Efficient implementation of class-based decomposition schemes for Naïve Bayes

Sang-Hyeun Park · Johannes Fürnkranz

Received: 1 January 2013 / Accepted: 17 December 2013 / Published online: 1 January 2014
© The Author(s) 2013

Abstract Previous studies have shown that the classification accuracy of a Naïve Bayes classifier in the domain of text-classification can often be improved using binary decompositions such as error-correcting output codes (ECOC). The key contribution of this short note is the realization that ECOC and, in fact, all class-based decomposition schemes, can be efficiently implemented in a Naïve Bayes classifier, so that—because of the additive nature of the classifier—all binary classifiers can be trained in a single pass through the data. In contrast to the straight-forward implementation, which has a complexity of $O(n \cdot t \cdot g)$, the proposed approach improves the complexity to $O((n + t) \cdot g)$. Large-scale learning of ensemble approaches with Naïve Bayes can benefit from this approach, as the experimental results shown in this paper demonstrate.

Keywords Naïve Bayes · Error-correcting output codes · Scalability

1 Introduction

A common approach for tackling multi-class classification is the decomposition of a given k -class problem into a set of n binary classification problems, which in turn can be learned by a binary base classifier. The best-known decomposition schemes are *one-against-all* and *one-against-one* (Friedman 1996; Fürnkranz 2002), where $n = k$ and $n = \frac{k(k-1)}{2}$ respectively. These particular schemes and, in general, every class-based decomposition scheme can be modeled within the *error-correcting output coding* (ECOC) framework (Dietterich and Bakiri 1995), and its later generalization known as *ternary ECOC* (Allwein et al. 2000). It consists of a code matrix $(m_{ij}) \in \{-1, 0, 1\}^{k \times n}$, whose k rows represent the classes, and the

Editor: Hendrik Blockeel.

S.-H. Park · J. Fürnkranz (✉)
Knowledge Engineering Group, Department of Computer Science, TU Darmstadt, Darmstadt, Germany
e-mail: juffi@ke.tu-darmstadt.de

S.-H. Park
e-mail: park@ke.tu-darmstadt.de

columns encode the training protocol for the n classifiers. Each entry m_{ij} encodes whether the examples of class i are used as positive examples (+1), as negative examples (−1), or are ignored (0) in the training of classifier j .

The original motivation for the development of the ECOC framework was to transfer the error-correcting properties that are known from signal theory (Macwilliams and Sloane 1983) to the problem of multi-class classification (Dietterich and Bakiri 1995), which could be empirically confirmed in several studies (Kong and Dietterich 1995; Kittler et al. 2003; Melvin et al. 2007). In particular, several authors have shown that the Naïve Bayes algorithm can benefit from the ECOC framework for text-classification (Berger 1999; Ghani 2000). Note, however, that this advantage is not guaranteed. One can show that there exist code types, for which the binary decomposition in conjunction with voting aggregation (which is in principle identical to Hamming decoding (Park and Fürnkranz 2012)) is equivalent to standard Naïve Bayes. For example, in Sulzmann et al. (2007) it was shown that a one-against-one decomposition of Naïve Bayes is equivalent to standard Naïve Bayes. Moreover, most of the above studies used a fairly basic Naïve Bayes algorithm, whose performance can certainly be improved. It is well-known that the probabilities obtained by Naïve Bayes tend to over-emphasize the winning class (Domingos and Pazzani 1997), and should be used with care. Calibration of the probabilities, with isotonic regression (Zadrozny and Elkan 2001, 2002) or, equivalently, via the ROC convex hull (Fawcett and Niculescu-Mizil 2007; Flach and Matsubara 2007), may lead to improved predictions, both in multi-class and binary classification settings. Similarly, algorithms that weaken the conditional independence assumption may lead to better predictions (Webb et al. 2005). Nevertheless, it is also well-known that for multi-class classification problems, Naïve Bayes can be optimal despite violations of its independence assumption and the resulting erroneous probability estimates (Domingos and Pazzani 1997; Zhang 2005).

However, the subject of this paper is not whether or in which cases a combination of ECOC and Naïve Bayes classifier can result in increased classification accuracy, or how it compares to the above-mentioned alternative approaches for improving predictive accuracy in Naïve Bayes. Instead, we show how this combination can be implemented in a single pass through the data, so that it is no more costly than standard Naïve Bayes. In this way, the question whether a certain output code can yield a gain in classification accuracy can be efficiently answered for specific practical problems. The key idea behind the approach is the realization that the binary decompositions of a Naïve Bayes classifier can be computed very efficiently from the estimated conditional probabilities of the original Naïve Bayes procedure.

First, we briefly recapitulate ECOC and Naïve Bayes in Sects. 2 and 3 and derive the efficient computation of ECOC ensembles with Naïve Bayes base classifiers in Sect. 4. Then, in Sect. 4.3, we demonstrate a suitable *precalculation* method for discrete, normal and kernel density estimation methods. Finally, we provide empirical support for the efficiency of the method in Sect. 5, and end with the conclusion in Sect. 6. For completeness, we also show a comparison of the accuracies of Naïve Bayes to its ECOC decomposition in the Appendix.

2 Error-correcting output codes

Error-correcting output codes (Dietterich and Bakiri 1995) are a well-known technique for converting multi-class problems into a set of binary problems. Each of the k original classes c_i receives a codeword $\mathbf{c}_i$ in $\{-1, 1\}^n$, thus resulting in a $k \times n$ coding matrix $\mathbf{M}$. Each of its columns $\mathbf{b}_i$, $i = \{1, \dots, n\}$ of the matrix corresponds to a binary classifier, which considers

all examples of a class corresponding to a (+1) entry as positive, and all examples of a class corresponding to a (−1) entry as negative. Ternary ECOC (Allwein et al. 2000) are an elegant generalization of this technique which allows (0)-values in the codes, which correspond to ignoring examples of this class.

