GENETICS-BASED MACHINE LEARNING SYSTEMS FOR CLASSIFICATION TASK

Helen Afanasyeva

Keywords: classifier systems, genetic algorithms, crowding, bucket brigade algorithm, classification task

Introduction

The necessity to analyse and formalise tasks dealing with the comparison and classification of various objects was recognised long ago. Various pattern recognition methods were employed: supervised and unsupervised learning, reinforcement learning, etc. Well-known statistic methods [1], inductive methods, evolutionary methods and neural networks can also be mentioned here.

This paper examines classifier system construction principles based on evolutionary methods (genetic algorithms in essence are evolutionary methods). Possibilities of this classification method are studied using a specific case study: mushroom classification task.

To describe some principles of genetics-based learning systems, the concept of a system of classifiers based on rules and messages is introduced. A classifier system is a machine learning system that learns syntactically simple string rules (called classifiers) able to solve tasks of control, classification and forecasting. Classifier systems consist of three main components:

- Rule and message system;
- Apportionment of credit system assignment;
- Genetic algorithm.

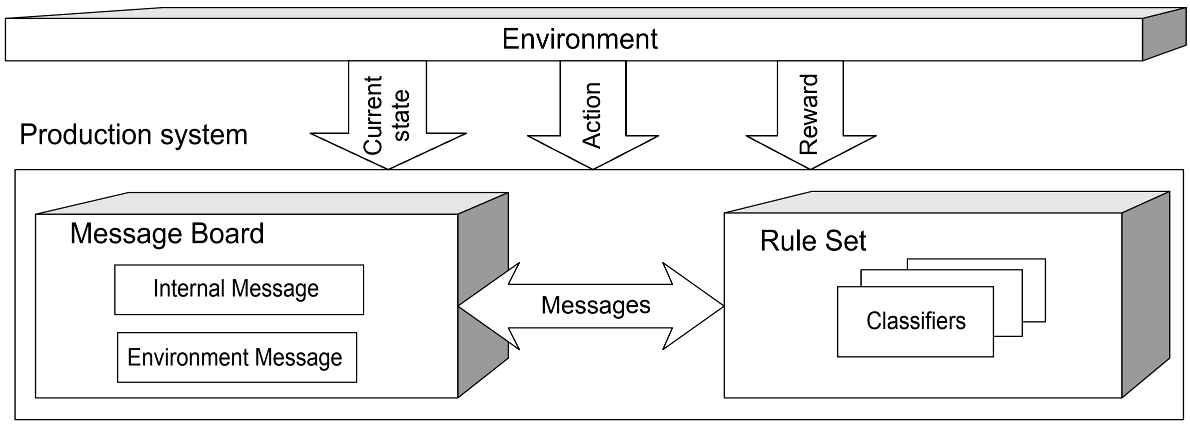

Fig. 1 Classifier system interpretation

Sets of rules and messages in the genetic system are a kind of production system [7]. Messages come from the environment or system classifiers. They are compared to classifier rules with regard to matching. A message in the classifier system is a string of fixed length consisting of symbols of a certain alphabet. The more frequently a message matches the condition part of the rule, the more valuable the rule with which it has coincided is. The rule value has to be ascertained in the course of work by means of competition and using credit assignment methods.

The exchange and accumulation of an internal currency provides a natural figure of merit for the application of genetic algorithms. Using a classifier's bank balance as a fitness function, classifiers may be reproduced, crossed and mutated. Thus, not only can the system learn by ranking extant rules, it can also discover new, possibly better rules as innovative combinations of its old rules.

Together, apportionment of credit via competition and rule discovery using genetic algorithms form a reasonable basis for constructing a machine learning system [2].

The Bucket Brigade Algorithm

Various methods for ranking classifiers according to their role in receiving the environment reward exist. The most popular is the method called by Holland [2] the Bucket Brigade Algorithm. This algorithm can easily be interpreted in economic terms where trading rights are sold and purchased by classifiers. Classifiers form a chain of middlemen from information manufacturer (the environment) and information consumer (the effector).

