PROBABILISTIC ERROR CORRECTING OUTPUT CODES

Abstract. Probabilistic error correcting output codes are a reduction from multiclass classification to binary classification. They are a modification of error correcting output codes [3] with an additional property: ϵ regret for binary classification implies at most $4\sqrt{2\epsilon}$ regret for multiclass classification.

The probabilistic error correcting output code can also be used to predict the probability of a class label. Used in this mode, the PECOC reduction has the property that ϵ regret for binary classification implies at most 4ϵ l_2 regret for class probability predictions.

1. 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, and some successful binary learning algorithms (most notably AdaBoost [4] and support-vector machines [8]) are extended to the multiclass case only with difficulty. This motivates a question: "Can multiclass problems be solved using binary classifier learners?"

A general approach to such reductions from multiclass to binary classification is based on error-correcting output codes (ECOC), first described in [3]. This reduction works by learning a binary classifier, which decides membership of a label amongst 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.

GENERAL WARNING: Attempts to understand this theorem in terms of sample complexity based learning will fail. This is an orthogonal theory addressing how best to preserve learning across reductions and not how to train (or how many samples are needed for training).

Theorem 1.1. [2] *For all distributions D over multiclass test examples, for all learned binary classifiers, the error rate of the ECOC reduction is at most 4 times the binary error rate.*

We discuss the precise definition of the binary error rate later. Intuitively, it is the binary error rate of the learned classifier on the distribution over binary examples induced by the ECOC reduction and the multiclass distribution D.

The proof of this statement (which appeared with a weaker quantification over the training set rather than the testing distribution in [5],[1]) 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 that the multiclass error rate is at most 4 times the binary error rate.

This theorem is vacuous when the binary error rate is 0.25 or more. The probabilistic error correcting output codes (PECOC) we present here address this problem.

Allwein, Shapire, and Singer [1] modified ECOC by considering 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[1] distance is used instead of the Hamming distance.

There are several drawbacks of Margin ECOC that we address in this paper:

(1) It only works on margin-based classifiers. This limitation seems significant because several non-margin based classifiers (for example, decision trees) work quite well.
(2) The margin is not a very good estimate of the probability of a class. The common heuristic "find a large minimum margin" causes a distortion where the optimization increases the margin of small-margin examples at the expense of the margin on large-margin examples. The existence of the Platt scaling [7] (i.e., fitting a sigmoid to a margin to get a probabilistic prediction) can be thought of as a strong empirical evidence of the probabilistic deficiency of margins.

Note that a modified version of Margin ECOC which uses Platt scaling on the margins is still not quite satisfactory because the margin is not generally a sufficient statistic with the necessary information to make good probabilistic predictions.

PECOC functions in the same way as Margin ECOC, except we use a reduction from probabilistic classification to classification [6] to estimate the binary class probabilities and decode a probability for each class label. If we define *regret* as the difference between the loss achieved and the best possible loss (i.e., the loss rate of the "Bayes optimal" classifier), the PECOC reduction guarantees the following regret transform:

Theorem 1.2. *(PECOC Regret Transform) For all distributions D over test examples (x, y), if the learned binary classifier has regret ϵ the* PECOC *reduction satisfies:*

$$\forall l \in \{1, ..., k\} : \ (PECOC(x) - D(l|x))^2 \leq 2\epsilon$$

Intuitively, the regret of the learned binary classifier is defined with respect to the distribution on binary examples induced by D and PECOC as a sampling process.

When the most likely class under the PECOC reduction is reported, this implies the following regret for multiclass classification:

Corollary 1.3. *(PECOC Regret Transform) For all distributions over test examples, if the learned binary classifier has regret ϵ the* PECOC *multiclass classifier has regret at most $4\sqrt{2\epsilon}$.*

A regret transform has stronger implications than an error transform, such as for ECOC. For our example, on a problem with an unremovable error rate of 1/4

[1]"Loss" is in quotes here because the notion of loss here is a specification of the optimization used by the binary learning algorithm rather than the loss given by the problem.

