Cost-Sensitive Reference Pair Encoding for Multi-Label Learning

Yao-Yuan Yang, Kuan-Hao Huang, Chih-Wei Chang, and Hsuan-Tien Lin

CSIE Department, National Taiwan University
{b01902066,r03922062}@ntu.edu.tw, cwchang@cs.cmu.edu, htlin@csie.ntu.edu.tw

Abstract. Label space expansion for multi-label classification (MLC) is a methodology that encodes the original label vectors to higher dimensional codes before training and decodes the predicted codes back to the label vectors during testing. The methodology has been demonstrated to improve the performance of MLC algorithms when coupled with off-the-shelf error-correcting codes for encoding and decoding. Nevertheless, such a coding scheme can be complicated to implement, and cannot easily satisfy a common application need of cost-sensitive MLC—adapting to different evaluation criteria of interest. In this work, we show that a simpler coding scheme based on the concept of a reference pair of label vectors achieves cost-sensitivity more naturally. In particular, our proposed cost-sensitive reference pair encoding (CSRPE) algorithm contains cluster-based encoding, weight-based training and voting-based decoding steps, all utilizing the cost information. Furthermore, we leverage the cost information embedded in the code space of CSRPE to propose a novel active learning algorithm for cost-sensitive MLC. Extensive experimental results verify that CSRPE performs better than state-of-the-art algorithms across different MLC criteria. The results also demonstrate that the CSRPE-backed active learning algorithm is superior to existing algorithms for active MLC, and further justify the usefulness of CSRPE.

Keywords: multi-label classification, cost-sensitive, active learning

1 Introduction

The *multi-label classification* (MLC) problem aims to map an instance to multiple relevant labels [1, 2], which matches the needs of many real-world applications, such as object detection and news classification. Different applications generally require evaluating the performance of MLC algorithms with different criteria, such as the Hamming loss, 0/1 loss, Rank loss, and F1 score [3].

Most existing MLC algorithms are designed to optimize one or few criteria. For instance, *binary relevance* (BR) [3] learns a binary classifier per label to predict its relevance, and naturally optimizes the Hamming loss. *Classifier chain* (CC) [4] extends BR by ordering the labels as a chain and using earlier labels of the chain to improve the per-label prediction, and optimizes the Hamming loss like BR. *Label powerset* (LP) [3] optimizes the 0/1 loss by solving a multi-class classification problem that treats each label combination as a hyper-class.

These *cost-insensitive* algorithms cannot easily adapt to different criteria, and may suffer from bad performance when evaluated with other criteria.

Cost-sensitive MLC (CSMLC) algorithms are able to adapt to different criteria more easily. In particular, CSMLC algorithms take the criterion as an additional piece of input data and aim to optimize the criterion during the learning process. Two state-of-the-art CSMLC algorithms are *probabilistic classifier chain* (PCC) [5] and *condensed filter tree* (CFT) [6]. PCC estimates the conditional probability of the labels to infer the Bayes-optimal decision with respect to the given criterion. While PCC can tackle any criterion in principle, the Bayes-optimal inference step can be time-consuming unless an efficient inference rule of the criterion is derived in advance. CFT can be viewed as an extension of CC for CSMLC by re-weighting each example with respect to the criterion when training each binary classifier. Nevertheless, the re-weighting step depends on going back and forth within the chain, making CFT still somewhat time-consuming and hardly parallelizable.

The *multi-label error-correcting code* (ML-ECC) [7] framework is a more sophisticated algorithm that goes beyond the per-label classifiers to improve classification performance. ML-ECC uses error-correcting code (ECC) to transform the original MLC problem into a bigger MLC problem by adding error-correcting labels during encoding. Classifiers on those labels, much like ECC for communication, can be used to correct prediction errors made from the original per-label classifiers and improve MLC performance. While ML-ECC is successful in terms of the Hamming loss and 0/1 loss [7], it is not cost-sensitive and cannot easily adapt to other evaluation criteria. In fact, extending ML-ECC for CSMLC problem appears to be highly non-trivial and has not yet been deeply studied.

In this work, we study the potential of ECC for CSMLC by considering a special type of ECC, the one-versus-one (OVO) code, which is a popular code for multi-class classification [8]. We extend the OVO code to a cost-sensitive code, *cost-sensitive reference pair encoding* (CSRPE), which preserves the information of the criterion in each code-bit during encoding. We further propose a method to convert the criterion into instance weights during training, and a method to take the criterion into account during decoding. To make the whole CSRPE algorithm efficient enough to deal with exponentially many possible label vectors, we study the possibility of sampling the code-bits and zooming into a smaller subset of label vectors during prediction. The resulting algorithm is as efficient as a typical random forest (when coupled with decision trees) in training, and can be easily implemented in parallel. Extensive experimental results demonstrate that CSRPE outperforms existing ML-ECC algorithms and the state-of-the-art CSMLC algorithms across different criteria.

In addition, based on the proposed CSRPE, we design a novel algorithm for *multi-label active learning* (MLAL). Retrieving ground-truth labels is usually expensive in real-world applications [2]. The goal of MLAL is to actively query the labels for a small number of instances while maintaining good test MLC performance. Nevertheless, current MLAL algorithms [9–11] are not capable of taking the evaluation criterion into consideration when querying. In this paper, we formulate the *cost-sensitive multi-label active learning* (CSMLAL) setting, and propose a novel algorithm that leverages the code space computed by CSRPE to conduct cost-sensitive querying. Experimental results justify that the proposed algorithm is superior to other state-of-the-art MLAL algorithms.

