Statistical Instance-based Ensemble Pruning for Multi-class Problems

Gonzalo Martínez-Muñoz[1,2], Daniel Hernández-Lobato[1], and Alberto Suárez[1]

[1] Universidad Autónoma de Madrid, EPS, Madrid 28049 Spain
[2] Oregon State University, Corvallis, OR 97331 USA

Abstract. Recent research has shown that the provisional count of votes of an ensemble of classifiers can be used to estimate the probability that the final ensemble prediction coincides with the current majority class. For a given instance, querying can be stopped when this probability is above a specified threshold. This instance-based ensemble pruning procedure can be efficiently implemented if these probabilities are pre-computed and stored in a lookup table. However, the size of the table and the cost of computing the probabilities grow very rapidly with the number of classes of the problem. In this article we introduce a number of computational optimizations that can be used to make the construction of the lookup table feasible. As a result, the application of instance-based ensemble pruning is extended to multi-class problems. Experiments in several UCI multi-class problems show that instance-based pruning speeds-up classification by a factor between 2 and 10 without any significant variation in the prediction accuracy of the ensemble.

Key words: Instance based pruning, ensemble learning, neural networks

1 Introduction

Ensemble methods generate a collection of diverse classifiers by introducing variations in the algorithm used to train the base predictors or in the conditions under which learning takes place [1–5]. The classification of an unlabeled instance by the ensemble is obtained by combining the predictions of the individual classifiers. In majority voting, each classifier in the ensemble is asked to predict the class label of the instance considered. Once all the classifiers have been queried, the class that receives the greatest number of votes is returned as the final decision of the ensemble. The time needed to classify an instance increases linearly with the size of the ensemble. However, in many cases, it is not necessary to compute the predictions of every classifier to obtain a reliable estimate of the prediction of the complete ensemble. Assuming that the classifiers of the ensemble are generated independently when conditioned to the training data[1],

[1] Note that this is different from assuming that the classifiers are unconditionally independent.

the class labels predicted by the ensemble members can be seen as independent identically distributed random variables. With this assumption, it is possible to compute the probability that the current majority class coincides with the final ensemble prediction, on the basis of the known class votes. Therefore, for a given instance, the querying process can be halted when this probability is above a specified threshold. This dynamical instance-based ensemble pruning can be used in practice to speed-up the classification process [6, 7].

For statistical IB-pruning to be effective, the probability that the final decision of the ensemble will not change when taking into account the remaining votes needs to be rapidly computed. A possible implementation is to store precomputed values of these probabilities in a lookup table. The difficulty is that both the size of the table and the cost of evaluating the probability values grow very quickly with the number of class labels. Hence, applying IB-pruning to classification problems with more than two classes remains a challenging task.

In this work we introduce several techniques that can be applied to reduce the size of the lookup table and to optimize the process of computing these probability values. With the proposed optimizations, IB-pruning can be effectively used to reduce the time of classification by ensembles also in multi-class problems. Specifically, in this work problems with up to 11 different classes are considered. In the problems investigated, the number of classifiers queried is reduced by a factor between ≈ 2 and ≈ 10. The empirical rates of disagreement between the class predicted by the dynamically pruned ensembles and the complete ones are close to the confidence level specified and often below it. Furthermore, the actual differences in classification error between these two ensembles are very small; in fact, much smaller than the disagreement rates.

The article is structured as follows: Statistical instance-based ensemble pruning is briefly reviewed in Section 2. Section 3 describes the optimizations that make the application of IB-pruning to problems with more than two classes practicable. Section 4 illustrates the improvements in classification speed that can be achieved when IB-pruning is used in a variety of multi-class problems. Finally, the conclusions of this work are summarized in Section 5.

2 Statistical Instance-Based Ensemble Pruning

Consider an ensemble composed of T classifiers $\{h(\mathbf{x})_i\}_{i=1}^{T}$. The class assigned to an unlabeled instance, $\mathbf{x}$, when majority voting is used to combine the outputs of the ensemble classifiers is

$$\arg\max_{y} \sum_{t=1}^{T} \mathcal{I}(h_t(\mathbf{x}) = y), \quad y \in \mathcal{Y}, \tag{1}$$

where $h_t(\mathbf{x})$ is the prediction of the t-th ensemble member, $\mathcal{I}$ is an indicator function and $\mathcal{Y} = \{y_1, \ldots, y_l\}$ is the set of possible class labels.

