Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System

GECCO 2003

Behrouz Minaei, William Punch

Genetic Algorithm Research and Application Group
Department of Computer Science and Engineering
Michigan State University

Topics

- Problem Overview
- Classification Methods
  - Classifiers
  - Combination of Classifiers
- Weighting the features
- Using GA to choose best set of weights
- Experimental Results
- Conclusion & Next Steps

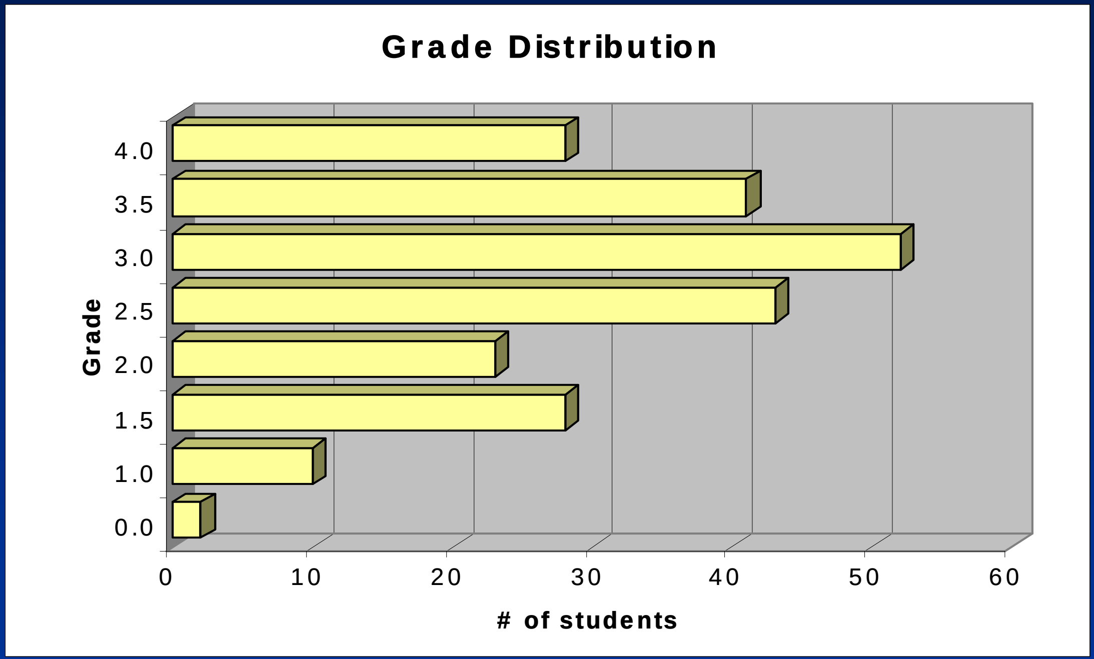
Problem Overview

- This research is a part of the latest online educational system developed at Michigan State University (MSU), the *Learning Online Network with Computer-Assisted Personalized Approach* (*LON-CAPA*).

- In LON-CAPA, we are involved with two kinds of large data sets:
  - Educational resources: web pages, demonstrations, simulations, individualized problems, quizzes, and examinations.
  - Information about users who create, modify, assess, or use these resources.

- Find *classes* of students. Groups of students use these online resources in a *similar* way.
- *Predict* for any individual student to which class he/she belongs.

Data Set: PHY183 SS02



- 227 students
- 12 Homework sets
- 184 Problems

Class Labels (3-ways)

2-Classes

1	**Passed**	Grade > 2.0	164	72.20%
2	**Failed**	Grade <= 2.0	63	27.80%

3-Classes

1	**High**	Grade >= 3.5	69	30.40%
2	**Middle**	2.0 < Grade < 3.5	95	41.80%
3	**Low**	Grade <= 2.0	63	27.80%

9-Classes

Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Grade	0	0.5	1	1.5	2	2.5	3	3.5	4
# of students	2	0	10	28	23	43	52	41	28
Percentage	0.90%	0.00%	4.40%	12.40%	10.10%	18.90%	22.90%	18.00%	12.40%

Extracted Features

1. Total number of correct answers. (Success rate)
2. Success at the first try
3. Number of attempts to get answer
4. Time spent until correct
5. Total time spent on the problem
6. Participating in the communication mechanisms