Algorithm 1 PECOC-train (Set of k-class examples S, $n \times k$ binary matrix M, probabilistic binary classifier learning algorithm PB)

(1) For each row i of M:
  Let $h_i = \mathrm{PB}(\{(x, M(i,y)) : (x,y) \in S\})$
(2) return $\{h_i\}$

we might learn a binary classifier with error rate of $1/4 + \epsilon$. The regret is ϵ, so the theorem implies that the regret of the derived multiclass classifier would be at most $4\sqrt{2\epsilon}$. Working with the regret (rather than the absolute error rate) allows us to make meaningful statements even when the problem has large amounts of noise.

One implication of this theorem is that asymptotic consistency of binary classification implies asymptotic consistency of multiclass classification. This is true for neither Margin ECOC nor ECOC.

The paper has the following structure:

(1) We define the basic PECOC reduction.
(2) We prove the relative error transform of the PECOC reduction.
(3) We prove the inconsistency of ECOC and Margin ECOC. Consequently, no theorem similar to our main theorem can be proved for these techniques.

2. The PECOC Reduction

We work in an assumption-free multiclass classification setting. A multiclass learning algorithm takes as input a sequence of training examples in $(X \times \{1, ..., k\})^*$, where X is some arbitrary feature space and $\{1, ..., k\}$ is a set of labels. The learning algorithm outputs a multiclass classifier $c : X \to \{1, ..., k\}$ which labels arbitrary new elements of X. The PECOC reduction uses as a black box a binary learning algorithm B.

The PECOC reduction uses an $n \times k$ binary coding matrix M with columns corresponding to multiclass labels. 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 Probing reduction [6] from probabilistic binary classification to binary classification is then used to estimate the probability of this answer. 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 classifiers needed is less than $2k$.

To make a prediction, for each label the PECOC reduction adds up the probability of the subset containnig the label for each probabilistic classifier and outputs the label with the largest probability.

The training algorithm is given in Figure 1. There are two methods of predicting: getting a probabilistic prediction as in Figure 2 or taking an argmax to make a hard prediction as in Figure 3.

It might appear that PECOC-train requires several invocations of the oracle learning algorithm A, but a standard trick [2, 1] allows this to be collapsed to one classifier. The trick is to augment the feature space with the number of the call, $X \to X \times \{1, ..., n\}$, and then learn a classifier c on (a random subset of) the union of all training data. With this classifier, we can define $c_i(x) \equiv c(x, i)$, and all of our results hold for this single invocation classifier.

Algorithm 2 PECOC-predict (Probabilistic binary classifiers $\{h_i\}$, $n \times k$ binary matrix M, test example $x \in X$)

1. For each row i of M:
   (a) Let $p_i = c_i(x)$.
2. For each label $l \in \{1, \dots, k\}$:
   (a) Let $\hat{p}(l) = \Sigma_{i=1}^{n} M(i,l)p_i + (1 - M(i,l))(1 - p_i)$
3. For each label l, return $\frac{2\hat{p}(l)}{n} - 1$.

Algorithm 3 PECOC-hard-predict(Probabilistic Binary classifiers $\{h_i\}$, $n \times k$ binary matrix M, test example $x \in X$)

1. Let $\{\hat{p}(l)\}$ =**PECOC-predict**($\{h_i\}$, M, x)
2. return $\arg\max_l \hat{p}(l)$

The implication of this observation is that we can regard PECOC-train as a machine which maps multiclass samples to binary samples (using probing as a subroutine). Thus, we can regard PECOC-train as having type PECOC-train : $(X \times \{1, ..., k\})^* \rightarrow (X' \times \{0, 1\})^*$, and the learned binary classifier is simply B(PECOC-train(S)).

3. The Main Theorem

Before stating the theorem, we define exactly what error and regret means. Given any distribution D on examples $X \times Y$, the error rate of a classifier is given by

$$e(D, c) = \Pr_{x,y \sim D}(c(x) \neq y)$$

The regret is the difference between the error rate and the best possible error rate:

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

(Note that we mean the minimum over *all* c' not over some subset.)

We also must define exactly what the induced distribution on the binary classifier is. To do this, we simply define how to draw one sample from the induced distribution. To draw from the induced distribution, first draw a multiclass sample (x, y) from D. Then apply PECOC-train to the singleton set $\{(x, y)\}$ to get a sequence of binary examples $(x', y')^n$. Now, to draw from D', sample uniformly from this set. We overload notion and denote the induced distribution as PECOC-train(D).

