Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition Letters 26 (2005) 2206–2214
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Cost-conscious classifier ensembles

Cigdem Demir *,[1], Ethem Alpaydin

Department of Computer Engineering, Bogazici University, Istanbul TR-34342, Turkey

Received 6 July 2004; received in revised form 16 March 2005
Available online 23 May 2005

Communicated by K. Tumer

Abstract

Ensemble methods improve the classification accuracy at the expense of testing complexity, resulting in increased computational costs in real-world applications. Developing a utility-based framework, we construct two novel cost-conscious ensembles; the first one determines a subset of classifiers and the second dynamically selects a single classifier. Both ensembles successfully switch between classifiers according to the accuracy-cost trade-off of an application.

Keywords: Ensemble techniques; Utility theory; Computational cost; Voting; Selection

1. Introduction

Different types of costs in machine learning have been extensively investigated to date (Turney, 2000). Among them are the cost of feature extraction, the cost of misclassification errors, and the cost of computation. The *cost of feature extraction* arises from the effort of sensing data and obtaining a feature set from the data; an example is the cost of a blood test in medical diagnosis (Turney, 1995). The *cost of misclassification errors* stems from different levels of losses observed in the case of different decisions as defined by a loss matrix (Duda et al., 2001). To deal with this type of cost, learning algorithms take into account the loss due to every possible misclassification error as to minimize the overall expected risk. For example, the misclassification error cost may be incorporated into decision trees (Breiman et al., 1984). The *cost of computation* includes the "dynamic cost" due to the time and space complexity that incurs during training and testing a classifier (Turney, 2000). The time and space complexity depends on the complexity of the classifier and, in turn, on the dimension of the feature set.

* Corresponding author. Fax: +1 518 276 4033.
E-mail addresses: demir@cs.rpi.edu (C. Demir), alpaydin@boun.edu.tr (E. Alpaydin).
[1] Present address: Department of Computer Science, Rensselaer Polytechnic Institute, NY 12180, USA.

0167-8655/$ - see front matter
doi:10.1016/j.patrec.2005.03.028

A real-world application typically requires testing its trained system for numerous instances. Although testing a single instance requires smaller amounts of time and space than training the entire system, the total amount of time and space consumed during the use of the system with numerous tests is critically important in real-world applications, especially when (i) the total power consumption is restricted and/or (ii) the real-time response during testing an instance is required to be sufficiently fast. For example, it is important to increase the battery charge period of a personal digital assistant (PDA) by using minimal amount of space and time for optical character recognition. As another example, in the case of speech-to-speech translation, the response time for testing each instance should be short enough to provide continuous multi-lingual dialog.

In machine learning, a powerful technique to increase the classification accuracy is to combine a set of classifiers (Alpaydin, 2004). Among such ensemble techniques are voting (Kittler et al., 1998), stacked generalization (Wolpert, 1992), and mixture of experts (Jacobs et al., 1991). Although these approaches successfully increase the accuracy, the use of multiple classifiers leads to increased testing complexity with significantly increased power consumption and response time. Thus, the trade-off between the classification accuracy and the cost of computation needs to be carefully considered in real-world applications.

In this paper, we present a framework based on utility theory that takes into account the computational cost of each classifier while combining multiple classifiers to increase the accuracy. Using this framework, we construct two cost-conscious ensemble algorithms that evaluate the importance the application attaches to the accuracy at the cost of increased testing complexity. The first algorithm determines a subset of classifiers that leads to the highest expected utility for a given application. The second one dynamically selects the classifier that yields the highest expected utility for the classification of a given instance based on its difficulty level. Our experiments demonstrate that the proposed algorithms successfully switch between the classifiers according to the trade-off between the accuracy and the testing cost of an application.

2. Utility-based framework

Decision-making requires selecting an action A_i among several alternatives when the consequences of each action are not exactly known (Lindley, 1971). The consequences of an action depend on the state of nature C_k taking place after the action is selected. Therefore, the actual state of nature cannot be known a priori to selecting the action. However, there may be an input vector x that is used to calculate the probability $P(C_k|x)$. Decision has to be made in favor of the action with the highest expected utility. For the input vector x, the expected utility of selecting an action A_i is defined as follows, with $u(C_k, A_i)$ being the utility (negative risk) of taking the action A_i when the state of nature is C_k.