Classifiers

- Non-Tree Classifiers (Using MATLAB)
  - Bayesian Classifier
  - 1NN
  - kNN
  - Multi-Layer Perceptron
  - Parzen Window
  - Combination of Multiple Classifiers (CMC)
  - *Genetic Algorithm* (GA)

- Decision Tree-Based Software
  - C5.0 (RuleQuest <<C4.5<<ID3)
  - CART (Salford-systems)
  - QUEST (Univ. of Wisconsin)
  - CRUISE [use an *unbiased* variable selection technique]

GA Optimizer vs. Classifier

- Apply GA directly as a classifier
- Use GA as an optimization tool for resetting the parameters in other classifiers.
  - Most application of GA in pattern recognition applies GA as an optimizer for some parameters in the classification process.
  - Many researchers used GA in feature selection and feature extraction.
  - GA is applied to find an optimal set of feature weights that improve classification accuracy.

- Download GA Toolbox for MATLAB from:
  - http://www.shef.ac.uk/~gaipp/ga-toolbox/

Fitness/Evaluation Function

- 5 classifiers:
  1. Multi-Layer Perceptron — 2 Minutes
  2. Bayesian Classifier
  3. 1NN
  4. kNN
  5. Parzen Window
  - CMC — 3 seconds
- Divide data into training and test sets (10-fold Cross-Val)
- **Fitness function: performance achieved by classifier**

$$\text{Error Rate in each round} = \frac{\text{Total missclassified of test examples}}{\text{Total number of test examples}}$$

Individual Representation

- The GA Toolbox supports binary, integer and floating-point chromosome representations.
- Chrom = crtrp(NIND, FieldDR) creates a random real-valued matrix of N x d, NIND specifies the number of individuals in the population
- FieldDR is a matrix of size 2 x d and contains the boundaries of each variable of an individual.
- FieldDR = [0 0 0 0 0 0; % lower bound
-                    1 1 1 1 1 1]; % upper bound
- Chrom = 0.23   0.17   0.95   0.38   0.06   0.26
-               0.35   0.09   0.43   0.64   0.20   0.54
-               0.50   0.10   0.09   0.65   0.68   0.46
-               0.21   0.29   0.89   0.48   0.63   0.89

GA Parameters

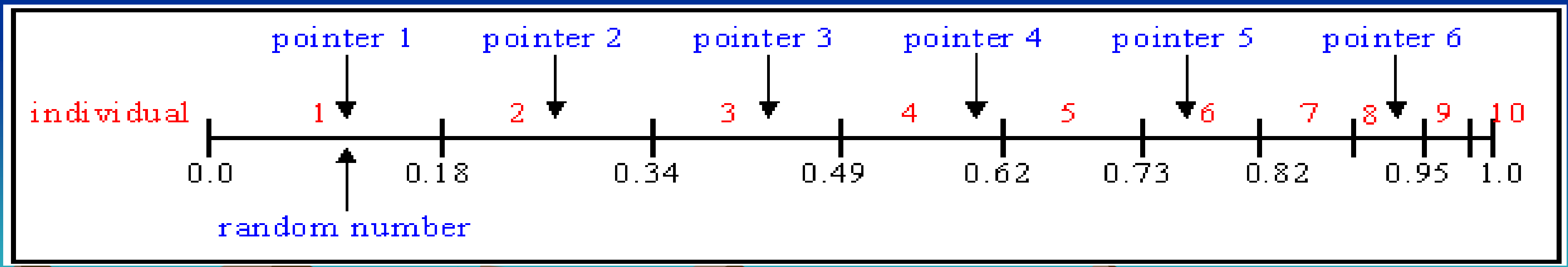
- GGAP = 0.3
- XOVR = 0.7
- MUTR = 1/600
- MAXGEN = 500
- NIND = 1000
- SEL_F = ‘sus’
- XOV_F = ‘recint’
- MUT_F = ‘mutbga’
- OBJ_F = ‘ObjFn2’ / ‘ObjFn3’ / ‘ObjFn9’

