SENSITIVE ERROR CORRECTING OUTPUT CODES

Abstract. Sensitive error correcting output codes are a reduction from cost sensitive classification to binary classification. They are a modification of error correcting output codes [3] which satisfy an additional property: ϵ regret for binary classification implies at most 2ϵ l_2 regret for cost-estimation. This has a number of important implications:

1) Any 0/1 regret minimizing online algorithm 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 ϵ regret for binary classification implies at most $4\sqrt{\epsilon}$ regret for cost sensitive classificaiton.

3) Using the canonical embeding of multiclass classifcation into cost sensitive classification, this reduction implies that ϵ binary regret implies at most 2ϵ l_2 error in the estimation of class probabilities. When thresholded, 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 (in the most general case) each prediction may have a different associated loss. For example, consider the following problem: Given information in the form of the current weather and radio reports, we must predict which of several routes to take. The goal in this task is to minimize the expected time of travel. We could think of this as a multiclass problem (predict which route is fastest), but this can make the problem more difficult[1] due to throwing away information.

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 (SECOC) from cost sensitive classification to binary classification with the property that small regret on the created binary problem(s) implies small regret on the original cost sensitive classification 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

[1]For example, it's easy to design a problem where there are several paths that are good except they are occasionally very slow. If there is another path which is never the best but never slow it may provide the best expected time of all choices. If this is altered into a multiclass problem of "predict the best path", the "never the best but never slow" option will never be the correct label.

significant empirical evidence (see [7, 2, 12]) that analysis in this style results in algorithms providing good performance.

The basic SECOC reduction can be reused in several ways.

(1) General Online Learning: Any 0/1 regret minimizing online algorithm 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) General Regret Minimizing Boosting: Since the binary learning algorithm might be a version of boosting which can boost the regret to 0 (as in [6]), the composition of boosting with SECOC can boost to zero regret on arbitrary (i.e. totally unstructured) loss functions.
(3) 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.
(4) Multiclass Probability Estimation: Using the canonical embeding of multiclass classifcation into cost sensitive classification, this reduction implies that small binary regret implies small l_2 error in the estimation of class probabilities.
(5) Multiclass classification: When the multiclass probability estimates are thresholded, this implies a small 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 [13] have been done wrong.

General Warning: Attempts to understand the contents of this paper in terms of sample complexity results such as PAC learning [9] or uniform convergence [11] will fail. This is an orthogonal theory where we define leaning problems as measures over certain sets and analyze the transformation of losses under mappings between these measures.

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 error rate of the individual binary classifiers. The proof of this statement is essentially the observation that there exist codes where 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 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 of [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 earlier result because the regret bounds are weaker, roughly according to ϵ error rate $\rightarrow \sqrt{\epsilon}$ 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[2] 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 expected cost (rather than the margin) of choices. This approach allows us to accomplish several things:

(1) We can use arbitrary classifiers rather than margin-based classifiers.
(2) We 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 [8] (i.e., fitting a sigmoid to a margin to get a probabilistic prediction) can be thought of as a strong empirical evidence of the deficiency of margins as probability estimates.
(3) 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.

Outline: The next sections of this paper do the following:

(1) We define the basic SECOC reduction.
(2) We define and then prove the main theorem.
(3) We prove corollaries of the main theorem for different applications.
(4) We prove some structural lower bounds: that ECOC can not achieve the regret transform we prove and that no tighter bound can be proved for SECOC.

2. The SECOC Reduction

We work in an assumption-free learning setting. To define the reduction, we first define the problems that we reduce to and from. We will actually reduce to importance weighted binary classification. Since importance weighted binary classification can be reduced to binary classification [12], a reduction all the way to binary classification is achievable.

[2]"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.

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 $(x, \vec{c}) \in S$, let $|\vec{c}| = \sum_y c_y$ and $c_s = \sum_{y \in s} c_y$.
 (b) For each t in $[t_{\min}, t_{\max}]$:
  Let $b_{st} = \mathrm{B}(\{(x,\ I(c_s \geq t|\vec{c}|), |c_s - t|\vec{c}||) : (x, \vec{c}) \in S\})$.
(2) return $\{b_{st}\}$

Algorithm 2 SECOC-predict (Binary classifiers $\{b_{st}\}$, test example $x \in X$)

For each label y return $2\,(E_s E_t I(y \in s) b_{st}(x) + I(y \notin s)(1 - b_{st}(x)) - 1$

Definition 2.1. (Importance Weighted Binary Classification) 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 correct binary label and $[0,\infty)$ is the importance of correct classification. The goal in solving binary classification is to find a binary classifer $b : X \to \{0,1\}$ which minimizes the expected importance weighted loss, $E_{(x,y,i)\sim D} iI(b(x) \neq y)$ where $I()$ is 1 when the argument is true and 0 otherwise.

