Error-Correcting Output Coding for Text Classification

Adam Berger
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
aberger@cs.cmu.edu

Abstract

This paper applies error-correcting output coding (ECOC) to the task of document categorization. ECOC, of recent vintage in the AI literature, is a method for decomposing a multiway classification problem into many binary classification tasks, and then combining the results of the subtasks into a hypothesized solution to the original problem. There has been much recent interest in the machine learning community about algorithms which integrate "advice" from many subordinate predictors into a single classifier, and error-correcting output coding is one such technique. We provide experimental results on several real-world datasets, extracted from the Internet, which demonstrate that ECOC can offer significant improvements in accuracy over conventional classification algorithms.

1 Introduction

Error-correcting output coding is a recipe for solving multi-way classification problems. It works in two stages: first, independently construct many subordinate classifiers, each responsible for removing some uncertainty about the correct class of the input; second, apply a voting scheme to decide upon the correct class, given the output of each weak learner. Recent experimental work has shown that ECOC offers improvements over standard k-way classification methods in domains ranging from cloud classification [Aha and Bankert, 1997] to speech synthesis [Bakiri and Dietterich, 1999], and a number of theories have been proposed for its success [James, 1998]. In this paper, we explore the application of error-correcting output coding to document categorization.

The idea of "classifying by consensus" using a large number of independently-constructed classifiers has appeared in a number of other guises recently in the machine learning literature. The technique of bagging, for instance, involves generating multiple training sets by sampling with replacement, learning a classifier from each generated set, and allowing the learned classifiers to vote on the correct class for a unlabeled object [Breiman, 1996a]. Boosting can be viewed as a special case of bagging where the sampling is adaptive, concentrating on misclassified training instances [Freund and Schapire, 1997]. Voting methods have also been applied to combining multiple neural networks trained on the same data [Perrone, 1993] and applying different types of classifiers to the same problem [Quinlan, 1993].

Why consensus algorithms work so well in practice is still an open question. As a step in that direction, theoretical work has recently established that combining multiple runs of a classification algorithm can reduce its variance [Breiman, 1996b]. Unlike most voting algorithms, the constituent classifiers in error-correcting output coding aren't all solving the same problem; in fact, they are each solving a distinct binary classification problem. [Kong and Dietterich, 1995] have shown that this property of the ECOC algorithm bestows on it, in addition to the variance-reduction property of all voting methods, the ability to correct for bias in the constituent classifiers.

This paper applies ECOC to the problem of text categorization: given a database of documents, each annotated with a label or set of labels, learn a mapping $\Lambda : x \to \{y\}$ from documents to labels. Text categorization by computer—such as the automatic assignment of index terms to medical research papers [Yang and Chute, 1994]—has been a central concern in the field of bibliometrics for many years, but the recent flood of online text has increased the interest in and applications for text categorization. Internet-related classification research has addressed the problem of learning to collect interesting postings to electronic discussion groups based on a user's predilections [Lang, 1995], automatically classifying web pages by content [Craven *et al.*, 1998], and suggesting web pages to a user based on his or her expressed preferences [Pazzani *et al.*, 1996].

We focus here on a restricted version of the general classification problem—namely, we imagine documents have exactly one correct labeling, meaning that the Λ mapping is a function. The databases we employ for experimental purposes in Section 4 have this and an additional convenient characteristic: each label is well represented in the data. Under these conditions, the method of Naive Bayes classification is highly competitive. However, Section 4 demonstrates that in this setting, error-correcting output coding consistently outperforms Naive Bayes. Further experiments reported there suggest that ECOC will be of utility in the sparse-data domain as well.

This paper will proceed as follows. The next section introduces the technique of error-correcting output coding and its application to text classification. An ECOC classifier relies on a binary "coding matrix," and Section 3 discusses some considerations in selecting this matrix. Section 4 describes a series of experiments to validate the claim that ECOC offers improvements on standard classification techniques. Section 5 relates ECOC to Naive Bayes and k-nearest neighbor, another high-performance classification algorithm, and Section 6 concludes by outlining some directions for future work in ECOC-based text categorization.

2 Error-correcting output coding

We describe the technique of error-correcting output coding with a simple example: the task of classifying newswire articles into the $m = 4$ categories {`politics`, `sports`, `business`, `arts`}. To begin, one assigns a unique n-bit vector to each label (where $n > \log_2 m$):

label	coding
`politics`	0110110001
`sports`	0001111100
`business`	1010101101
`arts`	1000011010

One can view the ith bitvector as a unique coding for label i. For this reason (and others, which will soon become apparent), we'll refer to the set of bitvectors as a *code* and denote it by $\mathcal{C}$. The ith row of $\mathcal{C}$ we will write as $\mathcal{C}_i$, and the value of the jth bit in this row as $\mathcal{C}_{ij}$.