The predictions of the classifiers in the ensemble can be summarized in the vector $\mathbf{T} \equiv \{T_1, T_2, \ldots, T_l; \sum_{i=1}^{l} T_i = T\}$ where T_i is the number of members in the ensemble that predict class y_i for the instance to be classified. Similarly, the vector that stores the prediction of the first $t \leq T$ classifiers in the ensemble is $\mathbf{t} \equiv \{t_1, t_2, \ldots, t_l;\ \sum_{i=1}^{l} t_i = t;\ t_i \leq T_i, i = 1, 2, \ldots, l\}$. Assuming the individual classifiers of the ensemble are built independently when conditioned to the training data, the probability that the class labels predicted by the subensemble of size $t < T$ and by the complete ensemble of size T coincide is

$$\tilde{\mathcal{P}}(\mathbf{t}, T) = \sum_{\mathbf{T}}{}^{*} \left[\frac{(T-t)!}{(t+l)_{T-t}} \prod_{i=1}^{l} \frac{(t_i+1)_{T_i-t_i}}{(T_i-t_i)!} \right], \qquad (2)$$

where $(a)_n = a(a+1)\cdots(a+n-1)$ is the Pocchammer symbol, or rising factorial, with a and n nonnegative integers, and the asterisk indicates that the summation runs over all values of $\mathbf{T}$ such that $\sum_{i=1}^{l} T_i = T$, $\{T_i \geq t_i,\ i = 1, 2, \ldots, l\}$, and $T_{k_{\mathbf{t}}^*} > T_j$ for $j \neq k_{\mathbf{t}}^*$, where $k_{\mathbf{t}}^*$ is the majority class after querying the first t classifiers. See [6] for further details.

If it is acceptable that the coincidence between these two predictions is not necessarily certain, but occurs with a high confidence level π, (2) can be used to stop the querying process after the predictions of t classifiers in the ensemble are known, when the vector of class predictions of the current subensemble $\mathbf{t}$ is such that $\tilde{\mathcal{P}}(\mathbf{t}, T) \geq \pi$. Since the fraction of examples that are assigned a different class label by the pruned ensemble of size t and the complete subensemble is at most $1 - \pi$, the differences in error should be smaller than this disagreement rate. In practice, the changes in class labels affect both correctly labeled examples and misclassified examples in approximately equal numbers, and the differences in error rates are much smaller than this upper bound. Therefore, for a given instance the partial ensemble prediction can be used instead of the complete ensemble to save time in the classification process at the expense of small differences in the assignment of class labels, which generally do not translate into large differences in generalization performance.

To determine whether the polling process can be stopped it is necessary to know the values of $\tilde{\mathcal{P}}(\mathbf{t}, T)$, whose computation can be costly. Since for a given number of classes l and ensemble size T, these probabilities only depend on $\mathbf{t}$, they can be pre-computed and stored in a lookup table. The difficulty is that for $l < T$ both the size of the lookup table that stores pre-computed probability values and the cost of computing each entry in the table are nearly exponential in the number of different class labels l. The number of different entries in this table is $\binom{T+l}{T}$, namely the number of different ways in which T objects can be assigned to $l+1$ classes. The extra class corresponds to the unknown predictions of the classifiers that have not been queried.

The cost of evaluating (2) depends on the number of vectors $\mathbf{T}$ involved in the summation. This number is different for each entry of the table and is given by the number of values of $\mathbf{T}$ such that $T_{k_{\mathbf{t}}^*}$ is the majority class. As a consequence of the constraints $T_i \geq t_i$ its value decreases as the value of the components of $\mathbf{t}$ increase. Thus, the maximum number of different vectors $\mathbf{T}$, $C_{max}(T, l)$, is obtained for the entry corresponding to $\mathbf{t} = \mathbf{0}$. This is the worst possible scenario which sets an upper bound on the time-complexity of evaluating (2).