SECOC reduces from cost sensitive classification.

Definition 2.2. (Cost Sensitive Classification) A cost sensitive 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 choice. The goal in solving cost sensitive classification is to find a multiclass classifer $m : X \to \{1, ..., k\}$ which minimizes the expected cost, $E_{(x,\vec{c})\sim D} c_{m(x)}$.

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 given a binary learning algorithm.

The SECOC reduction uses a code, just as the ECOC reduction. An $n \times k$ binary coding matrix M with columns corresponding to multiclass labels defines the code. Each of n rows defines a binary classification problem by specifying a subset of labels and asking whether the label is in the subset. The columns of M 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 exist and 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 $2k$.

To define SECOC, we create a set of importance weighted classification problems for each subset s of the labels. Since importance weighted classification can be reduced to binary classification [12] this reduction can actually be done all the way to binary classification. The particular importance weighted classification problems are defined so that from their optimal solution we can compute the expected cost of each subset.

To make a prediction, for each label the SECOC reduction adds up the expected cost of the subsets containing the label and outputs the label with the largest probability.

The training and prediction algorithms are given in Figure 1 and Figure 2.

It might appear that SECOC-train requires several invocations of the oracle learning algorithm B, but a standard trick [2, 1] allows this to be collapsed to one classifier. The trick is to augment the feature space with the name of the call, and then learn a classifier on (a random subset of) the union of all training data. With this classifier, we can define $h_{st}(x) \equiv h(x, s, t)$, and all of our results hold for this single invocation classifier.

The implication of this observation is that we can regard SECOC-train as a machine which maps cost sensitive samples to importance weighted binary samples. Thus, we can regard SECOC-train as having type SECOC-train : $(X \times [0, \infty)^k)^* \rightarrow (X' \times \{0, 1\} \times [0, \infty))^*$, and the learned binary classifier is simply $B(\text{SECOC-train}(S))$.

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

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

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

$$e(D, m) = E_{(x,\vec{c})\sim D}\, c_{m(x)}.$$

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 subset.)

We also must define the induced distribution on the (combined) binary classifier. To draw a sample from this distribution, we first draw a cost sensitive sample $(x, \vec{c})$ from D, and then apply SECOC-train to the singleton set $\{(x, \vec{c})\}$ to get a sequence of importance weighted binary examples. 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).

We will use the following notation throughout the paper. For a cost vector $\vec{c} \in [0, \infty)^k$ and a subset of labels $s \subseteq \{1, \ldots, k\}$, let $c_s = \sum_{y\in s} c_y$. Also let $|\vec{c}| = \sum_{y=1}^{k} c_y$.

Given s and t, it will be useful to talk about the distribution D_{st} over $X \times \{0, 1\} \times [0, \infty)$ induced by the process in Algorithm 1. To draw from D_{st}, we draw $(x, \vec{c})$ from D and output $(x, I(c_s \geq t|\vec{c}|), |c_s - t|\vec{c}||)$.

Theorem 3.1. (*SECOC Regret Transform*) *For any importance weighted binary learning algorithm B and a cost-sensitive dataset S in $(X \times [0, \infty)^k)^*$, let $b = B(\text{SECOC-train}(S))$. The* SECOC *reduction satisfies the following regret transform for all test distributions D over $X \times [0, \infty)^k$: For every label $y \in \{1, \ldots, k\}$,*

$$E_{(x,\vec{c})\sim D}\left(\text{SECOC-predict}(b, x) - \frac{E_{\vec{c}|x} c_y}{E_{\vec{c}|x}|\vec{c}|}\right)^2 \leq 4(t_{\max}-t_{\min})r(\text{SECOC-train}(D), b),$$

where $t_{\max} = \max_{D,s}(c_s/|\vec{c}|)$ and $t_{\min} = \min_{D,s}(c_s/|\vec{c}|)$.

First note that the dependence on B and S can be removed by proving the theorem for all b.

