Sensitive Error Correcting Output Codes

John Langford[1] and Alina Beygelzimer[2]

[1] Toyota Technological Institute, Chicago IL 60637, USA
jl@tti-c.org
[2] IBM T. J. Watson Research Center, Hawthorne NY 10532, USA
beygel@us.ibm.com

Abstract. We present a reduction from cost-sensitive classification to binary classification based on (a modification of) error correcting output codes. The reduction satisfies the property that ϵ regret for binary classification implies l_2-regret of at most 2ϵ for cost estimation. This has several implications:

1. Any regret-minimizing online algorithm for 0/1 loss is (via the reduction) a regret-minimizing online cost-sensitive algorithm. In particular, this means that online learning can be made to work for arbitrary (i.e., totally unstructured) loss functions.
2. The output of the reduction can be thresholded so that ϵ regret for binary classification implies at most $4\sqrt{\epsilon Z}$ regret for cost-sensitive classification where Z is the expected sum of costs.
3. For multiclass problems, ϵ binary regret translates into l_2-regret of at most 4ϵ in the estimation of class probabilities. For classification, this implies at most $4\sqrt{\epsilon}$ multiclass regret.

1 Introduction

Background: The goal of classification is to predict labels on test examples given a set of labeled training examples. Binary classification, where the number of labels is two, is the most basic learning task as it involves predicting just a single bit for each example. Due to this simplicity, binary learning is (perhaps) better understood theoretically than any other prediction task, and several empirically good binary learning algorithms exist.

Practical machine learning, on the other hand, often requires predicting more than one bit. Furthermore, each prediction may generally have a different associated loss, and ignoring this information can make the problem more difficult.[3]

Motivation: Reductions allow us to translate performance on well-studied binary problems into performance on the more general problems arising in practice. We provide a reduction (called SECOC) from cost-sensitive classification to

[3] For example, consider the following problem: Given some relevant information, we must predict which of several routes to take. It is easy to design a problem where there are several paths that are typically good but occasionally very slow. If there is another path that is never the best but never slow, it may provide the best expected time of all choices. If the problem is altered into a multiclass problem of predicting the best route, this path will never be taken.

binary classification with the property that small regret on the created binary problem(s) implies small regret on the original cost-sensitive problem. This is particularly compelling because *any* loss function on single examples can be expressed with cost-sensitive classification. Therefore, this reduction can be used (at least theoretically) to solve a very broad set of learning problems. In addition, there is convincing empirical evidence that SECOC works well in practice, which gives further support to this style of analysis. Experiments in Section 7 show that SECOC results in superior performance on all tested multiclass learning algorithms and problems, compared to several other commonly used algorithms.

The basic SECOC reduction can be reused in several ways.

1. General Online Learning: Any regret-minimizing online algorithm for $0/1$ loss is (via the reduction) a regret-minimizing online cost sensitive algorithm. In particular, this means that online learning can be made to work for arbitrary (i.e., totally unstructured) loss functions.
2. Cost Sensitive Classification: The output of the reduction can be thresholded so that a small regret for binary classification implies a small regret for cost-sensitive classification. This implies that any consistent binary classifier is a consistent cost-sensitive classifier.
3. Multiclass Problems: Using the canonical embedding of multiclass classification into cost-sensitive classification, this reduction implies that small binary regret translates into small l_2 error in the estimation of class probabilities. By thresholding the estimates, we get a bound on multiclass regret. Note that this implies that *any* consistent binary classifier is (via the reduction) a consistent multiclass classifier. This is particularly important because generalization of SVMs to multiple classes have been done wrong, as shown in [8].

These applications are discussed in Section 4.

General Comment: It is important to understand that analysis here is orthogonal to the sample complexity analysis in, for example, PAC learning [12] or uniform convergence [14]. We consider measures over sets of examples and analyze the transformation of losses under mappings between these measures induced by the algorithms. This allows us to avoid making assumptions (which cannot be verified or simply do not hold in practice) necessary to prove sample complexity bounds. Instead, we show relative guarantees in an assumption-free setting – we bound the performance on general problems arising in practice in terms of performance on basic problems that are better understood.

Context: SECOC is a variation of error-correcting output codes (ECOC) [3]. Later, in Section 5, we show that this variation is necessary in order to satisfy a regret transform. The ECOC reduction works by learning a binary classifier, which decides membership of a label in subsets of labels. Given a sequence of subsets, each label corresponds to a binary string (or a *codeword*) defined by the inclusion of this label in the sequence of subsets. A multiclass prediction is made by finding the codeword closest in Hamming distance to the sequence of binary predictions on the test example.

For the ECOC reduction, a basic statement [5] can be made: with a good code, for all training sets, the error rate of the multiclass classifier on the training set is at most 4 times the average error rate of the individual binary classifiers. The proof of this statement is essentially the observation that there exist codes in which the distance between any two codewords is at least $\frac{1}{2}$. Consequently, at least $\frac{1}{4}$ of the classifiers must err to induce a multiclass classification error, implying the theorem.

