Fuzzy Modeling of the Complexity vs. Accuracy Trade-off in a Sequential Two-Stage Multi-Classifier System

MARK LAST[1]
Department of Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva 84105
ISRAEL

HORST BUNKE
Institut fur Informatik and angewandte Mathematik
University of Bern
Neubrückstrasse 10, CH-3012, Bern
SWITZERLAND

ABRAHAM KANDEL
Department of Computer Science and Engineering
University of South Florida
4202 E. Fowler Avenue, ENB 118, Tampa, FL 33620
USA

Abstract: - The paper presents a new approach to sequential multi-stage combination of instance-based classifiers. Each classifier in the sequence requires more computational effort than the preceding classifier due to using a larger subset of features. A more complex classifier is activated only if the confidence level of the preceding classifier is below a pre-defined threshold. The optimal threshold is found by maximizing a customizable fuzzy-based measure, called *Performance Index* (*PI*), which expresses the task-specific trade-off between classification accuracy and computational complexity. The approach is evaluated on a two-stage combination of *k*-Nearest Neighbor classifiers. The features to be used by the first classifier in the combination are found by a novel feature selection method, called "IFN + Relief." The *PI* measure is shown empirically to be an efficient tool for integrating accuracy and complexity considerations in the design of a multi-stage classification system.

Keywords: - Classifier combination, sequential combination, feature selection, instance-based learning, nearest neighbor classifier, fuzzy optimization, info-fuzzy network (IFN)

[1] Corresponding author

1 Introduction

Classifier combination has become a very active area of research [5]. The driving force behind these activities is the expectation that the performance of a combined system can exceed the performance of each individual classifier. The main motivation behind existing combination architectures (e.g., parallel combination, serial combination, hierarchical concatenation, etc.) is improvement of classification accuracy. In this paper, we are rather concerned with the improvement of *computational efficiency* through classifier combination. Reducing the amount of computation associated with classifying new instances is especially important for "lazy" learning methods, like k-nearest neighbors, which defer the processing of training examples until each new instance becomes available. We propose a serial classifier combination scheme that follows the reevaluation approach (see [13]). This approach is motivated by the observation that in many applications the majority of the patterns to be classified are 'well-behaved'. This means they can be classified using a relatively simple and fast classifier, while just a few 'hard' cases require reevaluation by a more sophisticated classifier, which has higher classification accuracy. However, if the sophisticated classifier is applied to all patterns, including the easy ones, more computing resources than necessary are spent.

Our architecture for serial classifier combination, initially introduced by us in [8], includes two sequential classifiers of the same type, namely C_1 and C_2. The C1 classifier uses a selected subset of the features that are used by C2. Whenever the decision reached by C_1 is below a given level of confidence, C_1 rejects the input and activates C_2. For most classifiers the computation time grows with the dimensionality of the feature space. Therefore, C_1 will be faster than C_2. If a sufficiently large portion of all patterns will be classified by C_1, and not passed onto C_2, substantial savings in computation time can be expected. Variation of the confidence threshold in C_1 allows trading computation time for classification accuracy.

The best trade-off between computational complexity and classification accuracy depends on the considered classification task. In a general case, the overall performance depends on both computational complexity and accuracy in some implicit and possibly nonlinear fashion. Such relationships as "A 50% reduction in the classification time is worth a 1% loss in accuracy" are easily handled by humans, but are hard to implement on computers. The fuzzy set theory is known as an efficient tool for automating the human perception of information (see [6]). In this paper, we evaluate the performance of a multi-stage classifier by a new fuzzy-based measure, called *Performance Index*.

The rest of our paper is organized as follows. In Section 2, we describe the feature selection method used in this work. A two-stage combination of *k*-Nearest Neighbor Classifiers is presented in Section 3. Empirical evaluation of the system is carried out in Section 4 and conclusions regarding possible extensions of our approach are drawn in Section 5.

2 Feature Selection Algorithm

The architecture described in Section 1 requires for each classifier C_i to use a subset of features F_i out of all available features F, such that $F_1 \acute{I} F_i \acute{I} F_n = F$. Many different approaches to feature selection have been proposed in the literature (see [9]). Our method of feature selection is based on two feature filter algorithms, Relief [4] and IFN [7] [10], which are briefly described in the next sub-sections.