The second step in constructing an ECOC classifier is to build an individual binary classifier for each column of the code—10 classifiers in all, in this case. The positive instances for classifier j are documents with a label i for which $\mathcal{C}_{ij} = 1$. The third classifier, for instance, has the responsibility of distinguishing between documents whose label is `sports` or `arts` and those whose label is `politics` or `business`. Heeding to convention, we refer generically to any algorithm for predicting the value of a single bit as a "plug-in classifier" (PiC). A PiC, then, is a predictor of whether a document belongs to some fixed subset of the classes.

To summarize, training an ECOC classifier consists of learning a set $\Lambda = \{\lambda^1, \lambda^2 \ldots \lambda^n\}$ of independent binary classifiers. With Λ in hand, one can hypothesize the correct class of an unlabeled document x as follows. Evaluate each independent classifier on x, generating a n-bit vector $\Lambda(x) = \{\lambda^1(x), \lambda^2(x), \ldots \lambda^n(x)\}$. Most likely, the generated bitvector $\Lambda(x)$ will not be a row of $\mathcal{C}$, but it will certainly be closer (in Hamming distance Δ, say) to some rows than to others. Categorizing the document x involves selecting $\text{argmin}_i\, \Delta(\mathcal{C}_i, \Lambda(x))$, the label i for which $\mathcal{C}_i$ is closest to $\Lambda(x)$. (If more than one row of $\mathcal{C}$ are equidistant to $\Lambda(x)$, select one arbitrarily.) For instance, if the generated bitvector $\Lambda(x) = \{1010111101\}$, the document would receive the label `business`.

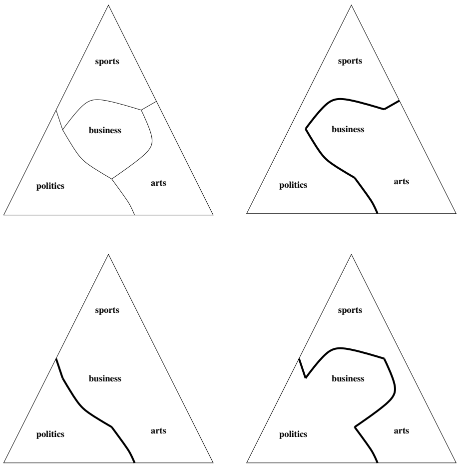

Figure 1: Decision boundaries for the first three plug-in classifiers corresponding to the code given above. Clockwise from upper left: all decision boundaries, bit 1, bit 2, bit 3.

Algorithm 1
Training an ECOC document classifier

Input: Documents $\{x_1, x_2, \ldots x_D\}$;
Labelings $\{y_1, y_2, \ldots y_D\}$ (with m distinct labels);
Desired code size $n \geq \log_2 m$

Output: m by n coding matrix $\mathcal{C}$;
n classifiers $\{\lambda^1, \lambda^2 \ldots \lambda^n\}$

1. Generate a m by n 0/1 coding matrix $\mathcal{C}$
2. Do for $j \in [1, 2 \ldots n]$
   - Construct two superclasses, S_j and $\bar{S}_j$. S_j consists of all labels i for which $\mathcal{C}_{ij} = 1$, and $\bar{S}_j$ is the complement set.
   - Construct a binary classifier λ^j to distinguish S_j from $\bar{S}_j$.

To the extent that rows of $\mathcal{C}$ are well-spaced in Hamming distance, the classifier will be robust to a few errant PiCs. This is the idea behind error-correcting codes as well: to transmit a point in the m-dimensional cube reliably over a noisy channel, map it to one of a set of well-separated "fixed points" in a higher-dimensional cube; to recover the original point, find the closest fixed point to the point actually received and take its preimage in the original cube.

In general, $\lambda^j(x)$ may not be a 0/1 value, but a real-valued probability, measuring the classifier's confidence that document x belongs in the j'th superclass. In this case, one can search for the nearest neighbor according to some L_p distance, rather than Hamming distance. In

Algorithm 2
Applying an ECOC document classifier

Input: `Trained ECOC classifier:` m `by` n `coding matrix` $\mathcal{C}$ `and` n `classifiers` $\{\lambda^1, \lambda^2 \ldots \lambda^n\}$`;` `Unlabeled document` x

Output: `Hypothesized label` y `for` x

1. `Do for` $j \in [1, 2 \ldots n]$
   - `Compute` $\lambda^j(x)$`---the confidence with which PiC` j `believes` $x \in S_j$`.`
2. `Calculate` $\Delta(\Lambda(x), \mathcal{C}_i) = \sum_{j=1}^{n} \mid \lambda^j(x) - \mathcal{C}_{ij} \mid$ `for` $i \in [1, 2 \ldots m]$`.`
3. `Output` $\operatorname{argmin}_i \Delta(\Lambda(x), \mathcal{C}_i)$

the experiments reported in Section 4, the plug-in classifiers output a probability, and we compute the nearest neighbor according to L_1 distance.

