Ensembles of Balanced Nested Dichotomies for Multi-Class Problems

Lin Dong,[1] Eibe Frank,[1] and Stefan Kramer[2]

[1]Department of Computer Science, University of Waikato, New Zealand
{ld21, eibe}@cs.waikato.ac.nz
[2]Department of Computer Science, Technical University of Munich, Germany
kramer@in.tum.de

Abstract. A system of nested dichotomies is a hierarchical decomposition of a multi-class problem with c classes into $c-1$ two-class problems and can be represented as a tree structure. Ensembles of randomly-generated nested dichotomies have proven to be an effective approach to multi-class learning problems [1]. However, sampling trees by giving each tree equal probability means that the depth of a tree is limited only by the number of classes, and very unbalanced trees can negatively affect runtime. In this paper we investigate two approaches to building balanced nested dichotomies—*class-balanced* nested dichotomies and *data-balanced* nested dichotomies—and evaluate them in the same ensemble setting. Using C4.5 decision trees as the base models, we show that both approaches can reduce runtime with little or no effect on accuracy, especially on problems with many classes. We also investigate the effect of caching models when building ensembles of nested dichotomies.

1 Introduction

Many real-world classification problems are multi-class problems: they involve a nominal class variable that has more than two values. There are basically two approaches for tackling this type of problem. One is to adapt the learning algorithm to deal with multi-class problems directly, and the other is to create several two-class problems and form a multi-class prediction based on the predictions obtained from the two-class problems. The latter approach is appealing because it does not involve any changes to the underlying two-class learning algorithm. Well-known examples of this type of approach are error-correcting output codes [2] and pairwise classification [3], and they often result in significant increases in accuracy.

Recently, it has been shown that ensembles of nested dichotomies are a promising alternative to pairwise classification and standard error-correcting output codes. In experiments with a decision tree learner and logistic regression, their performance was less dependent on the base learner used, and they yield probability estimates in a natural and well-founded way if the base learner can generate two-class probability estimates [1].

A drawback of ensembles of nested dichotomies, at least compared to pairwise classification, is the significant increase in runtime. Although pairwise classification requires applying the base learner $c*(c-1)/2$ times for a learning problem with c classes, each learning problem is much smaller than the original problem because only data from the relevant pair of classes is considered [3]. Assuming a learning algorithm that scales linearly in the number of instances, and assuming that every class has the same number of instances, the overall runtime for pairwise classification is linear in the number of classes.[1]

Building a single system of nested dichotomies in the same setting also requires time linear in the number of classes in the worst case, but the algorithm must be applied a fixed, user-specified number of times to build an ensemble of trees (10 to 20 ensemble members were found to be generally sufficient to achieve maximum accuracy on the UCI datasets investigated in [1]).

In this paper we are looking at approaches to reducing the time required to build an ensemble of nested dichotomies (END). More specifically, we propose *class-balanced* or *data-balanced* systems of nested dichotomies (ECBNDs or EDBNDs, respectively). Using C4.5 as the base learner, we show that they can improve runtime, especially on problems with many classes, with little or no effect on classification accuracy. We also investigate the effect of caching models: the same two-class learning problem can occur multiple times in an ensemble and it makes sense to re-use two-class base models that have already been built for previous systems of nested dichotomies.

The paper is structured as follows. In Section 2 we discuss the basic method of building ENDs, our two modified versions of the algorithm (ECBNDs and EDBNDs), and the use of caching models. Section 3 presents empirical results, obtained from 21 multi-class UCI datasets, and several artificial domains with a varying number of classes. Section 4 summarizes our findings.