A crucial step in the design of an ECOC-classifier is the selection of the coding matrix. Many well-known reduction algorithms, such as *one-against-all* or *one-vs-one* can be realized by choosing appropriate coding matrices. Many other domain-independent coding schemes, such as random codes, have been investigated.

In our experiments, we follow prior work in Naïve Bayes text classification (Ghani 2000), and use *Bose-Chaudhuri-Hocquenghem (BCH) codes*, because they have properties which are favorable in practical applications. For example, BCH codes allow to specify the desired minimum Hamming distance of the codewords, which is directly related to the error-correcting ability of the ECOC framework. The greater this distance, the greater the number of errors which can be detected and corrected. Besides, related results in the literature support its usability for multi-class classification.

The set of all BCH codewords of a specific length and desired Hamming distance can be computed using an appropriate *generator polynomial*. Similar to Dietterich and Bakiri (1995), each of the k classes is initialized with a randomly selected vector which is then multiplied with the generator polynomial to yield the code word for this class. In our evaluation, we used the *bchpoly* routine of Gnu Octave to generate binary BCH codes of lengths 15, 31, 63, 127, 255, 511 and 1023 with maximal designed minimum Hamming distance respectively. In the usual notation, we used BCH codes (15, 5, 7), (31, 6, 15), (63, 7, 31), (127, 8, 63), (255, 9, 127), (511, 10, 255) and (1023, 11, 511), where the parameters describe (in this order) the codeword bit-length, the bit-length for coded information, and the minimal Hamming distance between any pair of codewords. A detailed description and further information on BCH codes can, e.g., be found in Bose and Ray-Chaudhuri (1960), MacWilliams and Sloane (1983).

3 Naïve Bayes

Though Naïve Bayes (NB) is capable of directly learning multi-class predictors, results in the literature indicate that its classification performance can be increased in combination with ECOC methods. Especially for text classification the combination seems to be promising (Berger 1999; Ghani 2000).

Naïve Bayes is essentially an application of the Bayes Theorem with the so-called “Naïve” independence assumption. In the following, we recapitulate the derivation, which, although commonly known, is helpful for the presentation of the alternative computation scheme which will be presented in the following section. Let each example $\mathbf{x}$ be characterized with g values $(a_1, \dots, a_g)$ for attributes $A_1, \dots, A_g$, and $C = \{c_1, \dots, c_k\}$ be the set of classes. Using Bayes Theorem, we can compute the probability that $\mathbf{x}$ belongs to class c_i as

$$\Pr(c_i | \mathbf{x}) = \Pr(c_i | a_1, \dots, a_g) = \frac{\Pr(c_i) \cdot \Pr(a_1, \dots, a_g | c_i)}{\Pr(a_1, \dots, a_g)}$$

Since the denominator of the right hand side is constant for a given $\mathbf{x}$, we can ignore this term, and focus on the numerator. More precisely, for the case of classification, the following holds:

$$\operatorname{argmax}_{c_i} \Pr(c_i | a_1, \dots, a_g) = \operatorname{argmax}_{c_i} \Pr(c_i) \cdot \Pr(a_1, \dots, a_g | c_i)$$

Using the *class-conditional independence assumption*, we can estimate the class-conditional probability with

$$\Pr(a_1, \dots, a_g \mid c_i) = \prod_{j=1, \dots, g} \Pr(a_j \mid c_i)$$

$\Pr(a_j \mid c_i)$ and $\Pr(c_i)$ are estimated from the training data.

4 Computation of ECOC for Naïve Bayes in a single pass

In this section, we describe the key idea of this note. We first show that all probability estimates that are conditioned on a mutually exclusive group of classes are additive (Sect. 4.1), and that this can be used for faster probability estimation in ECOC codes (Sect. 4.2). Finally, we discuss how the idea can be implemented for nominal and numeric data (Sect. 4.3).

4.1 Reduction to base probabilities

The key idea behind the efficient computation of arbitrary class-based decomposition schemes such as ECOC is that all constituent classifiers of a class-based decomposition can be reduced to the estimation of the parameters of the Naïve Bayes classifier, $\Pr(a_j \mid c_i)$ and $\Pr(c_i)$.

Recall that a column $\mathbf{b} = \{-1, +1\}^k$ of the coding matrix corresponds to the training set of a binary classifier, in which all classes that correspond to a -1 ($+1$) entry in $\mathbf{b}$ are labeled as negative (positive). Let $C_{\mathbf{b}}^+ = \{c_1, \dots, c_l\}$ be the set of classes defined as positive given by column $\mathbf{b}$. Then it holds that

$$\Pr(C_{\mathbf{b}}^+) = \sum_{c \in C_{\mathbf{b}}^+} \Pr(c) \quad (1)$$

and

$$\begin{aligned} \Pr(a_j \mid C_{\mathbf{b}}^+) &= \Pr(a_j \mid c_1 \vee \dots \vee c_l) \\ &= \frac{\Pr(a_j \wedge (c_1 \vee \dots \vee c_l))}{\Pr(c_1 \vee \dots \vee c_l)} \\ &= \frac{\Pr(a_j \wedge c_1) + \dots + \Pr(a_j \wedge c_l)}{\sum_{i=1}^l \Pr(c_i)} \\ &= \frac{\sum_{i=1}^l \Pr(a_j \wedge c_i)}{\sum_{i=1}^l \Pr(c_i)} \end{aligned} \quad (2)$$

since the events of $\Pr(c)$ are mutually exclusive. The probabilities $\Pr(C_{\mathbf{b}}^-)$ and $\Pr(a_j \mid C_{\mathbf{b}}^-)$ for the negative class can be reduced analogously.