$$\mathrm{EU}(A_i|x) = \sum_{k=1}^{K} P(C_k|x, A_i) \cdot u(C_k, A_i) \tag{1}$$

For the classification problem with multiple classifiers, we adapt Eq. (1) to select which classifier(s) to use as follows. In the equation, we consider A_i as the classifier and C_k as the class that an instance can belong to. We also make use of the utility function to incorporate the cost of testing an instance into the classification accuracy resulting from this testing. For that, we define the utility function as $u(C_k, A_i) = \mathrm{accuracy}(C_k, A_i) - \alpha \cdot \mathrm{cost}(C_k, A_i)$, where $\mathrm{accuracy}(C_k, A_i)$ and $\mathrm{cost}(C_k, A_i)$ are the accuracy and the testing cost of the classifier A_i when this classifier assigns an instance to class C_k and α indicates the importance of the testing cost in a given application. Therefore, here, the decision-making corresponds to selecting the classifier that leads to the highest expected utility. In a real-world application, the choice of α depends on the physical parameters of the application. For example, the available maximum power and/or the allowed longest response time set a lower bound for α, whereas the required reliability of the application sets an upper bound for α. The Akaike information criterion (Akaike, 1974) and minimum description length (Hansen and Yu, 2001) are similar to the utility criterion adopted in this work, with the number of free parameters used as the cost of computation. Although these

criteria are used in the evaluation of different individual learning models, the use of such criteria has, however, not previously been demonstrated in an ensemble algorithm (neither in a model combination algorithm nor in a dynamic model selection algorithm).

We use 0/1 loss function for accuracy(C_k, A_i). It is possible to generalize this accuracy using an arbitrary loss function that allows for different gains and losses for different correct classifications and misclassifications. In this work, a classifier has a fixed testing cost regardless of the class which it assigns an instance to. Therefore, for classifier A_i, cost(C_k, A_i) is the same for each class C_k. It is also possible to generalize the cost to include also the cost of extracting a representation; if the extraction cost is fixed, it can be included as an additive constant in the cost function. Using 0/1 loss function and the same cost function for a single classifier, the expected utility of classifying the instance x by the classifier A_i is defined as

$$\mathrm{EU}(A_i|x) = P(C_{\text{correct class}}|x, A_i) - \alpha \cdot \text{cost}_i \tag{2}$$

where cost_i indicates the cost of classifying an unlabeled instance by the classifier A_i and $P(C_{\text{correct class}}|x, A_i)$ is the probability that the classifier A_i would generate for the correct class if it were used.

Given a data set $X = \{x^t\}_{t=1}^N$, the expected utility of using the classifier A_i for the classification of the entire data set is estimated as follows:

$$\begin{aligned}\widehat{\mathrm{EU}}(A_i|X) &= \frac{\sum_{t=1}^N \mathrm{EU}(A_i|x^t)}{N} \\ &= \frac{\sum_{t=1}^N P(C_{\text{correct class}}|x^t, A_i)}{N} - \alpha \cdot \text{cost}_i\end{aligned} \tag{3}$$

3. Utility-based voting

The classical ensemble algorithms combine the decisions of the classifiers that improve the classification accuracy neglecting the concomitant increase in the cost of testing instances with all these classifiers. In this work, we introduce a voting algorithm that uses the decision of a classifier only if the improvement in the accuracy resulting from using this additional classifier compensates for the additional cost that this classifier introduces. We use the expected utility to quantify such a compensation.

Our algorithm starts with the single classifier that yields the highest expected utility (calculated by Eq. (3)) and incrementally adds a classifier one at a time. This additional classifier is selected such that the subsequent subset of selected classifiers leads to the highest expected utility (calculated by Eq. (4) below). Our algorithm terminates if no additional classifier improves the expected utility. This is a greedy algorithm and, like all greedy algorithms, it does not guarantee to yield the best possible subset of the classifiers. However, it is $O(L^2)$ time, L being the number of the classifiers, where there are 2^L possible subsets.

$$\begin{aligned}&\widehat{\mathrm{EU}}(A_0, \ldots, A_m|X) \\ &\quad = \frac{\sum_{t=1}^N P(C_{\text{correct class}}|x^t, A_0, \ldots, A_m)}{N} \\ &\qquad - \alpha \cdot \text{cost}_{A_0,\ldots,A_m}\end{aligned} \tag{4}$$

In Eq. (4), $P(C_{\text{correct class}}|x^t, A_0, \ldots, A_m)$ is the probability of the correct classification of the instance x^t when the decisions of the classifiers $A_0, \ldots, A_m$ are combined and $\text{cost}_{A_0,\ldots,A_m}$ is the testing cost of using these classifiers plus the cost of combining them. Note that there is no extra combination cost in voting; if we were to use a combination scheme like stacking, there would also be the cost of the combiner.