The value of $C_{max}(T,l)$ can be expressed in terms of $N(n,r,m)$, the number of different ways of decomposing a positive integer n into r non-negative integers that are all smaller or equal than m,

$$n_1 + n_2 + \cdots + n_r = n \qquad n_i \leq m \quad i = 1, 2, \ldots, r\,.$$

If $m \geq n$ the problem reduces to computing the different ways of assigning n objects to r classes with repetitions allowed, namely $\binom{n+r-1}{n}$. Otherwise, it is given by the formula

$$N(n,r,m) = \begin{cases} \displaystyle\sum_{i=0}^{\lfloor n/(m+1) \rfloor} (-1)^i \binom{r}{i} \binom{n - i(m+1) + r - 1}{n - i(m+1)} & \text{if } n \leq mr \\ 0 & \text{otherwise} \end{cases}\,. \tag{3}$$

To calculate $C_{max}(T,l)$ we need to count the number of different vectors $\mathbf{T}$ involved in the summation in which the minority votes can be distributed among the $l-1$ minority classes. Assume that the majority class gets i votes. The number of different ways of distributing the remaining votes among the remaining $l-1$ classes, so that all of the minority classes receive less than i votes is $N(T-i, l-1, i-1)$. Summing over all possible values of i, we obtain

$$C_{max}(T,l) = \sum_{i=\lceil T/l \rceil}^{T} N(T-i, l-1, i-1). \tag{4}$$

Figure 1 (left) displays in log scale the size of the table and the maximum number of operations needed to calculate each element in the table ($C_{max}(T,l)$) as the number of class labels l increases for $T = 101$. These two numbers grow very rapidly with the number of classes l. The increase from $l-1$ classes to l classes corresponds in both cases to a multiplicative factor of $\approx T/l$. In consequence, the problem quickly becomes intractable even for relatively small numbers of classes. For $T = 101$ and $l = 2, \ldots, 7$, the maximum number of vectors $\mathbf{T}$ involved in the summation (and table sizes) are: 51 (5253), 1734 (182104), 44867 (4780230), 937747(101340876),16489227(1807245622), 250881300(27883218168), respectively. Thus, even for $l = 3$, building the lookup table is costly: The table has $182,104$ entries and computing each entry demands as much as $1,734$ operations whose time-complexity is $\mathcal{O}(l)$.

3 Optimizations

In this section we describe several exact optimizations that can be applied to build the lookup table in a more efficient way[3]. First, each of the terms in the summation in (2) can be computed faster if the factors that are repeatedly used

[3] Source code available at `http://www.eps.uam.es/~gonzalo/publications/`

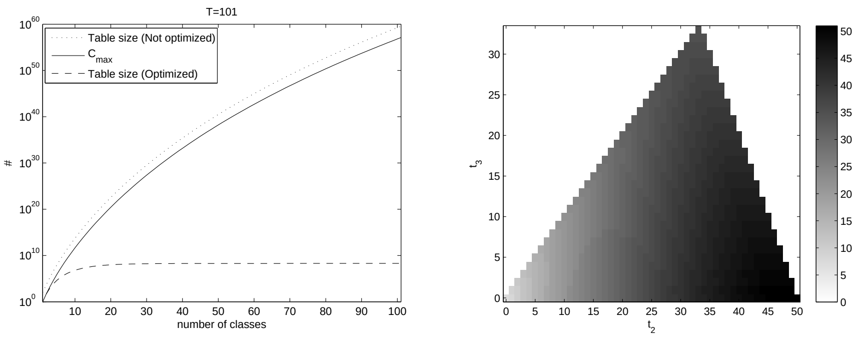


Fig. 1. (*Left*) Size of the lookup table (with and without optimizations) and complexity of the computation of the table entries ($C_{max}(T, l)$) as a function of number of classes of the lookup or $T = 101$. (*Right*) Lookup table for $T = 101$, $\pi = 0.99$ and $l = 3$

in the calculation are stored in auxiliary tables. Second, the size of the lookup table can also be reduced by storing the minimal amount of information that is needed to reach a decision on when to stop querying classifiers. Finally, some of the terms contribute with the same value to the sum in (2). Therefore, it is sufficient to compute this value only once and then multiply the result by the number of equivalent terms.

