Applying an instance selection method to an evolutionary neural classifier design

Dmitrii Khritonenko, Vladimir Stanovov, Eugene Semenkin

Siberian State Aerospace University, Krasnoyarsk, Russian Federation, 660037, Krasnoyarsky Rabochy Av., 31

E-mail: HDmitry.91@mail.ru

Abstract. In this paper the application of an instance selection algorithm to the design of a neural classifier is considered. A number of existing instance selection methods are presented. A new wrapper-method, whose main difference compared to other approaches is an iterative procedure for selecting training subsets from the dataset, is described. The approach is based on using training subsample selection probabilities for every instance. The value of these probabilities depends on the classification success for each measurement. An evolutionary algorithm for the design of a neural classifier is presented, which was used to test the efficiency of the presented approach. The described approach has been implemented and tested on a set of classification problems. The testing has shown that the presented algorithm allows the computational complexity to be decreased and the quality of the obtained classifiers to be increased. Compared to analogues found in scientific literature, it was shown that the presented algorithm is an effective tool for classification problem solving.

1. Introduction

Modern developments in information technologies allow huge amounts of data from different areas of human activity to be saved. This can include medical diagnostic data, descriptions of technological processes, the tracking and predicting of economic indexes, text and image recognition, among other types. Using this data allows us to discover unknown and non-trivial dependencies of the real world, and solve classification, clusterisation, prediction and regression problems. The number of measurements in these problems may reach hundreds of thousands and millions, which impose limitations on the set of intelligent data analysis technologies. The instances present in the data could be noisy, may duplicate each other, or contain a lot of useless information. The training instance selection technologies are capable of partially solving these problems. The approach described in this paper refers to wrapper-class and is capable of decreasing the computational complexity of training algorithms, partially eliminating the effect of over-fitting for intelligent data analysis, and increasing the accuracy of the designed classifiers by using an iterative procedure for redistributing instance selection probabilities. As a base model, on which the efficiency of the proposed algorithm was tested, a distributed program system of generating automated neural networks with evolutionary algorithms was chosen [1]. The choice is justified by the fact that the program system has shown its efficiency on a set of test and real problems [2, 3], however, its application in actual problems is not possible in an adequate length of time. The development of a simple and efficient instance selection procedure is a
topical issue today, and it will allow the effectiveness of using neural technologies in data analysis problems to be increased.

The rest of the paper is organized as follows: Section 2 contains a brief description of the instance selection method used. In Section 3 we describe a neural classifier design with the use of evolutionary algorithms. In Section 4 we present the results obtained with the developed methods, compare them to analogues and provide a short discussion. The last section is devoted to conclusions.

2. Instance selection method

Most of the existing instance selection methods are focused not only on data reduction, but also on increasing classification quality [4, 5, 6, 7]. However, these methods usually create a subsample only once, i.e. the goal is to create a subsample, and not to build a high-quality classifier.

The proposed IS method should be used with classifiers which require lots of iterations during the learning process. The main idea that stimulated the development was to create a method, allowing not only the time required to train a classifier to be decreased, but also the classification accuracy to be increased. The method used here has already been used in [8, 9] to train evolutionary fuzzy classifiers, which is why we will give only a short description here.

The method consists of several steps. Firstly, a probability value $p_i$ and a counter $U_i$ are assigned to each instance in the training set $i = 1 \ldots N$, where $N$ is the number of instances. The probability values are equal and represent the chance that an instance $i$ would be selected into the training subset, i.e. $p_i = p_j, i, j = 1 \ldots N$. The counter values $U_i$ are set to 1. Secondly, the probability values are used to select the instances and start the training process on the subsample. The size of the subsample is fixed and it is calculated with respect to the training sample size (in per cent). Thirdly, the training process continues for a certain number of generations, called the adaptation period. Finally, after the end of the adaptation period, the best individual in the population is used to change the $U_i$ values: if a training example has been classified correctly, then $U_i = U_i + 1$, otherwise $U_i = 1$. $U_i$ values are changed only for instances which are present in the subsample. Before creating a new subsample, the probabilities are updated using the following equation:

$$p_i = \frac{1/U_i}{\sum_{j=1}^{N} 1/U_j}. \quad (1)$$

Increasing the counter leads to a lower probability being chosen, while low counter values force the algorithm to choose a certain instance more often.

This procedure makes the algorithm adaptive, following two main strategies: exploration and exploitation. Instances which have not been selected before, have a counter value of 1, meaning that they have a quite large probability of being chosen, implementing the exploration of new areas of the feature space. Instances which have been chosen several times and have received high counter values get lower probabilities, because they appear to be easier to classify, i.e. they are not so interesting for the training process. However, instances which have been classified incorrectly, get their counters refreshed to $U_i = 1$, making them look like undiscovered instances. This implements the exploitation strategy, i.e. the algorithm uses the information about the classification quality to select the instances of larger interest. At the same time, if one of the instances, which have been classified correctly many times before, gets misclassified with a new model, it gets a higher probability of being chosen in the following adaptation periods. This allows the algorithm to focus not only on instances that are hard to classify, but also to keep the other instances classified correctly as well.