In both Eqs. (3) and (4), the cost of using a single classifier and a subset of classifiers is known in advance of testing. On the other hand, the probability of correct classification using either a single classifier or a subset of classifiers cannot be known in advance. To estimate it, consecutive to training a classifier A_i, we calculate $\sum_t P(C_{\text{correct class}}|x^t, A_i)/N$ by using the trained classifier on a separate validation data set that is not used during training and use the calculated value in Eq. (3) for the classification of further instances. Likewise, we calculate the value of $\sum_t P(C_{\text{correct class}}|x^t, A_0, \ldots, A_m)/N$ using the trained classifiers $A_0, \ldots, A_m$ on separate validation data sets and use the calculated value in Eq. (4).

4. Utility-based classifier selection

The algorithm defined in Section 3 works well for the data sets that consist of similar instances. This algorithm determines a subset of classifiers and uses this subset in the classification of the entire data set regardless of the different difficulty levels of its instances. Our experiments reveal that most of the time, only a single classifier is sufficient to classify an instance correctly. However, the type of the single classifier varies for different instances. Our second algorithm selects such a single classifier for the classification of a given instance. It aims to use simpler and cheaper classifiers for simpler instances, and to use more complex and probably more expensive classifiers for more difficult ones; this would decrease the overall testing cost without significantly decreasing the accuracy. For a given instance, this algorithm selects the classifier that yields the highest expected utility (calculated by Eq. (2)). In Eq. (2), $P(C_{\text{correct class}}|x, A_i)$ cannot be known in advance. Hence, we must estimate this probability before selecting a classifier and calculate the expected utility of each classifier using this estimated probability.

To do so, for each classifier, we use an estimator that outputs a probability value for a given instance. This estimator is trained to learn these probability values from the inputs of a separate validation data set that is not used during training the classifiers. New output values of this separate data set are formed as follows: After learning the parameters of a classifier, each validation instance is tested with the classifier and the probability generated for the correct class of the instance is used as the new output value.[2] This reduces the number of the outputs from the number of classes to 1 and, thus, reduces the complexity of an estimator compared to its corresponding classifier. Moreover, using simpler estimators further reduces the complexity. If the estimators were not cheaper than their corresponding classifiers, there would be no point in using them.

We further decrease the additional cost that the estimators introduce by using the estimators in ascending order of the complexity of their corresponding classifiers. We start with the estimator that belongs to the cheapest classifier and calculate its expected utility. For the next classifier, we compute its expected utility with the probability value of 1 without using its estimator. If even this maximum probability cannot improve the highest expected utility computed so far, the algorithm terminates and uses the classifier with the highest expected utility. Otherwise, the estimator of this classifier is used and its expected utility is calculated. For a single instance, the total cost of the algorithm is the sum of the costs of the estimators that have been used, and the cost of the selected classifier.

There are other dynamic selection algorithms available in the literature; these algorithms choose the models to optimize the accuracy-based criteria (Giacinto and Roli, 2000; Woods et al., 1997). Different than these algorithms, our dynamic selection algorithm chooses models based on a utility criterion to maximize the accuracy *while minimizing the computational cost*. To the best of our knowledge, this is the first of its kind that considers the overall computational cost in the dynamic classifier selection.

5. Experiments

We conduct the experiments on a data set of a real-world application. *Pendigits* is the preprocessed data set of handwritten digits collected from 44 different people on a touch-sensitive WACOM tablet (Alimoglu and Alpaydin, 1997). For the data sets on which the classifiers and the estimators are trained, we use 3748 and 1873 handwritten digits taken from 30 people, respectively. For the writer-dependent test set, we use the remaining 1873 handwritten digits taken from the same 30 people. For the writer-independent test set, we use 3498 handwritten digits taken from the remaining 14 people. This data set is available at the UCI repository (Blake and Merz, 1998).

[2] For example, in multilayer perceptrons, the calculated output of each node in the output layer is used as the probability of the corresponding class of this node; in multilayer perceptrons, the actual output of the node corresponding to the correct class is 1 and the actual output values of the other nodes are 0.

