Exploring the Space of Coding Matrix Classifiers for Hierarchical Multiclass Text Categorization

Janez Brank
Artificial Intelligence Laboratory
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
janez.brank@ijs.si

Abstract

One of the ways of approaching a multiclass classification problem is to transform it into several two-class (binary) classification problems. An ensemble of binary classifiers is trained for these tasks and their predictions are combined using a voting method into predictions for the original multiclass problem. Each of the new binary problems uses some of the original classes as positive training data, some classes as negative training data and the remaining classes (if any) are not used at all. The relationship between classes (of the original problem) and binary classifiers can be concisely represented by a matrix called the *coding matrix*. In this paper we explore some of the statistical properties of the space of coding matrix based classifiers in the context of a small hierarchical multiclass learning problem.

1 Introduction

Many machine learning methods were initially designed to handle binary (two-class) classification problems. If we want to use such a method to deal with a multi-class problem, one way to do this is to convert the multi-class problem into several binary problems, train binary classifiers for these new problems and then combine their predictions via a voting scheme to obtain predictions for the original multi-class problem.

Each of the new binary classification problems is defined by labelling some of the classes of the original multi-class problem as positive, some as negative, and the rest as unused. The union of instances of the positive (resp. negative) classes from the original multi-class problem then forms the positive (resp. negative) class of the new binary problem.

This mapping between classes of the original problem and the new binary problems can be concisely represented by a *coding matrix*. Let k be the number of classes in the original problem and let m be the number of new binary problems. Then the coding matrix M is a matrix from $\{-1, 0, 1\}^{k \times m}$, where the element M_{cj} tells us how class c of the original problem is used in the j'th new binary problem: $M_{cj} = 1$ means that instances from this class are used as positive examples, $M_{cj} = -1$ that they are used as negative examples and $M_{cj} = 0$ means that they aren't used at all when training the j'th binary classifier.

Thus, the coding matrix has one row for each class of the original multi-class problem. The c'th row of the matrix tells us how class c has been used while training the individual binary classifiers. Consider an instance x belonging to class c; let $y_j(x) \in \{-1, +1\}$ be the prediction of the j'th classifier; then, for those j where $M_{cj} \neq 0$, we would expect $y_j(x)$ to be equal to M_{cj} (unless the classifier y_j is making an error on this particular x), whereas for those j where $M_{cj} = 0$ we cannot make any advance judgments about what the predictions of y_j will be, because the j'th classifier hasn't seen members of class c during training. Thus, if x belongs to c, we would expect the sum $\sum_j y_j(x) M_{cj}$ to be high, and it's reasonable to take $z(x) := \arg\max_c \sum_j y_j(x) M_{cj}$ as the final prediction of the ensemble as a whole with regard to the original multi-class problem.

Coding matrices can be seen as a generalization of various traditional approaches to transforming multi-class problems into binary ones, such as the one-vs-rest approach (corresponding to $k = m$ and a matrix in which $M_{cj} = 1$ if $c = j$, and $M_{cj} = -1$ otherwise) and the one-vs-one approach (corresponding to $k = \binom{m}{2}$ and a matrix in which each column has exactly two nonzero entries).

The space of coding matrices is exponentially large in terms of k and m. Various approaches to constructing coding matrices have been considered in the literature, e.g. based on error correcting output codes [1] or by a greedy search through the space [2]. However, for small values of k and m, the space of coding matrices is tractable and we can afford to consider all possible coding matrices. In this paper we present some statistical properties of the space of coding matrix classifiers and their performance for a small dataset consisting of seven classes organized into a three-level hierarchy.

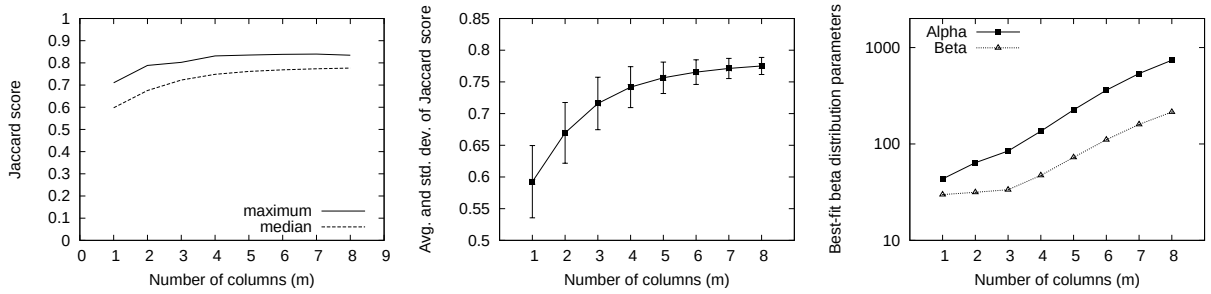

Figure 1. (a) Median and maximum Jaccard score over all m-column matrices; (b) average and std. dev. of the Jaccard score; (c) best-fit beta distribution parameters, α_m and β_m, as functions of m (note the logarithmic scale on the y-axis).

