# Some Possibilities of Improving the CORA Classification Algorithm

Eriks Tipans and Prof. Arkady Borisov

Riga Technical University, Kalku 1, LV-1658, Riga, Latvia
eriks@ibm.cs.ru.lv

**Abstract.** This paper examines some possibilities of improving the CORA classification algorithm developed by M. Bongard in sixties. The algorithm is based on finding *features* of objects one needs to classify. The theoretical part of this study explains two main shortcomings of the CORA classification algorithm: (1) the algorithm rejects features that at least once appear in the opposite class and thus loses potentially valuable information; and (2) the algorithm has extremely large learning time that can be due to the "combinatorial explosion". The study suggests two methods to overcome these difficulties and to improve the algorithm: (1) *the method of relatively good features* and (2) *the method of sequential covering*. The experimental results demonstrate that both the methods suggested ensure a sufficient algorithm performance's improvement as compared to the classic Bongard algorithm implementation. Best results are achieved when both methods are combined and sufficiently improve the classification precision and reduce learning time.

## 1 Introduction

The main task of this paper is to study the CORA classification algorithm, in more detail. Besides, it is necessary to try to explain possible shortcomings of the algorithm so as to improve its execution. This algorithm seems very promising as its implementation is rather simple but decision rules (concept generalizations) obtained as a result of its execution are easy to interpret for a human being in contrast to artificial neural networks.

## 2 Generalization by Features in Bongard's CORA Classification Algorithm

Let us consider the Bongard method, or CORA algorithm. The essence of the algorithm is as follows. The solution of the classification task is based on two procedures: learning and examination which are repeated in the course of generalization as many times as needed. In the course of learning a set of training examples is used that contains examples belonging to a concept to be generalized and also counter examples that do not belong to the concept in question. Learning results

in the development of a decision rule, which is able to indicate whether a particular example belongs to the generalized concept in question, or not. After a decision rule has been developed according to the training set, a stage of examination starts. At this stage, the system is presented a set of examples which certainly contains other examples that differ from those in the training set. If the decision rule demonstrates an acceptable quality of examination set classification in the course of examination, then the procedure of learning is considered to be finished. Otherwise, the decision rule is updated additionally processing a new complemented set of training examples or the same set.

The CORA-3 algorithm in Bongard's realization [1] is as follows.

**Learning Algorithm**

1. Three columns are taken from a matrix of training examples. It is then checked if the columns contain a combination of zeros and ones that is no less than K times met in vectors of one class and is never met in vectors of the other class. If a combination of this kind has been found, then the combination of zeros and ones available in these three columns will be called a *feature*. The examples that contain this combination of zeros and ones in the corresponding columns will be called examples of which this feature is characteristic. Features A and B are assumed to be equivalent if they are characteristic of the same examples. Feature A is stronger than feature B if it is characteristic of all the examples to which feature B is characteristic and also to some other (or at least to one) examples. It is clear that each feature characterizes objects of only one class.

2. The feature found is compared to all those found before. If this feature turns out to be stronger than any other feature found before, it is written in the place of the weakest feature. If some feature found before is stronger or equivalent to the new one, then the new feature is rejected. If none of these relationships holds, the new feature is written in a new place if there is such.

3. Similarly all column combinations taken by three are checked.

**Additional Learning Algorithm**

Let us call the number of features characteristic to the given example *the number of votes*, which this example has collected. Those features that in the end of Step 3 have collected no more than n votes will be called lagging features. The number n can be set different for each class depending on the situation.

4. Additional learning is a stage of learning the transition to which occurs not automatically but by a command. When additional learning is performed, K can be changed. Features characterizing no less than K lagging features will be called additional features.

5. Selection of additional features is performed according to Steps 1-3.

6. If the features selected have filled up the memory (64 features for each class in Bongard's realization), then learning is finished. Otherwise, the operator may return to Step 4 ( with new K and n) or stop at the result achieved.

7. The result of learning can be stored in the external memory.

**Recognition Algorithm**

8. Before recognition, either training has to be performed or learning results have to be read from the external memory (see Step 7).

9. When recognition is executed, it is checked how many features voting in favor of one or other class characterize the example being recognized. The example is classified in accordance with the majority of votes.