Table 1
A list of base classifiers and their representations along with their testing costs, average accuracy and the standard deviation of the accuracy for 10 different runs

Representation-classifier	Cost	Accuracy			
		Training (classifiers)	Training (estimators)	Dep. test	Indep. test
Counter4-mlp4	104	82.2 ± 1.0	80.7 ± 1.1	81.6 ± 1.0	79.0 ± 1.7
Dynamic-lp	160	97.8 ± 0.4	96.0 ± 0.2	96.5 ± 0.4	93.0 ± 0.2
Counter8-lp	640	99.1 ± 0.0	95.6 ± 0.0	95.2 ± 0.0	92.7 ± 0.1
Counter8-mlp20	1480	100.0 ± 0.0	95.7 ± 0.2	95.6 ± 0.3	93.9 ± 0.4
Static-mlp32	8512	100.0 ± 0.0	95.7 ± 0.2	95.6 ± 0.3	94.4 ± 0.2

"*lp*" is the linear perceptron and "*mlp4*" is the multilayer perceptron with 4 hidden units.

We use four different representations for this data set. The *dynamic* representation is a sequence of eight temporally equidistant (x, y) coordinates of the pen stroke; it has 16 dimensions. The *static* representation is a 16×16 bitmap image generated from the pen strokes. To obtain the *counter4* representation, we put a grid of 4×4 on this bitmap image, and the number of black pixels in each grid entry is used as a feature in the new feature set. The *counter8* representation is obtained similarly; we use a grid of 8×8 instead of the grid of 4×4. For these representations, we use linear perceptrons (LP) and multilayer perceptrons (MLP) as the classifiers. After experimenting with various classifiers for each representation, we select the classifiers that yield the most accurate results with their testing complexities falling in an acceptable range. In our experiments, we express the testing cost as the big-O values of the classifiers. These values are the number of multiplications used in testing and the number of weights (free parameters) in a perceptron. A linear perceptron with d inputs and K classes is $O((d+1) \times K)$, and a multilayer perceptron with d inputs, H hidden units, and K classes is $O((d+1) \times H + (H+1) \times K)$.

In Table 1, we present a list of the base classifiers and their representations. For example, the first classifier is a multilayer perceptron with four hidden units (*mlp4*) and uses the *counter4* representation. In this table, for each classifier, we also provide the average accuracy and its standard deviation of the classification for 10 runs and the testing cost as explained in the previous paragraph.

In Table 2, we report the subsets of the classifiers and the selection frequencies of these subsets for different exemplary α values in different orders. When the testing cost has no effect on the selection of the classifiers (i.e., $\alpha = 0$), the most expensive and the most accurate classifier (*static-mlp32*) is always used. As the importance of the testing cost increases (i.e., as α increases), our voting algorithm never selects the classifier *static-mlp32* and it starts selecting cheaper and less accurate classifiers such as *dynamic-lp*. When $\alpha = 0.3$, the cheapest classifier *counter4-mlp4* is used 90% of the time. As the testing cost becomes too important ($\alpha > 0.3$), the classifier *counter4-mlp4* is always selected.

In our selection algorithm, we use linear perceptrons for the estimators. Here, we consider learning the parameters of the estimators as a regression problem. Since the output dimension of each estimator is 1 and a linear perceptron is

Table 2
The subset of the classifiers selected by our voting algorithm and their selection frequencies for 10 different runs, for different values of α

α	Selected models
0.000	*dynamic-lp* + *static-mlp32* (60%) *dynamic-lp* + *static-mlp32* + *counter8-lp* (10%) *counter8-mlp20* + *dynamic-lp* + *static-mlp32* + *counter4-mlp4* (10%) *static-mlp32* + *dynamic-lp* (20%)
0.001	*dynamic-lp* (10%) *dynamic-lp* + *counter8-lp* (60%) *dynamic-lp* + *counter8-lp* + *counter4-mlp4* (30%)
0.005	*dynamic-lp* (70%) *dynamic-lp* + *counter4-mlp4* (30%)
0.010	*dynamic-lp* (100%)
0.300	*counter4-mlp4* (90%) *dynamic-lp* (10%)