When receiving a message, the classifiers matched participate in an auction where the bid is proportional to the strength and specificity of the classifier under consideration:

$$specificity = \frac{L - number\ of\ symbols\,\#}{L},\ where\, L\, is\, the\, message\, length \quad (1)$$

$$Bid(t+1) = C_{bid} \cdot (bid1 + bid2 \cdot specificity) \cdot S(t) \quad (2)$$

The classifier that has won the auction pays an amount that is the source of income for the message sender and receives the right to send a new message to the list of messages. Each classifier also pays a tax for existence, so that non-participating rules in the chains gradually lose their strength. The competitive nature of economy ensures that good rules (profitable) servive and bad ones (unprofitable) die off. So the classifier strength at the next time step can be calculated by formula:

$$S(t+1) = S(t) - C_{bid} \cdot S(t) \cdot (bid1 + bid2 \cdot specificity) - C_{tax} \cdot S(t) + R(t), \quad (3)$$

where $(0 < C_{bid} < 1)$ is the classifier bid,

$(0 < C_{tax} < 1)$ is the classifier tax => $C_{tax} \cdot S(t)$ is taxation,

$R(t)$ is the revenue obtained from the classifiers that will be activated [2].

The Simplified Classifier Systems

The classifier system under consideration can be simplified; it especially concerns the credit assignment method [2]. The distinguishing feature of such systems is that they do not have a message list. In those systems classifiers do not make any chains but the first and only classifier that won in the competition bids and receives an environment reward for the correct action. The remaining matched classifiers pay a certain tax for the bid. The tax value is much less than the bid. The bid values for the participants of the competition are calculated by formulae (1) and (2). New classifier string values are calculated as follows:

1) By formula (3) if it is the classifier that has won;

2) Using expression below for those classifiers that have matched but have not passed the competition:

$$S(t+1) = S(t) - CT_{bid} \cdot S(t) - C_{tax} \cdot S(t);$$

3) Using expression that follows for all the other classifiers:

$$S(t+1) = S(t) - C_{tax} \cdot S(t)$$

where $(0 < CT_{bid} < 1)$ is the bid tax;

$(0 < C_{tax} < 1)$ is the value of life tax => $C_{tax} \cdot S(t)$: taxation,

$R(t)$ the revenue received by the classifier from the environment for the correct action.

Genetic algorithms in classifier systems

As was previously mentioned, to derive a new generation in classifier systems, genetic algorithms are used. They are applied after the system has reached a stationary state and no quality improvement is observed. This is the time when new classifiers have to be introduced.

Crowding is a procedure that ensures the replacement of the weakest rules with those that can be derived from the strongest ones after genetic operators, crossover and mutation, have been applied to them. The specific feature of this procedure is that a whole set of potentially replaceable classifiers is formed but only one classifier is replaced which is closest by structure to the offspring that is to replace it. This enables one to preserve the experience previously accumulated by the system. The corresponding tests have demonstrated that direct application of genetic algorithms without using the crowding procedure actually does not work [2], [3].

For the execution of the algorithm that will be considered in detail later, the following parameters are of great importance. The strength value for new classifiers is calculated as the arithmetical mean of ancestors. However, the dependence on the specificity can also be added [6].

pmutation – probability mutation of a single gene;
pcrossover – crossover probability of a pair of individuals;
crowdingfactor – the number of sub-populations for selecting the most appropriate classifier out of the weakest for its replacement;
crowdingsubpop – the size of a sub-population for selecting the weakest individual;
matchcount – a parameter characterising the similarity degree of potentially replaceable object;
nselect **–** the number of pairs of individuals that have to be replaced at the present stage of the genetic algorithm;
bid1, bid2 **–** coefficients required to calculate the specificity influence on the bid.

Table 1. Algorithm for replacement procedure: crowding

Step	*Action required*
1	With the help of the roulette wheel, whose segments are proportional to the classifier strength, two (most probably) individuals with the largest strength values are selected and then copied to mate1 & mate2.
2	Crossover and mutation are applied to mate1 and mate2 with probability *pmutation* and *pcrossover*, respectively. As a result, child1 and child2 are derived.
3	Is executed for each of the offspring obtained. A classifier most similar to the offspring is selected *crowdingfactor* times from the *subpopulation* of size *crowdingsubpop*. As a result, 2 candidates for replacement are obtained: mort1 and mort2.
4	Replace mort1 with child1 and mort2 with child2, respectively.
5	Return to Step 1 *nselect* times.