Equations (1) and (2) simply show that all necessary values $\Pr(C_{\mathbf{b}}^+)$ and $\Pr(a_j \mid C_{\mathbf{b}}^+)$ can be computed using $\Pr(c_i)$ and $\Pr(a_j \mid c_i)$, as for the standard Naïve Bayes. Therefore, different decompositions within the ECOC-Framework can be applied with Naïve Bayes without employing further probability estimation steps from training data, since they would involve redundant computations.

This tight combination of Naïve Bayes and ECOC, i.e. applying standard Naïve Bayes learning and computing the appropriate probabilities using (1) and (2), will be called ECOC-NB in the following text, for convenience.

4.2 Complexity

Instead of n iterations over the dataset for estimating the corresponding estimations of $\Pr(a_j | C_b^+)$ and $\Pr(C_b^+)$ for an n -bit ECOC scheme, only one pass is necessary. The usual training complexity of $O(n \cdot t \cdot g)$ can thus be reduced to $O(t \cdot g)$, where n is the number of classifiers, t the number of training instances and g the number of features. This is possible by estimating the parameters $\Pr(a_j | c_i)$ and $\Pr(c_i)$ as for the regular Naïve Bayes classifier, and applying Eqs. (1) and (2) at classification time for each decomposed classifier and test instance.

Note, however, that although the training complexity is significantly decreased in comparison to the straight-forward application of the ECOC framework, the cost is moved to the prediction or testing phase, because we now have to perform more calculations for estimating the class-probabilities of an example. In particular, for a problem with a large number of attributes and classifiers, this can lead to a significant increase of testing complexity.

As we will show in the next section, this drawback can be solved by *precalculating* the probability distributions needed by the classifiers, i.e. precalculating the combined probability distribution $\Pr(a_j | C_b^+)$, instead of always aggregating over a series of part-probabilities according to (2) for each test instance. This approach, which will be described in the following sections, results in a *training* complexity of $O((n + t) \cdot g)$ and the same *testing* complexity as the standard approach.

4.3 Precalculation

We take a closer look into probability estimation methods for common attribute types and show how to precalculate the needed probability distributions.

4.3.1 Discrete/nominal attribute

For discrete attributes, i.e. $a_j \in A_j$, where A_j is a finite set of distinct values, the following frequency based model is usually used:

$$\Pr(a_j | c_i) = \frac{|a_j \wedge c_i|}{|c_i|}$$

where $|x|$ denotes the number of observed instances which satisfy x . Also: $\Pr(a_j \wedge c_i) = \frac{|a_j \wedge c_i|}{n}$ and $\Pr(c_i) = \frac{|c_i|}{n}$, where n is the number of observed instances, so far. So for Eq. (2),

$$\Pr(a_j | C_b^+) = \frac{\sum_{i=1}^l |a_j \wedge c_i|}{\sum_{i=1}^l |c_i|}$$

and by employing Laplace correction:

$$\Pr(a_j | C_b^+) = \frac{(\sum_{i=1}^l |a_j \wedge c_i|) + 1}{(\sum_{i=1}^l |c_i|) + |a_j|}$$

This leads to $(|A_j| + 1)k$ additions and $|A_j|$ divisions for generating the pseudo probability estimator. Note, this complexity is not dependent on the number of training instances.

4.3.2 Numeric attribute

For numeric attributes, the following two estimation procedures are commonly applied:

Normal density estimation The conditional probability $\Pr(a_j | c_i)$ is in this case usually modeled as a normal distribution:

$$f_N(x) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where the mean value $\mu = \frac{1}{t} \sum_{m=1}^t x_m$ and corresponding standard deviation

$$\sigma = \sqrt{\frac{\sum_{m=1}^t x_m^2 - (\sum_{m=1}^t x_m)^2 / t}{t}}$$

is updated for each incoming training instance by maintaining the number of observations t , the sum of observed attribute values $\sum_{m=1}^t x_m$ and the sum of squared values $\sum_{m=1}^t x_m^2$ for each attribute. These values can be analogously summed up for representing the pseudo probability distribution, whose computational cost is also independent of the number of instances but dependent on the number of attributes.

Kernel density estimation Here, the probability density model is in contrast to the previous two models not represented by a rather small number of model parameters. In essence, kernel density estimators maintain all observed data values x_m , and the probability estimate of a requested value x depends on its distance to these values. This results in a somewhat smooth and not necessary unimodal probability density function. The definition is

$$f_K(x) = \frac{1}{t \cdot h} \sum_{m=1}^t K\left(\frac{x - x_m}{h}\right)$$

where $K(\cdot)$ is some kernel (often a standard Gaussian function with mean zero and variance one) and h is a smoothing parameter, called the *bandwidth*.

In our context, the straight-forward method to combine these probability distributions is to merge the observations, i.e. to form the union of all observed values of an attribute in the relevant classes C_b^+ . This can be done in $O(t)$. In contrast to the previous estimation techniques, the overall worst-case complexity is therefore in this case only equivalent to the straight-forward ECOC method. Note, however, that the merging of the values of two partitions can be sped up in domains where numerical values repeat across training instances. The idea, as, e.g., realized in the WEKA software (Hall et al. 2009), is to maintain a list of all unique values of an attribute together with its number of occurrences. This allows additional savings if the total number of unique values w is much smaller than the total number of values $z = t \cdot g$ because the merging of the value lists in two partitions takes only $O(w)$ instead of $O(z)$. In the experimental, we also report the ratio w/z as *diversity*-value of a dataset.