Proof. We first analyze what happens when no regret is suffered, and then analyze the case with regret. For any choice of s and t, the optimal classifier is given by

$$b^*_{st} = \arg\min_b E_{(x,y,i)\sim D_{st}} iI(b(x) \neq y)$$
$$= \arg\min_b E_{(x,\vec{c})\sim D} I\left(b(x) \neq I(c_s \geq t|\vec{c}|)\right) \cdot |c_s - t|\vec{c}||\,.$$

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

$$E_{\vec{c}\sim D|x} \max\{(c_s - t|\vec{c}|), 0\}. \tag{3.1}$$

Otherwise, the expected cost is

$$E_{\vec{c}\sim D|x} \max\{(t|\vec{c}| - c_s), 0\}. \tag{3.2}$$

To simplify notation, let $Z_x = E_{\vec{c}\sim D|x}|\vec{c}|$. Equations 3.1 are 3.2 are continuous in t; the first decreases while the second increases monotonically with t, so we need only find the (only) equality point to describe the optimal behavior for all t. This equality point is given by

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

or

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

yielding

$$t = \frac{E_{\vec{c}\sim D|x} c_s}{Z_x},$$

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

For any choice of s, we have

$$E_t b^*_{st}(x) = E_t I\left(E_{\vec{c}\sim D|x} c_s \geq tZ_x\right) = \frac{E_{\vec{c}\sim D|x} c_s}{Z_x}$$

since the $E_t I(K \geq t) = K$ for all K in the same interval as t.

Since decoding is symmetric with respect to all labels, we need analyze only one label y. Furthermore, since step 1 in Algorithm 2 is symmetric with respect to set inclusion or complement set inclusion, we can assume that y is in every subset. This can be done by complementing all subsets not containing y and observing that doing so does not change the decoding properties of the code.

Consequently,

$$\hat{c}_y = E_s E_t b_{st}(x) = E_s \frac{\frac{1}{2} E_{\vec{c}\sim D|x}(c_y + |\vec{c}|)}{Z_x} = \frac{1}{2}\left(\frac{E_{\vec{c}\sim D|x} c_y}{Z_x} + 1\right),$$

where the second equality follows from the fact that every other label appears in s half the time in expectation over s. Consequently, step 1 outputs $\frac{1}{Z_x} E_{\vec{c}\sim D|x} c_y$ 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 3.1 and 3.2, notice that the importance weighted loss grows linearly with the distance of tZ_x from $E_{\vec{c}\sim D|x} c_s$, but on the other hand, each error has equal value in disturbing the expectation in line 1 of 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 $\frac{1}{Z_x} E_{\vec{c}\sim D|x} c_s$ first. (Any adversary can reduce the importance weighted regret by swapping errors at large values of $|t - \frac{1}{Z_x} E_{\vec{c}\sim D|x} c_s|$ for errors at small 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 expectation by Δ is to disturb the expectation for each s by Δ. The importance weighted regret of erring (with a "1") for $t = \Delta + \frac{1}{Z_x} E_{\vec{c}\sim D|x} c_s$ can be found by subtracting equation 3.2 from equation 3.1.

$$
\begin{aligned}
E_{\vec{c}\sim D|x}&(t|\vec{c}| - c_s) I(c_s < t|\vec{c}|) - E_{\vec{c}\sim D|x}(c_s - t|\vec{c}|) I(c_s \geq t|\vec{c}|) \\
&= E_{\vec{c}\sim D|x} \left(\left(\Delta + \frac{E_{\vec{c}\sim D|x} c_s}{Z_x} \right) |\vec{c}| - c_s \right) \\
&= \Delta Z_x .
\end{aligned}
$$

The same quantity holds for $t = -\Delta + \frac{E_{\vec{c}\sim D|x} c_s}{Z_x}$. Since we take an expectation with respect to the continuous variable t, this is really an observation about the *differential* regret. We must integrate to find the total regret. The minimum importance weighted regret suffered by an adversary disturbing the quantity by Δ is thus

$$
\int_{u=0}^{u=\Delta} u du Z_x = \frac{\Delta^2}{2} Z_x .
$$

We need to normalize this quantity by the average importance over t. Letting $d = t_{\max} - t_{\min}$, the average importance for fixed x and s is given by

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

This quantity is maximized (over all x and s) when $E_{\vec{c}\sim D|x} c_s = 0$, thus when $\frac{d^2 Z_x}{2d} = dZ_x/2$. Consequently, the normalized importance weighted regret is at least Δ^2/d.

