Optimizing Classification Ensembles via a Genetic Algorithm for a Web-Based Educational System[*]

Behrouz Minaei-Bidgoli[1], Gerd Kortemeyer[2], and William F. Punch[1]

[1] Genetic Algorithms Research and Applications Group (GARAGe)
Department of Computer Science & Engineering, Michigan State University
2340 Engineering Building, East Lansing, MI 48824, USA
{minaeibi,punch}@cse.msu.edu
http://garage.cse.msu.edu

[2] Division of Science and Math Education, Michigan State University
College of Natural Science, LITE lab, East Lansing, MI 48824, USA
korte@lite.msu.edu
http://www.lon-capa.org

Abstract. Classification fusion combines multiple classifications of data into a single classification solution of greater accuracy. Feature extraction aims to reduce the computational cost of feature measurement, increase classifier efficiency, and allow greater classification accuracy based on the process of deriving new features from the original features. This paper represents an approach for classifying students in order to predict their final grades based on features extracted from logged data in an educational web-based system. A combination of multiple classifiers leads to a significant improvement in classification performance. By weighing feature vectors representing feature importance using a Genetic Algorithm (GA) we can optimize the prediction accuracy and obtain a marked improvement over raw classification. We further show that when the number of features is few, feature weighting and transformation into a new space works efficiently compared to the feature subset selection. This approach is easily adaptable to different types of courses, different population sizes, and allows for different features to be analyzed.

1 Motivation

Several web-based educational systems with different capabilities and approaches have been developed to deliver online education in an academic setting. In particular, Michigan State University (MSU) has pioneered some of these systems to provide an infrastructure for online instruction. The research presented here was performed on a part of the latest online educational system developed at MSU, the *Learning Online Network with Computer-Assisted Personalized Approach* (*LON-CAPA*). LON-CAPA

[*] This work was partially supported by the National Science Foundation under ITR 0085921.

is involved with two kinds of large data sets: 1) educational resources such as web pages, demonstrations, simulations, and individualized problems designed for use on homework assignments, quizzes, and examinations; and 2) information about users who create, modify, assess, or use these resources. In other words, we have two ever-growing pools of data.

This paper investigates methods for extracting useful and interesting patterns from these large databases of students using online educational resources and their recorded paths within the system. We aim to answer the following research questions: Can we find *classes* of students? In other words, do there exist groups of students who use these online resources in a *similar* way? If so, can we predict a class for any individual student? With this information, can we then *help* a student use the resources better, based on the usage of the resource by other students in their groups?

We hope to find similar patterns of use in the data gathered from LON-CAPA, and eventually make predictions as to the most-beneficial course of studies for each learner based on their past and present usage. The system could then make suggestions to the learner as to how best to proceed.

2 Background on Using GAs for Feature Selection/Extraction

Genetic Algorithms (GA) have been shown to be an effective tool to use in data analysis and pattern recognition [1], [2], [3]. An important aspect of GAs in a learning context is their use in pattern recognition. There are two different approaches to applying GA in pattern recognition:

1. Apply a GA directly as a classifier. Bandyopadhyay and Murthy in [4] applied GA to find the decision boundary in N dimensional feature space.
2. Use a GA as an optimization tool for resetting the parameters in other classifiers. Most applications of GAs in pattern recognition optimize some parameters in the classification process. Many researchers have used GAs in feature selection [5], [6], [7], [8]. GAs have been applied to find an optimal set of feature weights that improve classification accuracy. First, a traditional feature extraction method such as Principal Component Analysis (PCA) is applied, and then a classifier such as k-NN is used to calculate the fitness function for GA [9], [10]. Combination of classifiers is another area that GAs have been used to optimize. Kuncheva and Jain in [11] used a GA for selecting the features as well as selecting the types of individual classifiers in their design of a Classifier Fusion System. GA is also used in selecting the prototypes in the case-based classification [12].