2.1 The Naive Bayes classifier

The PiC we relied most heavily on in constructing ECOC classifiers is the *Naive Bayes* classifier [Lewis, 1998]. Naive Bayes assumes that a document is generated by selecting a label y according to a prior distribution $p(y)$, and then independently selecting words w for the document according to a distribution $p(w \mid y)$. The probability of generating a document $W = \{w_1, w_2 \ldots w_N\}$ of N words from label y is thus

$$p(W \mid y) = \prod_{i=1}^{N} p(w_i \mid y) \tag{1}$$

Used for prediction, the Naive Bayes classifier selects for an unlabeled document W the most likely label, given by

$$\begin{aligned} \operatorname*{argmax}_y \ p(y \mid W) &= \operatorname*{argmax}_y p(y) p(W \mid y) \\ &= \operatorname*{argmax}_y \ p(y) \prod_{i=1}^{N} p(w_i \mid y) \end{aligned} \tag{2}$$

where the first equality follows from Bayes' Law.

2.2 Why should ECOC classification work?

Some standard classification algorithms such as backpropagation [Rumelhart *et al.*, 1986] are best suited to distinguishing between two outcomes. A natural way to combine such algorithms to predict from among $k > 2$ outcomes is to construct k independent predictors, assigning predictor i the task of deciding whether the ith outcome obtains. To build the classifier, construct m individual classifiers, where the positive examples for classifier λ^i are those documents with label i. To apply the classifier to an unlabeled document x, select $i^\star = \operatorname{argmax}_i \lambda^i(x)$—the label whose classifier produces the highest score. This is what some call the *one versus rest* strategy. This method is a special case of ECOC classification where $\mathcal{C}$ is the m by m identity matrix.

To see why one might expect ECOC classification to outperform a one-vs.-rest approach, consider the problem of learning to classifying fruit. Imagine that within the labeled set of examples used to train the individual one-vs.-rest classifiers, the only yellow fruit are bananas. So λ^{banana} will learn a strong association between a yellow color and bananas. Now provide a yellow grapefruit to the trained one-vs.-rest classifier. The value of $\lambda^{\text{grapefruit}}(x)$ will likely be close to one—after all, the object in question is round and grapefruit-sized, despite not being red like all the grapefruits encountered in training. But the value of $\lambda^{\text{banana}}(x)$ will be *very* close to one, and the system will misclassify the object as a banana. ECOC classification is less "brittle" than the one-vs.-rest approach: the distributed output representation means one errant subordinate classifier won't necessarily result in a misclassification. This is a circuitous way of saying that ECOC reduces variance of the individual classifiers.

Many classification algorithms, including decision trees, exponential models, and neural networks have the capability to directly perform multiway ($k > 2$) classification. A reasonable classification strategy with these algorithms is to construct a single, monolithic classifier. But the monolithic classifier faces a difficult task. Imagining the classes as clouds in a large-dimensional feature space, a single classifier must learn all the decision boundaries simultaneously, whereas each PiC of an ECOC classifier learns only a relatively small number of decision boundaries at once. Moreover, (assuming n is sufficiently large) an ECOC classifier learns each boundary many times, and is forgiving if a few PiCs place the input x on the wrong side of some decision boundaries [Kong and Dietterich, 1995].

3 Choosing a good code

Early work on error-correcting output coding looked to algebraic coding theory, and in particular to the family of linear codes, for a coding matrix $\mathcal{C}$. An n-bit *linear error-correcting code*, a subspace of the vertices on a n-dimensional cube, can be defined as the span of an n-column binary matrix $\mathcal{G}$, called a *generator matrix*. Error-correcting codes are often measured on the minimum distance between any two linear combinations of $\mathcal{G}$. BCH codes [MacWilliams and Sloane, 1977], a popular class of linear algebraic error-correcting codes, have the useful property that their codewords (all different linear combinations of rows of $\mathcal{G}$) are well separated. Using such a matrix for ECOC classification is for this reason an attractive possibility, and some ECOC classification work has used BCH codes as a coding matrix.