## 3 Cross Validation Method to Determine the Accuracy of Classification

The problem is stated as follows. How to test the accuracy of various classification methods so as the testing would be objective, the results would be accurate and would not be based on only two attempts and one could objectively compare different methods with each other? How large a training set and testing set should be? How many times one should perform training and testing to make the result statistically validated? The *cross validation* method seems to be most suitable for this purposes. This method is considered among researchers [5] as sufficiently objective to evaluate the accuracy of classification. The essence of the method is as follows. The whole set of examples is intermixed with the help of random number generator and separated into $k$ nearly equal parts. Each of these $k$ parts is in sequence employed as testing examples provided the classification algorithm learning was performed beforehand using those examples that are not contained in current testing example part.

According to various authors [5], it is most optimal to employ $k$=10. At this, an optimal relationship between the number of examples of a training set and of a testing set is ensured. In this case the method is called "10-fold cross validation". 90% of examples are used as a training example set, 10% of examples are employed as a testing example set and the cross validation process as a whole consists of 10 iterations. At each iteration, different 10% are taken for a testing set. An average error over those 10 iterations is called classification error.

## 4 Introduction of the Method of Relatively Good Features

In Bongard' s realization, a conjunction is sought that can no less than K times be met in vectors of one class and is never met in vectors of the other class. Due to this, the algorithm rejects those features that at least once appear in another class and thus loses potentially valuable information. It may be worth softening this strict requirement. For example, Gelfand and Guberman (1976) in the paper [2] dealt with the determination of earthquake epicenters describe an approach similar to CORA in which the conjunction is relatively frequently met in one class and is relatively infrequently met in the other class. An approach of this kind should be effective when classifying new objects, the error on the training set vectors, however, would increase. Let us call it a *method of relatively good features*. The method can be introduced as follows. In addition to an ordinary threshold $K$, which determines the minimal number of examples that the conjunction found must describe, threshold $Ko$ is set that

determines the maximal number of examples of the opposite class, which may be described by the conjunction, found. It is clear that $Ko << K$ should hold. A real employable value of $Ko$ has to be found experimentally.

## 5 Introduction of the Method of Sequential Covering

Practical experiments with the CORA classification algorithm show its essential shortcoming. The algorithm has an extremely large learning time especially if the number of attributes and the length of conjunction is large that can be due to the "combinatorial explosion". The reason for this is the following. The CORA algorithm searches for attribute combinations. If the conjunction length is $L$ but the number of attributes is $N$, then the algorithm has to examine

$$\frac{N!*2^L}{L!*(N-L)!} \qquad (1)$$

possible conjunctions. Moreover, it has also to examine possible conjunctions each of which is searched for over all training set examples. It is possible to reduce the number of operations indirectly, i.e. the above formula is valid, however, the number of training set examples in which new conjunctions has to be sought can be reduced as well as the number of conjunctions found and, hence, the number of conjunctions to be analyzed. This can be achieved by introducing the so-called *sequential covering method* described in [4] where its application to IF-THEN rule generation is presented. The essence of this method is as follows. While analyzing a training example set, the algorithm learns one rule. It then excludes from the training example set all the examples this rule describes. After that, when examining the training set examples remained, the algorithm learns the second rule and again excludes from the training example set all those examples, which are described by this rule. The process continues in iteration until all the examples are excluded from the training example set and the rules that cover all the examples are learned.

This method slightly modified can be adapted to the CORA algorithm. The original sequential covering method excludes examples provided they are described only once. This does not fit the CORA algorithm since it contradicts to the principles of additional learning according to which the examples that have collected less than $n$ votes are considered as lagging and therefore additional learning has to be performed to enable them to collect the votes missing. Due to this the authors of this paper propose to combine these principles and to exclude examples out of the training set when they have collected a specified number of votes, $n$. This will allow one to reject the stage of additional learning at all by transferring its functions to the sequential covering method thus gaining extra time saving. In this way the sequential covering method is "fused" with the stage of additional learning and is naturally involved in the ordinary learning process of the CORA algorithm thus simplifying this algorithm and at the same time improving its quality and operation speed.

## 6 Description of Learning and Testing Sets

The experiments have been performed on the basis of example sets taken from the UCI Repository of Machine Learning Databases and Domain Theories available in the Internet [6], [7].