Figures 1 and 2 show in pseudocode the simple combined algorithm. The first six lines of Fig. 1 correspond to standard Naïve Bayes training, whereas the remaining lines represent the precalculation scheme using Eqs. (1) and (2). The testing phase is in principle identical to the one of standard ECOC. The function `MAKETERNARY` maps the prediction of each classifier into the interval $[-1, 1]$ to be in line with the ternary ECOC framework using an appropriate mapping function, for instance $f(x) = (x - 0.5) \cdot 2$.

Require: ECOC Matrix $\mathbf{M} = (m_{ij}) \in \{-1, 0, 1\}^{k \times n}$,
training set $T = (\mathbf{x}, c)_m$

```

1: for each training instance  $(\mathbf{x}, c_i)$  do
2:    $ePrior.OBSERVE(i)$  #  $\Pr(c_i)$ 
3:   for each attribute  $a_j$  of  $\mathbf{x}$  do
4:      $eCond_{i,j}.OBSERVE(valueOf(a_j))$  #  $\Pr(a_j | c_i)$ 
5:   end for
6: end for
7:
8: for each classifier  $f_j$  do
9:    $\Pr(c_{\mathbf{b}_j}^+) = \sum_{i=1, m_{ij}>0}^k \Pr(c_i)$ 
10:   $\Pr(c_{\mathbf{b}_j}^-) = \sum_{i=1, m_{ij}<0}^k \Pr(c_i)$ 
11:  for each attribute  $a_k$  do
12:    precalculate  $\Pr(a_k | c_{\mathbf{b}_j}^+)$  and  $\Pr(a_k | c_{\mathbf{b}_j}^-)$  according to attribute type and estimation
    technique
13:  end for
14: end for

```

Fig. 1 ECOC-NB training scheme

Require: m_{ij} , $\Pr(c_{\mathbf{b}_j}^+)$, $\Pr(c_{\mathbf{b}_j}^-)$, $\Pr(a_k | c_{\mathbf{b}_j}^+)$, $\Pr(a_k | c_{\mathbf{b}_j}^-)$,
testing instance $\mathbf{x} = (a_1, a_2, \dots, a_m)$

```

1: for each binary classifier/column  $\mathbf{b}_j$  do
2:   # compute bit-prediction
3:    $b_j = \text{MAKETERNARY}(\Pr(c_{\mathbf{b}_j}^+) \cdot \prod_k \Pr(a_k | C_{\mathbf{b}_j}^+))$ 
4: end for
5: return  $\text{argmax}_i \sum_{j=1}^n m_{ij} \cdot b_j$  # weighted decoding

```

Fig. 2 ECOC-NB testing scheme

5 Evaluation

5.1 Experimental setup

ECOC-NB was implemented within version 3.7.1 of the WEKA framework (Hall et al. 2009). For the evaluation, we mainly focused on text-classification problems because the advantage of the combination of ECOC and Naïve Bayes has in the past been demonstrated in such domains. To that end, we used a freely available package of 19 text-classification datasets (19MclassTextWc¹), from which we selected 17 datasets. The remaining two datasets were excluded because one yielded a very low accuracy for all algorithms ($< 1\%$) and the other already exhibited a very high time complexity for standard Naïve Bayes, so that we did not attempt a complete evaluation of all variants of the ECOC-NB classifiers. This

¹http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html.

Table 1 Dataset characteristics. This table shows the number of instances t , the number of features g , the number of classes k and the diversity of a dataset

<i>Data</i>	#instances	#features	#classes	Diversity
<i>fbis</i>	2463	2000	17	0.0050
<i>la1</i>	3204	31472	6	0.0013
<i>la2</i>	3075	31472	6	0.0013
<i>oh0</i>	1003	3182	10	0.0038
<i>oh5</i>	918	3012	10	0.0038
<i>oh10</i>	1050	3238	10	0.0043
<i>oh15</i>	913	3100	10	0.0042
<i>re0</i>	1504	2886	13	0.0023
<i>re1</i>	1657	3758	25	0.0022
<i>tr11</i>	414	6429	9	0.0127
<i>tr12</i>	313	5804	8	0.0151
<i>tr21</i>	336	7902	6	0.0184
<i>tr23</i>	204	5832	6	0.0294
<i>tr31</i>	927	10128	7	0.0058
<i>tr41</i>	878	7454	10	0.0049
<i>tr45</i>	690	8261	10	0.0076
<i>wap</i>	1560	8460	20	0.0022

collection is composed of well-known benchmark datasets such as TREC and OHSUMED. For a detailed description, we refer to Forman (2003). Table 1 shows the dataset characteristics and the last column shows the diversity of each dataset.

We could not find datasets with discrete attributes with similar characteristics (a large number of attributes and a large number of classes), so we decided to perform experiments on discretized versions of these datasets. In particular, we converted each numerical attribute into a 10-valued discrete attribute using equal-frequency discretization in order to get a good idea of potential run-time savings in such a setting.

For all experiments, 10-fold Cross-Validation was applied and they were conducted on a 2.4 GHz AMD Opteron 250 system with 8GB RAM. For kernel density estimation, a Gaussian kernel with mean zero and variance one was used. No feature selection was applied, since we are mainly interested on the training complexity, which is more interesting with a high number of features. But this comes with the disadvantage that the accuracy performances may not represent the optimal values. So, in this regard, the following accuracy results should be viewed with reservation.