This theorem can be generalized [2] to quantify "for all measures" rather than "for all training sets". This generalization is not as significant as it might at first seem, because the measure implicit in a very large training set can approximate other measures (and it can do so arbitrarily well when the feature space is finite). Nevertheless, it is convenient to quantify over all measures, so that the statement holds for the process generating each individual example. Since there is always some process generating examples, the result is applicable even to adversarial processes.

The weighted all pairs algorithm [2] intuitively guarantees that a small error rate on created classification problems implies a small cost-sensitive loss. The core result here is similar except that small binary *regret* implies small cost-sensitive *regret*. Regret is the error rate minus the minimum error rate. Consequently, the results here can have important implications even when, for example, the binary error induced from a multiclass problem is 0.25. SECOC does not supercede this result, however, because the regret bounds are weaker, roughly according to ϵ error rate going to $\sqrt{\epsilon}$ regret.

ECOC was modified [1] to consider margins of the binary classifiers—numbers internal to some classification algorithms that provide a measure of confidence in a binary prediction. Decoding proceeds in the same way as for ECOC except a "loss"-based[4] distance is used instead of the Hamming distance. Roughly speaking, SECOC uses the motivation behind this approach although not the approach itself. Instead of working with margins, we define binary classification problems, for which the optimal solution computes the relative expected cost (rather than the margin) of choices. This approach allows us to accomplish several things: First, we can use arbitrary classifiers rather than margin-based classifiers. We also remove the mismatch between the margin and the motivations. Optimizations of hinge loss (for SVMs) or exponential loss (for Adaboost) cause a distortion where the optimization increases the margin of small-margin examples at the expense of the margin on large-margin examples. The efficacy of Platt scaling [10] (i.e., fitting a sigmoid to a margin to get a probabilistic prediction) can be thought of as strong empirical evidence of the deficiency of margins as probability estimates. Finally, we can generalize the approach to tackle all cost-sensitive problems rather than just multiclass problems. This generalization comes at no cost in multiclass performance.

[4] "Loss" is in quotes because the notion of loss is a specification of the optimization used by the binary learning algorithm rather than the loss given by the problem, as is used in the rest of the paper.

2 The SECOC Reduction

We work in an assumption-free learning setting. The SECOC reduction reduces cost-sensitive classification to importance weighted binary classification, which in turn can be reduced to binary classification using the Costing reduction [16]. We first define all the problems involved.

Definition 1. *An* importance weighted binary classification problem *is defined by a measure D on a set $X \times \{0,1\} \times [0,\infty)$, where X is some arbitrary feature space, $\{0,1\}$ is the binary label, and $[0,\infty)$ is the importance of correct classification. The goal is to find a binary classifier $b : X \to \{0,1\}$ which minimizes the expected importance weighted loss, $E_{(x,y,i)\sim D}\left[iI(b(x) \neq y)\right]$, where $I(\cdot)$ is 1 when the argument is true and 0 otherwise.*

Cost-sensitive classification defined below is sufficiently general to express any loss function on a finite set.

Definition 2. *A* cost-sensitive k-class problem *is defined by a measure D on a set $X \times [0,\infty)^k$, where X is some arbitrary feature space, and the additional information $[0,\infty)^k$ is the cost of each of the k choices. The goal is to find a classifier $h : X \to \{1,...,k\}$ which minimizes the expected cost, $E_{(x,\mathbf{c})\sim D}\left[c_{h(x)}\right]$.*

A cost-sensitive learning algorithm typically takes as input a sequence of training examples in $(X \times [0,\infty)^k)^*$ as advice in constructing $h(x)$.

The SECOC reduction is a cost-sensitive learning algorithm that uses a given binary learning algorithm as a black box. As with the ECOC reduction, SECOC uses a code defined by an $n \times k$ binary coding matrix M with columns corresponding to multiclass labels. For example, the columns can form a subset of any k codewords of a Hadamard code of length n, which has the property that any two distinct codewords differ in at least $n/2$ bit positions. Such codes are easy to construct when k is a power of 2. Thus, for Hadamard codes, the number n of classification problems needed is less than $2k$.

For each subset s of labels, corresponding to a row of M, we create a set of importance weighted classification problems parameterized by $t \in [t_{\min}, t_{\max}]$, where $t_{\min} = 0$ and $t_{\max} = 1$ are always correct, but significant efficiency improvements arise from appropriately chosen smaller ranges. Intuitively, the problem defined by the pair (s,t) is to answer the question "Is the cost of s greater than t times the total cost?" From the optimal solution of these problems we can compute the expected relative cost of each subset s. SECOC-Train (Algorithm 1) has the complete specification.

We write E_s to denote an expectation over s drawn uniformly from the rows of M, and E_t to denote expectation over t drawn uniformly from the interval $[t_{\min}, t_{\max}]$.

To make a label cost estimate, SECOC-Predict (Algorithm 2) uses a formula of the expected prediction of the subsets containing the label.