The sets of examples located at this address are well known and cited by many authors. Each of the example sets displays real case studies. The present study employs example sets from two knowledge areas: (1) USA Congress voting results and (2) Tic-tac-toe game. Below one can find a detailed information on each of the sets.

1. USA Congress voting results. The file employed was *house-votes-84.data.* In the congress voting database, each voting case has 16 binary attributes that represent the answers to 16 questions and a class that shows which party has won in voting (0 – democrats or 1 – republicans). 232 examples were used. In 124 cases of them democrats have won (class 0) but in 108 cases republicans have won (class 1). In what follows this database will be called *Votes*.

2. Tic-tac-toe game. The file employed was *tic-tac-toe.data.* The database includes all the 958 positions possible for this game. It is assumed that the crosses go first. If the crosses win, then an example belongs to class 1, if they loose or in the case of drawn play it belongs to class 0. To describe the state of each game square, binary combinations of two attributes were employed as follows: 01 - there is a cross in the square; 11 - there is a naught in the square; 00 - the square is empty. Hence, after the encoding each database example was described by 18 attributes. In what follows this database will be called Tic-tac-toe.

## 7 Experiments Performed with the CORA Classification Algorithm

In all further experiments, the author's own software program CORA was used to implement the CORA classification algorithm. To check the classification accuracy in all the experiments, the cross-validation method was used and also the learning time spent was measured.

### 7.1 Determination of Optimal Learning Parameters

The purpose of this experiment is to test the efficiency of the algorithm at different conjunction lengths and different minimal values of threshold $K$ , to determine which conjunctions should be given priority when making selection of conjunctions: those of shorter length or those of larger length as well as to find optimal learning parameters for the given example set.

During the experiment different conjunction lengths were examined, first from 1 through 2 and finally from 1 through 5. The values of threshold $K$ from 10 through 50 with step 10 have been tested. It was also examined which conjunctions should be given priority when making selection of conjunctions: those of shorter length or those of larger length (for example, 1-5 means that priority is given to shorter disjunctions

whereas 5-1 means that priority is given to longer disjunctions). In total, 40 experiments have been performed with the example set Votes at different values of the above parameters (see Table 1).

**Table 1.** Determination of optimal parameters. Example set: Votes

| No. | Conjunction length | Minimal threshold K | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|
| 1. | 1-2 | 10 | 10.870 | 15.4 | 0.0 |
| 2. | 1-2 | 20 | 10.870 | 12.2 | 0.1 |
| 3. | 1-2 | 30 | 10.870 | 10.1 | 0.0 |
| 4. | 1-2 | 40 | 10.870 | 8.9 | 0.1 |
| 5. | 1-2 | 50 | 10.870 | 5.1 | 0.0 |
| 6. | 1-3 | 10 | 5.652 | 45.1 | 1.9 |
| 7. | 1-3 | 20 | 6.087 | 26.6 | 1.1 |
| 8. | 1-3 | 30 | 6.087 | 19.8 | 0.7 |
| 9. | 1-3 | 40 | 7.391 | 14.5 | 0.5 |
| 10. | 1-3 | 50 | 10.000 | 7.6 | 0.4 |
| 11. | 1-4 | 10 | 4.348 | 82.3 | 21.7 |
| 12. | 1-4 | 20 | 4.783 | 41.6 | 7.9 |
| 13. | 1-4 | 30 | 5.217 | 31.6 | 4.2 |
| 14. | 1-4 | 40 | 5.217 | 20.1 | 2.8 |
| 15. | 1-4 | 50 | 8.696 | 8.1 | 2.1 |
| 16. | 1-5 | 10 | 4.348 | 103.6 | 104.2 |
| 17. | 1-5 | 20 | 4.783 | 52.5 | 33.8 |
| 18. | 1-5 | 30 | 5.217 | 39.9 | 14.8 |
| 19. | 1-5 | 40 | 5.217 | 24.3 | 12.4 |
| 20. | 1-5 | 50 | 8.696 | 8.3 | 10.6 |
| 21. | 2-1 | 10 | 10.870 | 16.3 | 0.1 |
| 22. | 2-1 | 20 | 10.870 | 13.1 | 0.1 |
| 23. | 2-1 | 30 | 10.870 | 10.9 | 0.0 |
| 24. | 2-1 | 40 | 10.870 | 9.6 | 0.0 |
| 25. | 2-1 | 50 | 10.870 | 5.5 | 0.0 |
| 26. | 3-1 | 10 | 5.652 | 54.0 | 7.8 |
| 27. | 3-1 | 20 | 6.087 | 33.4 | 3.7 |
| 28. | 3-1 | 30 | 6.087 | 24.7 | 1.7 |
| 29. | 3-1 | 40 | 7.391 | 18.0 | 0.8 |
| 30. | 3-1 | 50 | 10.000 | 10.1 | 0.4 |
| 31. | 4-1 | 10 | 4.348 | 90.4 | 131.9 |
| 32. | 4-1 | 20 | 4.348 | 47.5 | 40.1 |
| 33. | 4-1 | 30 | 4.783 | 35.5 | 15.7 |
| 34. | 4-1 | 40 | 4.783 | 24.2 | 5.5 |
| 35. | 4-1 | 50 | 8.696 | 11.3 | 3.4 |
| 36. | 5-1 | 10 | 4.348 | 110.3 | 861.8 |
| 37. | 5-1 | 20 | 4.783 | 57.5 | 235.5 |
| 38. | 5-1 | 30 | 3.913 | 43.3 | 78.8 |

