Stable Learning in Coding Space for Multi-Class Decoding and Its Extension for Multi-Class Hypothesis Transfer Learning

Bang Zhang[1], Yi Wang[1 2], Yang Wang[1], Fang Chen[1 2]
[1]National ICT Australia*
[2]School of Computer Science and Engineering, The University of New South Wales, Australia
{bang.zhang,yi.wang,yang.wang,fang.chen}@nicta.com.au

Abstract

Many prevalent multi-class classification approaches can be unified and generalized by the output coding framework [1, 7] which usually consists of three phases: (1) *coding,* (2) *learning binary classifiers, and* (3) *decoding. Most of these approaches focus on the first two phases and predefined distance function is used for decoding. In this paper, however, we propose to perform learning in coding space for more adaptive decoding, thereby improving overall performance. Ramp loss is exploited for measuring multi-class decoding error. The proposed algorithm has uniform stability. It is insensitive to data noises and scalable with large scale datasets. Generalization error bound and numerical results are given with promising outcomes.*

1. Introduction

Multi-class classification is a fundamental machine learning task as which many real-world problems can be casted. For instance, categorization of Internet resources, such as document, image and video, plays a critical role for online information retrieval. Robot vision, auto-driving car and augmented reality application require robust categorization module for distinguishing different visual categories.

Despite its popularity in real-world applications, multi-class classification has received relatively less attention compared with its binary counterpart. Therefore, many state-of-the-art multi-class classification approaches take advantage of the developments for binary problem by reducing a multi-class problem into a collection of binary problems. For instance, one-versus-one and one-versus-all [15] strategies are widely used in practice with good performances.

Output coding [1, 7] is a general framework that unifies and generalizes such type of approaches. It usually decomposes the learning process into three phases: (1) coding matrix design, which converts a multi-class classification problem into a set of binary problems, (2) training binary classifiers, which tackles the converted binary problems, (3) decoding, which makes multi-class classification decisions based on binary classification outputs with the aid of a decoding scheme.

Many efforts have been spent on either how to achieve optimal coding [7, 17] or how to optimize coding and binary learning simultaneously [13, 23, 22]. Very few efforts have been spent on decoding phase. Pre-defined similarity or distance functions are usually applied directly to multi-class decoding.

In this work, we propose a learning-based adaptive decoding method. There are several practical advantages for conducting problem-dependent learning in coding space.

Firstly, learning-based adaptive decoding helps to refine coding for better generalization performance. Most of the existing coding schemes are pre-defined and only generate binary or ternary codes. It constrains the coding space in which later binary learning works. Hence, the learnability is restrained. The adaptive decoding presented in this work learns continuous codes which refine coding based on binary learning outcomes.

Secondly, one of the main drawbacks of pre-defined decoding schemes is that binary classifiers are trained on different tasks separately. There is no guarantee that their real-valued outcomes are properly scaled for pre-defined decoding. Moreover, nonlinear relationships between binary classifiers are usually neglected [24], which can be crucial for better generalization performance. As elaborated later, the proposed method can tackle these via learning and kernelization technique.

Thirdly, learning in coding space in turn helps to improve binary classifier. This is beneficial to tackling some difficult machine learning problems. As an extension, we show that our learning-based decoding method benefits multi-class hypothesis transfer learning (HTL) [10, 11, 19]. HTL

*NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

is a transfer learning framework that only utilizes source hypotheses trained on a source domain for enhancing target domain learning. It assumes that neither source domain examples nor the relationship between target domain and source domain is accessible. Such settings mimic real world situations in which legacy binary classifiers exist but the datasets on which they are trained have already become unavailable.

By performing learning in coding space, auxiliary source domain hypotheses can be readily incorporated into learning process. The knowledge can be transferred from multiple source domains to multiple target domains by alternatively refining coding and target domain classifiers. Unlike previous HTL approaches, the proposed method considers the relationship between domains by adopting the output coding framework. It helps to discover related target domain and source hypothesis. It also prevents negative transferring from unrelated source hypothesis.

Additional to the aforementioned virtues, the proposed approach extends ramp loss [5, 4] to measure multi-class decoding error. Unlike hinge loss which is prone to outliers, ramp loss provides a clipped error measurement capping the linear penalty increment, the source of the outlier sensitivity. Besides, it helps to reduce the number of support vectors, which are critical to scalability. Uniform stability and a tight generalization bound are held by the proposed algorithm.

In order to optimize ramp loss, a non-convex loss, ConCave-Convex Procedure (CCCP) [21] which is closely related to Difference of Convex (DC) programming [18] is utilized.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 describes the proposed method and gives its generalization error bound. The multi-class HTL is given in Section 4. Experiments are conducted in Section 5. Finally, the work is concluded by Section 6.