In the following performance tables, some cells are empty, because for this particular combination of dataset and BCH bit-length, the BCH code generation process could not generate a valid ECOC matrix, which satisfies some machine learning relevant properties: The code generation process randomly picks k BCH codewords of the specified length as the ECOC matrix and checks for every column, if there is at least one (+1) and one (−1) symbol, respectively. Furthermore, no two columns must be identical or inverse to each other. The code generation process is aborted after 100,000 iterations. In this context, it holds that the lower the number of classes is, the lower is the probability to generate a suitable ECOC-matrix with high bit-length, since the number of valid combinations reduces with decreasing number of classes or rows in the coding matrix.

Note, that we used weighted decoding (Dietterich and Bakiri 1995) instead of Hamming decoding, because it performed slightly better with respect to accuracy in our setting in some preliminary tests. In the following, we will show the run-time results, which are our main focus. For completeness, accuracy results can be found in the [Appendix](#).

5.2 Run-time evaluation

Table 2 shows the training times for normal density estimation, Table 3 the corresponding results for kernel density estimation, and Table 4 for the discretized datasets. For the numerical datasets, for bit-lengths 15, 31, 63 and 127 the second column shows the training time of the straight-forward ECOC implementation, which should serve as a sanity check and for exposition purposes. The training time increase for the straight-forward ECOC method compared to Naïve Bayes corresponds very closely to the number of used ECOC bits respectively classifiers. Furthermore, one can clearly observe the very mild increase of the training time for ECOC-NB for increasing bit-length.

Also, using kernel density estimators, we can observe only a relative slight increase for increasing number of classifiers (Table 3). As previously mentioned, the worst-case training complexity is still the same as the baseline in this case, but in many cases, when the dataset has a very small ratio of distinct values compared to the number of instances, run-times will be significantly smaller. This is also the case here, the last column of Table 1 shows the ratio of the sum of distinct values over all attributes to the number of instances times the number of features, which are all far away from the worst-case scenario. In addition, the tight combination of ECOC and Naïve Bayes may benefit also from the reduced overhead on the programming language level, e.g. fewer function calls and I/O operations.

Note for discrete and normal density estimation, the difference of training time between ECOC-NB to NB is independent of the number of instances t . For instance, if dataset *fbis* had far more instances, the training time of ECOC-NB with 31-bit BCH codes will still only last about 1 sec longer than standard Naïve Bayes, for instance.

6 Conclusion

We report a simple combined computation of ECOC ensembles with Naïve Bayes as base learner. Compared to the straight-forward method with a training complexity of $O(n \cdot t \cdot g)$ its complexity using normal and discrete density estimation methods is reduced to $O((n + t) \cdot g)$.

In conjunction with kernel density estimators the worst-case complexity remains the same, but, in contrast, it can benefit from a low number of distinct feature values. We show some empirical evaluations supporting this statement and expect similar training complexity reduction also on the majority of real-world datasets, which, in our experience, typically exhibit such a low diversity.

A possible disadvantage of the decomposition approach is the need for tuning parameters such as the bit-length. However, with the efficient computation scheme proposed in this work, the cost of such a parameter tuning has become feasible. Furthermore, ECOC-NB can benefit naturally from sophisticated or more specialized code types in the future, which is an active research topic (e.g., Pujol et al. 2006; Escalera et al. 2010). Finally, we note that the results of this paper also facilitate an implementation of ECOC-NB for learning from sufficient statistics, in very much the same way as it possible for the regular Naïve Bayes classifier (Koul et al. 2008).

In summary, we have shown that the combination of Naïve Bayes with Error-Correcting Output Codes is almost as fast as a conventional Naïve Bayes classifier. ECOC are thus a viable technique for trying to improve the classification performance of Naïve Bayes on large-scale datasets.

Acknowledgements We would like to thank Eyke Hüllermeier and Jan-Nikolas Sulzmann for helpful suggestions and discussions. This work was supported by the German Science Foundation (DFG).

Table 2 Training time comparison of Naïve Bayes and ECOC-NB with different BCH code lengths (with precalculation and normal density estimators, in seconds). For bit-lengths 15, 31, 63 and 127 the second column shows the corresponding training time for the straight-forward ECOC implementation

<i>Data</i>	NB	15-Bit BCH	31-Bit BCH	63-Bit BCH	127-Bit BCH	255-BCH	511-BCH	1023-BCH				
<i>fbis</i>	11.16	11.34	165.24	12.03	340.76	11.84	630.94	12.58	1355.48	12.78	14.04	20.39
<i>lal</i>	101.74	101.20	1596.21	110.98	3219.63	—	—	—	—	—	—	—
<i>la2</i>	91.02	92.41	1400.38	98.21	2865.61	—	—	—	—	—	—	—
<i>oh0</i>	4.23	4.39	57.30	4.54	122.39	4.57	235.56	4.81	489.79	5.27	7.07	—
<i>oh5</i>	3.80	3.79	48.54	3.86	104.57	3.92	201.35	4.12	414.65	4.66	6.75	—
<i>oh10</i>	4.57	4.73	61.87	4.87	132.27	4.92	253.59	5.15	524.22	5.69	7.84	—
<i>oh15</i>	3.98	3.94	52.04	4.06	109.39	4.18	210.48	4.39	430.19	4.84	6.96	—
<i>re0</i>	6.07	6.14	76.51	6.16	156.57	6.24	309.98	6.48	657.10	7.03	8.54	15.11
<i>re1</i>	9.08	9.00	118.84	8.97	244.10	9.25	484.02	9.85	1013.32	10.98	12.93	21.55
<i>tr11</i>	4.37	4.45	61.24	4.51	124.40	4.59	249.29	5.03	506.52	6.05	—	—
<i>tr12</i>	2.83	2.85	39.66	2.92	81.02	3.03	161.02	3.39	338.22	—	—	—
<i>tr21</i>	4.35	4.46	62.16	4.51	128.97	—	—	—	—	—	—	—
<i>tr23</i>	1.78	1.82	24.92	1.85	50.94	—	—	—	—	—	—	—
<i>tr31</i>	16.10	16.49	229.83	16.35	467.13	16.32	940.35	—	—	—	—	—
<i>tr41</i>	11.23	11.31	153.79	11.35	313.71	11.46	629.27	12.15	1314.06	13.06	15.62	—
<i>tr45</i>	9.82	9.94	134.89	9.80	279.80	10.16	558.91	10.87	1174.95	11.63	14.58	—
<i>wap</i>	23.78	23.19	309.03	22.93	651.38	23.68	1294.39	24.63	2714.96	26.04	29.62	40.95