Single Classifier Trick: Multiple invocations of the oracle learning algorithm B can be collapsed into a single call using a standard trick [2,1]. The

Algorithm 1 SECOC-Train (Set of k-class cost-sensitive examples S, importance weighted binary classifier learning algorithm B, range $[t_{\min}, t_{\max}]$ for t)

1. For each subset s defined by the rows of M:
   (a) For $(x, \boldsymbol{c}) \in S$, let $|\boldsymbol{c}| = \sum_y c_y$ and $c_s = \sum_{y \in s} c_y$.
   (b) For each t in $[t_{\min}, t_{\max}]$:
   Let $b_{st} = \mathrm{B}(\{(x,\ I(c_s \geq t|\boldsymbol{c}|), |c_s - |\boldsymbol{c}|t|) : (x,\ \boldsymbol{c}) \in S\})$.
2. return $\{b_{st}\}$

Algorithm 2 SECOC-Predict (classifiers $\{b_{st}\}$, example $x \in X$, label y)

return $2\left(t_{\min} + (t_{\max} - t_{\min}) E_s E_t \left[I(y \in s) b_{st}(x) + I(y \notin s)(1 - b_{st}(x))\right]\right) - 1$

trick is just to augment the feature space with the name of the call, and then learn a classifier b on (a random subset of) the union of all training data. With this classifier, we can define $b_{st}(x) \equiv b(\langle x, s, t\rangle)$, and all of our results hold for this single invocation classifier. The implication of this observation is that we can view SECOC-Train as a machine that maps cost-sensitive examples to importance weighted binary examples. We denote the learned binary classifier by $B(\text{SECOC-Train}(S))$.

3 The Main Theorem

Before stating the theorem, we need to define loss and regret. Given any distribution D on examples $X \times \{0, 1\} \times [0, \infty)$, the importance weighted error rate of a binary classifier b is given by

$$e(D, b) = E_{(x,y,i)\sim D}\left[iI(b(x) \neq y)\right].$$

Similarly, given any distribution D on examples $X \times [0, \infty)^k$, the cost-sensitive loss of a multiclass classifier h is given by

$$e(D, h) = E_{(x,\boldsymbol{c})\sim D}\left[c_{h(x)}\right].$$

For each of these notions of loss, the regret is the difference between the achieved performance and best possible performance:

$$r(D, h) = e(D, h) - \min_{h'} e(D, h').$$

(Note that we mean the minimum over *all* classifiers h' not over some class.) The minimum loss classifier is also known as the "Bayes optimal classifier".

We must also define how SECOC transforms its distribution D into a distribution on the learned binary classifier. To draw a sample from this distribution, we first draw a cost-sensitive sample $(x, \boldsymbol{c})$ from D, and then apply SECOC-Train to the singleton set $\{(x, \boldsymbol{c})\}$ to get a sequence of importance weighted binary examples, one for each (s, t) pair. Now, we just sample uniformly from this set, adding the index in the sequence as a feature. We overload and denote the induced distribution by SECOC-Train(D).

Throughout the paper, for a cost vector $\boldsymbol{c} \in [0,\infty)^k$ and a subset of labels $s \subseteq \{1,\ldots,k\}$, let $c_s = \sum_{y \in s} c_y$. Also let $|\boldsymbol{c}| = \sum_{y=1}^{k} c_y$. The distribution $D|x$ is defined as D conditioned on x.

Theorem 1. (SECOC Regret Transform) *For all importance weighted binary learning algorithms B and cost-sensitive datasets S in $(X \times [0,\infty)^k)^*$, let $b = B(\text{SECOC-Train}(S))$. Then for all test distributions D on $X \times [0,\infty)^k$, for all labels $y \in \{1,\ldots,k\}$:*

$$E_{(x,\boldsymbol{c})\sim D}\left(\text{SECOC-Predict}(b,x,y)E_{\boldsymbol{c}'\sim D|x}\left[|\boldsymbol{c}'|\right] - E_{\boldsymbol{c}'\sim D|x}\left[c'_y\right]\right)^2$$

$$\leq 8(t_{\max} - t_{\min})^2 r(\text{SECOC-Train}(D), b),$$

where $t_{\max} = \max_{(x,\boldsymbol{c}):D(x,\boldsymbol{c})>0} \max_s (c_s/|\boldsymbol{c}|)$ *and* $t_{\min} = \min_{(x,\boldsymbol{c}):D(x,\boldsymbol{c})>0} \min_s (c_s/|\boldsymbol{c}|)$.

This theorem relates the average regret of the created binary importance weighted classifier to the relative estimation error.

For the proof, note that the dependence on B and S can be removed by proving the theorem for all b, which is of equivalent generality.

Proof. We first analyze what happens when no regret is suffered, and then analyze the case with regret. For any s and t, let $D(s,t)$ be the distribution on $X \times \{0,1\} \times [0,\infty)$ induced by drawing $(x,\boldsymbol{c})$ from D and outputting $(x, I(c_s \geq t|\boldsymbol{c}|), |c_s - t|\boldsymbol{c}||)$.