This paper is organized as follows. First, we define CSMLC problem formally and introduce the ML-ECC framework in Section 2. Our proposed CSRPE algorithm is described in Section 3. In Section 4, we define the CSMLAL problem and solve it with a novel algorithm based on CSRPE. The empirical studies of both CSRPE and its active learning extension are presented in Section 5. Finally, we conclude the paper in Section 6.

2 Preliminary

The goal of a MLC problem is to map the feature vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ to a label vector $\mathbf{y} \in \mathcal{Y} \subseteq \{0,1\}^K$, where $\mathbf{y}[k] = 1$ if and only if the k-th bit is relevant. During training, MLC algorithms use the training dataset $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ to learn a classifier $f \colon \mathcal{X} \rightarrow \mathcal{Y}$. During testing, for any test example $(\mathbf{x}, \mathbf{y})$ drawn from by some distribution that generated $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$, the prediction $f(\mathbf{x})$ is evaluated with a cost function $C \colon \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $C(\mathbf{y}, \hat{\mathbf{y}})$ represents the penalty of predicting $\mathbf{y}$ as $\hat{\mathbf{y}}$. The objective of MLC algorithms is to minimize the expected cost $\mathbb{E}_{(\mathbf{x},\mathbf{y})}[C(\mathbf{y}, f(\mathbf{x}))]$.

Traditional MLC algorithms are designed to optimize one or few cost functions. These algorithms may suffer from bad performance when other cost functions are used. On the contrary, *cost-sensitive multi-label classification* (CSMLC) algorithms take the cost function as an additional input and learn a classifier f from both $\mathcal{D}$ and C. Classifier f should adapt to different C easily.

The *multi-label error-correcting code* (ML-ECC) [7] framework is originally designed to optimize one cost function (the 0/1 loss). ML-ECC borrows the error-correcting code (ECC) from the communication domain. ML-ECC views the label vectors $\mathbf{y}^{(n)}$ as bit strings and encodes them to longer codes $\mathbf{b}^{(n)} = enc(\mathbf{y}^{(n)})$ with some ECC encoder $enc \colon \mathcal{Y} \rightarrow \{0,1\}^M$, where M is the code length. An MLC classifier h is trained on $\{(\mathbf{x}^{(n)}, \mathbf{b}^{(n)})\}$ to predict the codes instead of the label vectors. The code-bits store redundant information about the label vector to recover the intended label vector even when some bits are mispredicted by h. In prediction, the corresponding ECC decoder $dec \colon \{0,1\}^M \rightarrow \mathcal{Y}$, is used to convert the predicted vector from h back to the label vector $f(\mathbf{x}) = dec(h(\mathbf{x}))$. In other words, ML-ECC learns the classifier $f = dec \circ h$. Such an ECC decoder is often designed based on special nearest-neighbor search steps in the code space [7].

In the original work of ML-ECC [7], several encoder/decoder choices are discussed and experimentally evaluated. Nevertheless, none of them take the cost information into account. In fact, to the best of our knowledge, there is currently no work that deeply studies the potential of ECC for CSMLC. Next, we illustrate our ideas on making a special ECC cost-sensitive.

3 Proposed Approach

We start from a special cost-insensitive ECC, the *one-versus-one* (OVO) code. The OVO code is the core of the OVO meta-algorithm for *multi-class classification* (MCC). The meta-algorithm trains many binary classifiers, each representing the duel between *two* of the classes, and let the binary classifiers vote for the majority decision for MCC.

To study the OVO code for MLC, we can naïvely follow the label powerset algorithm [3] to reduce the MLC problem to MCC and then apply the OVO meta-algorithm to further reduce MCC to binary classification. As a consequence, each label vector $\mathbf{y} \in \mathcal{Y}$ is simply treated as a distinct hyper-class, and each binary classifier within the OVO meta-algorithm represents a duel between *two* label vectors. More specifically, the i-th classifier is associated with two label vectors $\mathbf{y}_\alpha^i$ and $\mathbf{y}_\beta^i$, called the reference label vectors. There are $\binom{2^K}{2}$ such classifiers, each can be trained with examples in D that match either $\mathbf{y}_\alpha^i$ and $\mathbf{y}_\beta^i$. During prediction, the $\binom{2^K}{2}$ binary classifiers can then vote for all the label vectors $\in \mathcal{Y}$ towards the majority decision.

The steps of applying OVO to MLC above can be alternatively described as a special ML-ECC algorithm, similar to how OVO is viewed as a special ECC for MCC [12]. OVO as ML-ECC encodes each label vector to a code of length $\binom{2^K}{2}$ with the following encoder $enc_{ovo}(\mathbf{y})[i] = \begin{cases} 1 & \text{if } \mathbf{y} = \mathbf{y}_\alpha^i \\ 0 & \text{if } \mathbf{y} = \mathbf{y}_\beta^i \\ 0.5 & \text{otherwise} \end{cases}$. The i-th bit in the code represents whether the label vector matches either of the reference vectors. The the special "bit" value of 0.5 for representing other irrelevant label vectors. Then, decoding based on majority voting is equivalent to nearest-neighbor search in the code space over all possible encoded $\mathbf{y} \in \mathcal{Y}$ in terms of the Hamming distance (d_{ham}), as the Hamming distance is a linear function of the vote that each $\mathbf{y}$ gets. More precisely, denote the predicted code as $\hat{\mathbf{b}} = h(\mathbf{x})$, the decoder of OVO is simply $dec_{ovo}(\hat{\mathbf{b}}) = \text{argmax}_{\mathbf{y}\in\mathcal{Y}}(d_{ham}(\hat{\mathbf{b}}, enc_{ovo}(\mathbf{y})))$.