Let us describe Step 3 in more detail.

Cycle from 1 to crowdingfactor

/ a cycle for selecting a classifier most similar to the* offspring */

Select a classifier randomly. So far it is the worst. So far it is the most similar as well.

Cycle from 2 to crowdingsubpop

*/*a cycle for selecting a classifier that is the worst as to the classifier strength */*

Select a classifier randomly and compare it to the worst.

Bring back the worst classifier

Compare the worst classifier obtained with the most similar one as to the value of matchcount (similarity in #,1,0 in rules and in the result). Select the most similar of them. In the output we will have the worst as to the strength value classifier and the most similar to the replacing classifier.

Object area analysis

As a database for constructing a classifier system, a mushroom database from two families of mushrooms, agaricus and lepiota was used which contained data about 8124 individuals [5].

It was necessary to classify the mushrooms into two classes: edible and poisonous on the basis of 22 external features such as shape, colour, odour, habitat, etc. (see Table 2). It is known that previously similar classification was performed in different ways and 95% recognition accuracy was achieved. The mushrooms were represented by a 57-bit message and fitted a simplified classifier system with a single condition. The result was represented by one bit: 1-poisonous, 0 – edible.

The percentage of symbols "#" (i.e. symbols that are not significant for classification) was taken rather high, as probabilistic evaluations showed [4], and was 93% for the initial system and 95% after learning.

Table 2. Classification features

No	*Number of bits for coding*	*Description*	*No*	*Number of bits for coding*	*Description*
1	3	cap-shape	13	2	stalk-surface below ring
2	2	cap-surface	14	4	stalk-colour above ring
3	4	cap-colour	15	4	stalk-colour below ring
4	1	presence of bruises	16	1	veil-type
5	4	odour	17	2	veil-colour
6	2	gill-attachment	18	2	ring-number
7	2	gill-spacing	19	3	ring-type
8	1	gill-size	20	4	spore-print-colour
9	4	gill-colour	21	3	population
10	1	stalk-shape	22	3	habitat
11	3	stalk-root	23	1	poisonous/edible
12	2	stalk-surface above ring			

Experiments with model and results validation and verification

First, experiments were performed for the population size equal to 50. Here the percentage of mushroom classification at the 100th generation of genetic algorithms has reached 80% on average although in the most successful experiments it was also possible to reach 93% classification. The classifier system has employed all the data for constructing a learning sample. Fig.2 shows several experiments with the system and the mean value over the experiments.

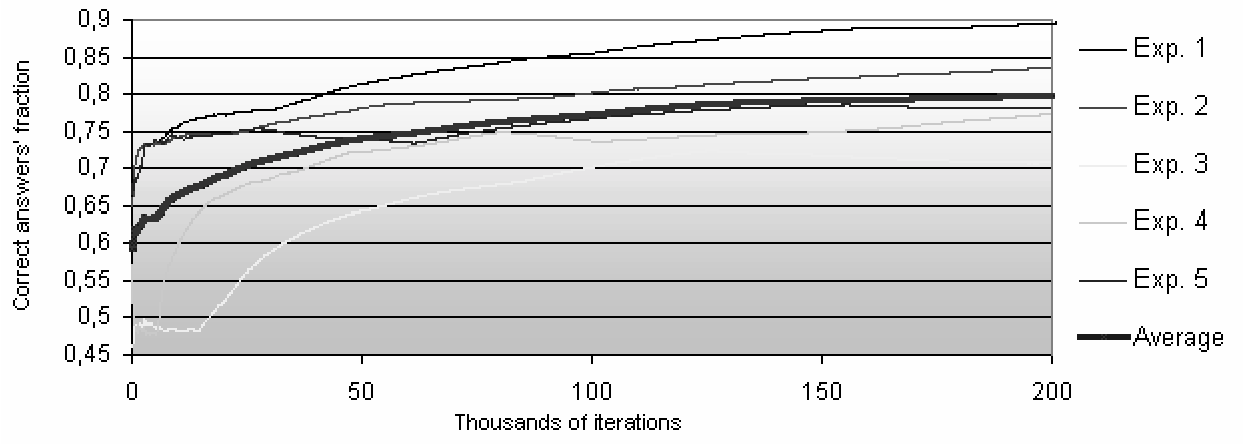

Figure 2 Learning system of 50 classifiers (with learning sample 8124)