For any choice of s and t, the optimal classifier is given by

$$b^*_{st} = \arg\min_b E_{(x,y,i)\sim D(s,t)}\left[iI(b(x) \neq y)\right]$$

$$= \arg\min_b E_{(x,\boldsymbol{c})\sim D}\left[|c_s - |\boldsymbol{c}|t| \cdot I\left(b(x) \neq I(c_s \geq t|\boldsymbol{c}|)\right)\right].$$

For any x, the optimal value of $b(x)$ is either 0 or 1. When it is 0, the expected cost is

$$E_{\boldsymbol{c}\sim D|x} \max\{(c_s - t|\boldsymbol{c}|), 0\}. \tag{1}$$

Otherwise, it is

$$E_{\boldsymbol{c}\sim D|x} \max\{(t|\boldsymbol{c}| - c_s), 0\}. \tag{2}$$

To simplify notation, let $Z_x = E_{\boldsymbol{c}\sim D|x}|\boldsymbol{c}|$. Equations 1 and 2 are continuous in t; the first decreases while the second increases monotonically with t, so we need only find the single equality point to describe the optimal behavior for all t. This equality point is given by

$$E_{\boldsymbol{c}\sim D|x} \max\{(c_s - t|\boldsymbol{c}|), 0\} = E_{\boldsymbol{c}\sim D|x} \max\{(t|\boldsymbol{c}| - c_s), 0\},$$

or

$$E_{\boldsymbol{c}\sim D|x}(c_s - t|\boldsymbol{c}|) = 0,$$

yielding

$$t = \frac{E_{\boldsymbol{c}\sim D|x}\left[c_s\right]}{Z_x},$$

and thus $b^*_{st}(x) = I(E_{\boldsymbol{c}\sim D|x}\left[c_s\right] \geq tZ_x)$.

For any choice of s, we have

$$E_t\left[b^*_{st}(x)\right] = E_t I\left(E_{\boldsymbol{c}\sim D|x}\left[c_s\right] \geq tZ_x\right) = \frac{\frac{E_{\boldsymbol{c}\sim D|x}[c_s]}{Z_x} - t_{\min}}{t_{\max} - t_{\min}}$$

since $E_{t\in[t_{\min},t_{\max}]} I(K \geq t) = \frac{K - t_{\min}}{t_{\max} - t_{\min}}$ for all $K \in [t_{\min}, t_{\max}]$.

Since decoding is symmetric with respect to all labels, we need analyze only one label y. Furthermore, since SECOC-Predict (Algorithm 2) is symmetric with respect to set inclusion or complement set inclusion, we can assume that y is in every subset (i.e., complementing all subsets not containing y does not change the decoding properties of the code.) Consequently,

$$\begin{aligned}\hat{c}_y = E_s E_t\left[b^*_{st}(x)\right](t_{\max} - t_{\min}) + t_{\min} &= E_s \frac{E_{\boldsymbol{c}\sim D|x}\left[c_s\right]}{Z_x} \\ = \frac{\frac{1}{2}E_{\boldsymbol{c}\sim D|x}(c_y + |\boldsymbol{c}|)}{Z_x} &= \frac{1}{2}\left(\frac{E_{\boldsymbol{c}\sim D|x}\left[c_y\right]}{Z_x} + 1\right),\end{aligned}$$

where the third equality follows from the fact that every label other than y appears in s half the time in expectation over s. Consequently, SECOC-Predict outputs $\frac{1}{Z_x}E_{\boldsymbol{c}\sim D|x}\left[c_y\right]$ for each y, when the classifiers are optimal.

Now we analyze the regret transformation properties. The remainder of this proof characterizes the most efficient way that any adversary can induce estimation regret with a fixed budget of importance weighted regret.

Examining equations 1 and 2, notice that the importance weighted loss grows linearly with the distance of tZ_x from $E_{\boldsymbol{c}\sim D|x}\left[c_s\right]$, but on the other hand, each error has equal value in disturbing the expectation in SECOC-Predict (Algorithm 2). There are two consequences for an adversary attempting to disturb the expectation the most while paying the least importance weighted cost.

1) It is "cheapest" for an adversary to err on the t closest to $\frac{1}{Z_x}E_{\boldsymbol{c}\sim D|x}\left[c_s\right]$ first. (Any adversary can reduce the importance weighted regret by swapping errors at larger values of $|t - \frac{1}{Z_x}E_{\boldsymbol{c}\sim D|x}\left[c_s\right]|$ for errors at smaller values without altering the estimation regret.)

2) It is "cheapest" to have a small equal disturbance for each s rather than a large disturbance for a single s. (The cost any adversary pays for disturbing the overall expectation can be monotonically decreased by spreading errors uniformly over subsets s.)