The calculation of each term inside the summation in (1) requires the computation of the product of one factor of the form

$$(T - t)!/(t + l)_{T-t} \ . \tag{5}$$

and of l factors of the form

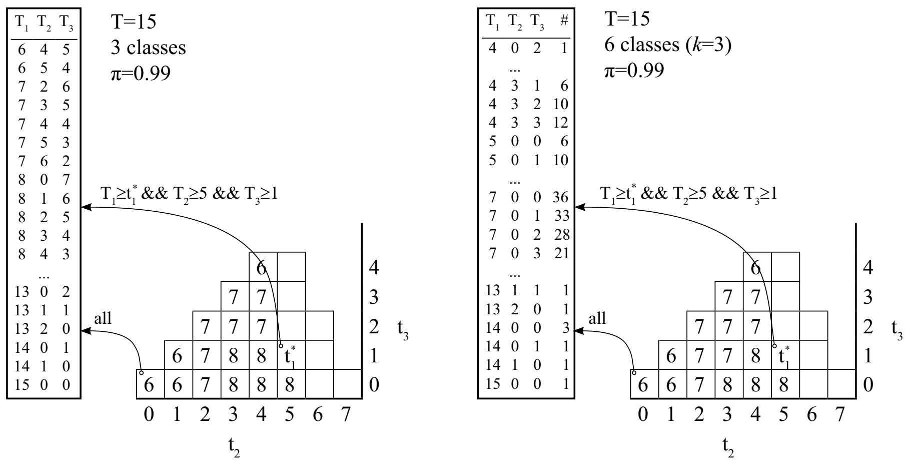
$$(t_i + 1)_{T_i - t_i}/(T_i - t_i)! \tag{6}$$

The number of different values that these factors can have is determined by T_i, t_i and t. The variables T_i and t take values in the range $[0, T]$ and $t_i \leq T_i$. Therefore, for a given T there are only $(T + 1)T/2$ different outcomes for (6) and $T + 1$ for (5). This is a fairly small number of values that can be easily precomputed and stored in an auxiliary table. Thus, each term in the summation that appears in (2) can be readily computed as a product of l of these factors. In addition, to avoid numerical difficulties with the computations that involve factorials, (6) and (5) are calculated using the prime-factor decomposition of the numbers involved.

Regarding the reduction of the size of the lookup table, the following optimizations can be made: Instead of storing probabilities, it is sufficient to generate a table whose entries correspond to the distribution of votes in the minority classes only. The value stored in the table for each entry is the minimum number of votes for the majority class needed to estimate the complete ensemble prediction with a confidence level π, given the observed minority classes. The size of the lookup table can be further reduced if the votes in $\mathbf{t}$ are kept in memory sorted in decreasing order. The overhead associated to keeping the votes sorted when new predictions are known is very small, of $O(l)$ per classifier queried. Thus, the first element in $\mathbf{t}$ corresponds to the currently observed majority class. The remaining observations correspond to the minority classes which are used as indexes for the lookup table. Because $t_i \geq t_{i+1}$ for all i, the value of t_i is at most $\lceil T/i \rceil - 1$ for $i = 1, 2, \ldots, l$.

The dependence of the size of the table on the number of classes with and without the optimizations described is shown in the left plot of Fig. 1 for an ensemble of size $T = 101$. The right plot of Fig. 1 displays the lookup table for $l = 3$, $T = 101$ and $\pi = 99\%$. The value stored in the position of the table labeled by the pair (t_2, t_3) corresponds to the minimum number of votes of the majority class $t_1^*(t_2, t_3)$ that are needed to anticipate the ensemble prediction with a confidence level $\pi = 99\%$ when the two minority classes have received t_2 and t_3 votes, respectively. The triangular shape of the table is given by the constrains $t_1 > t_2 \geq t_3$. The critical values of the majority class are plotted using an inverted gray scale: a darker color indicates that higher values for the majority class are needed to stop the voting process, according to the scale displayed on the right-hand side of the figure.