First, we restate the main theorem, and then prove it.

Theorem 3.1. *(Full PECOC Regret Transform) For all distributions D over multiclass test examples (x, y), for all binary learning algorithms B and multiclass datasets S, let $c = B(Probing(PECOC\text{-}train(S)))$. The* PECOC *reduction satisfies:*

$$\forall l \in \{1, ..., k\} : \ E_{x,y\sim D}(PECOC\text{-}predict(Probing\text{-}predict(c, .), x) - D(l|x))^2 \leq 8r(Probing\text{-}train(PECOC\text{-}train(D)$$

For the proof, first note that the dependence on A and S can be removed by proving the theorem $\forall c$ (this is of equivalent generality).

3.1. **The Probing Lemma.** The proof uses the composability of reductions for two reductions. The first reduction is the probing regret transform theorem [6] which we restate here. The probing reduction divides into a training part Probing-train and a testing part Probing-test in the same way as PECOC, and a similar theorem holds for a binary classifier.

Lemma 3.2. *(Probing Regret Transform) For all distributions D over binary test examples (x, y), for all Binary learning algorithms B and binary datasets S, let $c = B(\textit{Probing-train}(S))$*

$$E_{x,y\sim D}(D(y=1|x) - \textit{Probing-predict}(c,x))^2 \leq 2r(\textit{Probing-train}(D), c)$$

where r is the regret of the learned binary classifiers.

3.2. **The PECOC Lemma.** The second lemma is specific to the PECOC reduction. It states that the ability to learn binary class probabilities up to an l_2 regret implies the ability to learn multiclass class probabilities up to an l_2 regret.

Lemma 3.3. *(PECOC Regret Transform module) For all Distributions D over multiclass test examples (x, y), for all probabilistic binary classifiers c*

$$\forall l: \quad E_{x,y\sim D}(D(y=l|x) - \textit{PECOC-predict}(l,c,x))^2 \leq 4E_{x,y\sim D'\equiv \textit{PECOC-train}(D)}(D'(y=1|x) - c(x))^2$$

Proof. We first analyze what happens when no regret is suffered, and then analyze the case with regret.

Let $p_l = D(y = l|x)$ be the probability of label l according to the underlying process generating the test example. We identify sets with their characteristic functions, so each row of the coding matrix corresponds to a subset. Then define

$$P_l(x) = \sum_{l\in s} p_l(x)$$

as the probability of a subset of labels s.

First observe that the code is symmetric with respect to all labels, so we need analyze only one label l. Since 2(a) in Figure 2 is symmetric with respect to 0s and 1s, we can assume that l is in every subset. This can be done simply by complementing all rows i for which $M(i, l) = 0$ and observing that the resulting set of codewords still satisfies the distance property. For any other label l', half the time l' is in s and half the time it is not in s. When the probabilistic classifiers have 0 error, the value of $\hat{p}(l)$ is given by

$$\hat{p}(l) = \sum_s P_s(x) = \sum_s \sum_{l\in s} p_l(x) = \frac{n}{2}(p_l(x) + \sum_{l'} p_{l'}(x)) = n\frac{p_l(x)+1}{2}$$

The last equality uses the fact that $\sum_{l=1}^n p_l = 1$. Consequently, step 3 outputs the exact probability when the probabilistic classifiers are correct.

Now we analyze the regret transformation properties. In general, $\hat{P}_s(x)$ returned by the probabilistic binary classifier is not correct, so we have $\sum_s \hat{P}_s(x) = \sum_s P_s(x) \pm n\epsilon = n\frac{p_l(x)+1}{2} \pm n\epsilon$, so step 3 outputs $p_l(x) \pm 2\epsilon$. Since the probing error transform is concave, the most efficient method to induce a final error of 2ϵ is for each individual classifier to err by ϵ to create $n\epsilon$ error in the sum and 2ϵ error in the returned value. For this error scenario, the probabilistic binary classifiers suffer a regret of ϵ^2 and the probabilistic multiclass classifier suffers a regret of $4\epsilon^2$

This implies the lemma. □

Since one of the rows can be made an all-one row (with membership probability 1), we actually have a slightly tighter bound of

$$\frac{n}{2}(p_l(x)+1)+(n-1)\sqrt{2\epsilon}.$$

3.3. **The theorem proof.** Composing the two lemmas gives the proof.