Simple GA

% Create population Chrom = crtrp(NIND, FieldD);
% Evaluate initial population, ObjV = ObjFn2(data, Chrom);
gen = 0;
while gen < MAXGEN,
% Assign fitness-value to entire population FitnV = ranking(ObjV);
% Select individuals for breeding SelCh = select(SEL_F, Chrom, FitnV, GGAP);
% Recombine selected individuals (crossover)
SelCh = recombin(XOV_F, SelCh,XOVR);
% Perform mutation on offspring
SelCh = mutate(MUT_F, SelCh, FieldD, MUTR);
% Evaluate offspring, call objective function ObjVSel = ObjFn2(data, SelCh);
% Reinsert offspring into current population
[Chrom ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel);
gen = gen+1;
MUTR = MUTR+0.001;
end

Selection - Stochastic Universal Sampling

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



Crossover

OldChrom = [0.23 0.17 0.95 0.38 0.82 0.19;   % parent1
0.43 0.64 0.20 0.54 0.43 0.32]   % parent2

Intermediate Recombination:

NewChrom = recint(OldChrom)

New values are produced by adding the scaled difference between the parent values to the first parent.

An internal table of scaling factors, Alpha, is produced, e.g.

Alpha = [0.13 0.50 0.32 0.16 0.23 0.06;   % for offspring1
0.12 0.54 0.44 0.26 0.27 0.10]   % for offspring2

NewChrom = [0.40 0.92 0.86 0.33 0; % Alpha(1,:) parent1&2
0.11 0.59 0.98 0.04]% Alpha(2,:) parent1&2

Mutation

OldChrom = [0.23 0.81 0.17 0.66 0.28 0.30;
0.43 0.96 025 0.64 0.10 0.23]

NewChrom = mutbga(OldChrom, FieldDR, [0.02 1.0]);

mutbga produces an internal mask table, MutMx = [0 0 0 1 0 1;
0 0 1 0 0 0]

An second internal table, delta, specifies the normalized mutation step size,

delta = [0.02 0.02 0.02 0.02 0.02 0.02;
0.01 0.01 0.01 0.01 0.01 0.01]

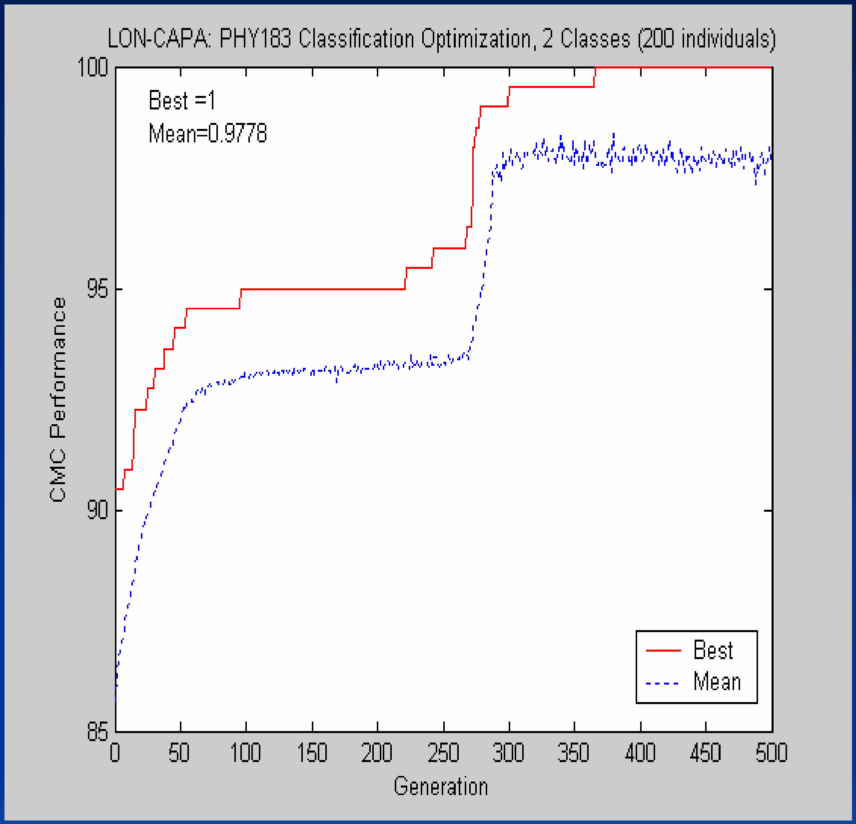
NewChrom =0.23 0.81 0.17 0.66 0.28 0.30
0.52 0.96 0.25 0.64 0.68 0.39