Using observation (2), this implies that the expectation over s is disturbed by at most Δ with the same (average) budget of importance weighted errors. Consequently, the predicted value for the relative cost is disturbed by at most 2Δ using the equation from line 1 of Algorithm 2. □

4. Applications

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 classification to importance weighted binary classification. For example, "Costing" [12] uses repeated rejection sampling to alter the measure. When SECOC is composed with these reductions we get:

Algorithm 3 PECOC-train (Set of k-class multiclass examples S, importance weighted binary classifier learning algorithm B)

(1) Let $S' = \{(x, \forall i\ c_i = I(i \neq y)) : (x,y) \in S\}$For each subset s defined by the rows of M:
(2) return SECOC-Train($S', B, \left\lfloor \frac{k-1}{2} \right\rfloor, \left\lfloor \frac{k}{2} \right\rfloor$)

Corollary 4.1. (*SECOC Regret Transform*) *For any importance weighted binary learning algorithm B and a cost-sensitive dataset S in $(X \times [0,\infty)^k)^*$, let $b = B(Costing(SECOC\text{-}train(S)))$. For all test distributions D over $X \times [0,\infty)^k$: For every label $y \in \{1,\ldots,k\}$,*

$$E_{(x,\vec{c})\sim D}\left(SECOC\text{-}predict(Costing(b),x,\vec{c}) - \frac{E_{\vec{c}|x}c_y}{E_{\vec{c}|x}|\vec{c}|}\right)^2 \leq 4(max{-}min)r(Costing(SECOC\text{-}train(D)),b)E_{x,\vec{c}\sim D}|\vec{c}|$$

Proof. The Costing transformation has the guarantee that the 0/1 regret times the expected importance bounds the importance weighted regret. □

4.2. **Cost Sensitive Classification.** If SECOC-predict reports the class maximizing the returned value, we can guarantee the following regret for the resulting multiclass classifier.

Corollary 4.2. *(SECOC Multiclass Regret Transform) The SECOC multiclass classifier satisfies the following l_2-regret transform for all distributions D over test examples and all labels $y \in \{1,\ldots,k\}$:*

$$(SECOC - predict(b,x) - D(y \mid x))^2 \leq 2r(SECOC - train(D), b).$$

Proof. Suppose that the correct label has cost c_y and some other label has cost $c_{y'} < c_y$. In order to induce an error, the estimate of □

4.3. **Multiclass Probability Estimation.** SECOC can be used to predict the probability of class labels using the training algorithm in Figure 3. The prediction algorithm remains unchanged.

Corollary 4.3. (*Multiclass Probability Regret Transform*) *For any importance weighted binary learning algorithm B and a multiclass dataset S in $(X \times \{1,...,k\})^*$, let $b = B(PECOC\text{-}train(S))$. For all test distributions D over $X \times \{1,..,k\}$: For every label $y \in \{1,\ldots,k\}$,*

$$E_{(x,y)\sim D}\left(SECOC\text{-}predict(b,x,y) - D(y|x)\right)^2 \leq 4r(PECOC\text{-}train(D), b).$$

This corollary implies that probability estimates (up to an l_2 loss) are accurate whenever the classifier has small regret.

Proof. The proof just uses the main theorem 4.1. In this case, $\max - \min = \frac{1}{k-1}$ and the sum of the costs is always $k-1$ so the regret simplifies. □

4.4. **Multiclass Classification.** Using SECOC-Hard-predict, we achieve a consistent multiclass classifier.

Corollary 4.4. (*Multiclass Classification Regret Transform*) *For any importance weighted binary learning algorithm B and multiclass dataset S in $(X \times \{1,...,k\})^*$,* *let $b = B(PECOC\text{-}train(S))$. For all test distributions D over $X \times \{1, .., k\}$: For every label $y \in \{1, \ldots, k\}$,*

$$r\left(D, SECOC\text{-}hard\text{-}predict(b, x)\right) \leq 4\sqrt{r(PECOC\text{-}train(D), b)}$$

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 strongest statement. It should be noted that consistent generalizations of binary classifiers to multiclass classifiers has historically been problematic [13]. This result shows that *any* consistent binary classifier can be used as a consistent multiclass classifier.

Proof. The regret of a multiclass prediction is proportional to the difference in probability of the best prediction and the prediction made. We can weaken corollary 4.3 as:

$$E_{(x,y)\sim D}|\text{SECOC-predict}(b, x, y) - D(y|x)| \leq 2\sqrt{r(\text{PECOC-train}(D), b)}$$

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. □

4.5. **Online Learning and Loss.** It might appear that that SECOC only works for batch learning, but this is not true—the basic transformations are applied to individual examples as in line 1(b) of algorithm SECOC-train. Consequently, the transformation can be done online. The theorems apply to any measure on $(x, \vec{c})$, so they also apply to the uniform measure over past examples. Consequently, online regret minimizing binary predictors can be used with SECOC to minimize cost sensitive regret online.