However, subsequent ECOC work has established that ECOC classification should perform well when the coding matrix $\mathcal{C}$ is constructed randomly—specifically, by choosing each entry $\mathcal{C}_{ij}$ uniformly at random from $\{0, 1\}$. This section provide some statistical and combinatorical arguments for why this should be the case. Section 3.1 summarizes some results from [James, 1998] and Section 3.2 is new.

3.1 A statistical perspective

Definition: *Given a database $\mathcal{D}$ of (document, label) pairs (x, y_x) with empirical distribution $\tilde{p}(y, x)$, the* **Bayes classifier** *is $\beta(x) \equiv \mathrm{argmax}_i\, \tilde{p}(y_x = i \mid x)$.*

The Bayes classifier assigns to a document x the label which appears most often in the database $\mathcal{D}$ with x. In terms of classification accuracy on $\mathcal{D}$, the Bayes classifier is the best possible strategy. In the present setting, it is reasonable to assume documents don't occur multiple times with different labels in the collection, and so the Bayes classifier simply selects the label of the document in $\mathcal{D}$. During the training phase, all document labels are available and so we have access to the Bayes classifier. But in applying the classifier we do not. Yet the Bayes classifier will still turn out to be a useful concept, as the following definition and theorem from [James, 1998] suggest.

Definition: *A classification algorithm Λ built from subordinate classifiers $\{\lambda^1, \lambda^2, \ldots\}$ is* **Bayes consistent** *if, whenever the λ^i are Bayes classifiers, so too is Λ.*

Loosely speaking, a Bayes consistent classifier constructed from accurate PiCs will be accurate. This is a property one would like to achieve in an ECOC classifier. The next theorem states the conditions under which this is achievable.

Theorem 1 *Assuming $\mathcal{C}$ was constructed randomly, the ECOC classifier becomes consistent as $n \to \infty$.*

This theorem is *not* saying that with enough bits, an ECOC classifier will do arbitrarily well. Consistency of an ECOC classifier doesn't guarantee correctness—since the PiCs aren't themselves producing Bayes estimates. Still, this theorem suggests why a random construction of $\mathcal{C}$ performs well.

3.2 A combinatorial perspective

The example code presented earlier has the unfortunate property that the third and tenth columns are equal. Therefore, the corresponding classifiers will learn precisely the same task. This is a permissible situation, though hardly desirable. Not permissible is when two *rows* of $\mathcal{C}$ are equal, for then the code cannot distinguish between the corresponding labels. Fortunately, for a randomly-generated binary code with sufficiently many columns, the probability of such an event is miniscule: for a code with m labels and n bits, the probability is

$$1 - \prod_{i=1}^{m-1} \left(1 - \frac{i}{2^n}\right),$$

which is one for $n = \log_2 m$ but approaches zero quickly thereafter as n increases.

More generally, if two rows in $\mathcal{C}$ are close in Hamming distance, an ECOC classifier built from $\mathcal{C}$ is apt to confuse the corresponding labels. We'll write $\Delta(\mathcal{C}_i, \mathcal{C}_{i'})$ as the Hamming distance between rows i and i' of $\mathcal{C}$, and $\Delta_{\min}(\mathcal{C})$ as the minimum distance between any two codewords. If the PiCs produce binary outputs, then the ECOC classifier can always recover from at least $\lfloor \Delta_{\min}(\mathcal{C})/2 \rfloor$ incorrect PiC outputs. The following theorem is a statement about how much row separation one can possibly hope for in a coding matrix.

Theorem 2 *For any m by n binary matrix $\mathcal{C}$, there exist two rows which differ in at most $\frac{n}{2}\left(\frac{m}{m-1}\right)$ bits.*

Proof Let H be the minimum distance between any two rows of one such matrix $\mathcal{C}$. Select two rows $i, j \in [1, 2 \ldots m]$ with replacement. Select a column $k \in [1, 2 \ldots n]$. The probability that $\mathcal{C}_{ik} \neq \mathcal{C}_{jk}$ is

$$p_{\text{diff}} \geq \left(\frac{m-1}{m}\right) \frac{H}{n}$$

Now select a column $k \in [1, 2 \ldots n]$, and then select two rows $i, j \in [1, 2 \ldots m]$ with replacement. The probability that $\mathcal{C}_{ik} \neq \mathcal{C}_{jk}$ is no greater than $1/2$. Combining these inequalities to solve for H gives the result. $\square$

This shows that, as m becomes large, a relative spacing of one half is optimal. If we consider only square matrices, there exists an explicit construction which achieves this bound; namely, the Hadamard matrix. For general 0/1 matrices we are not aware of an explicit construction meeting this bound, but the following result suggests that a random construction is likely to have good separation.