2 Balanced nested dichotomies

A system of nested dichotomies is a statistical model that is used to decompose a multi-class problem into multiple two-class problems (e.g. [4] introduces it as a method for performing multi-class logistic regression). The decomposition can be represented as a binary tree (Figure 1). Each node of the tree stores a set of class labels, the corresponding training data and a binary classifier. At the very beginning, the root node contains the whole set of the original class labels corresponding to the multi-class classification problem. This set is then split into two subsets. These two subsets of class labels are treated as two "meta" classes and a binary classifier is learned for predicting them. The training dataset is split into two subsets corresponding to the two meta classes and one subset of training data is regarded as the positive examples while the other subset is regarded as the negative examples. The two successor nodes of the root inherit the two subsets of the original class labels with their corresponding training

[1] Pairwise classification is actually even more beneficial when the base learner's runtime is worse than linear in the number of instances.

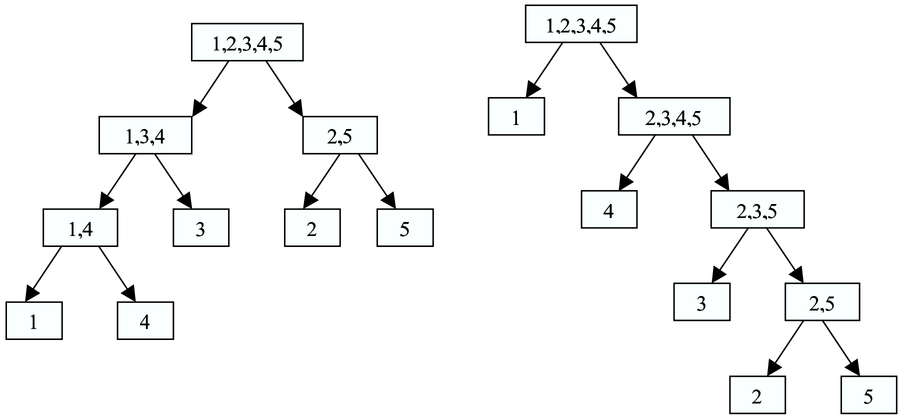

Fig. 1. Two different nested dichotomies for a classification problem with five classes.

datasets and a tree is built by applying this process recursively. The process finally reaches a leaf node if the node contains only one class label.

It is obvious that for any given c-class problem, the tree contains c leaf nodes (one for each class) and $c-1$ internal nodes. Each internal node contains a binary classifier. A nice feature of using a system of nested dichotomies for multi-class problems is that it yields class probability estimates in a straightforward fashion. Assuming the individual two-class classifiers built by the base learner produce class probability estimates for the corresponding two-class problems—one for each branch extending from the corresponding internal node—we can obtain a class probability estimate for a particular leaf node (i.e. the class label in the multi-class problem that is associated with this leaf node) by simply multiplying together the probability estimates obtained from the binary classifiers along the particular path from the root node to that leaf [1].

However, there is a problem with the application of nested dichotomies to standard multi-class problems: there are many possible tree structures for a given set of classes, and in the absence of prior knowledge about whether a particular decomposition is more appropriate, it is not clear which one to use. Figure 1 shows two different systems of nested dichotomies for a five-class problem. The problem is that two different trees will often lead to different predictions because the binary classifiers for each node are dealing with different two-class problems. The selection of the tree structure will influence the classification results. Given this observation and the success of ensemble learning in yielding accurate predictions, it makes sense to use all possible nested dichotomies for a given problem and average their probability estimates. Unfortunately this is infeasible because the number of possible systems of nested dichotomies for a c-class problem is $(2c-3)!!$ [1]. Hence it is necessary to take a subset.

Frank and Kramer [1] sample randomly from the space of all possible trees, giving each tree equal probability. However, it is not clear whether this is the best approach. In the absence of prior knowledge, any sampling scheme that does

Number of classes	Number of nested dichotomies	Number of class-balanced nested dichotomies
2	1	1
3	3	3
4	15	3
5	105	30
6	945	90
7	10,395	315
8	135,135	315
9	2,027,025	11,340
10	34,459,425	113,400
11	654,729,075	1,247,400
12	13,749,310,575	3,742,200

Table 1. Comparison of the number of possible trees.

not give preferential treatment to a particular class can be considered a suitable candidate. The problem with random sampling based on a uniform distribution over trees is that the tree depth is only limited by the number of classes, and deep trees can take a long time to build. Consider the case where the tree is a list, as in the second tree shown in Figure 1, and assume the two largest classes are separated out last (classes 2 and 5 in the example). Then all binary learning problems will involved the two largest classes, incurring a high computational cost for the process of building the binary classifiers in the tree.