Therefore SECOC can be used with online learning algorithms such as weighted majority [14] 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.

4.6. **Boosting Any Loss.** Boosting [4] is a technique for minimizing loss (and hence regret = loss - minimum loss) for binary classification by adaptively altering the measure over examples, learning with respect to the altered measures, and combining the learned predictions. Since SECOC can be reduced all the way to binary classification, boosting can be used on the binary classifiers to minimize regret. The main theorem guarantees that this is equivalent to minimizing regret with respect to any loss function.

Note that some of the created binary problems may be inherently noisy, and boosting tends to focus measure on these noisy examples. The basic guarantee of boosting: that doing ϵ better than random can be used to improve performance still holds, it is just that at some point it becomes impossible to do better than random with respect to the induced measure. Some work [6] has been done with alternate boosting algorithms to weaken the sensitivity to noise, and these approaches may be particularly valuable here due to the amount of induced noise

5. ECOC Can Not Transform Regret

Is it possible to do the same with ECOC? The answer is no, and the proof is actually quite simple. Intuitively, what we mean by inconsistency here is that even using an optimal binary classifier, the reduction fails to provide an optimal multiclass classifier.

Theorem 5.1. *(ECOC Inconsistency) For all $k > 2$, there exists a distributions D over multiclass test examples (x, y), such that for all codes M, with $c^* = \arg\min_c r(ECOC\text{-}train(D), c)$*

$$r(D, ECOC\text{-}test(c^*)) > \frac{1}{10}$$

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

A few observations about symmetry greatly simplify the proof. The first observation is that since only 3 labels have probability we can rewrite any code M as a new *weighted* code M' over the 3 labels where each subset has a weight w_s=the number of times that subset of the 3 labels exists in M after projection. The second observation is that the symmetry with respect to complementarity implies each row (and each code word) 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 labels be $0.25 + \frac{\epsilon}{2}$, the answer to every question will be "no" (i.e. 0).

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 at random.

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

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 code matrix. The existence of "don't cares" allows questions of the form "is the probability of this label greater than that label?" which are (in general) sufficiently powerful to support regret transformation consistency, with the exact quantity of regret transformation efficiency dependent on the coding matrix. We have not yet found a 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. The all-pairs code creates a classifier for every pair which (for an optimal classifier) decides "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 0 probability. In this situation, an adversarial classifier can freely choose to report either $p_i > p_j$ or $p_j > p_i$. Consequently, an adversarial binary classifier could make some label with 0 conditional probability "beat" all labels except for the correct label without paying any regret. 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.

5.1. **SECOC Theorem Can Not be Improved Much.**

6. Discussion

This is the final section where we discuss various details and modifications of the core algorithm.

6.1. **Variants.** There are several variants of the basic SECOC algorithm. 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 the limit of infinitely many picks.

For small values of k (the number of classes), it is possible to derive a better regret transform. 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*(average predicted cost) - 1. This removes a factor of 2 loosening in the last paragraph of the proof of the main theorem. Similar although smaller improvements are possible for small $k > 2$.

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.

6.2. **Practical Matters.** The SECOC algorithm provides nice theoretical guarantees, but has some obvious deficiencies. These deficiencies can be addressed without significantly weakening the theoretical guarantees.

(1) When k is very large, we can instead use a code defined by picking a random subset of $O(\frac{1}{\epsilon^2}\log\frac{2k}{\delta})$ codewords. Hoeffding bounds imply that the estimates vary by at most ϵ with probability $1-\delta$.
(2) Learning intractability. Intuitively, learning to distinguish a random subset may be significantly harder than learning to distinguish (say) one label from all other labels. This intuition is born out by experiments [1]. This observation suggest that SECOC should be applied on sparse codes (i.e. codes where each classifier learns to predict a small subset). The exact choice of code is a subtle affair.

6.3. **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. Understanding what, exactly, is possible here is open work.

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] 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.

[9] Leslie Valiant. Learning disjunctions of conjunctions, In *Proceedings of the 9th IJCAI*, 560–566, 1985.

[10] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[11] 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.

[12] 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.

[13] 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.

[14] Nick Littlestone and Manfred Warmuth, The Weighted Majority Algorithm, Foundations of Computer Science, 1992.