Theorem 3 *Define a* well row-separated *m by n binary matrix as one in which all rows have a minimum relative Hamming separation at least*

$$\frac{1}{2} - 4\sqrt{\frac{\log m}{n}}$$

The probability that a randomly-constructed binary matrix is not well row-separated is at most $1/m^4$.

Proof Given is a randomly-constructed $\mathcal{C}$. Fix two different rows r_1 and r_2. For $i \in [1, 2 \ldots n]$, define the random variable x_i as

$$x_i = \begin{cases} +1 & \text{if } r_1[i] = r_2[i] \\ -1 & \text{otherwise} \end{cases}$$

Let $S = \sum x_i$. For a randomly-constructed $\mathcal{C}$, $E[S] = 0$, which corresponds to an $n/2$ Hamming distance between the rows. We are interested in the probability that $S \gg 0$. Using Chernoff bounds,

$$\Pr(S > 4\sqrt{n \log m}) < e^{-8 \log m} = \frac{1}{m^8}.$$

There are $\binom{m}{2}$ rows in $\mathcal{C}$, and so the probability that no pair of rows is too close is

$$\Pr \leq \binom{m}{2} \frac{1}{m^8} \leq \frac{1}{m^4}$$

$\square$

Although attention in the ECOC literature has generally concentrated on finding a $\mathcal{C}$ with good row separation, a perhaps equally important desideratum is a large separation between columns. Columns that are close give rise to classifiers which are performing nearly the same task—in the extreme case, two equal columns corresponding to two identical classifiers. With only a slight change, Theorem 3 shows that random matrices are likely to have good *column* separation as well, providing another justification for constructing a code randomly.

In practice, a large column separation in $\mathcal{C}$ is not quite sufficient to ensure good performance, because of a degeneracy inherent in binary classification. Many classification algorithms treat 0 and 1 symmetrically, and so if two columns of $\mathcal{C}$ are complementary (or nearly so), the corresponding PICs will learn identical (or nearly identical) classification tasks. What we really want, then, is a matrix whose rows are pairwise well-separated, but not *too* well-separated. The following corollary to Theorem 3 shows that a randomly-selected matrix is, asymptotically, very likely to have this property.

Corollary: *Define a* strongly well-separated *m by n binary matrix as a matrix any two rows of which have a relative Hamming separation in the range*

$$\left[\frac{1}{2} - 4\sqrt{\frac{\log m}{n}}, \frac{1}{2} + 4\sqrt{\frac{\log m}{n}}\,\right]$$

The probability that a randomly-constructed binary matrix is not strongly well row-separated is at most $2/m^4$.

4 Experimental results

We applied error-correcting output coding classification to four real-world text collections, all extracted from the Internet[1]. All corpora were subject to the same preprocessing: remove punctuation, convert dates and monetary amounts and numbers to canonical forms, map all words to uppercase, and remove words occurring twice or less. Table 1 summarizes some salient characteristics of these datasets.

- **20 Newsgroups**: This is a collection of about 20,000 documents, culled from postings to 20 Usenet discussion groups [Lang, 1995]. The documents are approximately evenly distributed among the 20 labels.
- **Four universities**: This (misnamed) dataset contains web pages gathered from a large number of university computer science departments [Craven *et al.*, 1998]. The pages were manually classified into the categories {`course, department, faculty, staff, student, project, other`}.
- **Yahoo science**: Following [Baker and McCallum, 1998], we automatically extracted the entire *Yahoo* science hierarchy in early 1999, and formed a labeled collection containing 41 classes by collapsing the hierarchy to the first level.
- **Yahoo health**: This corpus was collected in the same way as the science collection, but has rather different characteristics. In particular, many of its 36 classes are highly confusable, presenting a difficult task for classification algorithms. For instance, three of the labels in this collection are `Health Administration`, `Hospitals And Medical Centers`, and `Health Care`.

[1]The 20 newsgroups and four universities datasets are publicly available at `www.cs.cmu.edu/~textlearning`.

collection	documents	labels	words
20 newsgroups	19997	20	60915
4 universities	8263	7	29004
Yahoo science	10158	41	69939
Yahoo health	5625	36	48110

Table 1: Particulars on the four training datasets used. Each dataset was partitioned five separate times into a $3/4-1/4$ training/test split, and the numbers are statistics from the last of these trials. The last column reports the number of distinct words in the collection, excluding those appearing once or twice.

Figure 2 plots ECOC classification accuracy against code size n for these four corpora. The codes $\mathcal{C}$ were constructed by selecting entries uniformly at random from $\{0, 1\}$, except in the case of the 4 universities dataset, for which the columns of $\mathcal{C}$ were a random permutation of the 126 unique, non-trivial 7-bit vectors. The plots also display the results of standard Naive Bayes classification.