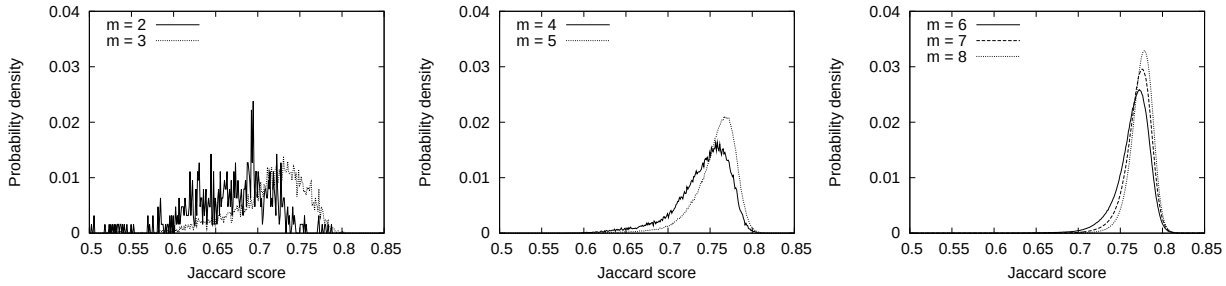

Figure 2. Distribution of Jaccard scores of m-column matrices, for various m.

2 Coding matrix space

In principle, since the matrix has $k \times m$ entries and each entry has three possible values, we could say that there exist 3^{km} possible coding matrices for the decomposition of a k-class problem into m binary problems. But this is in fact only a loose upper bound; it is not useful to distinguish matrices that differ only in the order of columns, nor to permit columns that do not have at least one $+1$ entry and at least one -1 entry. This leaves us with $u := 3^k - 2 \cdot 2^k + 1$ possible states of a column. If we furthermore require that all columns in the matrix be distinct (the effects of allowing multiple identical columns would be better achieved by introducing weighted voting when combining the predictions of binary classifiers), the number of distinct coding matrices (up to the reordering of columns) is $\binom{u}{m}$.

Sometimes the classes are organized hierarchically (a typical example are topical categories such as dmoz.org or Yahoo, but this also occurs in traditional text categorization datasets such as Reuters [4]); if a class p is a parent of class c in the hierarchy, this means that every instance of c is also an instance of p, and this in turn means that if $M_{pj} \neq 0$, then M_{cj} should be equal to M_{pj}, otherwise the j'th classifier would see class c as simultaneously positive and negative.

With this additional constraint, we can derive the following bound on the number of distinct coding matrices. Suppose that our class hierarchy consist of h levels, that all leaves are at level h and that all internal nodes have exactly b subtrees. Each column of the matrix now corresponds to a labelling of this hierarchy with the labels -1, $+1$ and 0, subject to the constraints described in the preceding paragraph. Let u_h be the number of distinct labellings for a h-level hierarchy; it can be shown that $u_h = 2 + (u_{h-1})^b$ and $u_1 = 3$. On the other hand let v_h be the number of distinct labellings that use only the labels $+1$ and 0 but not -1; it can be shown that there are v_h such labellings, where $v_h = 1 + (v_{h-1})^b$ and $v_1 = 2$. Thus the total number of different valid nontrivial labellings is $u := u_h - 2v_h + 1$, and the number of distinct coding matrices is again $\binom{u}{m}$ for this new value of u.

3 Experiments

3.1 Experimental setup

We use a small hierarchy of classes extracted from the large topic hierarchy of the Open Directory Project (ODP; see `dmoz.org`). There is a root topic with 2 subtopics, each of which has another 2 subtopics; thus we have 7 classes arranged in a 3-level hierarchy. Section 2 tells us (taking $h = 3$, $b = 2$) that there are 36 possible states of each column of the coding matrix; thus there are $\binom{36}{m}$ distinct matrices of m columns (up to the reordering of columns). Each topic except the root had 100 training documents and 100 test documens.

m	1	2	3	4	5	6
$\binom{36}{m}$	36	630	7 140	58 905	376 992	1 947 792

This means that for small values of m the number of all m-column matrices is small enough that we can afford to investigate them all; for higher values of m we can at least examine a considerable subset of all m-column matrices by random sampling. We use linear SVM [3] to train binary classification models corresponding to individual columns of the matrix.