In this paper we focus on the second approach and use a GA to optimize a combination of classifiers. Our objective is to *predict* the students' final grades based on their web-use features, which are extracted from the homework data. We design, implement, and evaluate a series of pattern classifiers with various parameters in order to compare their performance on a dataset from LON-CAPA. Error rates for the individual classifiers, their combination and the GA optimized combination are presented.

Two approaches are proposed for the problem of feature extraction and selection. The *filter model* chooses features by heuristically determined "goodness/relevant" or knowledge, while the *wrapper model* does this by the feedback of classifier evaluation, or experiment. Research has shown the wrapper model outperforms the filter model comparing the predictive power on unseen data [13]. We propose a wrapper model for feature extraction through setting different weights for features and getting feedback from ensembles of classifiers.

3 Dataset and Class Labels

As test data we selected the student and course data of a single LON-CAPA course, BS111 (Biological Sciences), which was held at MSU in spring semester 2003. This course integrated 24 homework sets, including 229 problems, all of which are online. All 402 students used LON-CAPA for this course. Some students who dropped the course in mid-semester have initial homework scores, but no final grades. After removing those students, there remained 352 valid samples. The grade distribution of the students is shown in Fig 1.

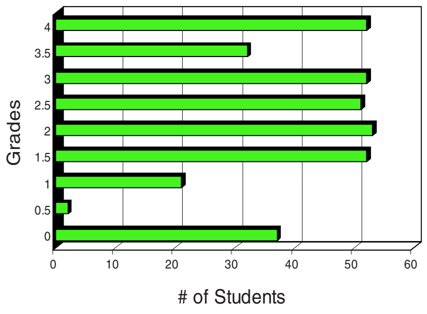

Fig. 1. LON-CAPA: BS111 SS03, Grades distribution.

Table 1. Selecting 9-class labels.

Class	Grade	# of Std.	Percentage
1	0.0	37	10.51%
2	0.5	2	0.57%
3	1.0	21	5.97%
4	1.5	52	14.77%
5	2.0	53	15.06%
6	2.5	51	14.49%
7	3.0	52	14.77%
8	3.5	32	9.09%
9	4.0	52	14.77%

We can group the students regarding their final grades in several ways, three of which are:

The nine possible labels can be the same as students' grades, as shown in Table 1

1. We can group them into three classes, "*high*" representing grades from 3.5 to 4.0, "*middle*" representing grades from 2.5 to 3, and "*low*" representing grades less than 2.5, as shown in Table 2.
2. We can also categorize students with one of two class labels: "*Passed*" for grades above 2.0, and "*Failed*" for grades less than or equal to 2.0, as shown in Table 3.

An essential step in performing classification is selecting the features used for classification. The BS111 course had an activity log with approximately 368 MB. After cleansing, we found 48 MB of useful data. We mined from these logged data 1,689,656 transactions from which the following features were extracted:

Table 2. Selecting 3-Classes labels regarding to students' grades in course BS111 SS03.

Class	Grade	# of Students	Percentage
High	Grade $\geq$ 3.5	84	23.86%
Middle	2.0 < Grade < 3.5	103	29.26%
Low	Grade $\leq$ 2.0	165	46.88%

Table 3. Selecting 2-Classes labels regarding to students' grades in course BS111 SS03.

Class	Grade	# of Students	Percentage
Passed	Grade > 2.0	187	53.13%
Failed	Grade $\leq$ 2.0	165	46.88%

1. Total number of tries for doing homework. (Number of attempts before correct answer is derived)
2. Total number of correct answers. (Success rate)
3. Getting the problem correct on the first try vs. those with high number of tries. (Success at the first try)
4. Getting the problem correct on the second try.
5. Getting the problem correct between 3 and 9 tries.
6. Getting the problem correct with high number of tries (10 or more tries).
7. Total time that passed from the first attempt, until the correct solution was demonstrated, regardless of the time spent logged in to the system.
8. Total time spent on the problem regardless of whether they got the correct answer or not.