The naïve OVO for ML-ECC above suffers from several issues. First, the code length $\binom{2^K}{2}$ is prohibitively long for large K, making it inefficient to compute. Second, many of the $\binom{2^K}{2}$ classifiers may not be associated with enough data during training. Last but not least, OVO is not cost-sensitive and cannot adapt to different cost functions easily. We resolve the issues in the designs below.

Cost-sensitive encoding. The OVO code is designed to optimize 0/1 loss ($C(\mathbf{y}, \hat{\mathbf{y}}) = [\![\mathbf{y} \neq \hat{\mathbf{y}}]\!]$, where $[\![\cdot]\!]$ is the indicator function) for MLC. In the OVO code, each bit of $enc_{ovo}(\mathbf{y})$ is learned from only the instances with $\mathbf{y}$ being exactly the same as $\mathbf{y}_\alpha^i$ or $\mathbf{y}_\beta^i$. For instances with $\mathbf{y}$ being neither $\mathbf{y}_\alpha^i$ nor $\mathbf{y}_\beta^i$, these instances will be dropped from training. This suits the design of optimizing 0/1 loss. Now, we take a different perspective to view the OVO code.

When considering 0/1 loss, what the OVO code does is to decide whether predict as $\mathbf{y}_\alpha^i$ or $\mathbf{y}_\beta^i$ suffers less 0/1 loss. For the case that $\mathbf{y}$ is neither $\mathbf{y}_\alpha^i$ nor $\mathbf{y}_\beta^i$, the costs for predicting as $\mathbf{y}_\alpha^i$ and $\mathbf{y}_\beta^i$ are the same. That is why OVO code ignores these cases during training. However, for other cost functions, the costs for predicting $\mathbf{y}$ as $\mathbf{y}_\alpha^i$ and $\mathbf{y}_\beta^i$ can be different. Hence, even if the label vector $\mathbf{y}$ is neither $\mathbf{y}_\alpha^i$ nor $\mathbf{y}_\beta^i$, the vector can still provide information for training.

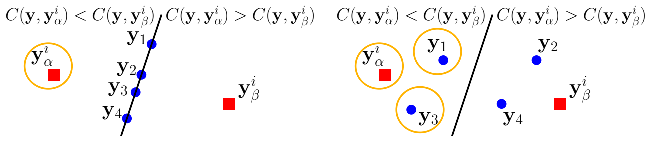
To generalize the encoding function towards cost-sensitivity, we hold the same idea that each bit should predict which reference label vector incurs less cost. The encoding function is designed as $enc_{cs}(\mathbf{y})[i] = \begin{cases} 1 & \text{if } C(\mathbf{y},\mathbf{y}_\alpha^i) < C(\mathbf{y},\mathbf{y}_\beta^i) \\ 0 & \text{if } C(\mathbf{y},\mathbf{y}_\alpha^i) > C(\mathbf{y},\mathbf{y}_\beta^i) \\ 0.5 & \text{otherwise} \end{cases}$.

Training classifiers for cost-sensitive codes. With the encoding function defined, we learn a classifier h to predict the encoded vectors outputted from enc_{cs}. Although enc_{cs} gives the classifier a better ground truth, different label vectors are not equally important for the classifier. For example, if $C(\mathbf{y},\mathbf{y}_\alpha^i)$ and $C(\mathbf{y},\mathbf{y}_\beta^i)$ differ by a lot, there would be a high cost if the classifier gives the wrong prediction, thus making $\mathbf{y}$ very important. In contrast, if there exists a label vector $\mathbf{y}$ s.t. $C(\mathbf{y},\mathbf{y}_\alpha^i) \approx C(\mathbf{y},\mathbf{y}_\beta^i)$, then $\mathbf{y}$ is relatively unimportant because a misclassified $\mathbf{y}$ would not incur a high cost. Thus, we design a weight function to emphasize the importance for each label vector as $weight(\mathbf{y})[i] = |C(\mathbf{y},\mathbf{y}_\alpha^i) - C(\mathbf{y},\mathbf{y}_\beta^i)|$.

Dataset $\{(\mathbf{x}^{(n)}, enc_{cs}(\mathbf{y}^{(n)}), weight(\mathbf{y}^{(n)}))\}_{n=1}^N$ is used to train the classifier h to predict the encoded vector. Normally, h should be trained on the full-length encoded vectors. But the exponentially growing code length $\binom{2^K}{2}$ makes training on the full encoding infeasible. However, many classifiers would result in learning similar problems during training. This could allow us to use fewer bits and preserves the same amount of information. For example, let the i-th reference label vectors be $\mathbf{y}_\alpha^i = (1,0,1,0)$ and $\mathbf{y}_\beta^i = (1,0,0,1)$, and the j-th reference vectors be $\mathbf{y}_\alpha^j = (1,1,1,0)$ and $\mathbf{y}_\beta^j = (1,1,0,1)$. The i-th and j-th classifier are actually learning similar things: learning to predict whether the last two labels of the label vector should be $(1,0)$ or $(0,1)$. Observing the redundancy in the encoded vectors, it is clear that the length of the encoded vector can be decreased and thus learning becomes feasible. For simplicity, we uniformly sample some bits for from encoded vectors. In Section 5, we demonstrate that the number of needed bits are much smaller than $\binom{2^K}{2}$.

Cost-sensitive decoding. OVO code decodes by letting each bit votes on either of the reference label vectors. Following the idea for encoding, this is also a special case of decoding by considering the 0/1 loss. To match with our proposed cost-sensitive encoding, the decoding approach is redesigned to utilize the information more effectively.