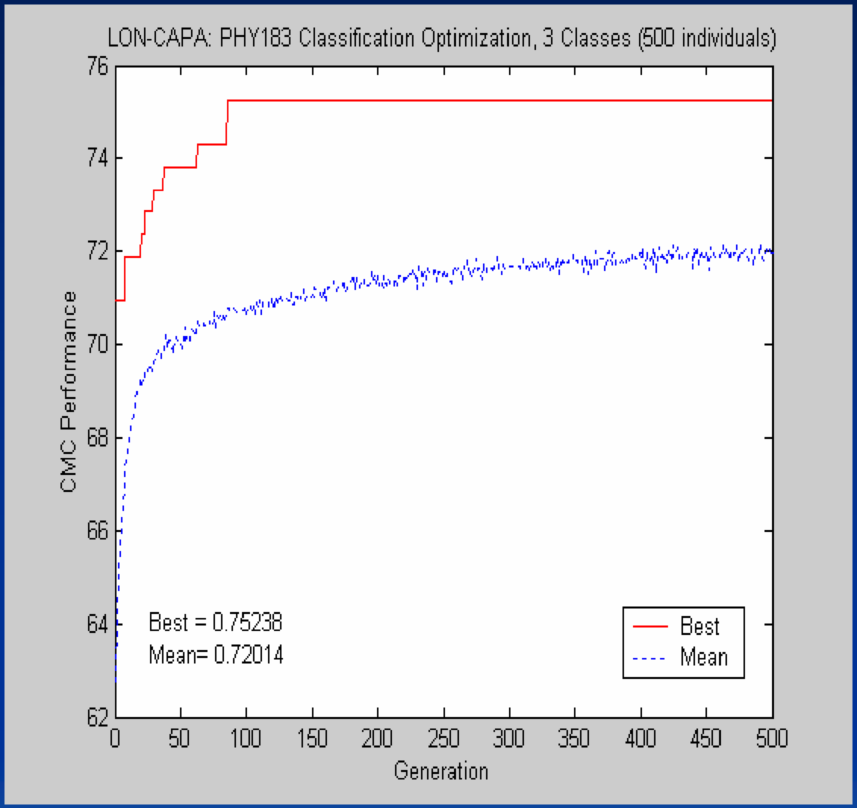
NewChrom - OldChrom shows the mutation steps = 0 0 0 4.61
0 0 -7.51 -5.01

Results without GA

		Performance %		
Classifier		2-Classes	3-Classes	9-Classes
Tree Classifier	C5.0	80.3	56.8	25.6
	CART	81.5	**59.9**	**33.1**
	QUEST	80.5	57.1	20.0
	CRUISE	81.0	54.9	22.9
Non-tree Classifier	Bayes	76.4	48.6	23.0
	1NN	76.8	50.5	29.0
	kNN	**82.3**	50.4	28.5
	Parzen	75.0	48.1	21.5
	MLP	79.5	50.9	-
	CMC	**86.8**	**70.9**	**51.0**