2.1 The Relief Algorithm

The Relief algorithm, proposed by Kira and Rendell in [4], selects features by their relevance to the target concept (function). The basic idea of Relief is similar to the guiding principle of the *k*-Nearest Neighbors algorithm: instances that are closer to a given instance are more likely to belong to the same class. If a dimension (feature) is relevant, the closest instances of the same class are expected to be closer to a given instance *along that dimension* than the closest instances of all the other classes.

Relief normalizes the values of feature relevance to the [0,1] range. The average relevance level of a relevant feature should be close to one, while the relevance levels of irrelevant features should be close to zero. According to [4], the distinction between relevant and irrelevant features is best determined by manual inspection, which means that the algorithm is usually implemented in a "semi-automatic" mode.

2.2 The Information-Fuzzy Network (IFN) Algorithm

An Info-Fuzzy Network (IFN) [10] is a network-like classification model, which differs from the structure of a standard decision tree (see [11]) in two aspects: it is restricted to a single testing feature across

all nodes of the same hidden layer and it has interconnections between the terminal (leaf) nodes and the final nodes, which represent the classes under consideration.

The network is built by a greedy stepwise procedure: at each step, a new feature is selected to maximize the total decrease in the conditional entropy, as a result of splitting the nodes of the last layer. The nodes of a new hidden layer are defined for a Cartesian product of split nodes of the previous hidden layer and the values of a new selected feature. If there is no feature decreasing the conditional entropy of the target attribute, the network construction stops and the algorithm outputs the list of features associated with the network layers.

2.3 IFN + Relief

Relief does not provide an automatic threshold for separating the relevant features from the irrelevant ones. On the other hand, the IFN method clearly determines the number of relevant features, which is equal to the number of network layers. Consequently, we have proposed a novel combination of these two algorithms called "IFN + Relief." According to the new method, the *number* of the selected features is determined by IFN, while the *selection* of features is based on the relevance level calculated by Relief. The empirical results presented by us in [8] show that the new method tends to be superior, or at least comparable, to each one of the stand-alone methods (IFN *and* Relief).

3 Two-Stage Combination of *k*-Nearest Neighbor Classifiers

In this section, we describe the implementation and the main settings of the two-stage classifier, based on the *k*-NN algorithm, and suggest a new measure for evaluating the performance of a multi-stage classification system.

3.1 Two-Stage Classifier

In this paper, we propose a serial combination of two *k-Nearest Neighbor (k-NN) classifiers* (see [11]). The input to the system includes the training set and a subset of features selected by the "IFN+Relief" algorithm from that set. The first classifier C_1 attempts to classify a new instance by using the selected features only. The number of the nearest neighbors in the majority class is then compared to a user-defined threshold, which represents a minimum percentage of the total number of nearest neighbors (*k*). If the percentage of examples in the majority class is greater than the threshold, the system outputs the classification made by C_1. Otherwise (if the majority is less than or equal to the threshold), the second, more expensive, classifier C_2 is applied by using the full set of the available features. In the second case, the system outputs the classification made by C_2.

The system implementation involves the choice of the following parameters:

- *Number of nearest neighbors (k).* To study the effect of the majority threshold (see below) on the system performance, we have set the number of nearest neighbors to 10.
- *Feature selection algorithm*. We have applied a novel feature selection method called "IFN + Relief" (see sub-section 2.3 above).
- *Majority threshold.* This is the lowest percentage of the training instances belonging to the most common class, which makes the system to accept the decision of C_1. As the threshold goes up, the run time of the system will increase, since more and more instances are classified by the second classifier (C_2). At the same time, the classification performance of the system is expected to improve and to get closer to the accuracy of C_2. In the next sub-section, we propose a fuzzy-based method for finding a threshold point, which provides the optimal trade-off between accuracy and complexity for a given application.