4 Classification Ensembles

Pattern recognition has a wide variety of applications in many different fields, such that it is not possible to come up with a single classifier that can give good results in all cases. The optimal classifier in every case is highly dependent upon the problem domain. In practice, one might come across a case where no single classifier can achieve an acceptable level of accuracy. In such cases it would be better to pool the results of different classifiers to achieve the optimal accuracy. Every classifier operates well on different aspects of the training or test feature vector. As a result, assuming appropriate conditions, combining multiple classifiers may improve classification performance when compared with any single classifier.

The scope of this study is restricted to comparing some popular non-parametric pattern classifiers and a single parametric pattern classifier according to the error estimate. Four different classifiers using the LON-CAPA dataset are compared in this study. The classifiers used in this study include *Quadratic Bayesian classifier, 1-nearest neighbor (1-NN)*, *k-nearest neighbor (k-NN)*, *Parzen-window*[1]. These are some of the common classifiers used in most practical classification problems. After

[1] The classifiers are coded in MATLAB™ 6.5.

some preprocessing operations the optimal *k*=3 is chosen for *kNN* algorithm. To improve classification performance, a fusion of classifiers is performed.

Normaliztion. Having assumed in Bayesian and Parzen-window classifiers that the features are normally distributed, it is necessary that the data for each feature be normalized. This ensures that each feature has the same weight in the decision process. Assuming that the given data is Gaussian, this normalization is performed using the mean and standard deviation of the training data. In order to normalize the training data, it is necessary first to calculate the sample mean μ , and the standard deviation σ of each feature in this dataset, and then normalize the data using the equation (1).

$$x_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

This ensures that each feature of the training dataset has a normal distribution with a mean of zero and a standard deviation of unity. In addition, the *k*NN method requires normalization of all features into the same range.

Combination of Multiple Classifiers. Clearly, the data here suggest that in combining multiple classifiers we can improve classifier performance. There are different ways one can think of combining classifiers:

- The simplest way is to find the overall error rate of the classifiers and choose the one which has the least error rate on the given dataset. This is called an *offline classification fusion*. This may appear to be a classification fusion; however, in general, it has a better performance than individual classifiers.
- The second method, which is called *online classification fusion*, uses all the classifiers followed by a vote. The class getting the *maximum votes* from the individual classifiers will be assigned to the test sample.

Using the second method we show that classification fusion can achieve a significant accuracy improvement in all three cases of 2-, 3-, and 9-Classes. A GA is employed to determine whether classification fusion performance can be maximized.

5 GA-Optimized Ensembles of Classifications

Our goal is to find a population of best weights for every feature vector, which minimize the classification error rate. The feature vector for our predictors are the set of eight variables for every student: Number of attempts before correct answer is derived, Success rate, Success at the first try, Success at the second try, Success with number of tries between three and nine, Success with high number of tries, the time at which the student got the problem correct relative to the due date, and total time spent on the problem. We randomly initialized a population of eight dimensional weight vectors with values between 0 and 1, corresponding to the feature vector and experimented with different number of population sizes. We found good results using a population with 200 individuals. Real-valued populations may be initialized using the GA MATLAB Toolbox function *crtrp*. For example, to create a random population of

200 individuals with eight variables each: we define boundaries on the variables in *FieldD* which is a matrix containing the boundaries of each variable of an individual.

```
FieldD = [ 0 0 0 0 0 0 0 0 ;  % lower bound
           1 1 1 1 1 1 1 1];  % upper bound
```

We create an initial population with `Chrom = crtrp(200, FieldD)`, So we have for example:

```
Chrom = 0.23 0.17 0.95 0.38 0.06 0.26 0.31 0.52
        0.35 0.09 0.43 0.64 0.20 0.54 0.43 0.90
        0.50 0.10 0.09 0.65 0.68 0.46 0.29 0.67
        0.21 0.29 0.89 0.48 0.63 0.81 0.05 0.12
```