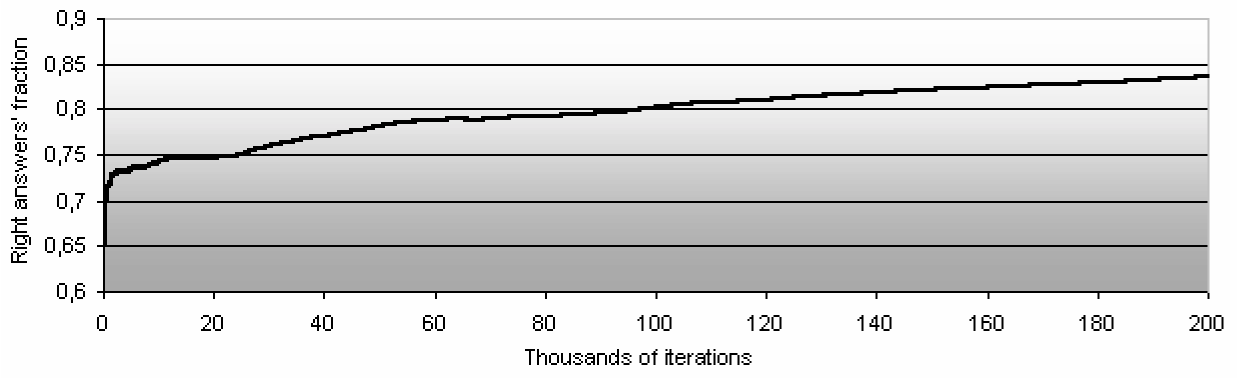

Figure 3 System learning of 50 classifiers (learning sample 2000)

Then the classifiers were trained using part of the database where 2000 individuals were presented (nearly equal number of edible and poisonous mushrooms like in the whole population).

As can be seen from Fig.3, in this case a larger and, which is most important, faster convergence can be observed, approximately 85% of correct answers in the training sample. The system quickly adapts to the initial data, that is why further experiments on the data out of the non-training sample (6124 individuals) yielded worse results than in training over the whole population (See Fig.2).

Here the system holds, on average at the level of 75% of correct answers; however, 5-6% fluctuations to both sides can be observed. That is why it is recommended to add noise to the initial data for system learning that will be accounted when calculating the bid at the auction.

Genetic algorithms were applied every 2000 iterations. It was supposed that during this time the system would certainly reach a stationary state. 100 classifier generations have been generated. Then the number of classifiers was increased up to 200, and 200 generations of classifiers were constructed (see Fig.4).

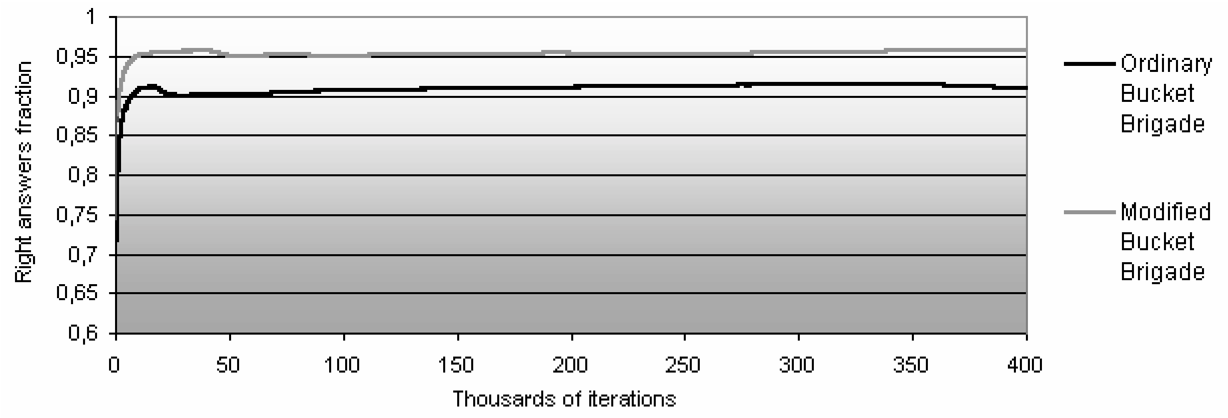

Figure 4 Classifier system with 200 classifiers

Under the same conditions the system's ability to recognition has increased to 90% at learning over all data available. At the same time, program execution time has essentially increased. It was noted that the most successful classifiers for the system under consideration could be selected faster out of a greater variety.