As we are going to use an evolutionary approach with a set of individuals in a population, the best solution for the subsample might not be the best solution for the whole sample. However, the population may still contain a solution with high generalization ability, despite not being the best when checked on the subsample. That is why, after each adaptation period, every individual in the population is checked on the whole training sample, and the best solution for the training sample is remembered, and included in the subsample, as well as the best solution for the subsample.
3. Evolutionary designing of neural classifiers

The problem concerning the automated designing of artificial neural networks is not new, and there is more than one way of solving this problem. One of the approaches developed is the use of evolutionary algorithms [1]. Evolutionary algorithms are an optimization tool in arbitrary search spaces and they have proved their efficiency on a set of real problems. To solve the problem of the evolutionary design of neural classifiers, a genetic programming algorithm (GP) with several modifications is used in this work. The training is performed with different single-criterion unconstrained optimization methods (backpropagation, genetic algorithms, CMA-ES [10] and others).

To start the genetic programming algorithm, it is required to fill the terminal and functional set. As terminal set elements, different types of artificial neurons are used (8 types totally), attributes and sets of attributes of the problem solved, and some constants. The functional set consists of elements joining in one layer operator (plus sign), and operators responsible for layer interconnections (shown in arrows at Fig. 1).

![Figure 1. Example of ANN coding in GP.](image)

Where $N_k$ is a neuron with the $k$-th number, $In_i$ is an attribute of the problem with the number $i$. It should be mentioned that for this encoding, the insularity condition could be violated. That is why a special procedure for finding and eliminating the violations by changing the operators was used.

After determining the required sets, it becomes possible to automatically design neural networks. However, it is important to remember that the efficiency of all evolutionary algorithms depends on the set of parameters. Because of this, a modification called self-configuration was used [3]. It allows a set of parameters to be refused, the time required for the algorithm to work to be decreased, and the accuracy of the models to be increased. A more general scheme of adjusting the parameters of evolutionary algorithms is presented in [11]. The algorithm configurations are considered, as well as different numerical parameters. The approaches implemented today for single-criterion unconstrained optimization has shown positive results and will soon be transferred to the described algorithm. Together with self-configuration, the uniform crossover scheme has been added to the algorithm [12]. This procedure allows the accuracy of the models obtained to be increased, and it is an additional way of controlling the complexity of the received neural nets.

4. Testing results

To test the efficiency of the described instance selection method, a self-configuring evolutionary algorithm for the automated design of artificial neural networks (ANNEA) has been implemented in a program. A set of computational experiments has been performed on problems taken from the machine learning repositories KEEL [13] and UCI [14].

The following parameters were used for testing the algorithm presented in section 2: number of individuals: 100, number of generations: 10000, subsample size from 5 to 50 percent (with 5% step), adaptation period from 25 to 500 generations (with a 25 generation step). The program system, using
the described instance selection algorithm (ANNEA-IS) was compared to its basic version (ANNEA), as well as to analogues found in scientific literature.

Table 1. Numerical characteristics of problems.

| Dataset         | Number of instances | Number of attributes | Number of classes |
|-----------------|---------------------|----------------------|-------------------|
| Magic           | 19020               | 10                   | 2                 |
| Page-Blocks     | 5472                | 10                   | 5                 |
| Australian credit | 690                | 14                   | 2                 |
| German credit   | 1000                | 20                   | 2                 |
| Texture         | 5500                | 40                   | 11                |
| Twonorm         | 7400                | 20                   | 2                 |
| Segment         | 2310                | 19                   | 7                 |
| Ring            | 7400                | 20                   | 2                 |
| Penbased        | 10992               | 16                   | 10                |
| Satimage        | 6435                | 36                   | 6                 |

As an efficiency criterion, the classification error, averaged over all runs, has been used. A run means using stratified 10-fold cross-validation. The total number of runs is 10. In the table below, the testing results for the base algorithm, as well as its modified version, are presented.