We used the simple genetic algorithm (SGA), which is described by Goldberg in [14]. The SGA uses common GA operators to find a population of solutions which optimize the fitness values. We used the *Stochastic Universal Sampling* [14] as our selection method, mainly due to its popularity and functionality. A form of stochastic universal sampling is implemented by obtaining a cumulative sum of the fitness vector, *FitnV*, and generating *N* equally spaced numbers between 0 and sum(FitnV). Thus, only one random number is generated, all the others used being equally spaced from that point. The index of the individuals selected is determined by comparing the generated numbers with the cumulative sum vector. The probability of an individual being selected is then given by

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)} \tag{2}$$

where $f(x_i)$ is the fitness of individual x_i and $F(x_i)$ is the probability of that individual being selected.

The operation of *crossover* is not necessarily performed on all strings in the population. Instead, it is applied with a probability *Px* when the pairs are chosen for breeding. We selected *Px* = 0.7 since this would preserve a reasonably high level of the original population. Intermediate recombination combines parent values using the following formula [15]:

$$\mathit{Offspring} = \mathit{parent1} + \mathit{Alpha} \times (\mathit{parent2} - \mathit{parent1}) \tag{3}$$

where Alpha is a scaling factor chosen uniformly in the interval [-0.25, 1.25].

A further genetic operator, *mutation* is applied to the new chromosomes, with a set probability *Pm* as the rate of mutation. Mutation causes the individual genetic representation to be changed according to some probabilistic rule. Mutation is generally considered to be a background operator that ensures that the probability of searching a particular subspace of the problem space is never zero. This has the effect of tending to inhibit the possibility of converging to a local optimum, rather than the global optimum. We considered *Pm* = 1/800 as our mutation rate, due to its small value with respect to the population. The mutation of each variable is calculated as follows:

$$\mathit{Mutated\ Var} = \mathit{Var} + \mathit{MutMx} \times \mathit{range} \times \mathit{MutOpt(2)} \times \mathit{delta} \tag{4}$$

where delta is an internal matrix which specifies the normalized mutation step size; MutMx is an internal mask table; and MutOpt specifies the mutation rate and its shrinkage during the run.

During the reproduction phase, each individual is assigned a *fitness value* derived from its raw performance measure given by the objective function. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating whereas less fit individuals have a correspondingly low probability of being selected. The error rate is measured in each round of cross validation by dividing "the total number of misclassified examples" into "total number of test examples". Therefore, our *fitness function* measures the accuracy rate achieved by classification fusion and our objective would be to maximize this performance (minimize the error rate).

6 Experimental Results

Without using GA, the overall results of classification performance on our dataset for four classifiers and classification fusion are shown in the Table 4. Regarding individual classifiers, 1NN and *k*NN have the best performance in the case of 2-, 3-, and 9-Classes, of approximately 62%, 50% and 35% accuracy, respectively. However, the classification fusion improved the classification accuracy significantly in all three cases. That is, it achieved 72% accuracy in the case of 2-Classes, 59% in the case of 3-Classes, and 43% in the case of 9-Classes.

Table 4. Comparing the average performance (%) of ten runs of classifiers on BS111 dataset for 2-, 3-, and 9-Classes, using 10-fold cross validation, without GA optimization.

Classifier	2-Classes	3-Classes	9-Classes
Bayes	52.6	38.8	22.1
1NN	**62.1**	45.3	29.0
*k*NN	55.0	**50.6**	**34.5**
Parzen	59.7	42.9	22.6
Classification Fusion	***72.2***	***58.8***	***43.1***

For GA optimization, we used 200 individuals (weight vectors) in our population, running the GA over 500 generations. We ran the program 10 times and obtained the averages, which are shown, in Table 5.

Table 5. Comparing the classification fusion performance on BS111 dataset Using-GA and without-GA in the cases of 2-, 3-, and 9-Classes, 95% confidence interval.