Consequently, the optimal strategy for an adversary wanting to disturb the output of SECOC-Predict by Δ is to disturb the expectation for each s by $\frac{\Delta}{2(t_{\max}-t_{\min})}$. The importance weighted regret of erring (with a "1") for $t = \frac{\Delta}{2(t_{\max}-t_{\min})} + \frac{1}{Z_x}E_{\boldsymbol{c}\sim D|x}\left[c_s\right]$ can be found by subtracting equation 2 from equation 1:

$$E_{\boldsymbol{c}\sim D|x}(t|\boldsymbol{c}| - c_s)I(c_s < t|\boldsymbol{c}|) - E_{\boldsymbol{c}\sim D|x}(c_s - t|\boldsymbol{c}|)I(c_s \geq t|\boldsymbol{c}|)$$

$$= E_{\mathbf{c}\sim D|x}\left(\left(\frac{\Delta}{2(t_{\max}-t_{\min})}+\frac{E_{\mathbf{c}\sim D|x}\left[c_s\right]}{Z_x}\right)|\mathbf{c}|-c_s\right)$$

$$= \frac{\Delta}{2(t_{\max}-t_{\min})}Z_x.$$

The same quantity holds for $t=-\frac{\Delta}{2(t_{\max}-t_{\min})}+\frac{E_{\mathbf{c}\sim D|x}[c_s]}{Z_x}$. By observation (1) above, in order for the adversary to induce an estimation error of Δ an error must occur for every $t\in\left[\frac{E_{\mathbf{c}\sim D|x}[c_s]}{Z_x},\frac{E_{\mathbf{c}\sim D|x}[c_s]}{Z_x}+\frac{\Delta}{2(t_{\max}-t_{\min})}\right]$. If we consider a limit as the discretization of t goes to 0, the average regret is given by an integral of the differential regret according to:

$$\int_{u=0}^{u=\frac{\Delta Z_x}{2(t_{\max}-t_{\min})}} u\,du=\frac{\Delta^2 Z_x^2}{8(t_{\max}-t_{\min})^2}.$$

Solving for $\Delta^2 Z_x^2$ and taking an expectation over all x gives the theorem. □

4 Applications and Corollaries

In this section we discuss various uses of SECOC to solve problems other than relative cost estimation and corollaries of the main theorem.

4.1 Reduction all the way to Classification

The basic SECOC reduction above reduces to importance weighted binary classification. However, there are easy reductions from importance weighted binary classification to binary classification. For example, the Costing reduction [16] uses rejection sampling to alter the measure. When SECOC is composed with this reduction we get the following corollary:

Corollary 1. (SECOC Binary Regret Transform) *For any importance weighted binary learning algorithm B and cost-sensitive dataset S in $(X\times[0,\infty)^k)^*$, let $b=B(\text{Costing}(\text{SECOC-Train}(S)))$. Then for all test distributions D on $X\times[0,\infty)^k$ and all labels $y\in\{1,\ldots,k\}$:*

$$E_{(x,\mathbf{c})\sim D}\left(\text{SECOC-Predict}(b,x,y)E_{\mathbf{c}'\sim D|x}\left[|\mathbf{c}'|\right]-E_{\mathbf{c}'\sim D|x}\left[c'_y\right]\right)^2$$

$$\le 4(t_{\max}-t_{\min})r(\text{Costing}(\text{SECOC-Train}(D)),b)E_{(x,\mathbf{c})\sim D}\left[|\mathbf{c}|\right]$$

Proof. The basic result from importance weighted analysis is that for every importance weighted test measure D, we have

$$r(D,b)=r(\text{Costing}(D),b)E_{(x,y,i)\sim D}\left[i\right],$$

where the regret on the left is the importance weighted regret of b with respect to D, and the regret on the right is the regret with respect to the induced binary distribution.

Algorithm 3 SECOC-Hard-Predict (classifiers $\{b_{st}\}$, example $x \in X$)

return $\arg\min_y$ SECOC-Predict($\{b_{st}\}, x, y$)

Consequently, we need only compute an upper bound on the average importance over t and s. The average importance for fixed x and s is given by

$$\frac{1}{t_{\max} - t_{\min}} \int_{t_{\min}}^{t_{\max}} E_{\boldsymbol{c} \sim D|x} \left| t|\boldsymbol{c}| - c_s \right| dt.$$

This quantity is maximized (over all x and s) when $E_{\boldsymbol{c} \sim D|x}[c_s] = 0$. In this case the integral is $\frac{(t_{\max}-t_{\min})^2}{2(t_{\max}-t_{\min})} E_{\boldsymbol{c} \sim D|x}|\boldsymbol{c}| = \frac{(t_{\max}-t_{\min})}{2} E_{\boldsymbol{c} \sim D|x}|\boldsymbol{c}|$. Taking the expectation over x and s, we get the corollary. □

4.2 Cost Sensitive Classification

If we use the decoding function SECOC-Hard-Predict in Algorithm 3, we can choose a class in a regret transforming manner.