2.1 Class-balanced nested dichotomies

In light of these observations we consider two different sampling strategies in this paper. The first method is based on balancing the number of classes at each node. Instead of sampling from the space of all possible trees, we sample from the space of all *balanced* trees, and build an ensemble of balanced trees. The advantage of this method is that the depth of the tree is guaranteed to be logarithmic in the number of classes. We call this an ensemble of class-balanced nested dichotomies (ECBND).

The number of possible class-balanced nested dichotomies is obviously smaller than the total number of nested dichotomies. The following recurrence relation defines the number of possible class-balanced trees:

$$T(c) = \begin{cases} \frac{1}{2}\binom{c}{c/2}T(\frac{c}{2})T(\frac{c}{2}) & : \text{ if } c \text{ is even} \\ \binom{c}{(c+1)/2}T(\frac{c+1}{2})T(\frac{c-1}{2}) & : \text{ if } c \text{ is odd} \end{cases}$$

where $T(1) = 1$ and $T(2) = 1$.

Table 1 shows the number of possible systems of nested dichotomies for up to 12 classes for the class-balanced (CBND) and the unconstrained case (ND). It shows that a non-trivial number of CBNDs can be generated for classification problems with five or more classes.

Figure 2 shows the algorithm for building a system of class-balanced nested dichotomies. At each node the set of classes is split into equal size subsets (of

```
method buildClassBalancedNestedDichotomies(Dataset D, Set of classes C)

  if |C| = 1 then return
  P = subset of C, randomly chosen from all subsets of size ⌊|C|/2⌋
  N = C \ P
  D_p = all instances in D apart from those pertaining to classes in P
  buildClassBalancedNestedDichotomies(D_p, P)
  D_n = all instances in D apart from those pertaining to classes in N
  buildClassBalancedNestedDichotomies(D_n, N)
  D' = a two-class version of D created based on N and P
  classifierForNode = buildClassifier(D')
```

Fig. 2. Algorithm for generating class-balanced nested dichotomies.

course, if the number of classes is odd, the size will not be exactly equal), and the base learning algorithm is applied to the data corresponding to these two subsets. The algorithm then recurses until only one class is left. It is applied repeatedly with different random number seeds to generate a committee of trees.

2.2 Data-balanced nested dichotomies

There is a potential problem with the class-balanced approach: some multi-class problems are very unbalanced and some classes are much more populous than others. In that case a class-balanced tree does not imply that it is also *data-balanced* (i.e. that it exhibits about the same number of instances in the two successor nodes of an internal node). This can negatively affect runtime if the base learning algorithm has time complexity worse than linear in the number of instances. Hence we also consider a simple algorithm for building data-balanced nested dichotomies in this paper. Note that this method violates the condition that the sampling scheme should not be biased towards a particular class: based on this scheme, leaf nodes for larger classes will be located higher up in the tree structure. Despite this potential drawback we decided to investigate this scheme empirically because it is difficult to say how important this condition is in practice.