2. Related Work

Output coding framework [1, 7] converts a multi-class classification problem into a set of binary problems which can be readily solved by existing binary classification approaches. The quality of the coding matrix plays a critical role for the performance of the final solution. In early works [1], the error-correcting ability is the main measurement for the quality of the coding matrix. In other words, the codewords for different classes need to be as distinguishable as possible for a good performance. But later, it is realized that such coding strategy does not lead to superior performance comparing with simple strategies, such as random-half, one-versus-one and one-versus-all. It is because the most distinguishable coding matrix may not be learnable for binary classification algorithms [13, 22]. The optimal coding should be both distinguishable and learnable [25]. In order to find a balance between them, approaches have been developed for considering binary learning and coding simultaneously [13, 23, 22].

However, all of these approaches work on binary coding or ternary coding [1]. Decoding is achieved by simply applying a pre-defined similarity function. The work in [8] provides a good summary for decoding methods. As mentioned before, the coding space is dramatically constrained by current approaches. So in this paper, we propose to perform learning in coding space with continuous codes.

Several methods [10, 11, 19] have been developed for HTL. Specifically, [19] developed a multi-model knowledge transfer approach (Multi-KT) based on least square SVM. It can properly select and weight prior knowledge coming from different source domains. [10] proposed a multiple kernel transfer learning algorithm (MKTL). By adopting multiple kernel learning framework, it takes advantage of priors built over different features and with different learning methods. [11] developed a MULticlass Transfer Incremental LEarning (MULTIpLE) method. It also adopted least square SVM. It seeks for a balance between transferring to the new class and preserving what has already been learned on the source models.

3. Stable Learning for Multi-class Decoding

We review the output coding framework first. Let $S = \{z_i\}_{i=1}^m$ denote a set of m training examples, where $z_i = (x_i, y_i)$, $x_i \in \mathcal{X}$ is a feature vector and $y_i \in \mathcal{Y}$ is a class label. We usually have $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, \cdots, k\}$. The learning task is to learn a classifier: $\mathcal{G}(x) : \mathcal{X} \to \mathcal{Y}$ which can accurately assign a class label to an unseen example. In output coding framework, the classifier $\mathcal{G}$ is obtained from an ensemble of binary classifiers $\mathcal{H}(x) = [h_1(x), \cdots, h_l(x)]$ with the aid of a coding matrix M which has k rows and l columns. Each row of the coding matrix, M_y, represents a code word for the corresponding class y. Each column of the coding matrix represents a binary partition over all the classes. It defines a binary classification problem over the training examples, for which a binary classifier of the ensemble is trained. The length of codes equals to the number of binary classifiers and is denoted by l. The decoding process performs via a similarity metric function $f(\mathcal{H}(x), M_y) : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$. Then $\mathcal{G}$ can be represented as: $y^* = \arg\max_{y'} f(\mathcal{H}(x), M_{y'})$. $y' \in \{1, \cdots, k\}$ denotes a class label. The class whose code word is the closest to the ensemble output is predicted as the label of the input.

Traditional decoding process is conducted by directly applying a predefined similarity or distance measure on binary classifiers' outputs and code words. For example, Hamming distance is widely adopted for multi-class decoding: $\sum_{j=1}^{l} \frac{1 - M_{y',j} h_j(x)}{2}$, where $M_{y',j}$ represents the element in y'-th row and j-th column of the coding matrix M.

Alternatively, we suggest that a problem-dependent decoding metric function can be learned from examples. For instance, a general inner product function can be used:

$$f(\mathcal{H}, M_{y'}) = \mathcal{H}^T W_{y'} M_{y'} = \sum_{j=1}^{l} w_j h_j(x) M_{y',j}, \quad (1)$$

where $W_{y'}$ is a diagonal weighting matrix with $\{w_1, \cdots, w_l\}$ as its elements. $W_{y'}$ is the function parameter that needs to be learned from examples. It is noted that, with each class y' having a weighting matrix $W_{y'}$, the original coding matrix M is not necessary to be known for learning or applying such decoding function. The learning for decoding process can be regarded as to refine the coding matrix by learning an adaptive decoding matrix for the obtained binary classifier ensemble without necessarily knowing the coding matrix. Thus, the decoding problem becomes to learn an adaptive decoding matrix $W \in \mathbb{R}^{k \times l}$ for the decoding metric function;

$$f(\mathcal{H}, W_{y'}) = \mathcal{H}^T W_{y'}, \quad (2)$$

where $W_{y'}$ now represents the y'-th row of W. It can be shown that such metric function is equivalent to Hamming distance when the elements of $\mathcal{H}$ and W are in $\{-1, +1\}$. Other options for designing decoding metric function certainly exist and are worth for further study, *e.g.*, Mahalanobis distance and manifold distance. However, an obvious and crucial advantage of using inner product is that it allows us to make use of kernel trick readily. By doing so, the lack of the consideration of the nonlinear relationships between binary classifiers in traditional output coding framework can be easily tackled. It provides us the possibility to design proper kernels for modeling the correlations of binary classifiers.