Figure 1 is an illustration of the relation between encoded vectors under OVO encoding and our cost-sensitive encoding. In 0/1 loss, all instances that are predicted incorrectly incur the same cost making all label vectors except $\mathbf{y}_\alpha^i$ and $\mathbf{y}_\beta^i$ are on the decision boundary. Only $\mathbf{y}_\alpha^i$ and $\mathbf{y}_\beta^i$ are distinguishable under the current bit. Thus, original OVO voting only needs to be done on reference label vectors. When using our cost-sensitive encoding, all label vectors are generally separated into two groups by the boundary as Figure 1(b): the group that


$C(\mathbf{y}, \mathbf{y}^i_\alpha) < C(\mathbf{y}, \mathbf{y}^i_\beta)$
$C(\mathbf{y}, \mathbf{y}^i_\alpha) > C(\mathbf{y}, \mathbf{y}^i_\beta)$
$\mathbf{y}^i_\alpha$
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_3$
$\mathbf{y}_4$
$\mathbf{y}^i_\beta$
$C(\mathbf{y}, \mathbf{y}^i_\alpha) < C(\mathbf{y}, \mathbf{y}^i_\beta)$
$C(\mathbf{y}, \mathbf{y}^i_\alpha) > C(\mathbf{y}, \mathbf{y}^i_\beta)$
$\mathbf{y}^i_\alpha$
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_3$
$\mathbf{y}_4$
$\mathbf{y}^i_\beta$

(a) OVO voting (0/1 loss) (b) proposed voting

Fig. 1: An illustration of the decoding methods.

is closer to $\mathbf{y}^i_\alpha$ (left) (in terms of cost) and the group that is closer to $\mathbf{y}^i_\beta$. A predicted encoded bit not only provides the information about the reference label vector, but also the information about all other label vectors in the same group. Following this thought, if the prediction is $\mathbf{y}^i_\alpha$, all label vectors $\mathbf{y}$ such that $C(\mathbf{y}, \mathbf{y}^i_\alpha) < C(\mathbf{y}, \mathbf{y}^i_\beta)$ should be voted as well. If predicted otherwise, all label vectors in the other group are voted. By this voting approach, we can use the information encoded within the vectors to decode more effectively.

In fact, this voting approach echoes the Hamming decoding for ECC [12]. More specifically, with the predicted encoded vector $\hat{\mathbf{b}} = h(\mathbf{x})$, the decoding function is written as $dec_{cs}(\hat{\mathbf{b}}) = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} d_{ham}(\hat{\mathbf{b}}, enc_{cs}(\mathbf{y}))$ With this formulation, dec_{cs} is formulated as the classic nearest neighbor search problem, where efficient algorithms exist to speed up the decoding process [13].

Despite the efficient decoding algorithm, the number of possible predictions $|\mathcal{Y}|$ equals 2^K, which makes it computationally infeasible. Inspired by [14], we propose to only work with a subset of label vectors that are more likely to be the prediction. We define a relevant set $\tilde{\mathcal{Y}} \subseteq \mathcal{Y}$, which contains a subset of the label vectors from the label space, on which we perform the nearest neighbor search. The decoding function is written as $dec_{cs}(\hat{\mathbf{b}}) = \operatorname{argmax}_{\mathbf{y}\in\tilde{\mathcal{Y}}} d_{ham}(\hat{\mathbf{b}}, enc_{cs}(\mathbf{y}))$.

The use of the $\tilde{\mathcal{Y}}$ introduces a trade-off between the number of possible predictions and the prediction efficiency. A reasonable choice of $\tilde{\mathcal{Y}}$ would be $\{\mathbf{y}|(\mathbf{x}, \mathbf{y}) \in \mathcal{D}\}$, which are the distinct label vectors in the training set. Given that the training and testing sets come from the same distribution, the label vectors that appear in the testing set are likely to have appeared in the training set. We justify this choice of $\tilde{\mathcal{Y}}$ in Section 5.

The algorithm that combines enc_{cs}, *weight* and dec_{cs} is called *cost-sensitive reference pair encoding* (CSRPE). Our design is inspired by a cost-sensitive extension of OVO for MCC problem called *cost-sensitive one-versus-one* [8], but is refined by our special ideas for encoding and decoding in the MLC problem.

4 Active Learning for CSMLC

CSRPE is able to preserve cost information in the encoded vectors. In this section, we design a novel active learning algorithm for MLC based on CSRPE.

MLC algorithms intend to learn a classifier from a fully labeled dataset, in which every feature vector is paired with a label vector. In many real-world applications, obtaining a label vector to the corresponding feature vector is very expensive [2]. This gives rise to a new problem, active learning, which investigates how to obtain good performance with as little data labeled as possible.

In this paper, we consider the pool-based *multi-label active learning* (MLAL) setting [15] and formulate the cost-sensitive extension of MLAL called *cost-sensitive multi-label active learning* (CSMLAL). In CSMLAL, the algorithm is presented with two sets of data, the labeled pool $\mathcal{D}_l = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N_l}$ and the unlabeled pool $\mathcal{D}_u = \{\mathbf{x}^{(n)}\}_{n=1}^{N_u}$. During iterations $t = 1, \ldots, T$, the MLAL algorithm considers $\mathcal{D}_u$, $\mathcal{D}_l$, a MLC classifier f_t trained on $\mathcal{D}_l$ and cost function C to choose a instance $\mathbf{x}_t \in \mathcal{D}_u$ to query. After the queried label vector is retrieved as $\mathbf{y}_t$, $\mathbf{x}_t$ is removed from $\mathcal{D}_u$ and the pair $(\mathbf{x}_t, \mathbf{y}_t)$ is added to $\mathcal{D}_l$. With a small budget of T queries, the goal of the CSMLAL algorithm is to minimize the average prediction cost of f_t on the testing instances evaluated on C.