| | | | | | |
|---|---|---|---|---|---|
| 39. | 5-1 | 40 | 4.783 | 27.1 | 25.9 |
| 40. | 5-1 | 50 | 8.696 | 12.4 | 15.7 |

Each row of this table represents the results obtained at 10 training and testing iterations performing cross-validation. The shaded row denotes a variant (No.34) with optimal learning parameters: a low error level is achieved within short learning time. The same experiments have been performed with the Tic-tac-toe example set (see Table 2).

**Table 2.** Determination of optimal learning parameters. Example set: Tic-tac-toe

| No. | Conjunction length | Minimal threshold K | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|
| 1. | 1-2 | 10 | 100.000 | 0.0 | 0.2 |
| 2. | 1-2 | 20 | 100.000 | 0.0 | 0.2 |
| 3. | 1-2 | 30 | 100.000 | 0.0 | 0.2 |
| 4. | 1-2 | 40 | 100.000 | 0.0 | 0.2 |
| 5. | 1-2 | 50 | 100.000 | 0.0 | 0.2 |
| 6. | 1-3 | 10 | 26.316 | 31.1 | 2.3 |
| 7. | 1-3 | 20 | 29.579 | 14.2 | 2.1 |
| 8. | 1-3 | 30 | 32.211 | 11.4 | 2.0 |
| 9. | 1-3 | 40 | 50.421 | 4.9 | 2.0 |
| 10. | 1-3 | 50 | 61.263 | 2.1 | 2.0 |
| 11. | 1-4 | 10 | 9.474 | 351.5 | 133.9 |
| 12. | 1-4 | 20 | 7.053 | 175.9 | 37.7 |
| 13. | 1-4 | 30 | 9.789 | 120.5 | 25.2 |
| 14. | 1-4 | 40 | 32.105 | 44.5 | 17.9 |
| 15. | 1-4 | 50 | 58.211 | 5.6 | 16.0 |
| 16. | 1-5 | 10 | 10.947 | 1150.1 | 4337.3 |
| 17. | 1-5 | 20 | 4.316 | 477.5 | 780.1 |
| 18. | 1-5 | 30 | 6.316 | 303.7 | 284.1 |
| 19. | 1-5 | 40 | 27.895 | 104.1 | 123.9 |
| 20. | 1-5 | 50 | 57.789 | 10.0 | 95.0 |
| 21. | 2-1 | 10 | 100.000 | 0.0 | 0.2 |
| 22. | 2-1 | 20 | 100.000 | 0.0 | 0.2 |
| 23. | 2-1 | 30 | 100.000 | 0.0 | 0.2 |
| 24. | 2-1 | 40 | 100.000 | 0.0 | 0.2 |
| 25. | 2-1 | 50 | 100.000 | 0.0 | 0.2 |
| 26. | 3-1 | 10 | 26.316 | 31.1 | 2.3 |
| 27. | 3-1 | 20 | 29.579 | 14.2 | 2.1 |
| 28. | 3-1 | 30 | 32.211 | 11.4 | 2.1 |
| 29. | 3-1 | 40 | 50.421 | 4.9 | 2.1 |
| 30. | 3-1 | 50 | 61.263 | 2.1 | 2.0 |
| 31. | 4-1 | 10 | 9.474 | 351.5 | 159.9 |
| 32. | 4-1 | 20 | 7.053 | 178.6 | 45.4 |
| 33. | 4-1 | 30 | 9.789 | 125.4 | 30.0 |