3.4. **The corollary proof.** To prove the corollary, suppose the correct label has probability p_l and some other label has probability $p_{l'} < p_l$. In order to cause an error, the probability estimate $\hat{p}_l$ for the correct must be less than the probably estimate $\hat{p}_{l'}$. Again, due to the concavity of the square root, the most efficient way to create an error is for $\hat{p}_l = \frac{p_{l'}+p_l}{2} \simeq \hat{p}_{l'}$, implying that the adversary must pay the regret necessary to induce a probabilistic error of size $\frac{p_l-p_{l'}}{2}$. By theorem 3.1, this is $\epsilon = \frac{1}{8}\left(\frac{p_l-p_{l'}}{2}\right)^2$. On the other hand, the amount of multiclass regret induced by choosing l' rather than l is just $p_l - p_{l'}$. Solving for $p_l - p_{l'}$ finishes the corollary.

4. Lower Bounds?

5. Inconsistency of Other Approaches

The PECOC main theorem and proof are stronger than any previous statement because they are regret transformations rather than absolute error (or absolute loss) transformations. One obvious question is: Is it possible to do the same with these other approaches? The answer is no, and the proofs are actually quite simple.

5.1. **Inconsistency of ECOC.** 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 $c^* = \arg\min_c r(\text{ECOC-train}(D), c)$*

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

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 $\frac{1}{6}$. □

5.2. **Inconsistency of MarginECOC.** The title of this section is not entirely fair, because we do not prove that all possible variants of MarginECOC are inconsistent transformers. However, we can rule out some large sets. First, we can show an equivalent theorem where MarginECOC is restricted to use codes on $\{-1,+1\}$.

Theorem 5.2. *(Limited MarginECOC Inconsistency) For all $k>2$, there exists a distributions D over multiclass test examples (x,y), such that for all codes M on $\{-1,+1\}$, with $c^* = \arg\min_c r(\text{ECOC-train}(D), c)$*

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

Proof. Note that for boosting, if the weak classifier is the optimal classifier, every call to boosting will return the same classifier. Consequently, after renormalization, the margin will always be either -1 (when not correct due to noise) or 1 (when correct). All of the losses they analyze are equivalent (after rescaling and renormalizing) to 0/1 loss when the margin takes only two values. Therefore, the analysis of the previous theorem applies. □

MarginECOC 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. However, it is unclear how to get a good regret transform.

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 optimal classifier can freely choose to report either $p_i > p_j$ or $p_j > p_i$. Consequently, an adversarial optimal binary classifier could make some label with 0 probability and "beat" all labels except for one label with probability 1. Needless to say, this is not robust, because one error in one classifier can alter the result. This, is a counterexample to the main theorem of the paper.

6. Discussion

6.1. **Practical Matters.** The above 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 $\hat{p}[l]$ 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 PECOC 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 not fully addressed here.

6.2. **Optimal Codes.**

7. Conclusion

We presented PECOC:probabilistic error correcting output codes, which reduce probabilistic multiclass classification to probabilistic binary classification. These codes have several nice properties:

(1) PECOC, with a good choice of code, for *any* number of labels, satisfies a regret transform of 4ϵ where ϵ is the maximal l_2 regret of either probabilistic classifier.
(2) When composed with known probabilistic binary->binary reductions, PECOC yield a complete reduction to binary classification with a regret transform of 8ϵ where ϵ is the regret of the binary classifier.
(3) When the class with the largest probabilistic estimate is chosen, PECOC yields a reduction from multiclass classification to binary classification with a regret transform of $4\sqrt{2\epsilon}$ where ϵ is the regret of the binary classifier.

In addition, we have introduced a formal definition and provided example analyses of regret transforms which allow us to do reduction analyses while taking into account fundamental noise in the world.

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] 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.
[7] 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.
[8] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.