The optimizations introduced thus far allow to compute lookup tables for small and intermediate values of l, the total number of classes. In problems with a larger number classes, the rapid growth in the number of terms that need to be considered makes the calculation of (2) unfeasible. Nevertheless, this computation is manageable for instances that during the classification process have received votes in at most k classes. Instances that have votes in more than k classes can be classified by querying all the classifiers in the ensemble, without pruning. An instance that has received votes in only k classes when t classifiers have been queried is characterized by a vector $\mathbf{t}$ whose $l-k$ last components are zero: $t_{k+1} = t_{k+2} = \ldots = t_l = 0$. For these instances, it is sufficient to compute a lookup table with the same dimensions as a table designed for a problem with k classes. Note that if $t_i = 0$ then (6) is equal to one independently of the value of T_i. This observation can be used to simplify the computation of (2) in two ways: First, in each of the terms in the summation (2) it is not necessary to multiply by the factors of the form (6) for classes $i = k+1, k+2, \ldots, l$, because they are all equal to one. Second, when $\mathbf{t} = \{t_1, t_2, \ldots, t_k, 0, \ldots, 0\}$ all terms $\mathbf{T}$ of the form $\{T_1, \ldots, T_k, *, \ldots, *\}$ contribute with the same value in (2). For a particular value of $T_1, T_2, \ldots, T_k$, the number of terms $\mathbf{T}$ of the form $\{T_1, \ldots, T_k, *, \ldots, *\}$, is $N(T - \sum_{i=1}^{k} T_i, l-k, T_1 - 1)$; that is, the number of ways in which the remaining votes, $T - \sum_{i=1}^{k} T_i$, can be decomposed among the remaining $l-k$ classes so that none of these classes receives more votes than T_1. Thus, the terms of the form $\{T_1, \ldots, T_k, *, \ldots, *\}$ can be grouped and their contribution in (2) calculated by computing the value of one of them and then multiplying by their count $N(T - \sum_{i=1}^{k} T_i, l-k, T_1 - 1)$. Using this optimization the actual number of different terms that need to be computed in (2) grows



T_1	T_2	T_3
6	4	5
6	5	4
7	2	6
7	3	5
7	4	4
7	5	3
7	6	2
8	0	7
8	1	6
8	2	5
8	3	4
8	4	3
...		
13	0	2
13	1	1
13	2	0
14	0	1
14	1	0
15	0	0

T_1	T_2	T_3	#
4	0	2	1
...			
4	3	1	6
4	3	2	10
4	3	3	12
5	0	0	6
5	0	1	10
...			
7	0	0	36
7	0	1	33
7	0	2	28
7	0	3	21
...			
13	1	1	1
13	2	0	1
14	0	0	3
14	0	1	1
14	1	0	1
15	0	0	1

Fig. 2. Lookup table computation process for $T = 15$ and $l = 3$ (*left*) and for $T = 15$, $l = 6$ and $k = 3$ (*Right*)

slowly with l for fixed k. For example, for $T = 101$ and $k = 3$ the number of final vectors $\mathbf{T}$ to be computed grow from 44867 with $l = 4$ to 59821 with $l = 101$.

Fig. 2 illustrates the process of construction of the lookup table for $T = 15$, $l = 3$ and $k = 3$ (left part of the figure) and for $T = 15$, $l = 6$ and $k = 3$ (right part). On the left-hand side the list of final vectors $\mathbf{T}$ that have to be analyzed during the computation of the lookup table are shown. Note that not all of them have to be taken into account to compute every entry of the table. For instance, to calculate the entry $(t_2 = 0, t_3 = 0)$ all combinations (T_1, T_2, T_3) are needed. For entry $(t_2 = 5, t_3 = 1)$ only combinations such that $T_2 \geq 5$ and $T_3 \geq 1$ are used. In the case of $l = 6$ classes and $k = 3$ (right part of the figure) the procedure for computing the table is equivalent except that each term is multiplied by the number of equivalent vectors $\mathbf{T}$ that contribute with the same value (fourth column of the list of combinations, under the header #).