Many of the current MLAL algorithms are based on the idea of *uncertainty sampling*. They query the instance that current classifier f_t is most uncertain about. There are different uncertainty measures being developed. However, most of these measures consider only one specific C or even completely ignoring C. *Binary minimization* [9] was proposed to directly take the most uncertain bit in the label vector to represent the uncertainty of the whole instance. It queries based on one label at a time and arguably optimizes towards Hamming loss. Another work, in contrast, calculates an average over the uncertainty of all labels [10]. Yet another work uses the difference between the most uncertain relevant label and irrelevant label as an uncertainty measure [11]. This uncertainty is then combined with label cardinality inconsistency. However, this measure is designed heuristically and does not aim at any C.

We propose *cost-sensitive uncertainty* in the encoded vector space to evaluate the importance of instances. The cost-sensitive uncertainty can be separated into two parts, the *cost estimation uncertainty* and the *cost utility uncertainty*.

Cost estimation uncertainty. Cost estimation uncertainty measures how well CSRPE estimates the cost between label vectors. Let the predicted encoded vector $\hat{\mathbf{b}} = h(\mathbf{x})$ and $\tilde{\mathbf{b}} = enc_{cs}(dec_{cs}(\hat{\mathbf{b}}))$. Note that $\tilde{\mathbf{b}}$ is actually the nearest encoded vector of $\hat{\mathbf{b}}$. Ideally, if CSRPE estimates the cost information well, $\hat{\mathbf{b}}$ should be close to $\tilde{\mathbf{b}}$. If, unfortunately, the distance $d_{ham}(\hat{\mathbf{b}}, \tilde{\mathbf{b}})$ is large, this implies that CSRPE does not have a good cost estimation for this $\mathbf{x}$ and we hence need more information about it. In other words, we are uncertain about this $\mathbf{x}$. For this reason, we define $d_{ham}(\hat{\mathbf{b}}, \tilde{\mathbf{b}})$ as the *cost estimation uncertainty*.

Cost utility uncertainty. The cost utility uncertainty measures how uncertain the classifier f_t is under the current cost function. Let the prediction $\bar{\mathbf{y}} = f_t(\mathbf{x})$ and its encoding $\bar{\mathbf{b}} = enc_{cs}(\bar{\mathbf{y}})$. If the classifier f_t is certain about its prediction under current cost function, $\bar{\mathbf{b}}$ should be close to the cost estimation $\hat{\mathbf{b}} = h(\mathbf{x})$. If unfortunately, distance $d_{ham}(\hat{\mathbf{b}}, \bar{\mathbf{b}})$ is large, it implies that classifier f_t is uncertain under the current cost function. Therefore, we define $d_{ham}(\hat{\mathbf{b}}, \bar{\mathbf{b}})$ as the *cost utility uncertainty*.

The proposed cost-sensitive uncertainty is the combination of these two parts of uncertainty, namely $d_{ham}(\hat{\mathbf{b}}, \tilde{\mathbf{b}}) + d_{ham}(\hat{\mathbf{b}}, \bar{\mathbf{b}})$. The cost-sensitive uncertainty leads to a novel algorithm for CSMLAL. For each iteration, the algorithm selects the instance with the highest cost-sensitive uncertainty to query its label.

5 Experiments

We justify the proposed algorithm on ten public datasets [16] and three common evaluation criteria, including F1 score, Accuracy score, Rank loss. [3]. The experiment was run 20 times, each with a random 50-50 training-testing split. CSRPE has the flexibility to take any base learner. In CSMLC experiments, CSRPE is viewed as an ensemble MLC method, each bit with a binary classifier attached. Because ensemble of decision trees is arguably a popular ensemble method nowadays, we use decision trees as the base learner in these experiments. The parameters are searched with 3-fold cross-validation.

In CSMLAL experiments, the experiments are repeated for 10 runs. Since many of competitors designed their algorithms based on linear base learners, the base learner is changed to logistic regression for fair comparison. The parameters are searched with 5-fold cross-validation using the initial dataset.

More detailed experimental setup can be found in the full version [17]. In the following experimental results, we use ↑ (↓) to indicate that a higher (lower) value for the criterion is better.

Effect of Code Length. To justify our claim in Section 3 that the code length can be reduced by sampling, we conduct experiments to analyzing the performance of CSRPE with respect to the code length.

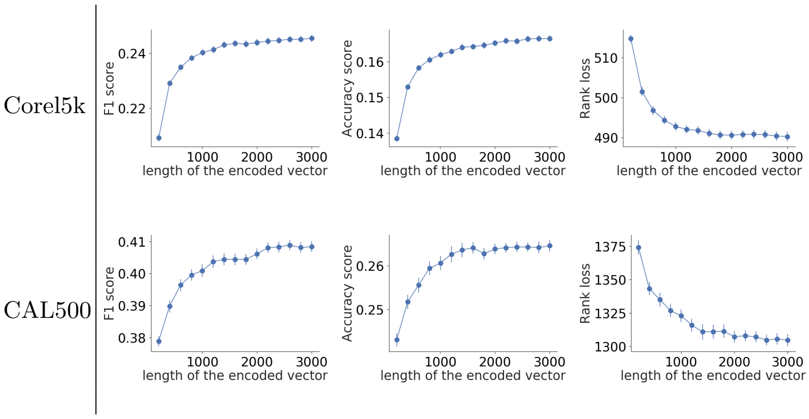


Fig. 2: Different criteria versus code length for CSRPE

Figure 2 shows the average performance and standard error versus code length. We select two of the datasets with larger label counts to showcase the effect of the code length on performance. The results of other datasets can be found in [17]. From the figures, CSRPE performs better as the number of bit increases. The performance of CSRPE generally converges when the code length reaches 3000 across all cost functions and datasets. The length is significantly smaller than the full encoding (2^K). This justifies our claim that full encoding is not needed to achieve top performance. In the following experiments, we set the code length as 3000.