| | | | | | |
|---|---|---|---|---|---|
| 34. | 4-1 | 40 | 32.105 | 48.5 | 20.5 |
| 35. | 4-1 | 50 | 58.211 | 12.4 | 17.2 |
| 36. | 5-1 | 10 | 10.947 | 1150.1 | 5612.4 |
| 37. | 5-1 | 20 | 4.316 | 477.7 | 1246.4 |
| 38. | 5-1 | 30 | 6.316 | 303.9 | 488.6 |
| 39. | 5-1 | 40 | 27.895 | 106.6 | 207.9 |
| 40. | 5-1 | 50 | 57.789 | 22.6 | 141.0 |

For this example set, variant No.13 is considered as optimal.

### 7.2 Experiments with Additional Learning

After optimal parameters for initial learning have been determined, experiments with additional learning can be performed. In this experiment series a threshold for additional learning is varied so as to find its optimal value for the both example sets (see Table 3 and 4).

**Table 3.** Experiments with additional learning. Example set: Votes

| No. | Conjunction length | Minimal threshold K | Threshold for additional learning | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|
| 1. | 4-1 | 40 | 35 | 4.348 | 30.6 | 8.2 |
| 2. | 4-1 | 40 | 30 | 4.348 | 35.8 | 8.9 |
| 3. | 4-1 | 40 | 25 | 4.348 | 39.0 | 10.9 |
| 4. | 4-1 | 40 | 20 | 4.348 | 46.0 | 12.7 |
| 5. | 4-1 | 40 | 15 | 4.348 | 61.8 | 17.8 |
| 6. | 4-1 | 40 | 10 | 3.913 | 88.0 | 31.9 |
| 7. | 4-1 | 40 | 5 | 4.348 | 126.1 | 83.5 |

**Table 4.** Experiments with additional learning. Example set: Tic-tac-toe

| No. | Conjunction length | Minimal threshold K | Threshold for additional learning | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|
| 1. | 1-4 | 30 | 25 | 6.421 | 149.7 | 47.7 |
| 2. | 1-4 | 30 | 20 | 7.053 | 175.9 | 53.3 |
| 3. | 1-4 | 30 | 15 | 7.684 | 246.1 | 81.6 |

For the *Votes* set, the additional learning threshold 10 turned out to be optimal but for the Tic-tac-toe set it was 25.

# 8 Experiments with the Modifications of the CORA Algorithm

The task of this experiment series is to find out whether the introduction of modifications of the standard CORA algorithm was useful.

## 8.1 Experiments with the Relatively Good Features Method

These experiments were performed to test the efficiency of the relatively good features method. In the experiments the maximal number of examples of the opposite class, Ko, which each conjunction found can describe was varied (see Table 5 and 6).

**Table 5.** Experiments with the relatively good features method. Example set: Votes

| No. | Conjunction length | Minimal threshold K | Maximal Ko | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|
| 1. | 4-1 | 40 | 1 | 3.043 | 29.8 | 13.6 |
| 2. | 4-1 | 40 | 2 | 4.783 | 43.1 | 21.3 |
| 3. | 4-1 | 40 | 3 | 6.087 | 42.4 | 31.7 |

**Table 6**. Experiments with the relatively good features method. Example set: Tic-tac-toe

| No. | Conjunction length | Minimal threshold K | Maximal Ko | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|
| 1. | 1-4 | 30 | 1 | 11.185 | 208.4 | 50.4 |
| 2. | 1-4 | 30 | 2 | 9.895 | 227.3 | 69.7 |
| 3. | 1-4 | 30 | 3 | 9.158 | 236.7 | 86.3 |
| 4. | 1-4 | 30 | 4 | 12.737 | 281.3 | 91.8 |
| 5. | 1-4 | 30 | 5 | 21.263 | 365.7 | 132.7 |

It can be seen that for the Votes training example set *Ko*=1 turned out to be optimal but for the Tic-tac-toe set it was equal to 3.