From an implementation standpoint, a larger value of n incurs a penalty in speed. (This may be an issue in high-throughput systems such as a text filtering systems designed to route relevant news articles to many users, each with their own preferences. However, Figure 2 suggests that, to a point, larger values of n offer more accurate classification. And beyond that point, accuracy doesn't tail off, as is the case in many other learning algorithms for classification, which are prone to overfitting when the number of parameters becomes large.

The four universities dataset was the only collection on which ECOC classification didn't significantly outperform Naive Bayes one-vs.-rest classification. The ECOC classifier's performance on this collection is almost poignant: error rate steadily decreases until $n = 126$, at which point there simply are no more unused, non-trivial 7-bit columns to add to $\mathcal{C}$.

In the collections we are considering, each label is well-represented in the data and models $p(w \mid y)$ can be well estimated. In this setting the standard Naive Bayes method is highly competitive [Lewis, 1998]. For this reason, we use a Naive Bayes classifier as the PiC in the ECOC classifiers corresponding to Figure 2. However, on datasets with poorly-represented labels, Naive Bayes can starve for a lack of data. With an eye towards such collections, we explored using a feature-based classifica-

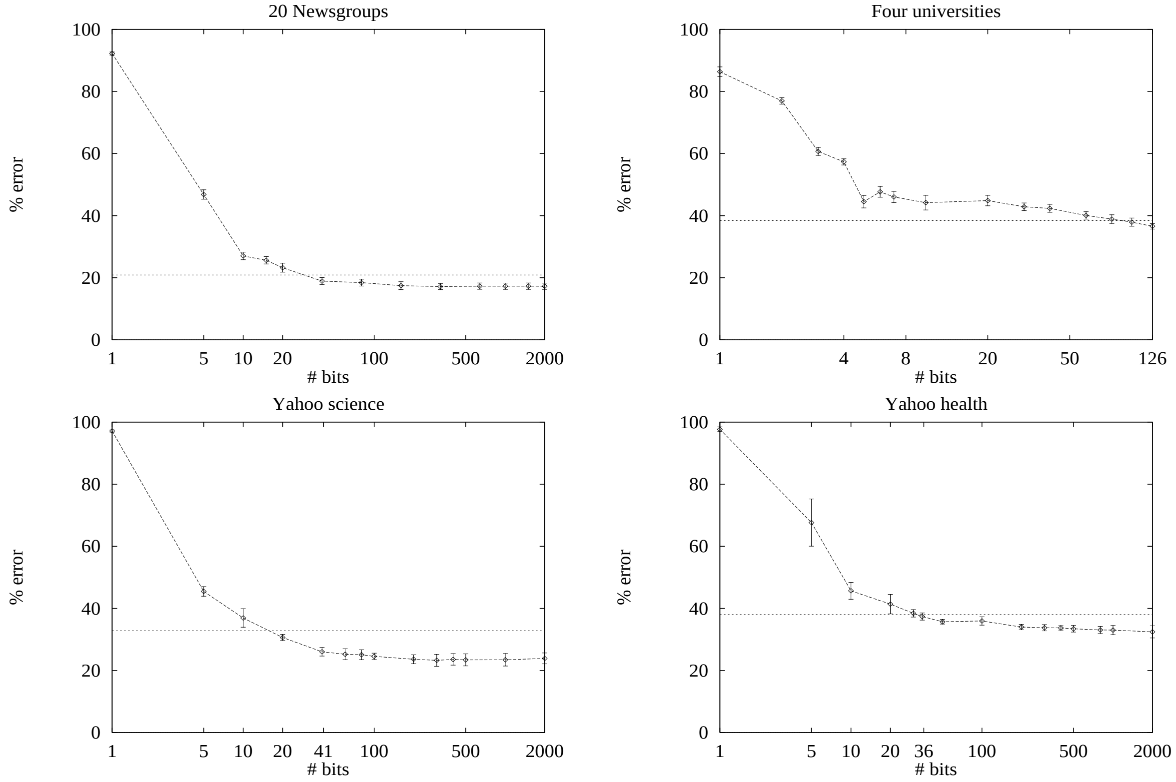

Figure 2: Performance of ECOC classification as a function of code size. Naive Bayes classifiers served as the PiCs. Each point reflects an average over five randomized training/test splits, and the bars measure the standard deviation over these trials. The horizontal line is the behavior of standard one-vs.-rest Naive Bayes. All points are averaged over five trials with a randomized $\mathcal{C}$ and randomized $3/4 - 1/4$ training/test split of the data.

tion approach as the ECOC PiC. Specifically, we trained binary decision trees to predict the individual bits in an ECOC code; the questions at the nodes of each tree were of the form *Did word w appear in the document?* We do not expect such a classifier to match the best reported performance on this dataset, since this algorithm only considers *whether* a word occurs in a document and not how often. However, Figure 3 does suggest that for sufficiently high n, combining decision trees into an ECOC classifier improves performance over a one-vs.-rest decision tree approach, which augurs well for the application of ECOC to larger, sparse datasets.