Figure 3 shows our algorithm for building a system of data-balanced nested dichotomies. It randomly assigns classes to two subsets until the size of the training data in one of the subsets exceeds half the total amount of training data at the node. One motivation for using this simple algorithm was that it is important to maintain a degree of randomness in the assignment of classes to subsets in order to preserve diversity in the committee of randomly generated systems of nested dichotomies. Given that we are aiming for an ensemble of nested dichotomies it would not be advisable, even if it were computationally feasible, to aim for an optimum balance because this would severely restrict the number of trees that can be generated. Even with our simple algorithm diversity suffers when the class distribution is very unbalanced. However, it is difficult to derive a general expression for the number of trees that can potentially generated

```
method buildDataBalancedNestedDichotomies(Dataset D, List of classes C)

  if |C| = 1 then return
  C = random permutation of C
  D_p = ∅, D_n = ∅
  do
    if (|C| > 1) then
      add all instances from D pertaining to first class in C to D_p
      add all instances from D pertaining to last class in C to D_n
      remove first and last class from C
    else
      add all instances from D pertaining to remaining class in C to D_p
      remove remaining class from C
  while (|D_p| < ⌊|D|/2⌋) and (|D_n| < ⌊|D|/2⌋)
  if ((|D_p| ≥ ⌊|D|/2⌋) then
     add instances from D pertaining to remaining classes in C to D_n
  else
     add instances from D pertaining to remaining classes in C to D_p
  P = all classes present in D_p, N = all classes present in D_n
  buildDataBalancedNestedDichotomies(D_p, P)
  buildDataBalancedNestedDichotomies(D_n, N)
  D' = a two-class version of D created based on N and P
  classifierForNode = classifier learned by base learner from D'
```

Fig. 3. Algorithm for generating data-balanced nested dichotomies.

by this method because this number depends on the class distribution in the dataset.

2.3 Computational complexity

The motivation for using balanced nested dichotomies is that this reduces runtime. In the following we analyze the computational complexity of completely random and balanced nested dichotomies. Let c be the number of classes in the dataset, and n be the number of training instances. For simplicity, assume that all classes have an approximately equal number of instances in them (i.e. that the number of instances in each class is approximately n/c). We assume that the time complexity of the base learning algorithm is linear in the number of training instances, and that we can ignore the effect of the number of attributes.

In the worst case, a completely random system of nested dichotomies can degenerate into a list, and the total runtime for building a multi-class classifier based on this kind of structure and the above assumptions is

$$\sum_{i=0}^{c-2} \frac{c-i}{c} n = \frac{n}{c} \sum_{i=0}^{c-2} c - i = \frac{n}{c} \left((c-1)c - \sum_{i=0}^{c-2} i \right) = \frac{n}{c} \left((c-1)c - \frac{(c-2)(c-1)}{2} \right)$$
$$> \frac{n}{c} \left((c-1)c - \frac{c(c-1)}{2} \right) = \frac{(c-1)}{2} n.$$

Hence the worst-case time complexity is linear in the number of instances and classes.

Let us now consider the balanced case. Assuming c is even, we have $\log c$ layers of internal nodes. In each layer, all the training data needs to be processed (because the union of all subsets in each layer is the original dataset). Given that we have assumed that the base learner scales linearly in the number of instances, the overall runtime becomes $n \log c$, i.e. it is logarithmic in the number of classes and linear in the number of instances.

Assuming a base learning algorithm whose time complexity is worse than linear, the advantage of the balanced scheme becomes even more pronounced (because the size of the subsets of data considered at each node decreases more quickly in this scheme). Note also that the assumption of evenly distributed classes is not strictly necessary. This can be seen by considering the worst case, where one class has almost all the instances. The worst-case time complexity for the unbalanced case remains linear in the number of classes in this situation, and the one for the balanced case logarithmic in the number of classes.

However, in the case of a skewed class distribution it is possible to improve on the class-balanced scheme when the base learning algorithm's runtime is worse than linear. In that case it makes sense to attempt to divide the number of instances as evenly as possible at each node, so as to reduce the maximum amount of data considered at a node as quickly as possible. This is why we have investigated the data-balanced approach discussed above.

2.4 Caching models

There is a further opportunity to improve the training time for ensembles of nested dichotomies. It arises from the fact that ensemble members may share some two-class problems. Consider Figure 1. In both trees, a classifier has to be learned that separates classes 2 and 5. These classifiers will be identical because they are based on exactly the same data. It is not sensible to build them twice. Consequently we can cache models that have been built in a hash table and reduce computational complexity further.

As explained by Frank and Kramer [1], there are $(3^c - (2^{c+1} - 1))/2$ possible two-class problems for a c-class problem, i.e. growth is exponential in the number of classes. Hence we can expect that caching only makes a difference for relatively small numbers of classes. If we consider balanced dichotomies, the number of possible two-class problems is reduced. Consequently caching will be more beneficial in the balanced case.