Corollary 2. (Hard Prediction Regret Transform) *For any importance weighted binary learning algorithm B and multiclass dataset S in $(X \times \{1, ..., k\})^*$, let $b = B(\text{SECOC-Train}(S))$. Then for all test distributions D over $X \times \{1, .., k\}$:*

$$r\left(D, \text{SECOC-Hard-Predict}(b, x)\right) \le 4(t_{\max} - t_{\min}) \sqrt{2r(\text{SECOC-Train}(D), b)}$$

Proof. We can weaken Theorem 1 so that for all y:

$$E_{(x,y)\sim D} \left| \text{SECOC-Predict}(b, x, y) E_{\boldsymbol{c}' \sim D|x} \left[|\boldsymbol{c}'| \right] - E_{\boldsymbol{c}' \sim D|x} \left[c'_y \right] \right|$$

$$\le 2(t_{\max} - t_{\min}) \sqrt{2r(\text{SECOC-Train}(D), b)}$$

since for all X, $\sqrt{E(X)} \ge E\sqrt{X}$. When doing a hard prediction according to these outputs, our regret at most doubles because the relative cost estimate of the correct class can be increased by the same amount that the relative cost estimate of the wrong class can be decreased. □

4.3 Multiclass Probability Estimation

SECOC can be used to predict the probability of class labels with the training algorithm PECOC-Train (Algorithm 4) for any k a power[5] of 2. Similarly, the prediction algorithm PECOC-Predict (Algorithm 5) is a slight modification of SECOC-Predict (Algorithm 2).

[5] This limitation is not essential. See Section 6.

Algorithm 4 PECOC-Train (Set of k-class multiclass examples S, importance weighted binary classifier learning algorithm B)

1. Let $S' = \{(x, \forall i\ c_i = I(i \neq y)) : (x,y) \in S\}$.
2. return SECOC-Train $\left(S', B, \left[\frac{\lfloor \frac{k}{2} \rfloor - 1}{k-1}, \frac{\lfloor \frac{k}{2} \rfloor}{k-1}\right]\right)$

Algorithm 5 PECOC-Predict (classifiers $\{b_{st}\}$, example $x \in X$, label y)

return $1 -$ SECOC-Predict($\{b_{st}\}$, x, y)$(k-1)$

Corollary 3. (Multiclass Probability Regret Transform) *For any importance weighted binary learning algorithm B and a multiclass dataset S in $(X \times \{1, ..., k\})^*$ with k a power of 2, let $b = B(\text{PECOC-Train}(S))$. Then for all test distributions D over $X \times \{1, ..., k\}$ and all labels $y \in \{1, \ldots, k\}$,*

$$E_{(x,y)\sim D} \left(\text{PECOC-Predict}(b, x, y) - D(y|x)\right)^2 \leq \frac{8}{(k-1)^2} r(\text{PECOC-Train}(D), b).$$

This corollary implies that probability estimates (up to l_2 loss) are accurate whenever the classifier has small regret. When we reduce all the way to classification as in Corollary 1, the factor of $1/(k-1)^2$ disappears so the l_2 regret in class probability estimation is independent of the number of classes.

Proof. The proof just uses Theorem 1. In this case $|\boldsymbol{c}| = k - 1$, $E_{\mathbf{c}' \sim D|x}\left[c'_y\right] = 1 - D(y|x)$, and $t_{\max} - t_{\min} = \frac{1}{k-1}$.

$$E_{(x,y)\sim D} \left(\text{SECOC-Predict}(b, x, y)(k-1) - (1 - D(y|x))\right)^2$$
$$\leq \frac{8}{(k-1)^2} r(\text{PECOC-Train}(D), b).$$

Applying algebra finishes the corollary. □

4.4 Multiclass Classification

When a hard prediction is made with PECOC-Hard-Predict (Algorithm 6), we achieve a simple algorithm that translates *any* consistent binary classifier into a consistent multiclass classifier.

Corollary 4. (Multiclass Classification Regret Transform) *For any importance weighted binary learning algorithm B and multiclass dataset S in $(X \times \{1, ..., k\})^*$ with*

Algorithm 6 PECOC-Hard-Predict (classifiers $\{b_{st}\}$, example $x \in X$)

return $\arg\max_{y \in \{1,\ldots,k\}}$ PECOC-Predict($\{b_{st}\}, x, y$)

k a power of 2, let $b = B(\text{PECOC-Train}(S))$. For all test distributions D over $X \times \{1, ..., k\}$:

$$r\left(D, \text{PECOC-Hard-Predict}(b, x)\right) \leq \frac{4}{k-1}\sqrt{2r(\text{PECOC-Train}(D), b)}.$$

Note again that if we reduce to binary classification, the factor of $k-1$ is removed and the result is independent of the number of classes.

This guarantee can not be satisfied by ECOC (as we show in Section 5). A guarantee of this sort may be provable with other variants of ECOC (such as [1]), but this seems to be the tightest known regret transform. Since consistent generalization of binary classifiers to multiclass classifiers has historically been problematic (see [8] for a fix for SVMs), this result may be of interest.

Proof. The regret of a multiclass prediction is proportional to the difference in probability of the best prediction and the prediction made. Weakening corollary 3 gives, for all y,

$$E_{(x,y)\sim D}\left|\text{PECOC-Predict}(b, x, y) - D(y|x)\right| \leq \frac{2}{k-1}\sqrt{2r(\text{PECOC-Train}(D), b)}$$

since for all X, $\sqrt{E(X)} \geq E\sqrt{X}$. When doing a hard prediction according to these outputs, our regret at most doubles because the probability estimate of the correct class can be reduced by the same amount that the probability estimate of the wrong class increases. □

4.5 Online Learning and Loss

Notice that all basic transformations are applied to individual examples, as in line 1(b) of SECOC-Train. Consequently, the transformation can be done online. The theorems apply to any measure on $(x, \boldsymbol{c})$, so they also apply to the uniform measure over past examples; thus online regret minimizing binary predictors can be used with SECOC to minimize cost-sensitive regret online.