It is also worth noting that, unlike traditional binary [1] or ternary [8] coding matrix, the decoding matrix in Eq. 2 can have real-valued codes [6]. For binary coding, confusing classes, which can not be confidently colored into any of the two classes, often appears and degrades the performance. Therefore, neutral value 0 is introduced into the partition scheme of ternary coding to ignore confusing classes. Continuous coding goes one step further. It provides the flexibility to fine partition classes with weights. Besides, it helps to scale binary classifiers properly for decoding. For instance, despite its advantages, the outputs of the binary classifiers learned from one-versus-all strategy are not guaranteed to be properly scaled.

3.1. The Proposed Method

We now describe the proposed decoding algorithm. Given an ensemble of binary classifiers $\mathcal{H}(x) = [h_1(x), \cdots, h_l(x)]$ and a set of training examples $S = \{(x_i, y_i)\}_{i=1}^m$ [1], the goal is to design an algorithm $\mathcal{A}$ which can learn an effective decoding metric function $f(x, y') : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ from S:

$$f(x, y') = \mathcal{H}(x)^T W_{y'} = \sum_{j=1}^{l} W_{y',j} h_j(x). \quad (3)$$

Decoding matrix $W \in \mathbb{R}^{k \times l}$ is the function parameter need to be learned. The final decision is made via $\mathcal{G}(x) : y^* = \arg\max_{y'} f(x, y')$. The margin of an example (x, y) for class y' with respect to f can be defined as:

$$\rho_f(x, y') = f(x, y) - f(x, y'), \quad (4)$$

where y is the true label of x. It is noted that $\rho_f(x, y) = 0$.

We use $\epsilon(\mathcal{A}_S, z)$ to represent the loss of $\mathcal{A}_S$, which denotes the solution of $\mathcal{A}$ on S, with respect to an example $z = \{x, y\}$ in the domain $\mathcal{Z} = \{\mathcal{X}, \mathcal{Y}\}$. Such loss is usually measured by a loss function $\ell(f, z) : \mathcal{F} \times \mathcal{Z} \to \mathbb{R}^+$. It represents the loss of a decoding metric function f, which is generated by $\mathcal{A}$ on S, with respect to an example z. $\mathcal{F}$ is generally a linear space.

As an estimation of generalization error, the empirical error of algorithm $\mathcal{A}$ on S is usually measured by 0-1 loss:

$$\begin{aligned} R_{emp}(\mathcal{A}, S) &= \frac{1}{m} \sum_{i=1}^{m} \epsilon(A_S, z_i) = \frac{1}{m} \sum_{i=1}^{m} \ell_{0-1}(f, z_i) \\ &= \frac{1}{m} \sum_{i=1}^{m} \delta(\mathcal{G}(x_i) \neq y_i), \end{aligned} \quad (5)$$

where $\delta(\pi)$ equals to 1 when the predicate π is true and 0 otherwise. From the study on classification and regression problems, we know that minimising 0-1 loss is computational expensive due to its combinatorial nature. Thus, with the consideration of a trade-off between statistical and numerical properties of learning loss, a surrogate loss is usually adopted as a substitute of 0-1 loss. A popular surrogate loss is hinge loss $\ell^1_{hinge}(\rho) = \max(0, 1 - \rho)$ where ρ denotes margin. Although hinge loss has a nice numerical property: convexity, which leads to efficient optimization, its statistical property has flaws. Its loss penalty is linear proportional to margin. This makes hinge loss prone to outliers. Moreover, hinge loss treats any training examples within margin or misclassified as support vectors (SVs). This usually leads to a large number of SVs, which makes both training and testing expensive. It becomes worse when large scale data set is involved, since the number of SVs increases linearly with respect to the number of training examples.

In order to avert the drawbacks of hinge loss, we propose to extend ramp loss [5] as a surrogate for measuring

[1] We keep using the same symbols to denote training examples. But it is worth noting that the data set used for learning a decoding metric function can differ from the data set used for training binary classifiers.

empirical multi-class decoding error:

$$\begin{aligned}\ell_{ramp}^{(s_l,s_r)}(f,z) &= \min(1-\frac{s_l}{sr}, \max_{y'}\{b_{y,y'}^1 - \frac{\rho_f(x,y')}{s_r}\}) \\ &= \ell_{hinge\_1}^{s_r}(f,z) - \ell_{hinge\_\frac{s_l}{s_r}}^{s_r}(f,z) \\ &= \max_{y'}\{b_{y,y'}^1 - \frac{\rho_f(x,y')}{s_r}\} \\ &\quad - \max_{y'}\{b_{y,y'}^{\frac{s_l}{s_r}} - \frac{\rho_f(x,y')}{s_r}\},\end{aligned} \tag{6}$$