Classifier	2-Classes	3-Classes	9-Classes
Classification fusion of 4 Classifiers without GA optimization	71.19 ± 1.34	58.92 ± 1.36	42.94 ± 2.06
GA Optimized Classification Fusion, Mean individual (not best value)	81.09 ± 2.42	70.13 ± 0.89	55.25 ± 1.03
Improvement of Mean individual	***9.82 ± 1.33***	***11.06 ± 1.84***	***12.71 ± 0.75***

The results in Table 5 represent the mean performance with a two-tailed t-test with a 95% confidence interval. For the improvement of GA over non-GA result, a P-value indicating the probability of the Null-Hypothesis (There is no improvement) is also given, showing the significance of the GA optimization. All have p<0.001, indicating significant improvement. Therefore, using GA, in all the cases, we got more than a 10% mean individual performance improvement and about 11 to 16% best individual performance improvement. Fig. 2 shows the results of one of the ten runs in the case of 2-Classes. The dotted line represents the population mean, and the solid line shows the best individual at each generation and the best value yielded by the run (Due to the space limitation, only two graphs are shown).

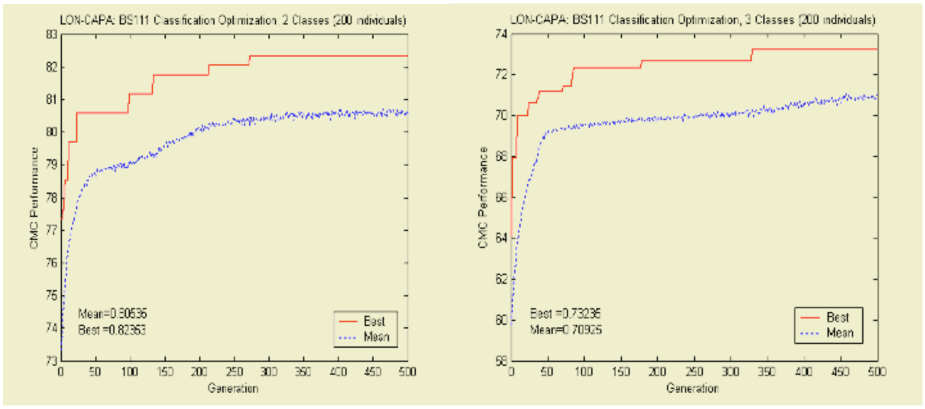

Fig. 2. GA-Optimized Combination of Multiple Classifiers' (CMC) performance in the case of 2- and 3-Classes, 200 weight vectors individuals, 500 Generations.

Finally, we can examine the individuals (weights) for features by which we obtained the improved results. This feature weighting indicates the *importance* of each feature for making the required classification. In most cases the results are similar to Multiple Linear Regressions or some tree-based software (like CART) that use statistical methods to measure feature importance. The GA feature weighting results, as shown in Table 6, state that the "Success with high number of tries" is the most important feature in all three cases. The "Total number of correct answers" feature is also important in some cases.

Table 6. Relative Feature Importance%, Using GA weighting.

Feature	2-Classes	3-Classes	9-Classes
Total Number of Tries	18.9	17.8	10.7
Total # of Correct Answers	**84.7**	**57.1**	27.4
# of Success at the First Try	14.4	**55.2**	34.2
# of Success at the Second Try	16.5	25.9	22.0
Got Correct with 3-9 Tries	21.2	38.8	11.1
Got Correct with # of Tries $\geq$ 10	**91.7**	**69.1**	**67.3**
Time Spent to Solve the Problems	32.1	14.1	28.3
Total Time Spent on the Problems	36.5	15.4	33.5