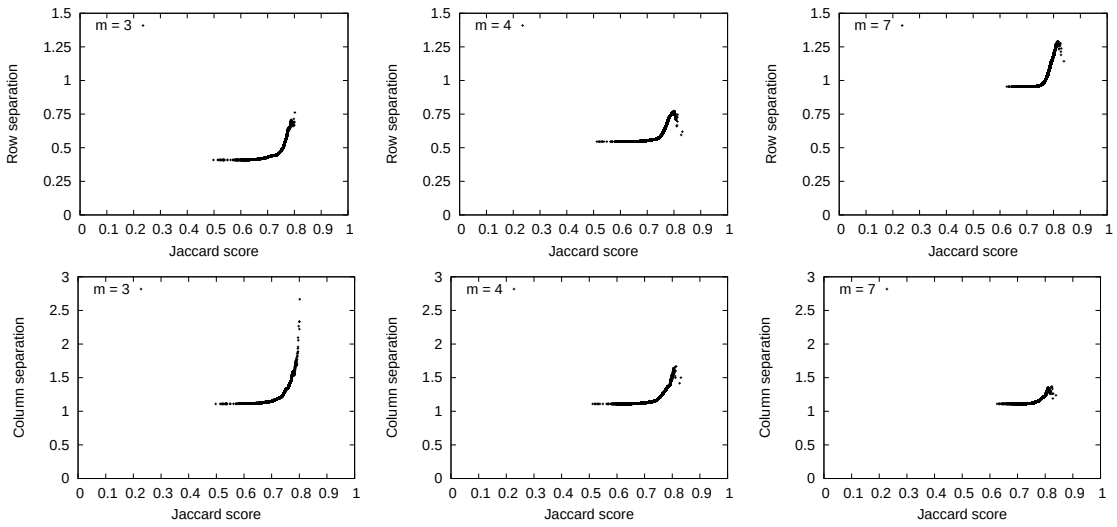

Figure 3. Scatterplots showing the relation between average row/column separation of a matrix and its Jaccard score, for $m = 3, 4, 7$.

To evaluate the classification performance of a coding matrix, we use an evaluation measure based on the Jaccard coefficient. For any class c, let $A(c)$ be the set consisting of c and all its ancestors. If an instance belonging to class c is predicted as belonging into the class c', we define the Jaccard score of this prediction to be $|A(c) \cap A(c')|/|A(c) \cup A(c')|$. We define the Jaccard score of the matrix as the average Jaccard score over all instances from the test set. This evaluation measure lies in the range $[0, 1]$, with higher values indicating better performance.

3.2 Distribution of matrix scores

A frequently used approach in the construction of coding matrices is to fill the matrix at random. If the matrix is effectively a random variable, then so is its Jaccard score, and it is interesting to investigate its probability distribution. Figure 1 shows the median, maximum and average Jaccard scores over all m-column matrices, as a function of m. The median tells us that if we choose a random coding matrix, we have a 50 % probability of getting a Jaccard score greater than or equal to the median; the maximum tells us the best performance we could get if we could examine all possible matrices.

As we can see from Fig. 1, the maximum score grows very slowly from $m = 4$ onwards, while the median keeps on growing. In other words, good matrices with a small number of columns do exist, but there are few of them and it's therefore harder to find them.

We can also describe the distribution of Jaccard scores with a histogram. We divided the range $[0, 1]$ into 1000 equal subintervals and counted the percentage of matrices whose score falls into each particular subinterval. The resulting histograms are shown on Fig. 2. As we can see from the charts there, the maximum score remains roughly the same from $m = 4$ to $m = 8$, but the mode of the distribution gradually moves upwards (i.e. as m increases, more and more matrices have higher scores).

The shape of these distributions approximately resembles that of the well-known beta distribution, $B(\alpha, \beta)$, whose probability density function is $f(x) = x^{\alpha-1}(1-x)^{\beta-1}/B(\alpha, \beta)$. For each m, we can fit a beta distribution over the histogram of Jaccard scores of m-column matrices and obtain a new pair of parameters, α_m and β_m. Fig. 4 shows some comparisons between the original distribution of Jaccard scores and the best-fit beta distribution; the mani difference is that the original distribution leans more towards higher values and has a higher mode than the best-fit beta distribution. Fig. 1(c) shows the value of the best-fit parameters α_m and β_m as a function of m. A very interesting relationship can be seen to emerge, as both α_m and β_m are approximately exponential functions of m.

3.3 Matrix score vs. other properties

An interesting question regarding coding matrices is whether the classification performance of (an ensemble based on) a matrix (e.g. as measured by its Jaccard score) is correlated to some other more easily measurable and controllable properties of the matrix; knowledge of such relationships could be used to construct good coding matrices more easily, or to guide a search through the space of coding matrices.