where $b_{y,y'}^1$ equals to 0 if $y = y'$, 1 otherwise, and similarly $b_{y,y'}^{\frac{s_l}{s_r}}$ equals to 0 if $y = y'$, $\frac{s_l}{s_r}$ otherwise. s_l and s_r represent left and right hinge points of ramp loss respectively. Its piece-wise form can be written as:

$$\ell_{ramp}^{(s_l,s_r)}(f,z) = \begin{cases} B & \text{if } \rho_f(x,y^*) \le s_l \\ b_{y,y^*}^1 - \frac{\rho_f(x,y^*)}{s_r} & \text{if } s_l \le \rho_f(x,y^*) \le s_r, \\ 0 & \text{if } \rho_f(x,y^*) \ge s_r \end{cases} \tag{7}$$

where $B = 1 - \frac{s_l}{sr}$ is the maximum output value of $\ell_{ramp}^{(s_l,s_r)}$.

With empirical error measured by the extended ramp loss, the proposed algorithm $\mathcal{A}$ can be defined as a regularized empirical error minimizer:

$$\mathcal{A}_S = \arg\min_{f\in\mathcal{F}} \left\{ \frac{1}{m}\sum_{i=1}^m \ell_{ramp}^{(s_l,s_r)}(f,z_i) + \frac{\lambda}{2}N(f) \right\} \tag{8}$$

$$\begin{aligned} &= \arg\min_{W\in\mathbb{R}^{k\times l}} \Bigg\{ \underbrace{\frac{\lambda}{2}|W|_F^2 + \frac{1}{m}\sum_{i=1}^m \ell_{hinge\_1}^{s_r}(f,z)}_{J_{convex}^S(W)} \\ &\quad \underbrace{-\frac{1}{m}\sum_{i=1}^m \ell_{hinge\_\frac{s_l}{s_r}}^{s_r}(f,z)}_{J_{concave}^S(W)} \Bigg\}, \end{aligned} \tag{9}$$

where $N(f)$ is a regularizer that measures the complexity of f and $|\cdot|_F$ represents Frobenius norm. It is noted that the learning objective can be decomposed to the summation of a convex component $J_{convex}^S(W)$ and a concave component $J_{concave}^S(W)$. Thus, the optimization problem defined by Eq. 9 can be solved by an iterative learning procedure: Concave-Convex procedure (CCCP) [21] which is closely related to Difference of Convex (DC) programming [18]. At each learning iteration, a convex upper bound can be found for the learning objective by replacing the concave component with its first order Taylor expansion at the current state. Then the learning problem reduces to a convex problem on which existing kernelization and optimization techniques can be applied. The learning procedure stops when convergence is reached.

3.2. Generalization Error Bound

Following the work in [3], we utilize concentration inequality (McDiarmid inequality [14] specifically) and algorithmic stability [3] to give a generalization error bound for the proposed decoding method.

We keep using $\mathcal{A}_S$ to denote the decoding metric that is learned by the proposed algorithm $\mathcal{A}$ from data set S. The generalization error is a random variable depending on S. It can be written as:

$$R(\mathcal{A},S) = \mathbb{E}_z[\epsilon(\mathcal{A}_S,z)], \tag{10}$$

where $\epsilon(\mathcal{A}_S,z)$ denotes the loss of a decoding metric f with respect to an example $z = \{x,y\}$. However, the $\mathbb{E}_z$ can not be calculated because the underlying distribution of z is unknown. Thus, empirical error on available data set S is usually utilized as an estimator for generalization error:

$$R_{emp}(\mathcal{A},S) = \frac{1}{m}\sum_{i=1}^m \epsilon(\mathcal{A}_S,z_i). \tag{11}$$

For the proposed method, multi-class ramp loss $\ell_{ramp}^{(s_l,s_r)}$, as defined in Eq. 7, is used to measure $\epsilon(\mathcal{A}_S,z)$. The goal of our analysis here is to give a bound for the difference between $R(\mathcal{A},S)$ and $R_{emp}(\mathcal{A},S)$, more precisely, a bound for: $\mathbb{P}_S[|R_{emp}(\mathcal{A},S) - R(\mathcal{A},S)| > \eta]$, where $\eta > 0$.

In the following, we briefly review the the analysis in [3] for generalization error bound given uniform stability. Then the uniform stability is shown to be hold by the proposed decoding algorithm.