used, the additional cost of using an estimator is equal to the number of dimensions of the corresponding feature set. In Fig. 1, we present the selection frequency of each classifier reported by our selection algorithm as a function of α. To reveal the significance of α in different orders, we use its logarithm in this figure. This figure demonstrates that the most expensive model (*static-mlp32*) is never used when $\alpha > 0$. For $0 < \alpha \leqslant 0.1$ ($\log \alpha \leqslant -1$), our algorithm selects a single classifier among the classifiers *dynamic-lp*, *counter8-lp*, and *counter8-mlp20*. As the importance of the testing cost increases, the algorithm selects the classifier *dynamic-lp* with higher frequencies. When $\alpha \approx 0.1$ ($\log \alpha \approx -1$), our algorithm starts selecting the least expensive and the least accurate classifier (*counter4-mlp4*). These results are obtained on the writer independent test sets for 10 different runs. The results obtained on the training sets and the writer dependent test set are similar. However, due to the limited space, we only provide the results obtained on the writer independent test set which is considered as the real test set.

In our experiments, we use 2/3 and 1/3 of the training samples for training the classifiers and estimators, respectively. To analyze the sensitivity of the estimator to the size of its training set, we use different-sized portions of the training set to train the estimators and the rest of the training set to train the classifiers. As observed in Fig. 2, increasing the size of the training set of the estimators (i.e., decreasing the size of the training set of the classifiers) results in decreasing the classifiers' performance. When the size of data used to train estimators increase, their performances first increase (since more amount of data improves their generalization ability) and then decrease (since the estimators estimate the accuracy of the classifiers which degrade with less remaining training data).

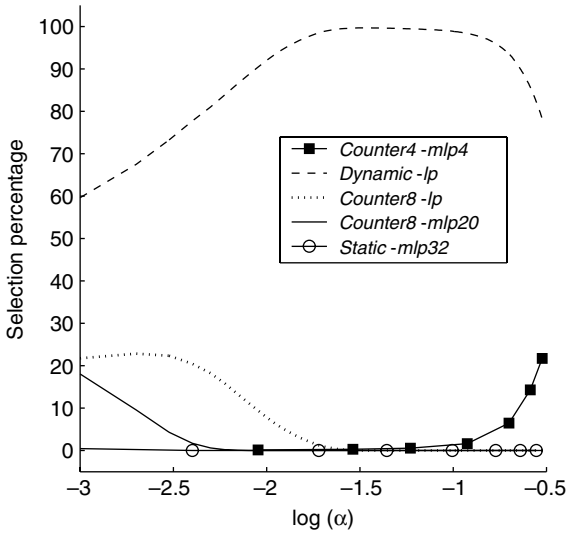

Fig. 1. The selection frequency of each classifier presented as a function of α on the writer independent test set.

In Table 3, we report the average utility values and their standard deviations computed using the base classifiers, our two algorithms (*vote-utility* and *selection*), and three other ensemble techniques (*vote-all*, *vote-max*, and *boosting*[3]). In this table, the utility values are computed as accuracy $- \alpha \cdot$ cost. This table demonstrates that when the testing cost has no effect, or, equivalently, the utility is simply equal to the accuracy (i.e., when $\alpha = 0$), all classical ensemble techniques (*vote-all*, *vote-max*, and *boosting*) and our algorithms are more accurate than the base classifiers indicating the advantage of ensemble methods. Although the classical ensemble techniques use all classifiers, *vote-utility* uses a few and *selection* uses only one. When $\alpha > 0$, our voting algorithm (*vote-utility*) never leads to a lower utility value than the base classifiers and the classical ensemble methods. Our selection algorithm yields the best results when $0 < \alpha < 0.01$ because this algorithm can potentially use different models in this range of α, as our experimental observations reveal. The larger α values forces our selection algorithm to select the same single classifier; this classifier is

[3] *Vote-all* averages the posterior probabilities computed by the individual classifiers for each class and uses this average in its decision. *Vote-max* selects the classifier that yields the maximum posterior probability for any class and uses the decision of this classifier (Kittler et al., 1998). *Boosting* takes a weighted vote where weights are taken proportional to model accuracies; in this work, the weights are calculated by using the algorithm described in (Kuncheva et al., 2002).