7 Conclusions and Future Work

We proposed a new approach to classifying student usage of web-based instruction. Four classifiers are used in grouping the students. A combination of multiple classifiers leads to a significant accuracy improvement in the 2-, 3- and 9-Class cases. Weighing the features and using a genetic algorithm to minimize the error rate improves the prediction accuracy by at least 10% in the all three test cases. In cases where the number of features is low, feature weighting worked much better than feature selection. The successful optimization of student classification in all three cases demonstrates the merits of using the LON-CAPA data to *predict* the students' final grades based on their features, which are extracted from the homework data. This approach is easily adaptable to different types of courses, different population sizes, and allows for different features to be analyzed. This work represents a rigorous application of known classifiers as a means of analyzing and comparing use and performance of students who have taken a technical course that was partially/completely administered via the web. In the present work, we propose an approach for predicting students' performance based on extracting the average of feature values over all of the problems in a course. For future work, we plan to implement such an optimized assessment tool for every student on any particular problem. Therefore, we can track students' behaviors on a particular problem over several semesters.

References

1. Raymer, M.L. Punch, W.F., Goodman, E.D., Kuhn, L.A., and Jain, A.K.: Dimensionality Reduction Using Genetic Algorithms. IEEE Transactions on Evolutionary Computation, Vol. 4, (2000) 164-171
2. Jain, A. K.; Zongker, D. Feature Selection: Evaluation, Application, and Small Sample Performance. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 19, No. 2, February (1997)
3. De Jong K.A., Spears W.M. and Gordon D.F.: Using genetic algorithms for concept learning. Machine Learning 13, (1993) 161-188
4. Bandyopadhyay, S., and Muthy, C.A.: Pattern Classification Using Genetic Algorithms. Pattern Recognition Letters, Vol. 16, (1995) 801-808
5. Bala J., De Jong K., Huang J., Vafaie H., and Wechsler H.: Using learning to facilitate the evolution of features for recognizing visual concepts. Evolutionary Computation 4(3) - Special Issue on Evolution, Learning, and Instinct: 100 years of the Baldwin Effect (1997)
6. Guerra-Salcedo C. and Whitley D.: Feature Selection mechanisms for ensemble creation: a genetic search perspective. In: Freitas AA (Ed.) Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop, 13-17. Technical Report WS-99-06. AAAI Press (1999)
7. Vafaie, H. and De Jong, K.: Robust feature Selection algorithms. Proceeding of IEEE International Conference on Tools with AI, Boston, Mass., USA. Nov. (1993) 356-363

8. Martin-Bautista M.J., and Vila M.A.: A survey of genetic feature selection in mining issues. Proceeding Congress on Evolutionary Computation (CEC-99), Washington D.C., July (1999) 1314-1321
9. Pei, M., Goodman, E.D., and Punch, W.F.: Pattern Discovery from Data Using Genetic Algorithms. Proceeding of 1st Pacific-Asia Conference Knowledge Discovery & Data Mining (PAKDD-97) (1997)
10. Punch, W.F., Pei, M., Chia-Shun, L., Goodman, E.D., Hovland, P., and Enbody R.: Further research on Feature Selection and Classification Using Genetic Algorithms. In 5th International Conference on Genetic Algorithm, Champaign IL, (1993) 557-564
11. Kuncheva, L.I., and Jain, L.C.: Designing Classifier Fusion Systems by Genetic Algorithms. IEEE Transaction on Evolutionary Computation, Vol. 33 (2000) 351-373
12. Skalak D. B.: Using a Genetic Algorithm to Learn Prototypes for Case Retrieval an Classification. Proceeding of the AAAI-93 Case-Based Reasoning Workshop, Washigton, D.C., American Association for Artificial Intelligence, Menlo Park, CA, (1994) 64-69
13. John, G.H., Kohavi, R., Pfleger K.: Irrelevant Features and the Subset Selection Problem. Proceedings of the Eleventh International Conference of Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA (1994) 121-129
14. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. MA, Addison-Wesley (1989)
15. Muhlenbein and Schlierkamp-Voosen D.: Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization, Evolutionary Computation, Vol. 1, No. 1 (1993) 25-49