In particular, this means that SECOC can be used with online learning algorithms such as weighted majority [9] in order to optimize regret with respect to *any* loss function.

Note that the notion of regret in online learning is typically defined with respect to some set of "experts" rather than the set of all possible experts as here. This distinction is not essential, because the weighted majority algorithm can be applied to an arbitrary measure over the set of all experts.

5 ECOC Can Not Transform Regret

Is it possible to get similar guarantees with ECOC? The answer is no. It is easy to show that even when ECOC is supplied with an optimal binary classifier, the reduction fails to provide an optimal multiclass classifier.

Theorem 2. (ECOC Inconsistency) *For all $k > 2$, there exists a distributions D over multiclass test examples (x, y) such that for all codes M, with $c^* = \arg\min_c r(\text{ECOC-Train}(D), c)$,*

$$r(D, \text{ECOC-Predict}(c^*)) > \frac{1}{8}$$

where ECOC-Train *and* ECOC-Predict *are as defined in the introduction.*

Proof. The proof is constructive. We choose a D which places probability on three labels: '1', '2', and '3'.

A few observations about symmetry simplify the proof. First, since only three labels have positive probability, we can rewrite any code M as a new *weighted* code M' over the three labels where each subset has a weight w_s corresponding to the number of times the subset of the three labels exists in M after projection. The second observation is that the symmetry with respect to complementarity implies that each row (and each codeword) has one '1' in it.

These observations imply that ECOC essentially uses the binary classifier to ask, "Is the probability of label $i > 0.5$?" for each $i \in \{1, 2, 3\}$. These answers are then combined with a weighted sum. If we let the probability of one label be $0.5 - \epsilon$ and the probability of the other two labels be $0.25 + \frac{\epsilon}{2}$ each, the answer to every question will be "no".

Since we have a weighted sum, the exact weighting determines the outcome (possibly with randomization to break ties). The exact distribution therefore picks a label at random to have probability $0.5 - \epsilon$, encodes that choice in the x value, and then draws the label from this associated distribution.

Under any code, the probability of predicting the label with greatest probability is at most $\frac{1}{3}$ implying a regret of $\frac{2}{3}(0.5 - \epsilon - (0.25 + \frac{\epsilon}{2}))$, which can be made arbitrarily close to $\frac{1}{6}$. $\square$

A margin-based version of ECOC [1] has the same lower bound whenever the coding matrices are limited to "-1" and "1" entries. This is because consistent binary classifiers might have margin 1 or -1 for each example, and the proof above holds.

However, this version of ECOC also allows "don't cares" in the coding matrix. The existence of "don't cares" allows questions of the form, "Is the probability of this label greater than that label?" In general, these are sufficiently powerful to support regret transformation consistency, with the exact quantity of regret transformation efficiency dependent on the coding matrix. We are not aware of any margin based code with "don't cares" with a better regret transform than SECOC with the Hadamard Code.

Take as an example the all-pairs code, which is consistent with margin-based ECOC. The all-pairs code creates a classifier for every pair of classes deciding (for an optimal classifier) "Is class i more probable than class j?" The problem with this question is that the classifier is applied when class i and class j each have zero probability. In this situation, an adversarial classifier can choose to report either $p_i > p_j$ or $p_j > p_i$ without paying any regret. Consequently, an adversarial binary classifier could make some label with 0 conditional probability beat all labels except for the correct label for free. This is not robust, because one error in one classifier (out of $k-1$ active classifiers) can alter the result. Consequently, the regret transform for this code scales with k.

6 Discussion

Variants There are several variants of the basic SECOC algorithm.

Random Code One simple variant code is "pick a random subset s and pick a random t". This code has essentially the same analysis as the Hadamard code presented here in expectation over the random choices.

Optimal codes For small values of k (the number of classes), it is possible to derive a better regret transform with other codes. For example, when $k = 2$ there is only one useful subset (up to symmetry in complementation), so the prediction algorithm can simply output the cost estimate for that one subset rather than 2 times the average predicted cost, minus 1. This removes a factor of 2 loosening in the last paragraph of the proof of the main theorem. When used for class probability prediction the above observation improves on the regret transform analysis of the probing algorithm [7] by a factor of $\sqrt{2}$. The reason for this improvement is (essentially) the use of a unified measure over the classification problem rather than many different measures for different problems.

Varying interval The range of t can be made dependent on s. This is useful when embedding multiclass classification into cost-sensitive classification for k not a power of 2. Roughly speaking, allowing the range of t to vary with s eliminates the use of classifiers for which the correct prediction is "always 0" or "always 1". Eliminating these classifiers improves the regret transform by reducing the size of the set over which regret is averaged.