3 Experiments

In the following we empirically investigate the effect of our proposed modifications on runtime and accuracy. We used 21 multi-class UCI datasets [5]. The number of classes varies from 3 to 26. We also performed some experiments with artificial data that exhibits a larger number of classes. For each scheme, we used

Dataset	Number of classes	Number of instances	Training time for ENDs w/o caching	with caching	
iris	3	150	0.03 ± 0.02	0.01 ± 0.00	
balance-scale	3	625	0.28 ± 0.06	0.09 ± 0.04	•
splice	3	3190	4.56 ± 0.32	1.37 ± 0.14	•
waveform	3	5000	38.78 ± 0.65	11.42 ± 0.96	•
lymphography	4	148	0.06 ± 0.02	0.04 ± 0.02	•
vehicle	4	846	1.87 ± 0.11	1.08 ± 0.16	•
hypothyroid	4	3772	3.13 ± 0.47	1.75 ± 0.29	•
anneal	6	898	0.99 ± 0.08	0.82 ± 0.13	•
zoo	7	101	0.07 ± 0.02	0.06 ± 0.02	
autos	7	205	0.40 ± 0.05	0.36 ± 0.05	
glass	7	214	0.30 ± 0.03	0.27 ± 0.03	
segment	7	2310	6.61 ± 0.27	5.87 ± 0.37	•
ecoli	8	336	0.25 ± 0.04	0.23 ± 0.04	
optdigits	10	5620	72.53 ± 3.30	68.70 ± 3.00	•
pendigits	10	10992	49.30 ± 2.00	47.07 ± 2.12	•
vowel	11	990	4.21 ± 0.11	4.04 ± 0.16	•
arrhythmia	16	452	21.14 ± 1.01	20.76 ± 1.09	
soybean	19	683	1.02 ± 0.07	0.99 ± 0.06	
primary-tumor	22	339	0.63 ± 0.06	0.63 ± 0.06	
audiology	24	226	0.74 ± 0.06	0.74 ± 0.05	
letter	26	20000	317.53 ± 11.44	315.74 ± 11.53	

Table 2. Effect of model caching on ENDs for UCI datasets.

10 ensemble members (i.e. 10 systems of nested dichotomies are generated and their probability estimates are averaged to form a prediction). J48, the implementation of the C4.5 decision tree learner [6] from the Weka workbench [7] was used as the base learner for the experiments.

All experimental results are averages from 10 runs of stratified 5-fold cross-validation (UCI datasets) or 3-fold cross-validation (artificial data). We also report standard deviations for the 50 (UCI data) or 30 (artificial data) individual estimates. Runtime was measured on a machine with a Pentium 4 3 GHz processor running the Java HotSpot Client VM (build 1.4.2_03) on Linux, and is reported in seconds. We tested for significant differences using the corrected resampled t-test [8].

3.1 Applying caching to ENDs

In this section we discuss the effect of caching individual classifiers in an ensemble of nested dichotomies. Table 2 has the average training time for ENDs with and without caching based on a hash table. Significant improvements in training time obtained by caching are marked with a •.

The results show that the runtime decreases significantly for 14 of the 21 UCI datasets (of course, accuracy remains identical). The improvement is especially obvious on datasets with a small number of classes and a large number of instances. With a small number of classes one is more likely to encounter the same