In [4] it is also recommended to make a slight modification of the Bucket Brigade algorithm in which the bid value would not depend on the classifier specificity. However, the specificity is still present at the auction when selecting an active classifier. With this, the average classification factor reached is 95%. This is because the specificity function value depends on both the specificity itself and parameters bid1 and bid2 and is always much less that 1. Therefore, this essentially reduces the bid that is compensated with a reward by the environment if the classification was correct. Hence, when the classification is not correct, the bid pays a kind of penalty for an incorrect answer that becomes larger in the modified algorithm.

The system's ability to classify objects that do not belong to the training sample is also about 95%. 10 most successful classifiers for one of experiments are shown in Table 3. On the basis of this data one can assess the classifier contribution to the classification.

Table 3. List of the most progressive classifiers

Classifier in the encoded form	*Strength*
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	
### 0# #### # #### ## ## # #### 1 ### ## ## #### #### # ## ## ### #### ### ##0 1	9.98
### ## ###1 # #### #0 ## # #1## # ### #0 ## #### #### # ## ## ### ###1 ### ### 0	9.96
### ## #### 1 #### ## ## # #### # ### 1# ## #### #### # ## ## ### ###1 ### ### 0	9.94
### ## #### # #0## ## ## # ##0# # ### ## ## #1## #### # 1# ## ### #### ### ##0 0	9.92
### ## #### # #### ## ## # ##0# # ### ## ## #### #### # ## ## ### ###1 ### ### 1	9.92
### ## #### # #### ## ## # #### 1 ### ## ## #### #### # ## ## ### #### ### ##0 1	9.17
### ## #### 1 #### ## ## # #### # ### 1# ## #### #### # ## ## ### #### ### ##0 0	9.15
### ## #### # #### ## ## # ##0# # ### ## ## #### #### # ## ## ### ###1 ### ### 1	7.33
### ## #### # #### ## ## # #### 1 ### ## ## #### #### # ## ## ### #### ### ##0 1	6.47
#0# ## ###1 # #### ## ## # #### 1 ### ## ## #### #### # ## ## ### #### ### ##0 0	6.25

Overall statistics over 200 classifiers of the 200th generation			
Max Strength	Min Strength	Sum Strength	Average Strength
9.9841	0.5416	646.1728	3.2309

It should be noted that trends to mark out attributes for classification do not coincide in different experiments. Table 4 shows the most successful coefficients employed to construct a model with 50/200 classifiers. As can be seen, the number of genes essential for the classification is approximately 3 in each classifier, which makes 5% of the total number of symbols in a rule. Attributes, however, exist (see Table 3) which do not affect the classification at all: 7,8,11,13,15,16,18,19 and 21, whereas attributes 16,17,18,19 and 21 are not important nearly in all the experiments, which shows their lack of importance for the classification as a whole. Attributes 9 and 22, on the contrary, are important nearly in every experiment.

Table 4. Coefficients used in the algorithm

Name	*Value*	*Description*
size	50/200	Number of classifiers
length	57	Message size
mutProb	0,02	Mutation probability of 1 gene
crossProb	1	Crossover probability
cbid	0.1	Bid coefficient
bid1	0.25	Coefficient required to calculate the specificity influence on the bid
bid2	0.125	Coefficient required to calculate the specificity influence on the bid
bidtax	0	Bid tax for the matched but non-active classifiers
lifetax	0.,00005	Tax for each life iteration for all classifiers
nselect	10	Number of pairs of individuals that have to be replaced at the present stage of genetic algorithm
crowdingfactor	3	Number of sub-populations for selecting the most suitable classifier out of the weakest for its replacement
crowdingsubpop	10	Sub-population size for selecting the weakest individual
reward	1	Reward for correct answer
badreward	0	Penalty for wrong answer
gaFlag	2000	Genetic algorithm application period
wildcardsProb	0.93	Percentage of symbols "#"
u	400000	Number of iterations