## 8.2 Experiments with the Sequential Covering Method

Let us perform experiments to test the efficiency of the sequential covering method. In these experiments the number of votes, *n*, at which examples are excluded out of a training set will be changed (see Table 7 and 8). The shaded rows represent the three best results in each set.

**Table 7.** Experiments with the sequential covering method. Example set: Votes

| No. | Conjunction length | Minimal threshold K | Number of votes *n* | Errors(%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|
| 1. | 1-4 | 20 | 20 | 4.783 | 41.3 | 7.5 |
| 2. | 1-4 | 20 | 15 | 4.783 | 39.9 | 6.6 |
| 3. | 1-4 | 20 | 10 | 4.348 | 33.9 | 4.2 |
| 4. | 1-4 | 20 | 5 | 3.913 | 20.8 | 1.4 |
| 5. | 1-4 | 15 | 20 | 4.348 | 54.9 | 11.4 |
| 6. | 1-4 | 15 | 15 | 4.348 | 48.2 | 9.3 |
| 7. | 1-4 | 15 | 10 | 3.913 | 43.5 | 4.8 |
| 8. | 1-4 | 15 | 5 | 5.217 | 25.2 | 1.4 |
| 9. | 1-4 | 10 | 20 | 3.913 | 73.6 | 18.4 |
| 10. | 1-4 | 10 | 15 | 3.478 | 64.3 | 14.3 |
| 11. | 1-4 | 10 | 10 | 3.913 | 52.2 | 6.5 |
| 12. | 1-4 | 10 | 5 | 5.217 | 29.3 | 1.6 |
| 13. | 1-4 | 5 | 20 | 3.913 | 108.4 | 37.9 |
| 14. | 1-4 | 5 | 15 | 3.913 | 85.1 | 27.5 |
| 15. | 1-4 | 5 | 10 | 7.826 | 55.1 | 8.1 |
| 16. | 1-4 | 5 | 5 | 11.739 | 22.2 | 2.2 |

**Table 8.** Experiments with the sequential covering method. Example set: Tic-tac-toe

| No. | Conjunction length | Minimal threshold K | Number of votes *n* | Percentage of errors | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|
| 1. | 1-4 | 20 | 20 | 6.737 | 167.1 | 34.2 |
| 2. | 1-4 | 20 | 15 | 6.842 | 157.6 | 31.7 |
| 3. | 1-4 | 20 | 10 | 6.421 | 131.8 | 26.1 |
| 4. | 1-4 | 20 | 5 | 6.842 | 83.5 | 18.7 |
| 5. | 1-4 | 15 | 20 | 7.684 | 210.8 | 45.9 |
| 6. | 1-4 | 15 | 15 | 7.684 | 197.8 | 40.8 |
| 7. | 1-4 | 15 | 10 | 6.737 | 165.8 | 31.4 |
| 8. | 1-4 | 15 | 5 | 6.211 | 102.9 | 20.3 |
| 9. | 1-4 | 10 | 20 | 9.053 | 291.1 | 78.0 |
| 10. | 1-4 | 10 | 15 | 8.842 | 255.1 | 61.8 |
| 11. | 1-4 | 10 | 10 | 7.789 | 214.8 | 43.9 |
| 12. | 1-4 | 10 | 5 | 5.684 | 131.3 | 24.6 |
| 13. | 1-4 | 5 | 20 | 9.684 | 422.4 | 154.0 |
| 14. | 1-4 | 5 | 15 | 9.053 | 371.3 | 114.1 |
| 15. | 1-4 | 5 | 10 | 7.579 | 273.2 | 74.8 |
| 16. | 1-4 | 5 | 5 | 6.526 | 165.7 | 35.2 |

### 8.3 Experiments with a Combination of Both Methods

The task of these experiments is to test the efficiency of a simultaneous introduction of the method of relatively good features and the sequential covering method. In these experiments the maximal number of examples of the opposite class, Ko, which each conjunction found can describe and the number of votes, *n*, at which examples are excluded out of a training set were varied (see Table 9 and 10).