Table 2. Testing results.

| Dataset         | ANNEA | ANNEA-IS |
|-----------------|-------|----------|
|                 | Training error, % | Test error, % | Time, min | Training error, % | Test error, % | Time, min |
| Magic           | 14.83 | 15.08 | 734 | 13.75 | **14.72** | 229 |
| Page-Blocks     | 3.72  | **4.23** | 180 | 3.96  | **4.23** | 51 |
| Australian credit | 8.71  | 10.99 | 67 | 10.07 | **9.97** | 19 |
| German credit   | 19.56 | 23.76 | 129 | 18.91 | **21.70** | 47 |
| Texture         | 6.93  | 8.05  | 805 | 5.90  | **7.27** | 146 |
| Twonorm         | 5.21  | 5.97  | 481 | 3.67  | **4.24** | 131 |
| Segment         | 5.89  | 6.14  | 312 | 4.69  | **5.10** | 81 |
| Ring            | 6.37  | 6.54  | 454 | 4.78  | **4.99** | 97 |
| Penbased        | 5.36  | 7.91  | 470 | 3.30  | **3.55** | 137 |
| Satimage        | 12.90 | 14.09 | 691 | 12.45 | **13.82** | 115 |

It should be noted that applying instance selection for the process of forming neural classifiers with evolutionary algorithms resulted in an increase in the classification accuracy and a decrease in the time required. For the worst case (Page-Blocks dataset), there is no statistical difference for the criteria used (in the sense of the Wilcoxon criterion). On average, the modelling accuracy increases by 14%, and the time required decreases by 1.75-5 times. Let us present the best parameters for the adaptation period and the sizes of the training subsamples for every problem used:

As can be seen from the table, the sizes of the subsamples do not reach the maximal size of 50%. This could be connected to the fact that for efficient learning the algorithm requires around 30% of the total size on average. Further increasing the subsample size definitely increases the model accuracy on the training sample, but leads to over-fitting, and, as a consequence, loss of generalization ability and test sample accuracy. Also it should be noted that for re-training on the new data, the algorithm requires a small adaptation period. A small adaptation period with a fixed number of generations
allows more training sample rotations to be made and a better description of the problematic areas by the training subset to be provided. This feature allows the algorithm to learn the dataset better.

Table 3. Best algorithm parameters.

| Dataset         | Subsample size, % | Adaptation period, generations |
|-----------------|-------------------|-------------------------------|
| Magic           | 35                | 175                           |
| Page-Blocks     | 25                | 200                           |
| Australian credit | 40              | 50                            |
| German credit   | 35                | 75                            |
| Texture         | 25                | 50                            |
| Twonorm         | 25                | 50                            |
| Segment         | 20                | 75                            |
| Ring            | 30                | 125                           |
| Penbased        | 25                | 75                            |
| Satimage        | 35                | 50                            |

The presented data is recommended for further usage with the instance selection algorithm when applied to other data analysis technologies and problems.

To compare the implemented program system with analogues, additional experiments were performed. To ensure objectivity in the comparison, the testing conditions were made maximally similar to each other (Table 4). In the case of the conditions being unknown, increased resources were used (Table 5).

Table 4. Comparison to other methods.

| Dataset         | ANNEA_IS | HEFCA_IS [9] | GP-Coach [15] | Parallel-Fuzzy GBML [16] | FARC-HD [17] |
|-----------------|-----------|--------------|---------------|--------------------------|--------------|
| Magic           | 14.72     | 15.08        | 20.18         | 14.89                    | 15.49        |
| Page-Blocks     | 4.23      | 3.25         | 8.77          | 3.62                     | 4.99         |
| Texture         | 7.27      | 4.45         | -             | 4.77                     | 7.11         |
| Twonorm         | 4.24      | 4.81         | 15.17         | 3.39                     | 4.72         |
| Segment         | 5.10      | 5.19         | 24.04         | 5.90                     | -            |
| Ring            | 4.99      | 5.08         | -             | 6.73                     | 5.92         |
| Penbased        | 3.55      | 3.81         | 17.80         | 3.07                     | 3.96         |
| Satimage        | 13.82     | 12.93        | 27.50         | 15.54                    | **12.68**    |

Table 5. Comparison to other methods on Australian and German credit.

| Algorithm name | Australian credit | German credit | Algorithm name | Australian credit | German credit |
|----------------|-------------------|---------------|----------------|-------------------|--------------|
| SC_GP          | 9.78              | 20.5          | Bayesian appr. | 15.3              | 32.1         |
| MGP            | 10.15             | 21.25         | Boosting      | 24.0              | 30.0         |
| 2SGP           | **9.73**          | **19.85**     | Bagging       | 15.3              | 31.6         |
| GP             | 11.11             | 21.66         | RSM           | 14.8              | 32.3         |
| Fuzzy classifier | 10.9             | 20.6          | CCEL          | 13.4              | 25.4         |
| C4.5           | 10.14             | 22.27         | CART          | 12.56             | 24.35        |
| LR             | 13.04             | 21.63         | MLP           | 10.14             | 23.82        |
| k-NN           | 28.5              | 28.49         | ANNEA_IS      | 9.97              | 21.7         |

Taking the data presented in tables 2, 4 and 5 into consideration, we may conclude the usefulness of applying the instance selection algorithm with the automated generation of neural classifiers. When compared with the base model, the algorithm has shown an improving effect for all ten problems. On
9 of them the accuracy of the resulting model has been improved, and the computational complexity has been decreased on all ten of them. Compared to analogues, the implemented program system with the described algorithm has shown the best result in 30% of the cases. The listed facts allow us to expect further effective usage of the developed algorithms in modern data analysis problems.