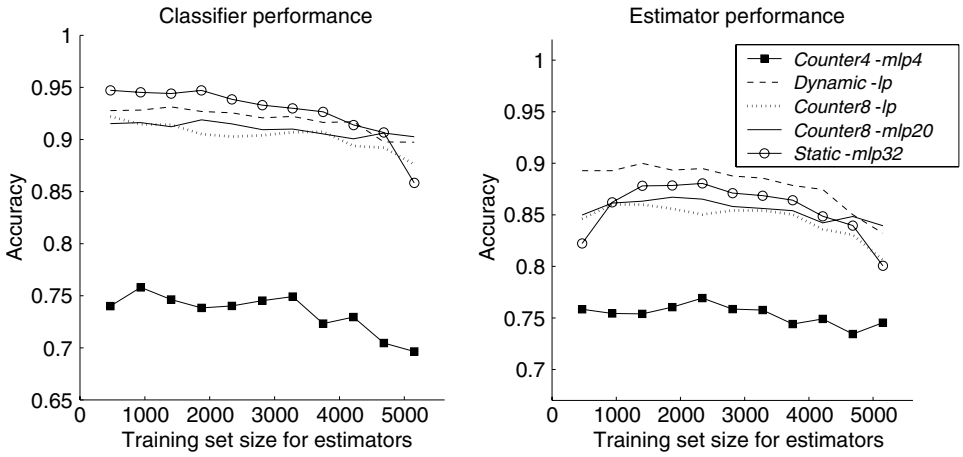

Fig. 2. The dependency of the classifier and estimator accuracies (for the writer independent test set) on the size of the training set that is used to train the estimators.

Table 3
The average utility values and their standard deviations calculated using the base classifiers, our voting and selection algorithms, and the other three ensemble techniques on the writer independent test sets

α	0.000	0.001	0.005	0.010	0.300
Counter4-mlp4	79.0 ± 1.7	78.9 ± 1.7	78.4 ± 1.7	77.9 ± 1.7	47.8 ± 1.7
Dynamic-lp	93.0 ± 0.2	92.8 ± 0.2	92.2 ± 0.2	91.4 ± 0.2	45.0 ± 0.2
Counter8-lp	92.7 ± 0.1	92.1 ± 0.1	89.5 ± 0.1	86.3 ± 0.1	−99.3 ± 0.1
Counter8-mlp20	93.9 ± 0.4	92.4 ± 0.4	86.5 ± 1.4	79.1 ± 0.4	−350.1 ± 0.4
Static-mlp32	94.4 ± 0.2	85.9 ± 0.2	51.9 ± 0.2	9.3 ± 0.2	−2459.2 ± 0.2
Vote-all	96.3 ± 0.2	85.4 ± 0.2	41.8 ± 0.2	−12.7 ± 0.2	−3172.5 ± 0.2
Vote-max	95.6 ± 0.6	84.7 ± 0.6	41.1 ± 0.6	−13.4 ± 0.6	−3173.2 ± 0.6
Boosting	96.1 ± 0.1	85.2 ± 0.1	41.6 ± 0.1	−12.9 ± 0.1	−3172.7 ± 0.1
Vote-utility	95.8 ± 0.5	94.2 ± 0.8	92.3 ± 0.3	91.4 ± 0.2	47.8 ± 1.6
Selection	95.2 ± 0.1	94.3 ± 0.2	92.5 ± 0.3	91.1 ± 0.3	38.5 ± 0.5

dynamic-lp for $0.01 \leqslant \alpha < 0.3$, and *counter4-mlp4* for $\alpha \geqslant 0.3$. Although our selection algorithm selects this single model correctly as shown in Fig. 1, the cost of the estimator used lowers the utility values. Thus, our selection algorithm works well when the importance of cost is not so high such that one single classifier does not dominate over all of the classifiers. In a real-world application, since the physical constraints of the application such as the available maximum power and the allowed longest response time in conjunction with the required accuracy set an α range, the success of the utility-based voting and selection algorithms will vary for different applications requiring different α ranges.

In Table 4, we present the computational times for the classification of the samples in the writer independent test set (in milliseconds). These results are obtained on a computer with a 1600 MHz Pentium M processor and 512 MB of RAM. In this table, it is observed that while the base classifiers and classical ensemble methods (*vote-all*, *vote-max*, and *boosting*) use the same amount of computational time regardless of the value of α (i.e., the importance of the cost), our proposed algorithms use less and less time with the increasing importance of the computational cost in an application. These proposed algorithms choose the models in accordance with their individual contribution to the computational cost and the value of

Table 4
The average and the standard deviation of the computational times for the classification of the samples in the writer independent test set (in milliseconds)