**Table 9.** Experiments with a combination of both methods. Example set: Votes

| No. | Conjunction length | Minimal threshold K | Number of votes *n* | Maximal Ko | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|---|
| 1. | 1-4 | 20 | 5 | 1 | 4.783 | 18.0 | 0.9 |
| 2. | 1-4 | 20 | 5 | 2 | 4.783 | 16.7 | 0.8 |
| 3. | 1-4 | 20 | 5 | 3 | 3.478 | 15.7 | 0.7 |
| 4. | 1-4 | 20 | 5 | 4 | 4.783 | 13.8 | 0.7 |
| 5. | 1-4 | 20 | 5 | 5 | 10.000 | 11.2 | 0.7 |
| 6. | 1-4 | 15 | 10 | 1 | 3.478 | 32.6 | 1.1 |
| 7. | 1-4 | 15 | 10 | 2 | 4.348 | 28.9 | 0.9 |
| 8. | 1-4 | 15 | 10 | 3 | 4.348 | 26.6 | 0.8 |
| 9. | 1-4 | 15 | 10 | 4 | 3.043 | 20.6 | 0.8 |
| 10. | 1-4 | 15 | 10 | 5 | 7.391 | 16.6 | 0.7 |
| 11. | 1-4 | 10 | 15 | 1 | 3.913 | 53.3 | 1.7 |
| 12. | 1-4 | 10 | 15 | 2 | 21.739 | 21.9 | 1.2 |
| 13. | 1-4 | 10 | 15 | 3 | 11.304 | 28.0 | 0.9 |
| 14. | 1-4 | 10 | 15 | 4 | 12.174 | 23.3 | 0.9 |
| 15. | 1-4 | 10 | 15 | 5 | 17.391 | 25.5 | 0.8 |

**Table 10.** Experiments with a combination of both methods. Example set: Tic-tac-toe

| No. | Conjunction length | Minimal threshold K | Number of votes *n* | Maximal Ko | Error (%) | Average number of conjunctions | Average time (s) |
|---|---|---|---|---|---|---|---|
| 1. | 1-4 | 20 | 10 | 1 | 6.737 | 147.5 | 26.8 |
| 2. | 1-4 | 20 | 10 | 2 | 5.263 | 133.8 | 25.1 |
| 3. | 1-4 | 20 | 10 | 3 | 5.474 | 147.5 | 24.5 |
| 4. | 1-4 | 20 | 10 | 4 | 6.632 | 164.9 | 25.6 |
| 5. | 1-4 | 20 | 10 | 5 | 8.632 | 177.5 | 24.0 |
| 6. | 1-4 | 15 | 5 | 1 | 7.158 | 110.1 | 17.7 |
| 7. | 1-4 | 15 | 5 | 2 | 5.474 | 110.8 | 16.4 |
| 8. | 1-4 | 15 | 5 | 3 | 10.211 | 103.0 | 15.0 |
| 9. | 1-4 | 15 | 5 | 4 | 17.474 | 91.6 | 13.3 |
| 10. | 1-4 | 15 | 5 | 5 | 21.368 | 80.5 | 11.8 |
| 11. | 1-4 | 10 | 5 | 1 | 6.842 | 137.5 | 19.9 |
| 12. | 1-4 | 10 | 5 | 2 | 12.000 | 110.3 | 18.7 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13. | 1-4 | 10 | 5 | 3 | 23.579 | 80.8 | 17.1 |
| 14. | 1-4 | 10 | 5 | 4 | 31.789 | 54.9 | 14.5 |
| 15. | 1-4 | 10 | 5 | 5 | 39.368 | 36.5 | 12.9 |

The experimental results show that better success was achieved when both CORA algorithm modifications suggested were combined. As a result of this combination, both the classification error and learning time were reduced.

Figures 1 and 2 present a summary of the experimental results.