Truly meaningful values of n lie in the range $[\log_2 m, 2^m]$. A code of size $n < \log_2 m$ cannot even assign a distinct bitvector to each label; at the other extreme, a code of size $n > 2^m$ must contain duplicate columns, which corresponds to two PiCs learning the same task. (A tighter upper bound is actually $(2^m - 1)/2$: the -1 comes about since the all-zero vector corresponds to a trivial classifier, and the denominator arises from the $0/1$ degeneracy mentioned above).

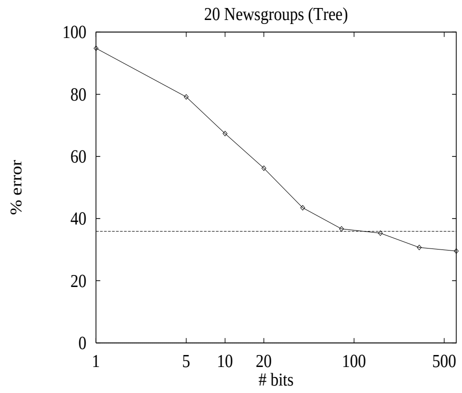

Figure 3: Performance of ECOC classification as a function of code size, for a decision tree PiC with a Bernoulli event model which takes no account of multiple appearances of a word in a document. Each point reflects a single trial using a randomized $3/4 - 1/4$ training/test partition of the 20 newsgroups collection. The horizontal line is the one-vs.-rest decision tree performance.

5 Discussion

The results of the previous section suggest that up to a point, classifier performance improves with n. A simple calculation shows why this should be so.

Assume for the moment that the PiCs only output binary values, and the errors committed by any two PiCs are independent of one another. Denote by p_i the probability of error by the ith PiC, and let $\hat{p} \equiv \max_i p_i$. If the minimum distance of $\mathcal{C}$ is $\Delta_{\min}$, then classification is robust to any $\lfloor \Delta_{\min}/2 \rfloor$ or fewer errors, and so the probability of a correct classification, as a function of n, is

$$p(n) \geq \sum_{k=0}^{\lfloor \Delta_{\min}/2 \rfloor} \binom{n}{k} \hat{p}^k (1-\hat{p})^{n-k} \tag{3}$$

The quantity on the right—the first $\lfloor \Delta_{\min}/2 \rfloor$ terms of the binomial expansion of $p + (1-p)$—is monotonically increasing in $\Delta_{\min}$, which itself increases with n for a randomly-constructed code. Section 4 shows that in practice, $p(n)$ eventually plateaus, which means that the assumption that the errors are uncorrelated is false. This is hardly surprising: after all, the individual classifiers were trained on the same data. One would expect a correlation between, for instance, the second and third columns of the code presented in Section 2.

5.1 Relation to Naive Bayes

We have already seen that the one-vs.-rest strategy is a special case of ECOC classification. It is not difficult to see that the standard Naive Bayes approach is an implementation of ECOC classification. Notice that Naive Bayes is clearly a one-vs.-rest technique: predicting from among m classes requires constructing m classifiers (each consisting of a prior $p(y)$ and a class-specific distribution $p(w \mid y)$), and selecting a label $y^\star$ via (2). But this just amounts to using as a code $\mathcal{C}$ the m by m identity matrix, and then applying Algorithm 2 using an L_p norm.

5.2 Relation to k-nearest neighbor

A popular approach to text classification, particularly competitive for very large and sparse datasets, is k-nearest neighbor (kNN). kNN relies on a map $\psi : x \to \vec{v}$ from documents x to N-dimensional vectors $\vec{v}$. The entries of the latter may be word counts or, more generally, a list of feature values. A kNN classifier stores the images of all training set documents in a database $\mathcal{V} = \{\vec{v}_1, \vec{v}_2, \ldots\}$. To classify an unlabeled document x, kNN finds the k vectors in $\mathcal{V}$ closest to $\psi(x)$, and takes a weighted vote of their labels.

kNN and ECOC have some superficial similarities. Both use for classification a data structure consisting of a set of vectors, and both search this data structure using a nearest-neighbor algorithm, linear in the size of the data structure. One distinction—of particular importance when the size of the training set becomes large—is that while ECOC's data structure consists of a single vector for each label, kNN must store a vector for each document in the training set.