Results of using GA

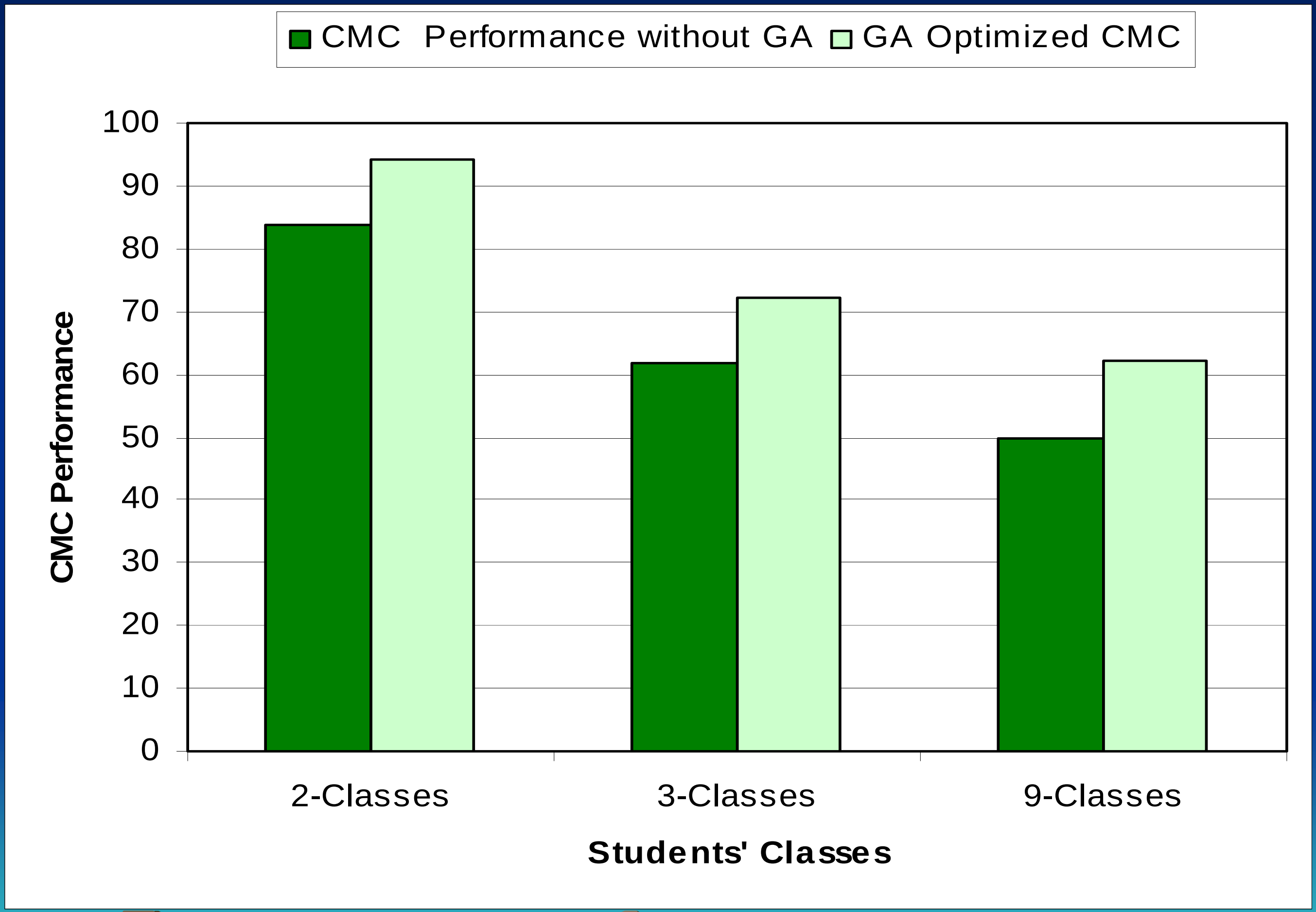




GA Optimization Results

	Performance %		
Classifier	2-Classes	3-Classes	9-Classes
CMC of 4 Classifiers without GA	83.87 ± 1.73	61.86 ± 2.16	49.74 ± 1.86
GA Optimized CMC, Mean individual	94.09 ± 2.84	72.13 ± 0.39	62.25 ± 0.63
Improvement	10.22 ± 1.92	10.26 ± 1.84	12.51 ± 1.75

GA Optimization Results



GA Optimization Results

Feature	Importance %
Total_Correct _Answers	100.00
Total_Number_of_Tries	58.61
First_Got_Correct	27.70
Time_Spent_to_Solve	24.60
Total_Time_Spent	24.47
Communication	9.21

Summary

- Four classifiers used to segregate the students
- A combination of multiple classifiers leads to a significant accuracy improvement in all 3 cases.
- Weighted the features and used genetic algorithm to minimize the error rate.
- Genetic Algorithm could improve the prediction accuracy more than 10% in the case of 2 and 3-Classes and more than 12% in the case of 9-Classes.

Next Steps

- Using Evolutionary Computation for extracting the features to find the optimal solution in LON-CAPA Data Mining
- Using Genetic programming to extract new features and improve prediction accuracy
- Apply EA to find the Frequency Dependency and Association Rules among the groups of problems *(Mathematical, Optional Response, Numerical, Java Applet, etc)*

Questions