5. Conclusions
In this work, an instance selection algorithm has been described, which can be used in intelligent data analysis technologies. The algorithm implements the wrapper strategy, i.e. uses the data received from a certain model when operating. In this work, the approach was considered when using neural classifiers generated by self-configured evolutionary algorithms. As a result of testing and comparing the implemented program system to analogues it has been shown that the proposed approach results in a significant increase in classification quality and a decrease in computational expenses. On average, the time required for the program system to work was decreased by 2.5 times, and the classification quality has increased by 14%. The presented approach could be used not only for the automated design of neural networks, but also for other intelligent data analysis methods, including ensembles.

Acknowledgments
The reported study was funded by RFBR and Government of Krasnoyarsk Territory according to the research project № 16-41-243064.

References
[1] Khritonenko D, Semenkin E 2013 Distributed self-configuring evolutionary algorithms for artificial neural networks design Vestnik SibGAU 4 (50) pp 113–116
[2] Khritonenko D, Semenkin E, Sugak E, Potilitsina E 2015 Solving the problem of city ecology forecasting with neuro-evolutionary algorithms Vestnik SibGAU 16 (1) pp 137–142
[3] Semenkin E, Semenkina M 2013 Classifier Ensembles Integration with Self-Configuring Genetic Programming Algorithm Lecture Notes in Computer Science: Adaptive and Natural Computing Algorithms 7824 pp 60-69
[4] Cano J R, Herrera F, Lozano M 2007 Evolutionary Stratified Training Set Selection for Extracting Classification Rules with trade off Precision-Interpretability Data & Knowledge Engineering archive 60 (1) pp 90-108
[5] Cano J R, Herrera F, Lozano M 2003 Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study IEEE Transactions on Evolutionary Computation 7 (6)
[6] Hart P E 1968 The condensed nearest neighbor rule IEEE Trans Inf Theory 14 pp 515–516
[7] Olvera-López J A, Ariel Carrasco-Ochoa J, Francisco Martinez-Trinidad J, Kittler J 2010 A review of instance selection methods Artif Intell Rev 34 pp 133–143
[8] Stanovov V, Semenkin E, Semenkina O 2015 Self-configuring hybrid evolutionary algorithm for fuzzy classification with active learning IEEE Congress on evolutionary computation
[9] Stanovov V, Semenkin E, Semenkina O 2015 Instance Selection Approach for Self-configuring Hybrid Fuzzy Evolutionary Algorithm for Imbalanced Datasets LNCS: Sixth International Conference on Swarm Intelligence 9140 (1) pp 451–459
[10] Hansen N 2010 Variable Metrics in Evolutionary Computation Habilitation à diriger des recherches, Université Paris-Sud
[11] Eiben A, Smit S 2011 Parameter tuning for configuring and analyzing evolutionary algorithms Swarm and Evolutionary Computation 1(1) pp 19–31
[12] Semenkin E, Semenkina M 2012. Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator Lecture Notes in Computer Science: Advances in Swarm Intelligence 7331 pp 414-421
[13] Alcalá-Fdez J, Sánchez L, García S, Jesus M J, Ventura S, Garrell J M, Otero J, Romero C, Bacardit J, Rivas V M, Fernández J C, Herrera F 2009 KEEL: A 132 software tool to assess
evolutionary algorithms for data mining problems *Soft Comput* 13(3) pp 307–318

[14] Asuncion A, Newman D 2007 UCI machine learning repository *University of California, Irvine, School of Information and Computer Sciences* [URL]: http://www.ics.uci.edu/~mlearn/MLRepository.html

[15] Berlanga F J, Rivera A J, del Jesus M J, Herrera F 2010 GP-COACH: Genetic programming-based learning of compact and accurate fuzzy rule-based classification systems for high-dimensional problems *Inf. Sci. 180(8)* pp 1183–1200

[16] Ishibuchi H, Mihara S, Nojima Y 2013 Parallel Distributed Hybrid Fuzzy GBML Models With Rule Set Migration and Training Data Rotation *IEEE Transactions on fuzzy systems 21(2)*

[17] Alcala-Fdez J, Alcala R, Herrera F 2011 A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning *IEEE Trans. Fuzzy Syst. 19(5)* pp 857–872