Theorem 1 (McDiarmid Inequality [14]) Given a set of examples $S = \{z_i = (x_i,y_i)\}_{i=1}^m$, a modified set S^i defined as $S^i = \{z_1,\cdots,z_{i-1},z_i^{'},z_{i+1},\cdots,z_m\}$ and a measurable function $F : \mathbb{Z}^m \to \mathbb{R}$, if the following condition is satisfied:

$$\sup_{S\in\mathbb{Z}^m, z_i'\in\mathbb{Z}} |F(S) - F(S^i)| \le c_i,$$

then

$$\mathbb{P}_S[F(S) - \mathbb{E}_S[F(S)] \ge \eta] \le \exp(\frac{-2\eta^2}{\sum_{i=1}^m c_i^2}).$$

Definition 1 (Uniform Stability [3]) An algorithm $\mathcal{A}$ has uniform stability β if the following condition holds:

$$\forall S\in\mathbb{Z}^m, \forall i\in\{1,\cdots,m\}, \sup|\epsilon(\mathcal{A}_S,\cdot) - \epsilon(\mathcal{A}_{S^{\setminus i}},\cdot)| \le \beta.$$

$S^{\setminus i}$ denotes the dataset which excludes the i-th element from S.

With theorem 1 and the uniform stability defined as the above, the following theorem about generalization error bound has been given in [3].

Theorem 2 (Exponential Bounds with Uniform Stability [3]) Given an algorithm $\mathcal{A}$ having uniform stability β with respect to a loss function ℓ which satisfies $0 \le \ell(\mathcal{A}_S,z) \le B$ for all sets S, then for any $m \ge 1$ and any $\delta \in (0,1)$, the following bound holds with probability at least $1-\delta$ over the random draw of the sample S:

$$R \le R_{emp} + 2\beta + (4m\beta + B)\sqrt{\frac{\ln(1/\delta)}{2m}}.$$

Now we derive the following theorem about the uniform stability for ramp loss.

Theorem 3 Given (1) a reproducing kernel Hilbert space $\mathcal{E}$ with kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ satisfying $\forall x \in \mathcal{X}, K(x,x) \leq \kappa^2 < \infty$, (2) a loss function $\ell(f,z) : \mathcal{F} \times \mathcal{Z} \rightarrow [0,\infty)$ which is monotonic and $1/s_r$-Lipschitz in its first argument, (3) $\mathcal{A}$ is a symmetric algorithm generating a solution $f_S \in \mathcal{F}$ on training set S by optimizing:

$$f_S = \arg\min_{f\in\mathcal{F}} \left\{ R^{\ell}_{emp}(\mathcal{A}, S) + \frac{\lambda}{2}\|f\|^2_K \right\},$$

where $R^{\ell}_{emp} = \frac{1}{m}\sum_{i=1}^{m} \ell(f, z_i)$ and $\lambda > 0$, then $\mathcal{A}$ has uniform stability $\beta(m) = \frac{8\kappa^2}{{s_r}^2 \lambda m}$.

The proof for theorem 3 is deferred to supplementary material. The direct application of theorem 2 and theorem 3 can provides us the following generalization error bound for the proposed decoding algorithm.

Example 1 Given the conditions of theorem 3, the following generalization error bound holds for algorithm $\mathcal{A}$:

$$R(\mathcal{A}, S) \leq R^{ramp}_{emp}(\mathcal{A}, S) + \frac{16\kappa^2}{{s_r}^2 \lambda m} + \left(\frac{32\kappa^2}{{s_r}^2 \lambda} + B \right) \sqrt{\frac{\ln 1/\delta}{2m}}.$$

Remark 1 The above gives a tight bound for the algorithm defined by Eq. 12 since β scales as $\frac{1}{m}$. From this bound, we can see that a larger s_r leads to a faster learning rate. Similar conclusions for structured estimation and clipped regularized risk minimizers are obtained in [4]. However, larger s_r also leads to a denser classifier which has more SVs [5]. For $B = 1 - \frac{s_l}{s_r}$ and $s_l < s_r$, we can see that larger s_l leads to a smaller B which indicates a better bound. It is possible to set s_l in $[0, s_r)$. It gives a smaller B and a shorter SVs interval. However, miss-classified examples will never become SVs. And this will degrade the generalization performance. The later empirical study verifies this. Note that SVM with ramp loss does not have the above bound because of the bias item in its predict function. There is no explicit B for hinge loss to have the above bound.

4. Extension for Multi-class Hypothesis Transfer Learning

Given target domain examples $S = \{z_i\}_{i=1}^{m}$ and a collection of auxiliary source domain hypotheses $\Delta\mathcal{H}(x) = [h_{l+1}, \ldots, h_{l+n_{sh}}]$, where n_{sh} denotes the number of source hypotheses, the aim of the multi-class HTL is to learn a set of binary hypotheses $\mathcal{H}(x) = [h_1(x), \cdots, h_l(x)]$ in target domains and an adaptive decoding metric function $f(x, y') = {\mathcal{H}'}^T W_{y'}$, where $\mathcal{H}' = [h_1(x), \ldots, h_{l+n_{sh}}(x)]$, so that the classification performance for target domains is improved by transferring the knowledge in $\Delta\mathcal{H}(x)$.