4 Experiments

The performance of the instance-based pruning procedure described is illustrated in experiments on benchmark multi-class problems from [8]. For each problem, we generate 100 random partitions of the data into two disjoint sets that contain 2/3 and 1/3 of the instances, respectively. The first set is used for constructing the ensembles while the second one is used for evaluation. For each of these partitions, we build a bagging ensemble composed of 101 neural networks [1]. In bagging, each network is trained on a different bootstrap sample of the data available. The neural networks are single layer feed forward networks with 5 units in the hidden layer and *soft-max* outputs. They are trained using a maximum of 1,000 epochs using the quasi-Newton method BFGS. These choices were

Table 1. Results for the analyzed datasets

	classes	Disagreement	Classification Error		# of NN	(Q1	Q2	Q3)
Dataset	(k)	(in %)	IB-Pruning	No pruning				
Dermatology	6	0.034±0.166	2.82±1.31	2.80±1.31	10.9	(8.0	8.0	10.0)
DNA	3	0.090±0.108	5.20±0.68	5.19±0.68	12.0	(7.0	7.0	9.0)
E-coli	8	0.196±0.485	23.84±11.22	23.80±11.27	34.7	(8.0	11.0	63.0)
	($k = 5$)	18.48% instances with votes for more than 5 classes						
Glass	6	0.310±0.651	31.18±5.01	31.17±4.93	31.4	(10.0	18.0	45.0)
Iris	3	0.080±0.394	4.78±2.59	4.82±2.68	9.8	(7.0	7.0	7.0)
LED24	10	0.149±0.139	28.40±1.79	28.40±1.79	21.9	(8.0	8.0	14.0)
	($k = 5$)	9.24% instances with votes for more than 5 classes						
New-Thyroid	3	0.056±0.274	3.94±2.13	3.94±2.11	9.7	(7.0	7.0	7.0)
Satellite	6	0.128±0.068	12.23±0.68	12.23±0.68	14.8	(8.0	8.0	10.0)
Segment	7	0.042±0.094	2.91±0.55	2.90±0.55	11.6	(8.0	8.0	8.0)
	($k = 5$)	0.74% instances with votes for more than 5 classes						
Vehicle	4	0.408±0.395	17.82±2.04	17.79±2.01	24.1	(7.0	11.0	29.0)
Vowel	11	0.364±0.305	23.32±2.44	23.24±2.40	50.2	(14.0	31.0	101.0)
	($k = 5$)	29.99% instances with votes for more than 5 classes						
Waveform	3	0.284±0.169	16.87±1.28	16.85±1.28	18.7	(7.0	9.0	17.0)
Wine	3	0.068±0.334	2.19±1.89	2.15±1.91	9.4	(7.0	7.0	7.0)

made so that a good overall accuracy is obtained in the classification problems investigated. The *nnet* R package [9] is used to train the neural networks. The performance of the ensembles is estimated on the test set. For each instance to be classified, the networks in the ensemble are queried at random without repetition. The vector of votes $\mathbf{t}$ is updated so that its elements remain sorted. The querying process is stopped when t_1 is equal or greater than the entry of the lookup table corresponding to the minority classes $\{t_2, \ldots, t_l\}$. For problems with $l \leq 6$ the complete lookup table for all possible values of $\mathbf{t}$ is computed. For problems with more than six classes, $k = 5$ is used to compute the lookup table. In these problems, if an instance receives votes for more than five classes after t queries, then the full ensemble is used to predict its class label. All the lookup tables are computed using $\pi = 99\%$ and $T = 101$.

Table 1 summarizes the results for the datasets investigated. The values reported correspond to averages over the 100 random partitions into training and test data. The standard deviations are also reported after the ± symbols. The second column of the table displays l, the number of classes of each problem and, if different, the k value used to generate the table. If $k \neq l$, the percentage of instances that require querying all the networks in the ensemble is also indicated. The third column displays the average disagreement rates between the predictions of the full ensemble and the predictions of the ensemble when IB-pruning is used. The generalization error and standard deviation of these predictions are given in the fourth and fifth columns, respectively. Finally, the sixth column displays the average number of neural networks used by IB-pruning to classify each instance and the quartiles between parenthesis.