6 Conclusion

We have described an application of error-correcting output coding to the problem of automatic text categorization. The recent explosion in availability of online text lends an extra importance, if not urgency, to this problem, and also suggests a source of experimental data. In fact, the experiments reported in Section 4 were all conducted on data gathered from the Internet. Those experiments offer compelling empirical evidence for the effectiveness of ECOC in text categorization.

This paper reports just some initial proof of concept experiments. There is yet much unexplored terrain, and it is our belief that coding theory has more to say about classification. For instance, a useful class of error-correcting codes for digital transmission is *erasure codes*, which are robust to some fraction of lost bits. If the PiCs produce probabilities, then one could view a classifier λ which is sufficiently indecisive ($\lambda(x) \approx 1/2$) as a "lost bit"; an ECOC classifier containing λ could ignore λ in attempting to recover the label of the document.

Although we presented evidence suggesting the benefits of random codes, there are settings in which one would expect a structured code to be preferable. For instance, performing a nearest-neighbor search in high dimensional space can be expensive, prohibitively so for high-throughput systems. However, one might still be able to reap the benefits of high-n error-correcting output coding without actually conducting the full search. Using a deterministic code with some structure, like a BCH code, may allow the user to replace the $\Theta(nm)$ exhaustive search with a $\Theta(n)$ search at a slight cost in accuracy. For just this reason, real-world digital encoding/decoding systems—such as modems, CD players, satellites, and digital cell phones—rely on structured codes.

Furthermore, the theoretical arguments which argue in favor of random codes are predicated on the assumption, untenable in most real-world data, that the errors made by the individual predictors are uncorrelated. In fact, textual data often contain strong correlations, which a classifier ignores at its own peril. For instance, the `astronomy` and `space` classes in the *Yahoo* science category have a strong overlap in word usage—evidenced by the confusion matrices of classifiers we have constructed on this data. A promising direction for improvement is to combine the ECOC approach with some form of word or document clustering, by designing a code which captures the inherent "clumpiness" of the data. In particular, a well-engineered code could reflect a hierarchical decomposition of the problem: first determine if the document belongs to either `astronomy` or `space`, and only then decide which of these classes is most appropriate.

Acknowledgments

The author thanks Tom Dietterich, Adam Kalai, John Lafferty, and Kamal Nigam for suggestions on an early draft, and the "theory lunch group" at CMU for suggestions leading to the material in Section 3.2. This research was supported in part by an IBM Cooperative Fellowship.

References

[Aha and Bankert, 1997] D. Aha and R. Bankert. Cloud classification using error-correcting output codes. *Artificial Intelligence Applications: Natural Resources, Agriculture, and Environmental Science*, 11:1:13–28, 1997.

[Baker and McCallum, 1998] D. Baker and A. McCallum. Distributional clustering for text classification. In *Proceedings of SIGIR*, 1998.

[Bakiri and Dietterich, 1999] G. Bakiri and T. Dietterich. Achieving high-accuracy text-to-speech with machine learning. *Data mining in speech synthesis*, 1999.

[Breiman, 1996a] L. Breiman. Bagging predictors. *Machine Learning*, 26:2:123–140, 1996.

[Breiman, 1996b] L. Breiman. Bias, variance, and arcing classifiers. Technical report, Statistics Department, Stanford University TR-460, 1996.

[Craven *et al.*, 1998] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[Freund and Schapire, 1997] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[James and Hastie, 1997] G. James and T. Hastie. The error coding method and PiCTs. *Journal of Computational and Graphical Statistics*, 7:3:377–387, 1997.

[James, 1998] G. James. *Majority vote classifiers: theory and applications*. PhD thesis, Stanford University, 1998.

[Kong and Dietterich, 1995] E. Kong and T. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning*, pages 313–321, 1995.

[Lang, 1995] K. Lang. Newsweeder: Learning to filter news. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, 1995.

[Lewis, 1998] D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning*, 1998.

[MacWilliams and Sloane, 1977] F. MacWilliams and N. Sloane. *The theory of error-correcting codes*. North Holland: Amsterdam, The Netherlands, 1977.

[Pazzani *et al.*, 1996] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence*, 1996.

[Perrone, 1993] M. Perrone. *Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization*. PhD thesis, Brown University, 1993.

[Quinlan, 1993] J. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufman, 1993.

[Rumelhart *et al.*, 1986] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Yang and Chute, 1994] Y. Yang and C. Chute. An application of expert network to clinical classification and Medline indexing. In *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care (SCAMC'94)*, volume 18 (Symp.Suppl), pages 157–161, 1994.