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 [2] with an additional property: ϵ *relative* error binary classification implies at most $2\sqrt{2\epsilon}$ *relative* error 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 *relative* error rate binary classification implies at most 2ϵ multiclass probabilistic classification.

1. Background

Binary classification is predicting one bit $\{0, 1\}$ from some arbitrary feature space based upon past examples. Similarly, multiclass classification is predicting a many state class $\{1, ..., n\}$ from an arbitrary feature space based upon past examples. Error correcting output codes [2] are a reduction from multiclass classification to binary classification. This reduction works by creating binary classifiers solving questions like: "Is the label in $\{0, 2, 4\}$ or the complement set?" Given a sequence of these classifiers, each label has a corresponding codeword given by inclusion or not in the sequence of query sets. A multiclass prediction is made by finding the codeword closest in Hamming distance to the set of binary predictions observed for some feature.

Theorem 1.1. [5]*(ECOC efficiency) For all distributions over examples D, the ECOC reduction errs at most 4 times the average binary error rate.*

The proof of this statement (similar statements with weaker quantification also appeared in [6],[1]) is essentially the observation that there exist codes where the distance between codewords is at most $\frac{1}{2}$. Consequently, at least $\frac{1}{4}$ of the classifiers must err to induce a multiclass classification error, implying the multiclass error rate is at most 4 times the binary error rate.

Here, we strengthen this theorem in the case where the distribution D is fundamentally noisy. In particular, note that the above theorem is vacuous when the average binary error rate is 0.25. The probabilistic error correcting output code we present here addresses this.

The PECOC reduction is conceptually similar to earlier work [1] which we call "MarginECOC" in this paper. Roughly speaking, MarginECOC is the same as ECOC with the modification that the margin of the binary classifier is thought of as a predicting the probability that the label is in a set or it's complement. Decoding proceeds in the same way as for ECOC except the l_1 distance between the margin based predictions and codewords is used instead of the hamming distance.

MarginECOC has a few drawbacks which PECOC addresses.

(1) It only works on margin based classifiers. This is significant because (for example) decision tree based binary classifiers often 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 "Platt scaling" [3] (fitting a sigmoid to a margin to get a probabilistic prediction) can be thought of as strong empirical evidence of the probabilistic deficiency of margins.

Note that a modified version of MarginECOC 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 MarginECOC, except we use the "Probing" reduction from probabilistic classification to classification[4] to estimate the probabilities. This reduction satisfies a relative error transform:

Theorem 1.2. *(PECOC Multiclass error efficiency) For all distributions over examples D, the PECOC reduction has a relative error rate at most $2\sqrt{2\epsilon}$ where ϵ is the average relative binary error rate.*

Here the "relative error rate" is the error rate minus the minimal error rate for the problem. Going back to our example, on a problem with an unremovable error rate of 0.25 we might learn a classifier with error rate 0.25. The relative error is 0 so the theorem implies that relative error of the derived multiclass classifier would be 0.

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

In addition, we can use the PECOCp reduction (in a slightly different manner) to produce an estimate of the conditional probability of a class. The details of this mode are given later, but a simple optimality guarantee holds.

Theorem 1.3. *(PECOCp Error Efficiency) For all distributions over examples D, the PECOCp reduction has a squared error of at most 2ϵ where ϵ is the average relative binary error rate.*

1.1. Structure.

The structure of the paper is as follows:

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

2. The PECOC Reduction

We work an assumption-free multiclass classification setting. A multiclass learning algorithm A_M takes as input a sequence of training examples $(X \times \{1, ..., n\})^*$ where X is some arbitrary feature space and $\{1, ..., n\}$ is a set of labels. The learning algorithm outputs a multiclass classifier $c_M : X \to \{1, ..., n\}$ which labels arbitrary new elements of X.

The PECOC reduction uses as a subroutine a binary learning algorithm A_B which takes as input a sequence of binary training examples $(X \times \{0, 1\})^*$ and produces a binary classifier $c_B : X \to \{0, 1\}$.

Algorithm 1 PECOC_train(Multiclass dataset S_M, Binary Classifier learning algorithm A_B)

(1) Let $n =$ the number of classes in S_M
(2) For each subset s of $\{1, ..., n\}$ which is half 0 and half 1
  (a) Let $S_s = \{(x, I(y \in s) : (x, y) \in S_m\}$
  (b) Let $c_s = \text{Probing}(S_s, A_B)$
(3) return $\{c_s\}$

Algorithm 2 PECOCp_predict(Probabilistic Binary classifiers $\{c_s\}$, Feature x)

(1) For each label l, let $\hat{p}[l] = 0$
(2) For each subset s of $\{1, ..., n\}$ which is half 0 and half 1
  (a) Let $p_s = c_s(x)$
  (b) $\forall l \in \{1, ..., n\}$ $\hat{p}[l] = \hat{p}[l] + I(l \in s)p_s + I(l \notin s)(1 - p_s)$
(3) return for each l, $\frac{2\hat{p}[l]}{|\{c_s\}|} - 1$