It is noted that the decoding metric function $f(x, y')$ is now parameterized by both W and $\mathcal{H}(x)$. If we assume $\mathcal{H}(x)$ is a set of binary hypotheses in the form of $h_j(x) = w_j x + b_j$, where $j \in [1, l]$, then the extended multi-class HTL algorithm becomes:

$$\begin{aligned} \mathcal{A}_{S,\Delta\mathcal{H}} &= \arg\min_{f\in\mathcal{F}} \left\{ \frac{1}{m}\sum_{i=1}^{m} \ell^{(s_l,s_r)}_{ramp}(f, z_i) + \frac{\lambda}{2} N(f) \right\} \\ &= \arg\min_{W\in\mathbb{R}^{k\times(l+n_{sh})},\mathcal{H}} \left\{ \frac{1}{m}\sum_{i=1}^{m} \ell^{(s_l,s_r)}_{ramp}(f, z_i) \right. \\ &\quad \left. + \frac{\lambda}{2}(|W|^2_F + \sum_{j=1}^{l} \|w_j\|^2) \right\}. \end{aligned} \tag{12}$$

The global optimal solution is difficult to be obtained if it is not impossible, since both W and $\mathcal{H}(x)$ need to be optimized. So we propose to adopt an alternative optimization strategy: (i) for fixed $\mathcal{H}(x)$ optimize W, (ii) for fixed W optimize $\mathcal{H}(x)$. The two steps alternate until convergence or the maximum number of iterations is reached. Although there is no guarantee that the global optimum could be achieved, each optimization step decreases the overall objective function.

Step (i) can be readily solved by the proposed learning-based decoding method. For step (ii), simultaneously optimizing all the binary hypotheses is difficult. Inspired by random coordinate descent approach, we propose to optimize binary hypotheses cyclically in a random order until convergence or maximum number of iterations is reached.

Any existing coding scheme can be used for initializing the algorithm. Target domain classifiers are firstly learned based on the initial coding. Learning-based decoding is then conducted with the support of auxiliary source hypotheses. Subsequently, target domain classifiers are updated. The alternative optimization continues until convergence or the maximum number of iterations is reached.

5. Experiments

We perform 3 experiments to evaluate the performance of the proposed method. The first experiment is conducted for comparing our learning-based decoding method with the state-of-the-art decoding approaches. The second experiment is to evaluate the impact of ramp loss with different parameter settings for verifying our theoretical analysis. The last experiment is performed for comparing our multi-class HTL method with the state-of-the-art HTL algorithms.

Unless otherwise mentioned, the following methodology is adopted for the experiments. The model parameters are selected via cross-validation for the best generalization performance. For the compared state-of-the-art techniques, default and optimized parameters given by the authors are used. 10-fold cross-validation is applied to acquire average accuracy and standard deviation for performance measurement.

		Glass	Vowel	Balance	Satimage	Letter	Vehicle
HD	SVM	55.75 ± 3.60	65.73 ± 2.62	85.57 ± 4.18	83.36 ± 2.02	85.11 ± 0.97	76.12 ± 2.40
	Boost	66.69 ± 3.16	61.30 ± 2.67	78.97 ± 5.02	83.25 ± 2.23	88.31 ± 1.58	72.34 ± 3.34
PD	SVM	57.04 ± 3.59	66.37 ± 2.69	83.36 ± 4.09	80.90 ± 1.74	77.82 ± 1.01	76.00 ± 1.45
	Boost	64.35 ± 2.72	58.48 ± 3.02	82.22 ± 4.19	83.90 ± 1.82	87.65 ± 2.01	72.58 ± 2.95
LAP	SVM	57.73 ± 3.14	68.40 ± 2.94	85.57 ± 4.18	83.36 ± 2.02	88.89 ± 1.05	76.12 ± 2.40
	Boost	66.69 ± 3.16	65.36 ± 2.17	80.15 ± 4.01	84.15 ± 2.22	90.12 ± 1.81	72.34 ± 3.34
ELW	SVM	59.30 ± 3.16	71.44 ± 3.16	85.57 ± 4.18	84.07 ± 2.00	89.44 ± 0.97	76.95 ± 1.76
	Boost	66.69 ± 3.16	71.77 ± 3.02	77.55 ± 4.46	85.37 ± 1.87	91.92 ± 1.58	73.15 ± 3.16
Prpsd.	SVM	$\mathbf{63.06 \pm 2.67}$	$\mathbf{74.59 \pm 2.39}$	$\mathbf{88.76 \pm 1.56}$	$\mathbf{90.22 \pm 1.02}$	$\mathbf{95.32 \pm 1.20}$	$\mathbf{79.87 \pm 1.49}$
	Boost	$\mathbf{68.89 \pm 2.71}$	$\mathbf{74.33 \pm 2.78}$	$\mathbf{83.09 \pm 2.78}$	$\mathbf{91.08 \pm 2.27}$	$\mathbf{96.77 \pm 1.26}$	$\mathbf{76.29 \pm 2.80}$

Table 1. Average performances (in percentage) of the state-of-the-art and the proposed decoding methods on six multi-class data sets from UCI. The best results are shown in bold.