α	0.000	0.001	0.005	0.010	0.300
Counter4-mlp4	13.0 ± 4.8	13.0 ± 4.8	13.0 ± 4.8	13.0 ± 4.8	13.0 ± 4.8
Dynamic-lp	13.0 ± 4.8	13.0 ± 4.8	13.0 ± 4.8	13.0 ± 4.8	13.0 ± 4.8
Counter8-lp	33.1 ± 4.8	33.1 ± 4.8	33.1 ± 4.8	33.1 ± 4.8	33.1 ± 4.8
Counter8-mlp20	80.1 ± 0.3	80.1 ± 0.3	80.1 ± 0.3	80.1 ± 0.3	80.1 ± 0.3
Static-mlp32	381.5 ± 3.0	381.5 ± 3.0	381.5 ± 3.0	381.5 ± 3.0	381.5 ± 3.0
Vote-all	528.1 ± 12.8	528.1 ± 12.8	528.1 ± 12.8	528.1 ± 12.8	528.1 ± 12.8
Vote-max	527.8 ± 12.3	527.8 ± 12.3	527.8 ± 12.3	527.8 ± 12.3	527.8 ± 12.3
Boosting	531.8 ± 11.7	531.8 ± 11.7	531.8 ± 11.7	531.8 ± 11.7	531.8 ± 11.7
Vote-utility	417.4 ± 32.7	57.0 ± 8.4	23.0 ± 10.6	18.1 ± 6.4	16.0 ± 5.2
Selection	156.4 ± 8.5	44.1 ± 5.1	25.0 ± 5.3	23.0 ± 4.8	17.1 ± 4.9

α such that the overall computational cost is reduced given the importance of the cost.

6. Conclusion

This work introduces a utility-based framework to incorporate the cost of computation into multi-classifier classification problems. In this work, we present two novel ensemble techniques, one for determining a subset of classifiers and the other for dynamically selecting a single classifier to maximize the expected utility. The experiments on a data set of handwritten digits demonstrate that our algorithms lead to higher utility values by switching to different types of classifiers in accordance with the importance of the testing cost.

One future research direction is to test our algorithms on different kinds of data sets including different classifiers. In classifier combination, it is also possible to use a different combination scheme such as stacking instead of voting. This allows combining models in a nonlinear way, but it has the disadvantage of introducing additional cost. Another possibility is to include the cost of extracting a representation.

Acknowledgement

This work has been supported by the Turkish Academy of Sciences, in the framework of the Young Scientist Award Program (EA-TÜBA-GEBIP/2001-1-1) and Boğaziçi University Scientific Research Project 02A104D.

References

Akaike, H., 1974. A new look at the statistical model identification. IEEE Trans. Automat. Contr. 19, 716–723.

Alimoglu, F., Alpaydin, E., 1997. Combining multiple representations and classifiers for pen-based handwritten digit recognition. ICDAR1997, Ulm.

Alpaydin, E., 2004. Introduction to Machine Learning. The MIT Press, Cambridge, MA.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth, California.

Blake, C.L., Merz, C.J., 1998. UCI Repository of machine learning databases. Available from: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Duda, O.R., Hart, E.P., Stork, G.D., 2001. Pattern Classification. Wiley-Interscience, New York.

Giacinto, G., Roli, F., 2000. A theoretical framework for dynamic classifier selection. 15th Internat. Conf. Pattern Recognition, vol. 2. IEEE Press, pp. 8–11.

Hansen, M.H., Yu, B., 2001. Model Selection and the principle of minimum description length. J. Amer. Statist. Assoc. 96, 746–774.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E., 1991. Adaptive mixtures of local experts. Neural Comput. 3, 79–87.

Kittler, J., Hatef, M., Duin, R.P.W., Matas, J., 1998. On combining classifiers. IEEE Trans. Pattern Anal. 20, 226–239.

Kuncheva, L.I., Skurichina, M., Duin, R.P.W., 2002. An experimental study on diversity for bagging and boosting with linear classifiers. Inform. Fusion 3, 245–258.

Lindley, D.V., 1971. Decision Making. Wiley-Interscience, New York.

Turney, P.D., 1995. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. J. Artificial Intelligence Res. 2, 369–409.

Turney, P.D., 2000. Types of cost in inductive concept learning. Workshop on Cost-Sensitive Learning, ICML2000, Stanford, pp. 15–21.

Wolpert, D.H, 1992. Stacked generalization. Neural Networks 5, 241–259.

Woods, K., Kegelmeyer, W.P., Bowyer, K., 1997. Combination of multiple classifiers using local accuracy estimators. IEEE Trans. Pattern Anal. 19, 405–410.