3.2 Performance Index

Accuracy and complexity are measured on completely different scales and finding the best trade-off between run-time complexity and classification accuracy is a subjective and application-specific task. We may only assume that the classifier performance is a non-increasing function of time complexity and a non-decreasing function of classification accuracy. In [6] we have presented a fuzzy-theoretic approach to automating the human perception of data, based on the Computational Theory of Perceptions [15]. In this paper, we propose a novel, CTP-based measure for evaluating performance of a multi-stage *k*-NN classifier. The new measure, called *Performance Index (PI)*, is defined by

$$PI = min\ [CI, AI] \qquad (1)$$

where *CI* and *AI* are the indices referring to the computational **c**omplexity and the classification **a**ccuracy, respectively (see below). The *min* operator is

common for representing an intersection of fuzzy sets [14]. It has two mathematical properties, which make it particularly useful for evaluating the classifier performance: it is non-decreasing in each set (better values of accuracy or complexity can never decrease the overall performance) and it has the following boundary conditions: *min [0, *] = 0* and *min [1, *] = **. These conditions are quite reasonable in our case. The first condition means that the complexity of a poor classifier is not important, while the second one implies that if one parameter (either accuracy, or complexity) has the optimal value, the performance is determined by the value of the other parameter only.

The *Complexity Index* (*CI*) is a function of the total number of feature-values, TFV_n, accessed by n sequential classifiers for classifying X instances. TFV_n can be found by the following formula:

$$TFV_n = \sum_{i=1}^{n} m_i x_i \qquad (2)$$

where n is the total number of classifiers, m_i is the number of features (out of M) used by classifier i ($\forall i: 0 \leq m_i \leq m_{i+1} \leq M$), and x_i is the number of instances accepted by classifier i ($\Sigma x_i = X$) .

The number of feature-values accessed by a single-stage k-NN Classifier is $TFV_1 = M*X$. Consequently, we normalize the total number of feature-values used by defining the *Complexity Ratio* R_C as follows:

$$R_C = TFV_n / TFV_1 = \frac{\sum_{i=1}^{n} m_i x_i}{M * X} \qquad (3)$$

We suggest the following non-linear function for representing the Complexity Index, based on the value of R_C:

$$CI = \frac{2}{1 + e^{bR_C}} \qquad (4)$$

where R_C is the *Complexity Ratio* and b is a scaling factor representing the user perception of a given complexity ratio. The above fuzzy-like function is equal to one, if TFV_n is equal to zero (no features used), and by changing the value of b it can be brought arbitrarily close to zero for any positive value of R_C. As shown below, the value of the optimal majority threshold depends only on the ratio between b and an additional scaling factor (α). Hence, from now on we will assume that $\beta = 1$.

The *Accuracy Index* (*AI*) can be calculated in a similar manner. We assume that *AI* reaches its maximum value of one, when the misclassification rate is equal to zero and it becomes arbitrarily close to zero for large values of the error rate. The formula for calculating *AI* is given by:

$$AI = \frac{2}{1 + e^{aE}} \qquad (5)$$

where E is the error (misclassification) rate and α is a scaling parameter having characteristics similar to β in the formula of *CI*.

What is the meaning of α and how should one determine its value for a given classification task? As shown experimentally in [8], the error rate is usually a non-increasing function of the majority threshold. That is, the more training instances among the k nearest neighbors are required to be from the majority class, the smaller is the error rate. On the other hand, the complexity ratio is never decreasing as the threshold goes up. Consequently, we may assume that in most datasets, *AI* tends to increase with the value of the threshold while *CI* goes down under the same conditions. The maximum value of *PI* is reached at a threshold point that corresponds to the optimum trade-off between the complexity and the accuracy from the user's point of view. This implies that the indices of complexity (4) and accuracy (5) are equal. Consequently:

$$a / b = R_C / E \qquad (6)$$

$$\text{and for } b = 1: \; a = R_C / E = R_E \qquad (7)$$

where R_E is defined as the *efficiency ratio* between the complexity ratio and the error rate. Thus, the parameter α can be interpreted as the optimal efficiency ratio of a multi-stage classifier. In the next section, we study the effect of α on the location of the optimal threshold point for a set of representative learning tasks.