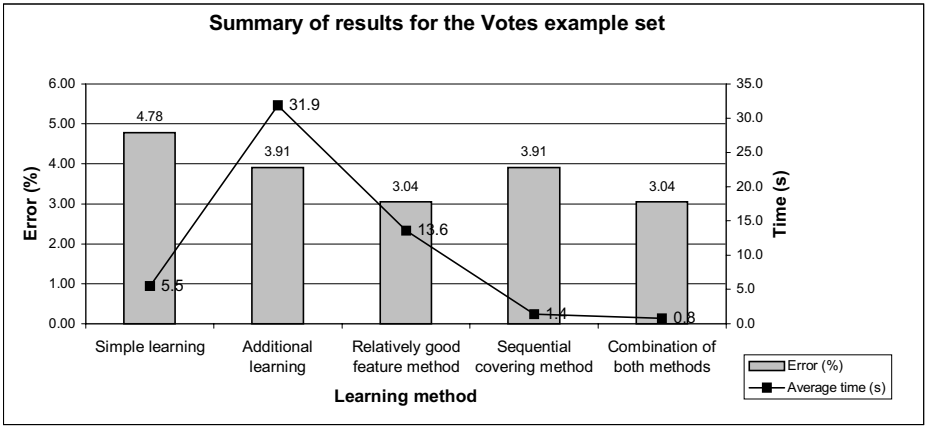


**Fig. 1.** Summary of experimental results for the Votes example set.

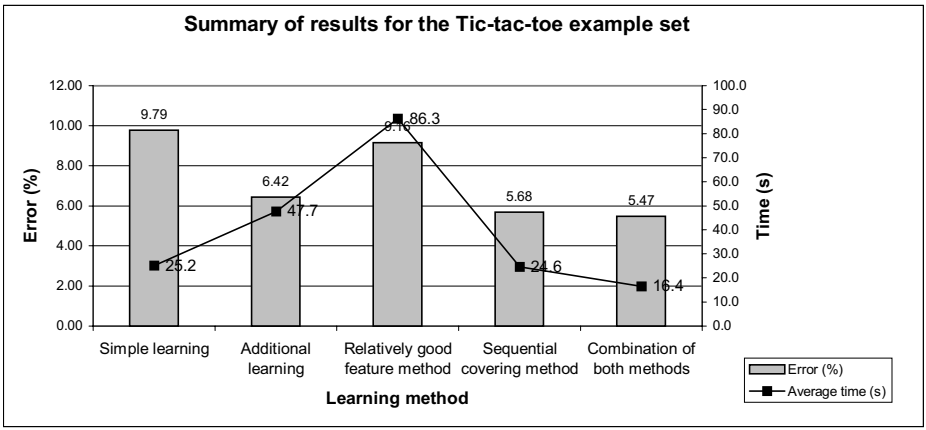


**Fig. 2.** Summary of experimental results for the Tic-tac-toe example set.

## 9 Analysis of the Results and Conclusions

The theoretical part of this study explains two main shortcomings of the CORA classification algorithm: (1) the algorithm rejects features that at least once appear in the opposite class and thus loses potentially valuable information; and (2) the algorithm has extremely large learning time that can be due to the "combinatorial explosion". The study suggests two methods to overcome these difficulties and to improve the algorithm: the method of relatively good features and the sequential covering method.

The experimental part of the study presents the experiments performed to examine the usefulness of those methods introduction and also checks their efficiency. The experiments performed made use of example sets from two knowledge areas: (1) USA Congress voting results forecasting (232 examples), and (2) Tic-tac-toe game (958 examples). The experimental results demonstrate that both the methods suggested ensure a sufficient algorithm performance's improvement as compared to the classic Bongard algorithm realization. Best results are achieved when both methods are combined and sufficiently improve the classification precision and reduce learning time.

## References


1. Bongard, M.: Problem of Recognition. Nauka Publishers, Moscow (1967) (In Russian)
2. Gelfand, M., Guberman, S.: Pattern Recognition Applied to Earthquake Epicenters in California. Physics of the Earth and Planetary Interiors 11 (1976) 227 - 283
3. Gladun, V.: Planning Decisions. Naukova Dumka, Kiev (1987) (In Russian)
4. Mitchell, T. M.: Machine Learning. The Mc Graw - Hill Companies Inc. (1997)
5. Weiss, S., Kapouleas, I.: An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods. Shawlik, I. W., Dietterich, T. G. (editors): Readings in Machine Learning. Morgan Kaufmann Publishers, San Mateo, California (1990) 177 - 183
6. URL: ftp://ftp.ics.uci.edu/pub/machine-learning-databases
7. URL: http://www.ics.uci.edu/~mlearn/MLRepository.html