Table 3 Training time comparison of Naïve Bayes and ECOC-NB with different BCH code lengths (with precalculation and kernel density estimators, in seconds). For bit-lengths 15, 31, 63 and 127 the second column shows the corresponding training time for the straight-forward ECOC implementation

<i>Data</i>	NB	15-Bit BCH	31-Bit BCH	63-Bit BCH	127-Bit BCH	255-BCH	511-BCH	1023-BCH				
<i>fbis</i>	11.85	13.53	181.11	13.51	376.83	13.62	750.80	15.88	1559.50	17.58	23.74	38.46
<i>lal</i>	107.12	120.18	1700.63	119.79	3474.00	—	—	—	—	—	—	—
<i>la2</i>	105.69	106.56	1517.53	107.69	3080.02	—	—	—	—	—	—	—
<i>oh0</i>	4.95	5.26	67.72	5.60	140.77	5.95	286.17	7.30	584.12	9.10	15.83	—
<i>oh5</i>	4.31	4.55	58.76	4.86	121.46	5.17	248.96	6.49	505.73	8.88	15.01	—
<i>oh10</i>	5.36	5.72	73.57	6.02	152.98	6.30	311.64	7.78	635.28	10.32	16.82	—
<i>oh15</i>	4.48	4.73	60.54	5.01	124.98	5.36	252.04	6.75	513.94	9.14	15.48	—
<i>re0</i>	6.94	7.25	91.68	7.59	190.76	7.83	388.70	9.58	792.19	12.27	18.31	32.40
<i>re1</i>	10.24	10.94	136.05	11.76	283.06	12.72	575.87	16.61	1175.56	22.14	36.46	63.40
<i>tr11</i>	4.76	5.25	66.61	5.79	138.48	6.62	280.17	9.19	568.41	13.53	—	—
<i>tr12</i>	3.12	3.49	43.84	3.92	90.84	4.69	184.31	6.64	374.89	—	—	—
<i>tr21</i>	4.76	5.26	69.61	5.75	143.73	—	—	—	—	—	—	—
<i>tr23</i>	1.92	2.29	27.81	2.72	57.59	—	—	—	—	—	—	—
<i>tr31</i>	17.75	18.41	256.99	19.18	526.89	19.90	1070.32	—	—	—	—	—
<i>tr41</i>	12.37	12.96	172.23	13.68	357.73	14.48	719.62	18.15	1489.90	23.13	36.71	—
<i>tr45</i>	10.75	11.52	152.41	12.30	313.86	13.30	640.29	17.14	1318.36	22.33	37.56	—
<i>wap</i>	25.87	27.24	354.87	28.60	731.24	30.24	1482.71	37.65	3110.42	46.45	70.85	119.63

Table 4 Training time comparison of Naïve Bayes and ECOC-NB with different BCH code lengths on the discretized datasets. For bit-lengths 15, 31, 63 and 127 the second column shows the corresponding training time for the straight-forward ECOC implementation

<i>Data</i>	NB	15-Bit BCH	31-Bit BCH	63-Bit BCH	127-Bit BCH	255-BCH	511-BCH	1023-BCH				
<i>fbis</i>	1.32	1.38	9.93	1.47	20.77	1.50	41.73	1.86	84.66	2.95	5.70	13.12
<i>lal</i>	13.10	14.02	153.54	13.76	343.46	—	—	—	—	—	—	—
<i>la2</i>	11.50	12.48	140.24	12.85	319.59	—	—	—	—	—	—	—
<i>oh0</i>	0.94	1.02	7.72	1.05	17.30	1.12	32.71	1.45	69.50	2.19	4.46	—
<i>oh5</i>	0.81	0.88	6.21	0.93	13.84	0.91	25.02	1.31	52.21	1.86	4.48	—
<i>oh10</i>	0.96	1.07	8.31	1.07	18.73	1.14	35.51	1.58	76.56	2.26	5.10	—
<i>oh15</i>	0.77	0.86	7.01	0.96	14.81	0.96	27.85	1.30	59.65	1.96	4.49	—
<i>re0</i>	1.22	1.34	8.69	1.32	18.64	1.40	35.91	1.83	74.00	2.30	4.60	11.06
<i>re1</i>	2.07	2.26	18.81	2.37	38.34	2.30	76.67	3.56	165.50	4.34	8.65	18.03
<i>tr11</i>	0.85	1.01	10.21	1.18	22.71	1.33	47.60	2.24	100.39	3.63	—	—
<i>tr12</i>	0.56	0.72	6.92	0.78	15.52	0.93	28.79	1.47	59.65	—	—	—
<i>tr21</i>	0.83	0.94	11.21	1.08	23.86	—	—	—	—	—	—	—
<i>tr23</i>	0.38	0.49	4.94	0.57	10.30	—	—	—	—	—	—	—
<i>tr31</i>	2.93	3.18	37.17	3.45	79.85	3.44	155.42	—	—	—	—	—
<i>tr41</i>	2.15	2.31	24.89	2.40	54.05	2.58	109.35	3.49	223.02	4.80	10.32	—
<i>tr45</i>	1.88	2.12	22.83	2.17	48.89	2.49	95.80	3.37	198.83	4.85	11.22	—
<i>wap</i>	4.46	4.64	45.62	5.05	105.40	5.11	219.85	6.46	434.70	8.73	16.24	30.09

Appendix: Accuracy results

Although not the main focus of this note, in the following we also show the accuracy results of the different methods for completeness. Essentially, our results confirm previous results in that the combination of Naïve Bayes and error-correcting output codes has a slight edge over conventional Naïve Bayes.