Dataset	Number of classes	Training time ENDs	ECBNDs		EDBNDs	
iris	3	0.01 ± 0.00	0.03 ± 0.02		0.02 ± 0.00	
balance-scale	3	0.09 ± 0.04	0.09 ± 0.04		0.09 ± 0.04	
splice	3	1.37 ± 0.14	1.45 ± 0.12		1.33 ± 0.20	
waveform	3	11.42 ± 0.96	11.31 ± 0.56		11.24 ± 1.10	
lymphography	4	0.04 ± 0.02	0.03 ± 0.02		0.03 ± 0.00	
vehicle	4	1.08 ± 0.16	0.51 ± 0.06	•	0.52 ± 0.05	•
hypothyroid	4	1.75 ± 0.29	0.86 ± 0.12	•	0.92 ± 0.22	•
anneal	6	0.82 ± 0.13	0.63 ± 0.09	•	0.50 ± 0.13	•
zoo	7	0.06 ± 0.02	0.06 ± 0.03		0.06 ± 0.02	
autos	7	0.36 ± 0.05	0.26 ± 0.04	•	0.25 ± 0.05	•
glass	7	0.27 ± 0.03	0.21 ± 0.04	•	0.20 ± 0.04	•
segment	7	5.87 ± 0.37	4.88 ± 0.34	•	4.98 ± 0.41	•
ecoli	8	0.23 ± 0.04	0.20 ± 0.03		0.21 ± 0.04	
optdigits	10	68.70 ± 3.00	55.17 ± 1.91	•	55.03 ± 2.16	•
pendigits	10	47.07 ± 2.12	37.95 ± 1.53	•	38.40 ± 1.52	•
vowel	11	4.04 ± 0.16	3.62 ± 0.11	•	3.70 ± 0.12	•
arrhythmia	16	20.76 ± 1.09	19.20 ± 1.08	•	17.39 ± 1.56	•
soybean	19	0.99 ± 0.06	0.87 ± 0.06	•	0.85 ± 0.07	•
primary-tumor	22	0.63 ± 0.06	0.54 ± 0.06	•	0.54 ± 0.06	•
audiology	24	0.74 ± 0.05	0.64 ± 0.08	•	0.63 ± 0.09	•
letter	26	315.74 ± 11.53	273.45 ± 16.75	•	274.07 ± 16.84	•

Table 3. Comparison of training time on UCI datasets.

binary classifier in different systems of nested dichotomies in the ensemble. With a large number of instances, more time is saved by avoiding rebuilding the same binary classifier. For instance, the training time on the waveform dataset, which has 5000 instances and only 3 classes, decreases dramatically from 38.78 seconds to 11.42 seconds by using hash tables. On the other hand, for the arrhythmia dataset, which has 16 classes and only 452 instances, the training time decreases only slightly, from 21.14 seconds to 20.76 seconds. From Table 2, we also see that there is no significant improvement for the training time when the number of classes exceeds 11. The chance to encounter the same binary classifier in those situations becomes limited as there are so many possible two-class problems. Moreover, the number of instances in these datasets (excluding the letter data) is small so that the time saved by using hash tables is not noticeable. We also performed experiments with artificial data with even more classes and there was essentially no difference in runtime on that data.

3.2 Comparing ENDs, ECBNDs, and EDBNDs

As we have seen, caching does not help when there are many classes. In the following we will see that using balanced nested dichotomies helps in those cases. We will first look at training time and then the effect on accuracy.

Training time Table 3 shows the training times for ENDs, class-balanced ENDs (ECBNDs), and data-balanced ENDs (EDBNDs), on the UCI datasets. Model

Number of classes	Number of instances	Training time ENDs	ECBNDs	EDBNDs
10	820	0.60 ± 0.09	0.58 ± 0.07	0.58 ± 0.07
20	1390	1.50 ± 0.12	1.42 ± 0.08	1.44 ± 0.09
30	1950	2.72 ± 0.11	2.31 ± 0.12 •	2.33 ± 0.12 •
40	2410	3.87 ± 0.16	3.18 ± 0.14 •	3.24 ± 0.13 •
50	3090	5.55 ± 0.23	4.54 ± 0.17 •	4.57 ± 0.20 •
60	3660	7.48 ± 0.29	5.86 ± 0.17 •	5.90 ± 0.25 •
70	4560	10.41 ± 0.36	8.23 ± 0.28 •	8.35 ± 0.30 •
80	5010	12.32 ± 0.43	9.56 ± 0.33 •	9.67 ± 0.31 •
90	5840	15.75 ± 0.53	12.62 ± 0.44 •	12.78 ± 0.34 •
100	6230	18.25 ± 0.61	13.61 ± 0.38 •	13.98 ± 0.44 •
150	9590	40.65 ± 1.90	27.63 ± 0.77 •	28.19 ± 0.65 •
200	12320	66.41 ± 2.95	42.37 ± 1.30 •	42.70 ± 1.30 •

Table 4. Comparison of training time on artificial datasets.

caching was applied in all three versions of ENDs. A • indicates a significant reduction in runtime compared to ENDs.