Table 2. Execution time

l	k	time (sec.)	C_{max}	table size
3	3	0.76	1,734	884
4	4	13.9	44,867	8,037
5	5	758.4	937,747	46,262
6	6	24,757	16,489,227	189,509
6	5	10,126	16,489,227	46,262
7	5	15,037	18,798,419	46,262
8	5	18,579	19,398,244	46,262
9	5	19,279	19,605,911	46,262
10	5	20,143	19,692,158	46,262
11	5	20,270	19,732,838	46,262
12	5	20,417	19,754,960	46,262

The disagreement rates between the pruned and the complete ensemble class estimates are under $1 - \pi$ in all the problems investigated. Furthermore, the differences in classification error between the complete ensemble and the IB-pruned ensemble are almost negligible, well below the disagreement rates. This means that the changes in the class labels occur in approximately the same numbers of correctly and incorrectly classified instances. A paired two-sided Wilcoxon sign test at 5% indicates that these differences in classification error are not statistically significant in any of the problems investigated. The average number of neural networks used to classify the different instance varies significantly in different problems. This value is below 10 on average for *Iris*, *New-Thyroid* and *Wine*. For *Vowel* approximately half of the networks need to be queried on average. These figures indicate an increment of the classification speed in a factor that varies between 2 for *Vowel* and 10.7 for *Wine*. The smaller improvement for *Vowel* is due to the fact that 30.2% of instances have votes for more than 5 classes and, in consequence, they are classified using all the classifiers in the ensemble, i.e. 101 neural networks.

Table 2 displays the costs of constructing the lookup table for different values of l and k. The computations have been performed in a 2.4 Ghz Intel Xeon processor. The times reported in the second column of the table are given in seconds. The third column displays the values of $C_{\max}$, the number of terms in the sum in (2) when $\mathbf{t} = \mathbf{0}$. Finally, the last column gives the sizes of the lookup tables constructed. The time needed to compute the lookup table grows very quickly with l, when $k = l$ (first four rows of the table). This is mainly due to the fast increase in the number of terms in the sum in (2), and, to a lower extent, to the larger table size. If the value k is kept fixed (last seven rows of the table), the size of the table remains constant as l increases. In addition, the growth of the maximum number of terms in the sum in (2) is much slower. The combination of these factors leads to a fairly slow increment in the time needed to build the table for increasing values of l and constant k.

5 Conclusions

Majority voting is often used to combine the decisions of the classifiers that make up an ensemble. In most cases, it is not necessary to query all classifiers in the ensemble to estimate the final class prediction with a high level of confidence. Assuming that, for a given instance, the predictions of the classifiers are independent random variables, it is possible to calculate the probability that the majority class obtained after querying t classifiers will not change after querying the remaining $T-t$ classifiers of the ensemble. If some uncertainty in the estimation of the class prediction of the complete ensemble is acceptable, the querying process can be stopped when this probability exceeds a specified threshold π, typically high, but smaller than 1. Because less classifiers are queried, this instance-based pruning procedure can be used to increase the classification speed, provided that the values of these probabilities are readily available. A possible solution is to compute them and then store them in a lookup table. The difficulty is that the size of the lookup table and the cost of computing the values to be stored in it grow very fast as the number class labels of the problem increases. In this article we propose several techniques that can be used to optimize this process, and allow the application of IB-pruning in multi-class problems. Experiments in problems with up to 11 classes show that the classification speed can be improved by a factor between ≈ 2 and ≈ 10, depending on the problem considered, without any significant variation in the classification error of the ensemble. The optimized tables are smaller and can be stored in the working memory of a standard desktop computer. They can be computed off-line for a given ensemble size, number of classes (l) and value of k and can then be used for *any* classification task and ensemble type, provided that the classifiers are trained independently.

References

1. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
2. Breiman, L.: Random forests. Machine Learning **45**(1) (2001) 5–32
3. Martínez-Muñoz, G., Suárez, A.: Switching class labels to generate classification ensembles. Pattern Recognition **38**(10) (2005) 1483–1494
4. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(10) (2006) 1619–1630
5. Martínez-Muñoz, G., Sánchez-Martínez, A., Hernández-Lobato, D., Suárez, A.: Class-switching neural network ensembles. Neurocomputing **71**(13-15) (2008) 2521–2528 Artificial Neural Networks (ICANN 2006).
6. Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A.: Statistical instance-based pruning in ensembles of independent classifiers. IEEE Transanctions on Pattern Analysis and Machine Intelligence **31**(2) (2009) 364–369
7. Fan, W.: Systematic data selection to mine concept-drifting data streams. In: KDD'04, ACM (2004) 128–137
8. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
9. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S. Springer, New York (2002)