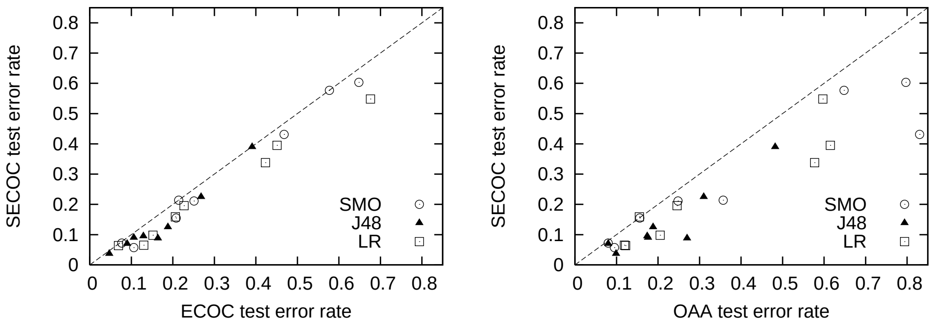
Clipping Our prediction algorithms can output relative costs or probability estimates above "1" or below "0". In such cases, clipping the prediction to the interval $[0, 1]$ always reduces regret.

Why regret isn't everything Other work [2] defines a reduction with the property that small *error* on the subproblem implies small *error* on the original problem. The definition of regret we use here is superior because the theorems can apply nontrivially even on problems with large inherent noise. However, the mathematical form of the regret transforms is weaker, typically by ϵ loss changing to $\sqrt{\epsilon}$regret. Tightening the regret transform by removal of the square root seems to require a different construction.

Difficulty of created learning problems A natural concern when using any reduction is that it may create hard problems for the oracle. And in fact, learning to distinguish a random subset may be significantly harder than learning to distinguish (say) one label from all other labels, as in the One-Against-All (OAA) reduction, as observed in [1, 5]. The best choice of code is a subtle affair. The method here is general and can be used with sparser coding matrices as well. Nevertheless, there is some empirical evidence in support of SECOC with Hadamard codes (presented in the next section).

7 Experimental Results

We compared the performance of SECOC, ECOC and One-Against-All (OAA) on several multiclass datasets from the UCI Machine Learning Repository [11] (ecoli, glass, pendigits, satimage, soybean, splice, vowel, and yeast). Hadamard matrices were used for both SECOC and ECOC. As oracles, we used a decision tree learner (J48), a (linear) support vector machine learner (SMO) and logistic regression (denoted LR), all available from Weka [15]. Default parameters were used for all three learners in all experiments. For datasets that do not have a standard train/test split, we used a random split with 2/3 for training and 1/3 for testing. The figures below show test error rates of SECOC plotted against those of ECOC and OAA (the axes are labeled).



For SECOC, we used six thresholds for each row of the matrix. SECOC resulted in superior (or equal) performance on every dataset tested, for every learner used. We do not report any statistical significance tests because the assumptions they are based on are not satisfied by the datasets. Instead we report *all* experiments performed; we believe that the observed consistency across different datasets and learners gives sufficient empirical evidence in support of SECOC. The code is available from the authors.

References

1. Erin Allwein, Robert Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.
2. Alina Beygelzimer, Varsha Dani, Tom Hayes, and John Langford. Reductions between classification tasks. *Electronic Colloquium on Computational Complexity*, TR04-077, 2004.
3. Thomas Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
4. Yoav Freund and Robert Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139, 1997.
5. Venkat Guruswami and Amit Sahai. Multiclass learning, boosting, and error-correcting codes. In *Proceedings of the 12th Annual Conference on Computational Learning Theory (COLT)*, 145–155,1999.
6. Adam Kalai and Rocco Servedio. Boosting in the Presence of Noise. In Proceedings of the 35th Annual ACM Symposium on the Theory of Computing (STOC), 195–205, 2003.
7. John Langford and Bianca Zadrozny. Estimating class membership probabilities using classifier learners. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.
8. Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory Support Vector Machines: Theory and Application to the Classification of Microarray Data and Satelite Radiance Data, Journal of the American Statistical Association, 99, 465 (2004) 67-81.
9. Nick Littlestone and Manfred Warmuth, The Weighted Majority Algorithm, Foundations of Computer Science, 1992.
10. John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, 61–74, 1999.
11. C. Blake and C. Merz, UCI Repository of Machine Learning Databases, `http://www.ics.uci.edu/∼mlearn/MLRepository.html`, University of California, Irvine.
12. Leslie Valiant. Learning disjunctions of conjunctions, In *Proceedings of the 9th IJCAI*, 560–566, 1985.
13. Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
14. Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of event to their probabilities. *Theory of Probability and its Applications*, 16(2), 264–280, 1971.
15. Ian H. Witten and Eibe Frank, Data Mining: Practical machine learning tools with Java implementations, Morgan Kaufmann, 2000, `http://www.cs.waikato.ac.nz/ml/weka/`.
16. Bianca Zadrozny, John Langford, and Naoki Abe. Cost-Sensitive Learning by Cost-Proportionate Example Weighting. In *Proceedings of the 3rd IEEE International Conference on Data Mining* (ICDM) 435–442, 2003.