
overall	28/0/0	11/5/12	12/6/10	26/2/0

TABLE VI: Comparison of CSEDR-$codec_{RAkEL}$ with CPLST, PRAkEL and CFT using Student's t-test with 95% confidence level (#win/#tie/#lose) on 7 datasets

CONCLUSION

We propose an algorithms, CSEDR, which enables the existing label space dimension reduction algorithms to acquire cost-sensitivity with generality, and embeds the codeword with cost information into arbitrary desired dimensions. CSEDR successfully bridges between dimension expansion and dimension reduction algorithms, and with our careful design of sub-problem dimension reduction trick, CSEDR operates with low computational burden under general criteria. Among the general cost-sensitive algorithms, CSEDR reduces cost-sensitive multi-label problem into multiple regression problems and enjoys lower time complexity than CFT and lower encoding dimension needed than PRAkEL, while being competitive and even better. To the best of our knowledge, CSEDR is the first algorithm that extends the problem domain of existing LSDR algorithms to both meet the upcoming new criteria as well as to compete with the state-of-the-art CSMLC algorithms.

REFERENCES

[1] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[2] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*, 75, 2008.

[3] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.

[4] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2001.

[5] F Briggs, B Lakshminarayanan, L Neal, XZ Fern, R Raich, SJK Hadley, AS Hadley, and MG Betts. New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–8, 2013.

[6] Yu-Ping Wu and Hsuan-Tien Lin. Progressive k-labelsets for cost-sensitive multi-label classification. *Machine Learning*, 2016. Accepted for Special Issue of ACML 2016.

[7] Hung-Yi Lo, Ju-Chiang Wang, Hsin-Min Wang, and Shou-De Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia*, 13(3):518–529, 2011.

[8] Yi Zhang and Jeff G Schneider. Multi-label output codes using canonical correlation analysis. In *AISTATS*, pages 873–882, 2011.

[9] Chun-Sung Ferng and Hsuan-Tien Lin. Multi-label classification with error-correcting codes. In *ACML*, pages 281–295, 2011.

[10] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[11] Chun-Liang Li and Hsuan-Tien Lin. Condensed filter tree for cost-sensitive multi-label classification. In *ICML*, pages 423–431, 2014.