Algorithm 3 PECOC_predict(Probabilistic Binary classifiers $\{c_s\}$, Feature x)

(1) Let $\hat{p}[l]$ =PECOCp_predict($\{c_s\}$,x)
(2) return $\arg\max_l \hat{p}[l]$

The reduction uses an optimized error correction code on a space with n elements. A code is defined by a set $\{s_1, ..., s_k\}$ of subsets of $\{1, ..., n\}$. For simplicity, we analyze the set of subsets of size $\binom{n}{\frac{n}{2}}$ because it is particularly simple to analyze (and gives a particularly tight analysis). The size of this set should not alarm because of the well known fact[7] that a randomized double exponential ($O\left(\frac{2^n}{\sqrt{n}}\right) \rightarrow O(\log n)$) reduction in the set produces a code with similar error rate tolerance (we discuss the implications of this and practical variants later).

The PECOC reduction uses the codewords to create a set of binary problems of the sort "Is the label in the subset or not?" The Probing reduction [4] from probabilistic binary classification to binary classification is then used to estimate the probability of this answer. The Probing reduction takes as input a sequence of binary training examples $(X \times \{0, 1\})^*$ and a binary learning algorithm. It outputs a probabilistic classifier $c_P : X \rightarrow [0, 1]$ which predicts the probability that the label is 1. 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 detailed in figure 1. There are two methods of predicting: getting a probailistic prediction as in figure 2 or taking an argmax to make a hard prediction as in figure 3.

3. Proof of the Main Theorem

Let p_l be the probability of label l according to whatever process D generates the test example. Then define:

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

as the probability of a subset s.

First observe that code is symmetric with respect to all labels, so we need analyze only one label l. Since 2.b. is symmetric w.r.t. the set complement, we can assume that l is in every set. 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 $\hat{p}[l]$ after step 2 is given by:

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

The last equality uses the fact that $\sum_{j=1}^n p_j = 1$. Consequently, step 3 outputs the exact probability when the probabilistic classifier is correct.

Now, we analyze the error transformation properties. In general, the $\hat{P}_s(x)$ returned by the probabilistic binary classifiers is not correct. In order for the probability returned by step 3 to be off by ϵ, the individual probabilistic classifiers must also err by ϵ on average. Since the probing error transform is concave, the most efficient method to induce a final error of ϵ is for each individual classifier to err by ϵ. The probing reduction guarantees that ϵ average relative binary error implies at most $\sqrt{2\epsilon}$ error in the magnitude of the class probability estimate. Consequently, we have:

$$\sum_s P_s(x) + \sqrt{2\epsilon} = |\{s\}| \frac{p_l(x) + 2\sqrt{2\epsilon} + 1}{2}$$

implying the main theorems 1 and 2.

4. Lower Bounds?

5. Inconsistency of ECOC

This proof relies on the observation (1) that ECOC can use at most 2^n bits to distinguish. However, predicting optimally requires an arbitrary number of bits.

6. Inconsistency of MarginECOC

This proof relies on the ill-definedness of "margin". Optimal binary margin based classifiers might (for example) always have a margin of -1 or 1 implying the ECOC inconsistency theorem.

7. Practical Variants

The above algorithm provides nice theoretical guarantees, but has some obvious deficiencies. These deficiencies can be addressed without significantly weakening the theoretical guarantees.

(1) Computational intractability. Learning a classifier for every half-size subset is computationally intractable for any reasonable size n. When n is not too large, we can instead use the set of $O(n)$ codewords defined by a Hadamard code. To derandomize the construction, we can simply use a Hadamard code of block length n, making $m = n$. Any two distinct codewords differ in exactly $n/2$ positions. (The code contains the set of all linear functions on $\mathrm{GF}(2)^{\log n}$.) For this code, $L_i(x) = \frac{n}{2}(p_i + 1) + \epsilon n$ for all i. Since one of

the codewords is $\{0\}^n$ (with membership probability 0), we actually have a slightly tighter bound

$$L_i(x) = \frac{n}{2}p_i + (\frac{n}{2} - 1) + \epsilon(n-1).$$

(2) Great Computational intractability. When n is very large, we can instead use a code defined by picking a random subset of $O(\frac{1}{\epsilon^2}\log\frac{2n}{\delta})$ codewords. Hoeffding bounds imply that the estimates $\hat{p}[l]$ vary by at most ϵ with probability $1-\delta$.

(3) 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.

8. Conclusion

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, September 2001.

[2] Thomas G. Dietterich and Ghulum Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes" Journal of Artificial Intelligence Research, 2:263-286, January 1995.

[3] Platt

[4] Probing

[5] WAP

[6] Venkat

[7] Shannon