Table 5 shows the accuracy performance for Naïve Bayes compared to ECOC-NB on the original datasets with various bit-lengths, using normal density estimation and kernel

Table 5 Accuracy of Naïve Bayes and ECOC-NB with different BCH code lengths. The *top part* of the table shows the results for *normal density estimators*, the *bottom part* for *kernel density estimators*. **Bold** values depict the best performance for each dataset and estimator, underlined values show the best performance for each dataset over both estimators

Data	NB	15	31	63	127	255	511	1023
<i>Normal Density Estimators</i>								
<i>fbis</i>	62.61	54.77	58.95	61.35	64.15	64.60	64.80	<u>65.33</u>
<i>la1</i>	75.06	75.25	76.03	–	–	–	–	–
<i>la2</i>	74.89	73.79	75.22	–	–	–	–	–
<i>oh0</i>	79.66	79.06	80.65	80.95	81.05	80.95	80.75	–
<i>oh5</i>	77.88	76.68	78.97	80.06	79.73	80.06	79.62	–
<i>oh10</i>	72.67	71.90	72.86	74.10	73.90	73.90	74.38	–
<i>oh15</i>	75.24	76.56	76.88	78.09	79.41	78.53	78.75	–
<i>re0</i>	57.51	59.71	62.70	64.70	67.56	65.29	69.55	68.55
<i>re1</i>	66.33	68.92	71.94	72.30	74.29	73.87	73.63	74.17
<i>tr11</i>	54.83	48.34	56.54	55.57	57.98	57.74	–	–
<i>tr12</i>	54.67	55.95	57.25	59.10	61.98	–	–	–
<i>tr21</i>	46.39	37.50	39.87	–	–	–	–	–
<i>tr23</i>	55.79	34.19	37.19	–	–	–	–	–
<i>tr31</i>	80.69	82.95	83.61	84.57	–	–	–	–
<i>tr41</i>	85.65	85.32	86.91	87.14	87.37	87.82	87.59	–
<i>tr45</i>	65.51	58.26	62.32	66.38	66.09	67.39	67.39	–
<i>wap</i>	72.76	61.35	66.54	65.51	67.50	68.27	67.76	68.40
<i>Kernel Density Estimators</i>								
<i>fbis</i>	56.27	43.36	49.49	50.14	51.23	51.56	52.45	51.93
<i>la1</i>	78.59	74.63	77.75	–	–	–	–	–
<i>la2</i>	75.02	74.31	75.35	–	–	–	–	–
<i>oh0</i>	79.95	79.06	80.06	79.76	80.26	80.16	80.36	–
<i>oh5</i>	74.29	72.55	73.30	74.50	74.17	75.05	74.39	–
<i>oh10</i>	69.43	67.05	68.10	69.24	68.86	69.43	69.62	–
<i>oh15</i>	70.32	69.22	70.10	69.98	70.31	70.20	69.98	–
<i>re0</i>	66.42	63.90	64.69	64.76	64.96	64.83	65.03	64.90
<i>re1</i>	69.16	70.91	71.52	72.84	72.54	72.96	72.36	72.72
<i>tr11</i>	49.30	42.29	49.06	46.88	48.08	48.33	–	–
<i>tr12</i>	47.62	48.59	48.34	52.42	52.44	–	–	–
<i>tr21</i>	44.33	39.30	41.36	–	–	–	–	–
<i>tr23</i>	53.33	30.31	32.81	–	–	–	–	–
<i>tr31</i>	66.66	70.12	70.87	71.73	–	–	–	–
<i>tr41</i>	76.66	77.46	79.62	80.87	80.75	80.98	80.98	–
<i>tr45</i>	54.93	49.71	54.78	53.91	54.64	55.65	55.22	–
<i>wap</i>	74.17	69.23	70.13	70.58	71.28	71.47	71.67	71.47

Table 6 Accuracy of Naïve Bayes and ECOC-NB with different BCH code lengths on the discretized datasets. *Bold* values depict the best performance for each dataset

<i>Data</i>	NB	15	31	63	127	255	511	1023
<i>fbis</i>	50.18	47.63	52.02	56.84	56.68	58.91	58.42	59.12
<i>la1</i>	70.73	78.28	85.58	–	–	–	–	–
<i>la2</i>	73.92	79.41	83.06	–	–	–	–	–
<i>oh0</i>	56.93	77.17	80.96	80.95	81.15	80.85	80.95	–
<i>oh5</i>	45.43	74.83	76.57	77.23	77.55	77.88	78.20	–
<i>oh10</i>	54.86	73.43	74.19	75.71	75.52	75.71	75.81	–
<i>oh15</i>	40.85	68.43	68.22	69.32	70.96	70.53	71.07	–
<i>re0</i>	54.19	61.71	62.91	64.37	64.70	64.77	64.84	64.90
<i>re1</i>	42.97	66.81	66.87	66.69	67.89	67.95	68.07	68.19
<i>tr11</i>	46.12	49.01	50.50	52.89	54.11	55.07	–	–
<i>tr12</i>	31.31	49.80	53.91	55.90	57.13	–	–	–
<i>tr21</i>	63.69	63.98	64.29	–	–	–	–	–
<i>tr23</i>	25.60	29.38	30.02	–	–	–	–	–
<i>tr31</i>	67.20	73.90	74.32	77.13	–	–	–	–
<i>tr41</i>	60.25	76.55	78.14	79.28	78.82	79.62	79.50	–
<i>tr45</i>	32.17	55.80	61.01	64.49	65.07	64.64	65.51	–
<i>wap</i>	48.65	71.22	72.88	75.06	74.49	75.58	75.71	75.64