An example of such property is the *row* (or *column*) *separation*, which is defined as the average Hamming distance between all pairs of rows (or columns). Informal arguments can be constructed why it is desirable for the matrix to have high row and column separation. Figure 3 explores the relationship between row/column separation and Jaccard score empirically, by plotting one symbol for each matrix, with the Jaccard score being used as its x-coordinate and the row or column separation as the y-coordinate. We can see that, in general, better matrices (as measured by the Jaccard score) have high row/column separation. But on the other hand,

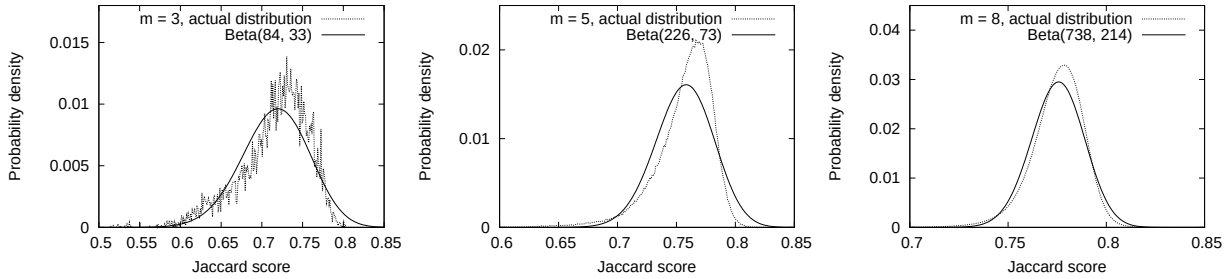

Figure 4. The distribution of Jaccard scores of m-column matrices, with a best-fit beta distribution superimposed on the same chart.

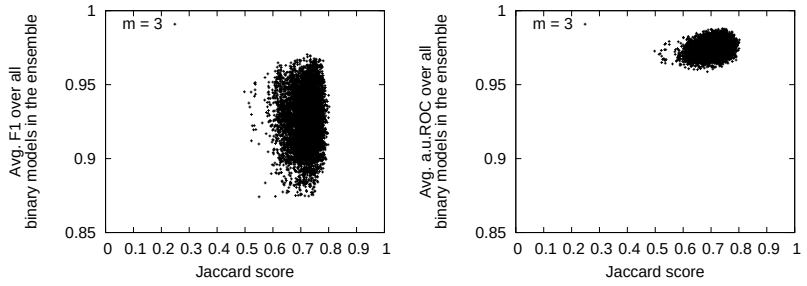

Figure 5. Scatterplots showing the relation between the average performance (F_1 on the left chart, a.u.ROC on the right one) of binary classifiers in the ensemble, and the Jaccard score of the ensemble as a whole.

the very best matrices do not actually have the highest row/column separation, which shows that merely maxmizing these two measures is not necessarily the best way to construct coding matrices.

Another interesting question is whether the performance of the ensemble is related to the performance of the individual binary classifiers in it. We evaluated each individual binary classifier from the point of view of its binary classification problem, and computed measures such as F_1 and the area under the ROC curve. We can then compute the average F_1 or a.u.ROC over all the columns of a matrix, and see whether this average is correlated to the Jaccard score of the matrix. Fig. 5 shows the results of this experiment; the correlation coefficients are 0.015 for the F_1 measure and 0.25 for the a.u.ROC, which shows that the performance of an ensemble isn't strongly correlated to the performance of the individual binary classifiers in it.

4 Conclusions and future work

We have investigated the performance of coding-matrix based ensembles on a small 7-class hierarchical text classification problem using exhaustive experiments on matrices of up to 8 columns. (1) Our experiments show that matrices with good performance can be found even with a small number of columns, but they are much more rare than if we allow more columns. (2) We have shown that the matrix performance scores are distributed approximately following a beta distribution, whose parameters are exponential in the number of columns. (3) We have shown that row/column separation is correlated with matrix performance, but maximizing separation does not lead to maximal performance. (4) We have shown that matrix performance is not correlated with the performance of the individual binary classifiers in the corresponding ensemble.

In the future, the experiments could be extended to a higher number of columns, where it would be necessary to use random sampling as the number of possible matrices is too high. Similarly it would be interesting to extend this research to problems with a larger number of classes, and to involve other easily computable matrix properties in addition to row/column separation.

References

[1] A. Berger. Error-correcting output coding for text classification. *Proc.* IJCAI, Stockholm, 1999.

[2] J. Brank, D. Mladenić, M. Grobelnik. *Generation of ontologies from very large databases*. SEKT report D1.13.1, August 2006.

[3] C. Cortes, V. Vapnik. Support-Vector Networks. *Mach. Learning*, 20(3):273–297, Sept. 1995.

[4] *Reuters Corpus, Volume 1*, English language, 1996-08-20 to 1997-08-19. Release date 2000-11-03, *http://trec.nist.gov*.