4 Experimental Evaluation of the System

According to its definition (see sub-section 3.1 above), the majority threshold may vary from the inverse of the number of classes m in a given dataset to one. For the lowest value of the threshold, the complexity ratio R_C is minimal and as the threshold goes up, R_C increases gradually accompanied by a possible decrease in the

testing error rate E. Thus, the efficiency ratio R_E tends to be smaller for the lowest threshold than for the threshold of one and the latter will usually be slightly below the R_E of a single-stage classifier, which uses the full set of features. The actual values of R_C, E, and R_E for these three configurations of the classification system are presented in Table 1 below for a representative set of benchmark datasets, available from the UCI Machine Learning Repository [1].

We are explaining the meaning of the numbers in Table 1 by using as an example the first row of the table, which is related to the Chess dataset. This dataset has two classes, which implies that the minimum majority threshold is 0.5. If the error rate corresponding to this threshold (0.082) is acceptable for the user, the system should use the minimum threshold, since it has the lowest complexity ratio (0.278). In terms of the *PI* parameters, it means that $\alpha < 3.402$. The complexity ratio of the highest threshold (1.0) is 0.376. If this complexity is low enough for the user, the maximum threshold should be chosen by the system, since it provides the lowest error rate (0.05). Any α above 7.533 is applicable to this case. If the user is not satisfied either with the error rate of 0.082, or with the complexity of 0.376, the best threshold should be associated with an efficiency ratio, which corresponds to the best trade-off between the accuracy and the complexity of the system. In Figures 1 - 3, we show the indices for $\alpha = 4$, 5, and 6 respectively.

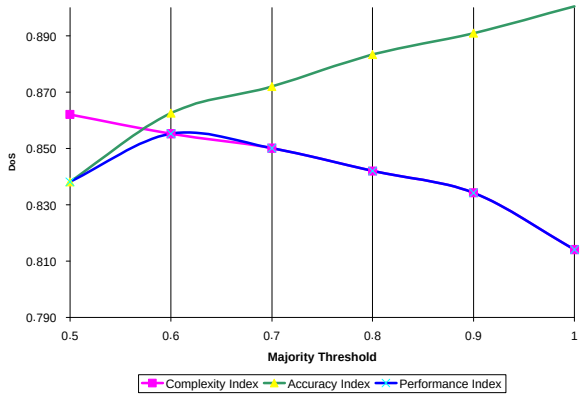

Figure 1 Chess Dataset - Complexity vs. Accuracy ($\alpha = 4$)

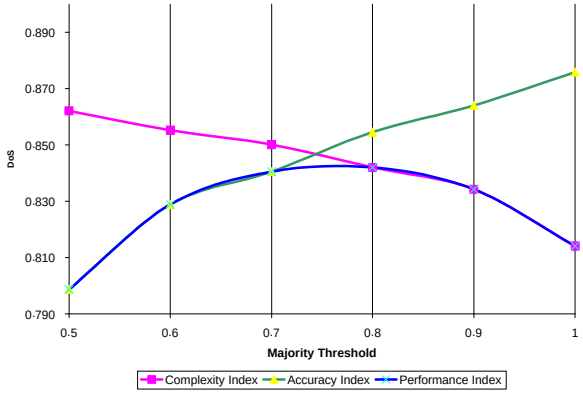

Figure 2 Chess Dataset - Complexity vs. Accuracy ($\alpha = 5$)

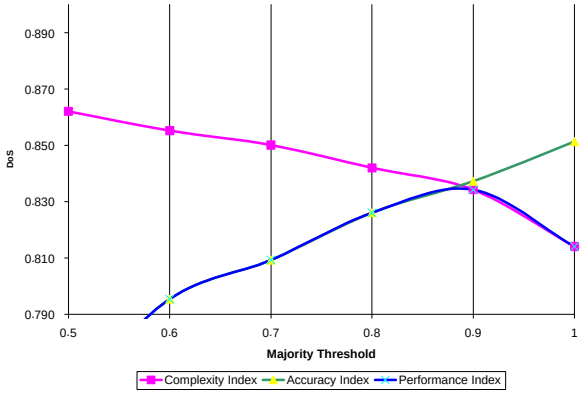

Figure 3 Chess Dataset - Complexity vs. Accuracy ($\alpha = 6$)

5 Conclusions

In this paper, we have presented a fuzzy-based method for designing the parameters of a sequential multi-stage classifier based on a user-specified trade-off between accuracy and complexity. The majority threshold found by the method optimizes the performance of a given classifier from the user's point of view. The overall Performance Index (*PI*) is defined as a function of both the computational complexity and the classification accuracy. The results of experiments on benchmark datasets demonstrate the applicability of the proposed approach to a variety of classification tasks.