5.1. Comparison with Previous Decoding Methods

We firstly conduct the comparison experiment between the state-of-the-art decoding methods and the proposed learning-based decoding algorithm. As shown in Table 1, six multi-class datasets from UCI Machine Learning Repository database [2] are used for the comparison. The compared decoding methods include Hamming decoding, probabilistic-based decoding [8], Laplacian decoding [8], exponential loss-weighted decoding [8] with continuous classifiers' outputs. Similar to the settings in [8], SVM and Adaboost (40 runs of decision stumps) are used as binary classifiers. RBF kernel is used for the proposed decoding method. Seven popular coding schemes are considered, including one-versus-one, one-versus-all, dense random, sparse random, DECOC [17] and ECOC-ONE [16]. For each decoding method, only the best coding result is reported. The comparison results are shown in Table 1. We can see that the proposed decoding method outperforms other decoding methods. Large improvements are gained for datasets $Letter$ and $Satimage$ which have a larger number of training examples compared with the others.

5.2. The Impact of Ramp Loss Settings

The second experiment is conducted to evaluate the impact of ramp loss with different parameter settings. The data used for this experiments is 15-scene image recognition data set [12]. There are 4485 images in total. 100 images are selected from each category for training. The remaining 2985 images are used for testing. 10-fold cross validation is applied. For visual description, we use spatial Principal component Analysis of Census Transform histograms (sPACT) [20, 9, 22] as feature.

The result is shown in Fig. 1. The left plot shows the changes of the number of SVs as a function of the number of training examples. It can be seen that, for Hinge loss, namely $\text{Ramp}_{(-\infty,1)}$, the number of SVs increases linearly with the number of training examples. It becomes sublinear when Ramp loss is used. We can observe that better scalability (less SVs) is achieved when s_l increases. Larger s_l leads to a shorter SVs internal. Subsequently, the number

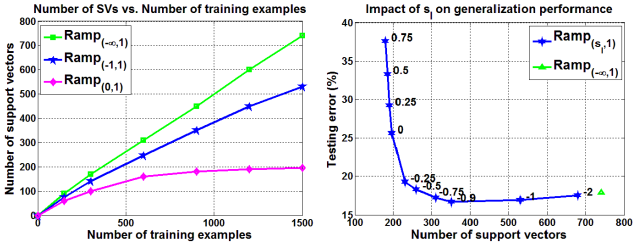

Figure 1. Impacts of s_l on learning scalability and generalization performance.

of SVs is dramatically reduced. It indicates that Ramp loss give us the ability to control the scalability of the algorithm by controlling the length of SVs interval.

The right plot in Fig. 1 shows the impact of s_l on both the generalization performance and scalability. Two trends can be observed: (1) At the early stage, the number of SVs increases as s_l decreases, and the generalization error is reduced correspondingly. However, the generalization error starts to increase when further decrement is put on s_l after it reaches -0.9. (2) The number of SVs is rapidly decreased when s_l increases. But when s_r becomes larger than -0.9, the generalization performance also starts to drop rapidly. These trends verify our conjecture that the second hinge point s_l gives us the flexibility to control the balance between generalization performance and scalability. Depend on specific problem and its training data, an optimal s_l is possible to be obtained in terms of the optimal balance between computational cost and generalization performance gain.

5.3. Multi-class HTL for Object Recognition

The last experiment is conducted on Caltech-256 dataset for verifying the proposed multi-class HTL method. We use similar experiment setting as the one used in [10]. We choose 10 classes (bonsai, gorilla, horse, motorbikes, mushroom, segway, skateboard, skunk, snowmobile, sunflower) as target classes. Maximum 30 examples are selected from each class for training, and 50 example are used for testing. 20 classes (palm-tree, cactus, fern, hibiscus, bat, bear, leopards, zebra, dolphin, killer-whale, mountain-

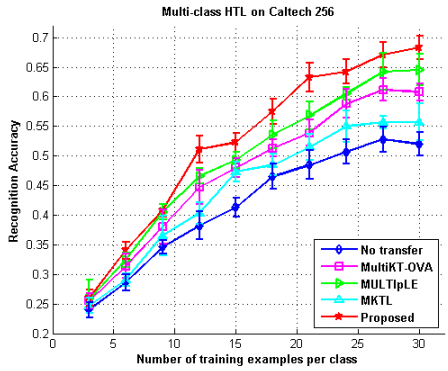

Figure 2. Recognition accuracy for Caltech-256 (10 target classes and 20 source classes) multi-class object categorization.

bike, fire-truck, car-side, bulldozer, camel, dog, raccoon, chimp, school-bus and touring-bike) are used as source classes. PHOG shape descriptor, SIFT appearance descriptor, Region Covariance and Local Binary Patterns are used as visual features for representing examples. One-versus-all strategy and SVM with RBF kernel are adopted for generating source domain hypotheses. For the proposed method, one-versus-all strategy is used for initializing target domain training. SVM with RBF kernel is used for training target domain classifiers. Balancing parameter λ and RBF kernel parameter γ are obtained by cross-validation.