The results show that using class-balanced nested dichotomies results in significantly reduced training times on 14 of the 21 datasets. Using the data-balanced scheme also helps: EDBNDs are significantly more efficient than ENDs on 14 datasets, just like ECBNDs. Compared to class-balanced trees, data-balanced trees are significantly more efficient on one dataset (arrhythmia). (This information is not included in Table 3.) This dataset has an extremely unbalanced class distribution and this is why the data-balanced approach helps.

The advantage of the balanced schemes is restricted to datasets with more than 3 classes. On three-class datasets, all nested dichotomies are class-balanced, so we would not expect any significant difference between ENDs and ECBNDs. The experimental results bear this out.

Table 4 shows the training times for our 12 artificial datasets. To generate these datasets we used a cluster generator and varied the number of clusters from 10 to 200. Instances in the same cluster were assigned the same class label. Each instance in these datasets consists of one boolean attribute and two numeric attributes. The attribute value ranges were set to be different but could overlap. Attribute values were generated randomly within each cluster. The number of instances in each cluster (i.e. class) was also randomly generated and varied between 20 and 110.

The results on the artificial datasets show that the balanced schemes exhibit a significant advantage in terms of running time when 30 or more classes are present in the data. There was no significant difference in running time for the two balanced schemes (class-balanced vs. data-balanced) on any of the datasets. This indicates that the class distribution in our artificial datasets is not skewed enough for the data-balanced approach to help.

Accuracy Improvements in runtime are less useful if they affect accuracy in a significant fashion. Hence it is important to evaluate the effect of our proposed

Dataset	Number of	Percent correct		
	classes	ENDs	ECBNDs	EDBNDs
iris	3	94.13 ± 3.84	94.13 ± 3.72	94.27 ± 3.81
balance-scale	3	79.92 ± 2.37	79.49 ± 2.41	79.78 ± 2.31
splice	3	94.75 ± 1.01	94.55 ± 0.98	93.07 ± 1.33 •
waveform	3	77.89 ± 1.88	77.53 ± 1.91	77.85 ± 2.06
lymphography	4	77.73 ± 7.47	76.63 ± 6.35	76.90 ± 6.93
vehicle	4	73.20 ± 2.92	72.36 ± 2.30	72.36 ± 2.30
hypothyroid	4	99.54 ± 0.26	99.51 ± 0.27	99.54 ± 0.28
anneal	6	98.63 ± 0.80	98.44 ± 0.75	98.53 ± 0.62
zoo	7	93.66 ± 5.67	93.87 ± 4.61	93.88 ± 4.50
autos	7	76.20 ± 6.11	74.83 ± 6.62	75.32 ± 7.10
glass	7	72.82 ± 7.42	73.51 ± 6.17	72.25 ± 6.84
segment	7	97.45 ± 0.83	97.35 ± 0.80	97.39 ± 0.87
ecoli	8	85.60 ± 4.11	85.36 ± 4.06	84.88 ± 4.13
optdigits	10	96.99 ± 0.49	97.14 ± 0.45	97.18 ± 0.50
pendigits	10	98.59 ± 0.27	98.76 ± 0.25	98.76 ± 0.26
vowel	11	88.31 ± 2.66	89.98 ± 2.47	89.24 ± 2.79
arrhythmia	16	72.59 ± 3.24	72.82 ± 4.11	71.51 ± 3.55
soybean	19	93.90 ± 1.63	94.49 ± 1.69	94.36 ± 1.78
primary-tumor	22	44.72 ± 5.04	46.28 ± 4.61	45.96 ± 4.62
audiology	24	78.46 ± 5.44	79.66 ± 5.12	79.48 ± 5.23
letter	26	94.33 ± 0.37	94.50 ± 0.36	94.51 ± 0.35