In our view, the proposed approach to trading accuracy for complexity in a classifier combination is still far from being fully explored and exploited. The approach is definitely not limited to a *k*-NN Classifier

and it can be applied to other classification methods, e.g., Bayes Classifier, neural networks, and decision trees. Moreover, the method and its extensions can be applied to classifying online streams of instances, like log and clickstream data on the web (see [2]).

Acknowledgment: This work was partially supported by SPAWAR Research Grant N00039-01-1-2248.

Table 1 Calculation of Efficiency Ratio Values

	Selected	Minimum Threshold			Maximum Threshold			All Features		
Dataset	Features	Complexity Ratio	Error Rate	Efficiency Ratio	Complexity Ratio	Error Rate	Efficiency Ratio	Complexity Ratio	Error Rate	Efficiency Ratio
Chess	10	0.278	0.082	3.402	0.376	0.050	7.533	1.000	0.094	10.622
Credit	3	0.214	0.151	1.423	0.770	0.113	6.815	1.000	0.113	8.852
Diabetes	3	0.375	0.286	1.313	0.898	0.227	3.956	1.000	0.227	4.407
Glass	2	0.222	0.535	0.415	1.000	0.380	2.630	1.000	0.380	2.630
Heart	3	0.231	0.235	0.981	0.873	0.200	4.367	1.000	0.212	4.722
Optdigits-Orig	13	0.013	0.513	0.025	0.978	0.025	38.553	1.000	0.024	41.130
Wine	2	0.154	0.150	1.026	1.000	0.033	30.000	1.000	0.033	30.000
Mean	**5.1**	**0.21**	**0.28**	**1.23**	**0.84**	**0.15**	**13.41**	**1.00**	**0.15**	**14.62**

References:

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

[2] G. Hulten, L, Spencer, and P. Domingos, Mining Time-Changing Data Streams, Proc. of KDD-2001, ACM Press, 2001.

[3] G. H. John, R. Kohavi, and K. Pfleger, Irrelevant Features and the Subset Selection Problem, Proc. of Machine Learning: Proc. of the 11th Int'l Conf., pages 121-129, 1994.

[4] K. Kira & L.A. Rendell, The Feature Selection Problem: Traditional Methods and a New Algorithm, Proc. of AAAI'92, pages 129-134, 1992.

[5] J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, Second International Workshop, MCS 2001, Springer, 2001.

[6] M. Last and A. Kandel, Automated Perceptions in Data Mining, 1999 IEEE International Fuzzy Systems Conference Proceedings, Part I, pp. 190 - 197, Seoul, Korea, August 1999.

[7] M. Last, A. Kandel, O. Maimon, Information-Theoretic Algorithm for Feature Selection, Pattern Recognition Letters, No. 22 (6-7), pp. 799-811, 2001.

[8] M. Last, H. Bunke, A. Kandel, A Sequential Two-Stage Approach to Classifier Combination, *Submitted to Publication*.

[9] H. Liu and H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, 1998.

[10] O. Maimon and M. Last, Knowledge Discovery and Data Mining, The Info-Fuzzy Network (IFN) Methodology, Kluwer Academic Publishers, 2000.

[11] T.M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[12] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[13] A. F. R. Rahman and M. C. Fairhurst, Serial Combination of Multiple Experts: A Unified Evaluation, Pattern Analysis & Applications, No. 2, pp. 292–311, 1999.

[14] L.-X. Wang, A Course in Fuzzy Systems and Control, Prentice-Hall, 1997.

[15] L. A. Zadeh, A New Direction in System Analysis: From Computation with Measurements to Computation with Perceptions, In N. Zhong, et al., editors, New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, pages 10-11, Springer-Verlag, 1999.