Influence of the Relevant Set. In Section 3, we claim that a good choice for relevant set $\tilde{\mathcal{Y}}$ is all distinct label vectors in the training dataset. To justify our claim, we demonstrate that the possible downside of this choice, which is the inability to predict all possible label vectors, will not degrade the performance much. In particular, we compare CSRPE with CSRPE-ext, which is CSRPE-ext with a larger relevant set that includes label vectors that appeared in either the training set or the testing set.

Table 1: Experiment results (mean ± ste) of CSRPE and CSRPE-ext

Dataset	**Rank loss ↓**		**F1 score ↑**		**Accuracy score ↑**	
	CSRPE	CSRPE-ext	CSRPE	CSRPE-ext	CSRPE	CSRPE-ext
Corel5k	490.17 ± 1.20	**485.73 ± 0.88**	.2455 ± .0012	**.2492 ± .0011**	.1664 ± .0009	**.1674 ± .0009**
CAL500	1304.6 ± 4.57	**1303.4 ± 4.18**	.4083 ± .0017	**.4109 ± .0013**	.2645 ± .0013	**.2690 ± .0014**
bibtex	104.94 ± 0.38	**102.78 ± 0.32**	.4663 ± .0008	**.4695 ± .0009**	.3926 ± .0011	**.3946 ± .0010**
enron	34.32 ± 0.182	**33.47 ± 0.206**	.5911 ± .0014	**.5921 ± .0016**	.4772 ± .0016	**.4777 ± .0017**
medical	**5.330 ± 0.068**	5.415 ± 0.081	.8203 ± .0023	**.8204 ± .0023**	**.7939 ± .0024**	.7934 ± .0022
genbase	**0.353 ± 0.030**	0.360 ± 0.032	**.9878 ± .0009**	.9876 ± .0009	**.9836 ± .0010**	.9828 ± .0012
yeast	8.451 ± 0.030	**8.448 ± 0.026**	.6670 ± .0012	**.6679 ± .0012**	**.5653 ± .0012**	.5650 ± .0012
flags	**3.010 ± 0.047**	3.050 ± 0.050	**.7222 ± .0041**	.7192 ± .0043	**.6056 ± .0058**	.6028 ± .0052
scene	0.679 ± 0.008	**0.645 ± 0.006**	.7860 ± .0020	**.7913 ± .0014**	**.7620 ± .0020**	.7563 ± .0017
emotions	**0.591 ± 0.001**	0.592 ± 0.002	.6655 ± .0035	**.6673 ± .0030**	**.5775 ± .0037**	.5774 ± .0036

The results, which shows the mean and standard error (ste) of the criteria, are listed in Table 1. The results demonstrate that CSRPE-ext is slightly better performing, but the improvement is at best marginal and insignificant. Even in the CAL500 dataset, where all the label vectors in training and testing sets are different, there is only a small performance difference between CSRPE and CSRPE-ext. The result verifies that our choice of $\tilde{\mathcal{Y}}$ as all the distinct label vectors in the training set are sufficiently good.

Comparison with Other MLC Algorithms. In this experiment, we compare the performance of various MLC and CSMLC algorithms. For the MLC competitors, we include different codes applied within ML-ECC framework. The competing codes include the *Hamming on repetition code* (HAMR), *repetition code (REP)*, and *RAKEL repetition code (RREP)* [7]. REP and RREP are equivalent to BR [3] and RAKEL [18], respectively. In addition, CC [4] is added to serve as a baseline competitor together with REP and RREP. For CSMLC algorithms, we compete with PCC [5] and CFT [6].

The results are shown in Table 2 and 3. The results show that CSMLC algorithms generally outperform traditional MLC algorithms. This justifies that it is important to take cost information into account. Among the CSMLC algorithms, CSRPE is superior over all other competitors with respect to F1 and Accuracy score. For Rank loss, PCC performs slightly better, but CSRPE still performs competitively with PCC and CFT. Such result justifies CSRPE as a top performing CSMLC algorithm.

Table 2: Experiment results (mean ± ste) on different criteria (best in bold)