Table 5. Comparison of accuracy on UCI datasets.

modifications on accuracy. Table 5 shows the estimated accuracy for ENDs, ECBNDs, and EDBNDs on the UCI datasets. We can see that there is no dataset with a significant difference in accuracy for ENDs and ECBNDs. This is the desired outcome. For EDBNDs, there is one three-class dataset (splice) where the accuracy is significantly reduced compared to ENDs. The splice data has a skewed class distribution, where one class has about half the instances and the rest is evenly distributed among the remaining two classes. We measured the diversity of the three types of ensembles on this dataset using the kappa statistic. This statistic can be used to measure agreement between pairs of ensemble members [9]. For EDBNDs, the mean kappa value over all pairs, measured on the training data, was 0.96, which was indeed higher than the mean kappa values for ENDs and ECBNDs (0.94 and 0.93 respectively). This indicates that reduction in diversity is the reason for the drop in performance.

Table 6 shows the same information for the artificial datasets. In this case there is not a single dataset where there is a significant difference in accuracy between any of the schemes.

4 Conclusions

Ensembles of nested dichotomies have recently been shown to be a very promising meta learning scheme for multi-class problems. They produce accurate classifications and yield class probabilities estimates in a natural way. In this paper

Number of classes	Number of instances	Percent correct		
		ENDs	ECBNDs	EDBNDs
10	820	78.08 ± 1.94	78.34 ± 2.35	78.32 ± 2.32
20	1390	77.79 ± 1.87	77.21 ± 1.44	77.47 ± 1.66
30	1950	77.09 ± 1.61	76.93 ± 1.53	76.85 ± 1.46
40	2410	76.64 ± 1.24	76.56 ± 1.39	76.46 ± 1.24
50	3090	76.26 ± 1.09	76.17 ± 1.26	76.25 ± 1.19
60	3660	76.43 ± 1.08	76.33 ± 1.04	76.37 ± 0.95
70	4560	73.58 ± 1.12	73.27 ± 0.97	73.50 ± 0.90
80	5010	75.85 ± 1.06	75.61 ± 0.94	75.71 ± 0.87
90	5840	76.41 ± 0.84	76.40 ± 0.91	76.41 ± 0.87
100	6230	76.59 ± 0.77	76.54 ± 0.73	76.50 ± 0.85
150	9590	75.92 ± 0.66	75.89 ± 0.72	75.86 ± 0.62
200	12320	75.89 ± 0.51	75.67 ± 0.51	75.73 ± 0.49

Table 6. Comparison of accuracy on artificial datasets.

we have shown that it is possible to improve the runtime of this meta learning scheme without affecting accuracy. A simple way to improve runtime for problems with a small number of classes is to cache two-class models and re-use them in different members of an ensemble of nested dichotomies. On problems with many classes we have shown that using class-balanced nested dichotomies significantly improves runtime, with no significant change in accuracy. We have also presented a data-balanced scheme that can help to improve runtime further when there are many classes and the class distribution is highly skewed.

References

1. Frank, E., Kramer, S.: Ensembles of nested dichotomies for multi-class problems. In: Proc Int Conf on Machine Learning, ACM Press (2004) 305–312
2. Dietterich, T., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research **2** (1995) 263–286
3. Fürnkranz, J.: Round robin classification. Journal of Machine Learning Research **2** (2002) 721–747
4. Fox, J.: Applied Regression Analysis, Linear Models, and Related Methods. Sage (1997)
5. Blake, C., Merz, C.: UCI repository of machine learning databases. University of California, Irvine, Dept. of Inf. and Computer Science (1998) [www.ics.uci.edu/∼mlearn/MLRepository.html].
6. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Los Altos, CA (1992)
7. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (2000)
8. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning **52** (2003) 239–281
9. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40** (1998) 139–157