Developed software description

The software program is written in C++. As a basis, the software program suggested in [2] specially for the simplified classifier systems as applied to multiplexer task, was employed. The program is written in the object-oriented manner. The results of program execution are recorded in two report files: (1) plot.txt for constructing graphs of successful answers and (2) report.txt where the general protocol of the program is represented (classifier lists: general, matched, active; sum total, the minimal, the maximal, and the mean strength value at the given iteration, results of genetic algorithm application, etc.) Printed in bold program components are shown in Table 5.

Table 5. Developed classes of objects

Class name	*Function*
Mushrooms	A class that adjusts the system to the specific subject area. It represents steps of the algorithm's highest level;
Coder	Subject area encoder to the binary system. This makes further processing more convenient;
Reward	Collects general statistics of the system as well as assigns rewards
ClassifiersContainer	Classifier list in which general operations over the list are represented;
Classifier	Classifier abstraction. Implements an access to the internal structure of each classifier.

Conclusions

Evolutionary methods are a comparatively new approach and are employed to solve different tasks including search for certain optimal strategies, robot adaptation to certain environments and even control of fighter strategic fights. In the present paper special consideration is given to studying such possibilities of classifier systems as the implementation of classification and construction of classifier systems.

This study is only a first step in examining possibilities of classification on the basis of symbolic features with the help of genetic algorithms. The study resulted in constructing a classifier system able to classify about 95% of mushrooms for 200 initial classifiers with optimal parameter selection.

Further tasks are as follows:

1. Introducing noise effect to reduce the system's dependence on the learning sample and improving the classification quality of the population.
2. Testing the derived classification algorithms on other subject areas.

Acknowledgement

I appreciate useful comments on the manuscript given by Professor Arkady Borisov from Riga Technical University.

References

1. S.A. Aivazian, V.S. Mkhitarian, Applied statistics and essentials of econometrics, Moscow Book-Publishing Association UNUTY, 1998. – 1022 p.,
2. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Redwood City, 1989.
3. L.B. Booker, D.E. Goldberg and J.H. Holland, Classifier Systems and Genetic Algorithms, Artifitial Intelligence vol. 40, 1989, pp. 235-282.
4. Bart de Boer, Classifiers Systems: A useful approach to machine learning? Master's Thesis, Leiden University, August 31, 1994.
5. Patrick M. Murphy, David W. Aha, Mushroom Database, UCI Repository of machine learning databases, University of California, Department (Internet: ics.uci.edu), 1980.
6. George F. Luger, William A. Stubblefield, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Addison Wesley Longman Inc., 1998. – 824 p.
7. Jackson P. Introduction to Expert Systems, Addison-Wesley, 1999.

Helen Afanasyeva,
Transport and Telecommunications Institute
1 Lomonosova Str., Riga LV-1019, Latvia,
jelena_riga@yahoo.com

Helen Afanasjeva
Genetics-based machine learning systems for classification task

This paper examines the possibility of using evolutionary learning methods for classification. Great attention is paid to studying special features of credit assignment methods and genetic algorithm application to classification tasks. All the statements are based on the data obtained as a result of specific task solving: lepiota and agaricus family mushroom division into edible and poisonous. To implement the system, special software was developed in C++. As a result, a system was constructed able to classify approximately 90% of the mushroom varieties suggested. After a

slight modification of the algorithm and optimisation of the parameters, the system was able to produce about 95% of correct answers.

Елена Афанасьева

Системы обучения, основанные на генетических алгоритмах для задач классификации

В данной работе исследуется применение эволюционных методов обучения для классификации. Большое внимание уделяется особенностям методов распределения кредитов и применению генетических алгоритмов применительно к задачам классификации. Все утверждения базируются на данных, полученных в процессе решения конкретной задачи –разделения грибов семейств lepiota и agaricus на съедобные и ядовитые. Для реализации системы было разработано специальное программное обеспечение на языке С++. В результате была построена система, способная распознавать примерно 90% предложенных разновидностей грибов. После некоторой модификации алгоритма и оптимизации многочисленных параметров система выдаёт около 95% правильных ответов.