			F1 score ↑				
Dataset	REP (BR)	RREP (RAKEL)	HAMR	CC	PCC	CFT	CSRPE
Corel5k	.0683 ± .0011	.1028 ± .0010	.0608 ± .0008	.0661 ± .0009	.1759 ± .0008	.1708 ± .0017	**.2455 ± .0012**
CAL500	.3388 ± .0014	.3527 ± .0011	.3152 ± .0012	.3354 ± .0024	.3540 ± .0018	.3815 ± .0016	**.4083 ± .0017**
bibtex	.3636 ± .0009	.3761 ± .0010	.3658 ± .0008	.3569 ± .0009	.3736 ± .0011	.3957 ± .0015	**.4663 ± .0008**
enron	.5441 ± .0026	.5336 ± .0025	.5459 ± .0023	.5492 ± .0022	.5508 ± .0014	.5530 ± .0013	**.5911 ± .0014**
medical	.7883 ± .0028	.7757 ± .0034	.7877 ± .0031	.7924 ± .0035	.8131 ± .0023	.7970 ± .0031	**.8203 ± .0023**
genbase	.9897 ± .0012	.9893 ± .0014	.9896 ± .0012	.9896 ± .0012	**.9911 ± .0007**	.9845 ± .0009	.9878 ± .0008
yeast	.6119 ± .0014	.6130 ± .0011	.6171 ± .0015	.5968 ± .0018	.6013 ± .0013	.6111 ± .0024	**.6670 ± .0012**
flags	.6954 ± .0045	.6965 ± .0044	.7005 ± .0044	.6973 ± .0048	.7075 ± .0038	.6725 ± .0055	**.7222 ± .0041**
scene	.5895 ± .0026	.5926 ± .0019	.6365 ± .0021	.6547 ± .0019	.7306 ± .0016	.6592 ± .0027	**.7860 ± .0020**
emotions	.5968 ± .0038	.5773 ± .0047	.6100 ± .0035	.6205 ± .0035	.6384 ± .0033	.6015 ± .0043	**.6655 ± .0035**
			Accuracy score ↑				
Dataset	REP (BR)	RREP (RAKEL)	HAMR	CC	PCC	CFT	CSRPE
Corel5k	.0471 ± .0007	.0696 ± .0006	.0408 ± .0009	.0471 ± .0007	.1135 ± .0005	.0790 ± .0019	**.1664 ± .0009**
CAL500	.2097 ± .0010	.2179 ± .0008	.1925 ± .0007	.2085 ± .0018	.2209 ± .0012	.2425 ± .0015	**.2645 ± .0013**
bibtex	.3063 ± .0009	.3103 ± .0009	.3094 ± .0008	.3031 ± .0010	.2940 ± .0010	.3235 ± .0011	**.3926 ± .0011**
enron	.4303 ± .0023	.4215 ± .0022	.4344 ± .0024	.4437 ± .0021	.4259 ± .0013	.4363 ± .0018	**.4772 ± .0016**
medical	.7559 ± .0034	.7431 ± .0033	.7604 ± .0033	.7643 ± .0035	.7716 ± .0025	.7570 ± .0031	**.7939 ± .0024**
genbase	.9859 ± .0014	.9852 ± .0015	.9856 ± .0014	.9858 ± .0014	**.9873 ± .0009**	.9792 ± .0012	.9835 ± .0010
yeast	.5047 ± .0014	.5065 ± .0012	.5120 ± .0015	.4954 ± .0021	.4872 ± .0017	.5027 ± .0019	**.5653 ± .0012**
flags	.5849 ± .0047	.5860 ± .0046	.5913 ± .0051	.5908 ± .0057	.5974 ± .0041	.5616 ± .0059	**.6056 ± .0058**
scene	.5791 ± .0025	.5816 ± .0020	.6258 ± .0017	.6457 ± .0018	.6821 ± .0019	.6467 ± .0029	**.7620 ± .0020**
emotions	.5179 ± .0037	.4959 ± .0045	.5320 ± .0034	.5417 ± .0035	.5433 ± .0035	.5216 ± .0036	**.5775 ± .0037**
			Rank loss ↓				
Dataset	REP (BR)	RREP (RAKEL)	HAMR	CC	PCC	CFT	CSRPE
Corel5k	618.1 ± .6695	597.2 ± .6664	623.5 ± .6474	636.0 ± .5374	421.2 ± .6626	**300.7 ± .7848**	490.2 ± 1.1959
CAL500	1500. ± 5.023	1477. ± 4.835	1537. ± 4.488	1520. ± 6.155	1179. ± 4.498	**1122. ± 4.470**	1305. ± 4.574
bibtex	132.6 ± .2981	124.1 ± .2511	131.5 ± .2819	136.8 ± .2886	**69.10 ± .2454**	112.06 ± .2811	104.9 ± .3814
enron	43.39 ± .2919	44.06 ± .2810	43.40 ± .2540	43.56 ± .3000	27.94 ± .1681	**27.20 ± .1365**	34.32 ± .1815
medical	5.454 ± .1184	5.733 ± .1088	5.601 ± .1232	5.469 ± .0997	**3.058 ± .0603**	4.117 ± .0741	5.330 ± .0676
genbase	.2461 ± .0281	.2422 ± .0273	.2525 ± .0257	.2423 ± .0308	**.1976 ± .0178**	.4686 ± .0310	.3863 ± .0341
yeast	9.609 ± .0358	9.565 ± .0290	9.443 ± .0312	10.324 ± .0448	9.378 ± .0365	9.473 ± .0363	**8.451 ± .0298**
flags	3.123 ± .0434	3.139 ± .0383	3.078 ± .0352	3.120 ± .0450	3.012 ± .0490	3.363 ± .0504	**3.010 ± .0470**
scene	1.136 ± .0066	1.149 ± .0055	1.031 ± .0046	1.098 ± .0080	0.726 ± .0060	0.892 ± .0069	**0.679 ± .0083**
emotions	1.789 ± .0182	1.906 ± .0220	1.764 ± .0165	1.741 ± .0207	**1.563 ± .0176**	1.834 ± .0281	1.591 ± .0198