We compare our method with three state-of-the-art HTL approaches: MKTL [10], MultiKT [19] and MULTIpLE [11]. Since MultiKT is a binary transfer learning approach, one-versus-all strategy is adopted for it to achieve multi-class HTL. Learning without knowledge transferring is also compared as a baseline. It adopts one-versus-rest strategy and only uses target domain examples for training classifiers. The experiment runs 10 times. Training and testing examples are randomly drawn from each category for each run. Recognition accuracies are averaged over runs and classes. The result is shown in Fig. 2 with the number of training examples per class increasing.

From Fig. 2, we can see that all the HTL approaches perform better than learning without knowledge transferring, and the proposed method consistently outperforms the other HTL approaches. The improvement becomes larger when the number of training examples per class increases. The reason is that, with more target domain training examples available, the proposed method improves not only target domain classifier learning but also coding space learning.

6. Conclusion

A learning-based decoding method is proposed for improving multi-class classification. It exploits ramp loss to measure multi-class decoding error and refines coding with real-valued codes. Both theoretical analysis and numerical results show its superiority over existing approaches. The method is further extended for multi-class HTL. Source domain hypotheses are exploited for leveraging target domain training via an alternative optimization process between coding space learning and target domain training. Experimental result shows that it outputs the state-of-the-art multi-class HTL approaches.

References

[1] E. L. Allwein and et al. Reducing multiclass to binary: A unifying approach for margin classifiers. *JMLR*, 1:113–141, 2001. 1, 2, 3

[2] A. Asuncion and D. Newman. Uci machine learning repository [http://www.ics.uci.edu/˜ mlearn/mlrepository. html]. university of california irvine. 2007. 6

[3] O. Bousquet and A. Elisseeff. Stability and generalization. *JMLR*, 2:499–526, 2002. 4

[4] O. Chapelle, C. Do, Q. Le, A. Smola, and C. Teo. Tighter bounds for structured estimation. *NIPS*, pages 281–288, 2008. 2, 5

[5] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML*, pages 201–208. ACM, 2006. 2, 3, 5

[6] K. Crammer and Y. Singer. Improved output coding for classification using continuous relaxation. *NIPS*, pages 437–443, 2001. 3

[7] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *JAIR*, 2(263):286, 1995. 1, 2

[8] S. Escalera and et al. On the decoding process in ternary error-correcting output codes. *TPAMI*, 32(1):120–134, 2010. 2, 3, 6

[9] G. Herman and et al. Region-based image categorization with reduced feature set. In *MMSP*, pages 586–591, 2008. 6

[10] L. Jie and et al. Multiclass transfer learning from unconstrained priors. In *ICCV*, pages 1863–1870, 2011. 1, 2, 6, 7

[11] I. Kuzborskij, F. Orabona, and B. Caputo. From n to n+ 1: Multiclass transfer incremental learning. In *CVPR*, 2013. 1, 2, 7

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006. 6

[13] L. Li. Multiclass boosting with repartitioning. In *ICML*, pages 569–576. ACM, 2006. 1, 2

[14] C. McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989. 4

[15] N. J. Nilsson. Learning machines. 1965. 1

[16] O. Pujol and et al. An incremental node embedding technique for error correcting output codes. *PR*, 41(2):713–725, 2008. 6

[17] O. Pujol, P. Radeva, and J. Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *TPAMI*, 28(6):1007–1012, 2006. 1, 6

[18] P. D. Tao et al. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133(1-4):23–46, 2005. 2, 4

[19] T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *CVPR*, pages 3081–3088, 2010. 1, 2, 7

[20] J. Wu and J. M. Rehg. Where am i: Place instance and category recognition using spatial pact. In *CVPR*, pages 1–8, 2008. 6

[21] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003. 2, 4

[22] B. Zhang and et al. Finding shareable informative patterns and optimal coding matrix for multiclass boosting. In *ICCV*, pages 56–63, 2009. 1, 2, 6

[23] B. Zhang and et al. Multi-class graph boosting with subgraph sharing for object recognition. In *ICPR*, pages 1541–1544, 2010. 1, 2

[24] B. Zhang and et al. Multilabel image classification via high-order label correlation driven active learning. *IEEE TIP*, 23(3), 2014. 1

[25] Y. Zhang and J. Schneider. Maximum margin output coding. *ICML*, 2012. 2