density estimation. In general, Naïve Bayes and ECOC-NB without kernel density estimators perform better on these datasets. Furthermore, as can be seen in the top part of Table 5, ECOC-NB yields better results than standard Naïve Bayes, except for the worse performance on datasets *tr21*, *tr23* and *wap*. Using kernel density estimators (bottom part), we can observe a different result. Both methods seem to be competitive, with some deviations in favour of both methods.

We can also view the choice of the density estimator as an additional parameter in the parameter tuning phase. So, if we focus on the best performance for each dataset across both tables (depicted by underlined values), only in 4 of 17 datasets, namely *la1*, *tr21*, *tr23* and *wap*, the traditional Naïve Bayes outperforms ECOC-NB

The advantage of the ECOC-based methods is more visible in the discretized versions of the dataset (Table 6). Here, larger codes always seem to work better. However, these results have to be taken with a grain of salt, as the discretization was performed on the entire dataset, and not on each individual fold, so that some information about the distribution of values in the test set could have leaked into the discretization. However, as the discretization was unsupervised (i.e., did not look at the class labels), and as we do not see any reason why this approach should prefer the decomposition-based methods, we do not think that this approach is problematic, in particular as our main goal is to evaluate the run-time of the methods.

References

Allwein, E. L., Schapire, R. E., & Singer, Y. (2000). Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1, 113–141.

- Berger, A. (1999). Error-correcting output coding for text classification. In *Proceedings of the IJCAI-99 workshop on machine learning for information filtering (IJCAI99-MLIF)*, Stockholm, Sweden.
- Bose, R. C., & Ray-Chaudhuri, D. K. (1960). On a class of error correcting binary group codes. *Information and Control*, 3(1), 68–79.
- Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2, 263–286.
- Domingos, P., & Pazzani, M. J. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3), 103–130.
- Escalera, S., Pujol, O., & Radeva, P. (2010). Error-correcting output codes library. *Journal of Machine Learning Research*, 11, 661–664.
- Fawcett, T., & Niculescu-Mizil, A. (2007). PAV and the ROC convex hull. *Machine Learning*, 68(1), 97–106.
- Flach, P. A., & Matsubara, E. T. (2007). A simple lexicographic ranker and probability estimator. In J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, & A. Skowron (Eds.), *Proceedings of the 18th European conference on machine learning (ECML-07)*, Warsaw, Poland (pp. 575–582). Berlin: Springer.
- Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3, 1289–1305.
- Friedman, J. H. (1996). Another approach to polychotomous classification. Technical Report, Department of Statistics, Stanford University, Stanford, CA.
- Fürnkranz, J. (2002). Round robin classification. *Journal of Machine Learning Research*, 2, 721–747.
- Ghani, R. (2000). Using error-correcting codes for text classification. In P. Langley (Ed.), *Proceedings of the 17th international conference on machine learning (ICML-00)* (pp. 303–310). Stanford: Morgan Kaufmann.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1), 10–18.
- Kittler, J., Ghaderi, R., Windeatt, T., & Matas, J. (2003). Face verification via error correcting output codes. *Image and Vision Computing*, 21(13–14), 1163–1169.
- Kong, E. B., & Dietterich, T. G. (1995). Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th international conference on machine learning (ICML-95)* (pp. 313–321). Stanford: Morgan Kaufmann.
- Koul, N., Caragea, C., Honavar, V., Bahirwani, V., & Caragea, D. (2008). Learning classifiers from large databases using statistical queries. In *Proceedings of the 2008 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology* (pp. 923–926).
- Macwilliams, F. J., & Sloane, N. J. A. (1983). *The theory of error-correcting codes*. Amsterdam: North-Holland.
- Melvin, I., Ie, E., Weston, J., Noble, W. S., & Leslie, C. (2007). Multi-class protein classification using adaptive codes. *Journal of Machine Learning Research*, 8, 1557–1581.
- Park, S. H., & Fürnkranz, J. (2012). Efficient prediction algorithms for binary decomposition techniques. *Data Mining and Knowledge Discovery*, 24(1), 40–77.
- Pujol, O., Radeva, P., & Vitrià, J. (2006). Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6), 1007–1012.
- Sulzmann, J. N., Fürnkranz, J., & Hüllermeier, E. (2007). On pairwise Naïve Bayes classifiers. In J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, & A. Skowron (Eds.), *Proceedings of the 18th European conference on machine learning (ECML-07)*, Warsaw, Poland (pp. 371–381). Berlin: Springer.
- Webb, G. I., Boughton, J., & Wang, Z. (2005). Not so naive Bayes: aggregating one-dependence estimators. *Machine Learning*, 58(1), 5–24.
- Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and Naïve Bayesian classifiers. In *Proceedings of the 8th international conference on machine learning (ICML-01)*, Williamstown, MA (pp. 609–616).
- Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining (KDD-02)*, Edmonton, Canada (pp. 694–699).
- Zhang, H. (2005). Exploring conditions for the optimality of Naïve Bayes. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(2), 183–198.