Table 3: CSRPE versus others based on *t*-test at 95% confident level

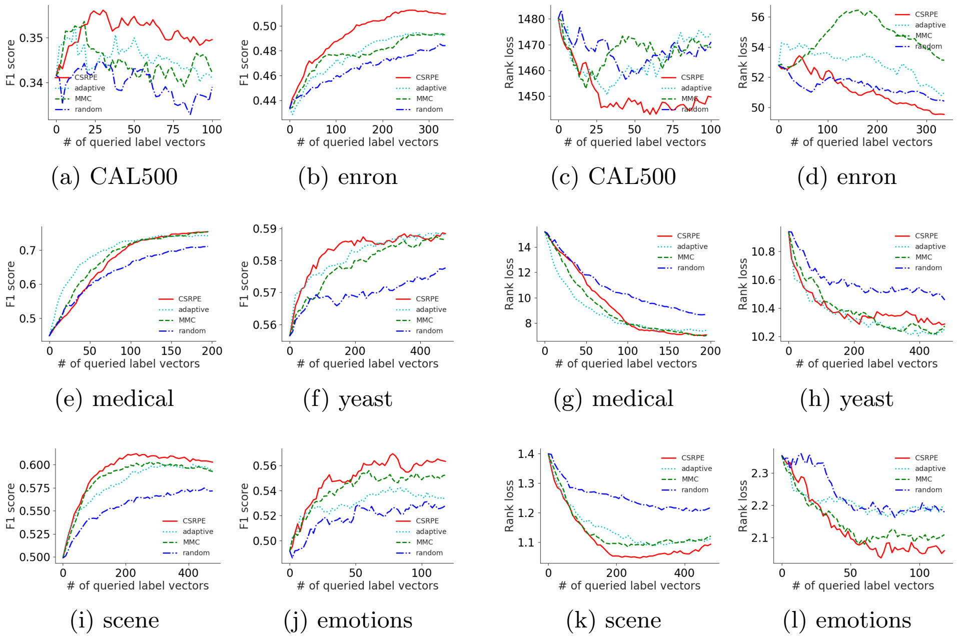
criteria (win/tie/loss)	F1	Rank.	Acc.	total
REP	9/1/0	7/2/1	9/0/1	27/7/6
RREP	9/1/0	9/0/1	9/1/0	31/5/4
HAMR	9/1/0	7/2/1	8/2/0	26/9/5
CC	9/1/0	7/2/1	8/2/0	30/6/4
CFT	9/1/0	6/1/3	9/1/0	30/4/6
PCC	9/0/1	2/2/6	8/1/1	22/7/11

Comparison with MLAL Algorithms In this experiment, we evaluate the performance of CSRPE under the CSMLAL setting. We compare it with several state-of-the-art MLAL algorithms, which includes *adaptive active learning* (adaptive) [11], *maximal loss reduction with maximal confidence* (MMC) [10], and random sampling as a baseline algorithm. We do not include a comparison with *binary minimization* [9] since MMC and adaptive are reported to outperform it.

Figure 3 shows the performance with respect to the number of instances queried. For F1 score and Rank loss, CSRPE performs better than other strategies on four out of six datasets. These results indicate that CSRPE is able to consider the cost information, thus enabling it to outperform other competitors on most of the datasets across different evaluation criteria.

6 Conclusion

In this paper, we propose a novel approach for cost-sensitive multi-label classification (CSMLC), called *cost-sensitive reference pair encoding* (CSRPE). CSRPE



(a) CAL500 (b) enron (c) CAL500 (d) enron

(e) medical (f) yeast (g) medical (h) yeast

(i) scene (j) emotions (k) scene (l) emotions

Fig. 3: CSMLAL results with F1 score and Rank loss

is derived from the one-versus-one algorithm and can embed the cost information into the encoded vectors. Exploiting the redundancy of the encoded vectors, we use random sampling to resolve the training challenge of building so many classifiers. We also design a nearest-neighbor-based decoding procedure and use the relevant set to efficiently make cost-sensitive predictions. Extensive experimental results demonstrate that CSRPE achieves stable convergence respect to the code length and outperforms not only other encoding methods but also state-of-the-art CSMLC algorithms across different cost functions. In addition, we extend CSRPE to a novel multi-label active learning algorithm by designing a cost-sensitive uncertainty measure. Extensive empirical studies show that the proposed active learning algorithm performs better than existing active learning algorithms. The results suggest that CSRPE is a promising cost-sensitive encoding method for CSMLC for either supervised or active learning.

Acknowledgments. We thank the anonymous reviewers and the members of NTU CLLab for valuable suggestions. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-15-1-4012, and by the Ministry of Science and Technology of Taiwan under MOST 103-2221-E-002-149-MY3 and 106-2119-M-007-027.

References

1. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. ECML PKDD discovery challenge **75** (2008)
2. Liu, Y.: Active learning with support vector machine applied to gene expression data for cancer classification. Journal of Chemical Information and Computer Sciences (2004) 1936–1941
3. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook. (2010) 667–685
4. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine learning **85**(3) (2011) 333–359
5. Dembczynski, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML. (2010)
6. Li, C.L., Lin, H.T.: Condensed filter tree for cost-sensitive multi-label classification. In: ICML. (2014)
7. Ferng, C.S., Lin, H.T.: Multilabel classification using error-correcting codes of hard or soft bits. IEEE Transactions on Neural Networks and Learning Systems **24**(11) (2013) 1888–1900
8. Lin, H.T.: Reduction from cost-sensitive multiclass classification to one-versus-one binary classification. In: ACML. (2014)
9. Brinker, K.: On active learning in multi-label classification. In: From Data and Information Analysis to Knowledge Engineering. (2006) 206–213
10. Yang, B., Sun, J.T., Wang, T., Chen, Z.: Effective multi-label active learning for text classification. In: ICDM. (2009)
11. Li, X., Guo, Y.: Active learning with multi-label svm classification. In: IJCAI. (2013)
12. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research **1** (2001) 113–141
13. Liu, T., Moore, A.W., Gray, A.: New algorithms for efficient high-dimensional nonparametric classification. Journal of Machine Learning Research **7** (2006) 1135–1158
14. Huang, K.H., Lin, H.T.: Cost-sensitive label embedding for multi-label classification. Machine Learning (2017) 1725–1746
15. Settles, B.: Active learning literature survey. University of Wisconsin, Madison (2010)
16. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: Mulan: A java library for multi-label learning. Journal of Machine Learning Research **12** (2011) 2411–2414
17. Yang, Y.Y., Huang, K.H., Chang, C.W., Lin, H.T.: Cost-sensitive random pair encoding for multi-label classification. arXiv preprint arXiv:1611.09461 (2016)
18. Tsoumakas, G., Vlahavas, I.P.: Random k -labelsets: An ensemble method for multilabel classification. In: ECML. (2007)