Parallel Implementation of Evolutionary Learning of a Fuzzy System with Non-Singleton Fuzzification*

S. A. Karatach, V. G. Sinuk

BSTU after V. G. Shukhov, Belgorod, Russia

Abstract. Fuzzy systems with fuzzy inputs can be used in tasks where it is necessary to make predictions for data objects that have fizzy characteristics. However, building an optimal block of rules for such a system may be non-trivial, including due to the requirement to have a certain depth of knowledge in the subject area. In this situation, there is a need to automate the process of compiling the rule base, that is, to build a machine learning algorithm. In this paper, we propose to use a genetic (evolutionary) algorithm as such an algorithm. It describes both the specifics of using this family of algorithms for training a fuzzy system, and the features of parallel implementation of the learning process using CUDA technology.

Keywords: Parallel implementation CUDA Evolutionary learning Fuzzy system

DOI 10.14357/20718632230212

Introduction

Providing a high level of accuracy in solving classification and regression problems using models rooted in mathematical statistics may be difficult to achieve in some applications where data objects are described using features with uncertain nature. In this case, the mathematical model formalized for working with numerical information causes a narrowing of the interpretation field for the original fuzzy information. For solving problems in such areas, inference systems based on the theory of fuzzy logic [1] are more suitable models, where the inputs are specified either by terms of linguistic variables, or when fuzzy sets are used as a representation of noisy inputs, or the result of the fuzzy system fuzzification procedure.

The construction of a fuzzy inference system involves drawing up a base of rules and defining term sets of linguistic variables [2, 3] for each feature that is fed to the input of the system or issued by it as a solution.

The experience of experts in this field of application can be used to create a base of rules. However, as the number of features describing the input objects of the problem increases, the number of possible rules tends to grow exponentially. This makes it much more difficult to compile the optimal rule base manually and causes the need to create an algorithm for automated construction of a fuzzy inference system.

A fuzzy inference system can be defined as a single parameter vector that includes hyperparameters of the system, parameters of membership functions, and a rule base. The generated vector represents a point in the hyperspace of searching for solutions, each of which defines the entire system in one way or another. Using a specially selected quality function, you can evaluate the accuracy of a given system at each acceptable point in

* The work was carried out with the financial support of the RFBR (project no. 2007-00030).
This work was realized using equipment of High Technology Center at BSTU named after V. G. Shukhov.

the solution domain and, eventually, build a hypersurface. As a result, we can distinguish several features of the parameter vector and its relation to the constructed hypersurface:

– Some components of the vector allow only discrete values, which precludes differentiability of the hyperspace at each point.

– There is no relationship between the increment of the quality function and transitions from one hyperspace point to the neighboring one, since a shift of one model parameter to the neighboring value can dramatically change the value of the quality function.

– The surface formed by the quality function is non-convex and has a set of local optimum points.

The combination of these features excludes the possibility of using convex optimization algorithms (in particular, various modifications of gradient descent). To perform a search in hyperspace with such properties, it is more appropriate to use a genetic algorithm - one of the representatives of the family of evolutionary algorithms [1].

The main difference in research in the direction of adapting a genetic algorithm for training a fuzzy inference system is the method of encoding the parameter vector (or various groups of its components) that defines the entire system in the chromosome. In the work of Thrift, Hwang, and Thompson [4], a rule base is trained for a fixed set of membership functions. In their work [5], Karr and Gentry train the parameters of membership functions, and the quality evaluation function is calculated for a list of all possible rules. However, since the membership functions have a direct connection to the rule base, a more correct adaptation of the genetic algorithm searches for optimal values of both the parameters of the membership functions and the optimal rule base.

In subsequent works [6], membership functions and a list of all possible rules were encoded in the chromosome. This approach has its drawbacks. The most obvious one is the unnecessarily high amount of computation required to perform fuzzy inference during quality function evaluation. The next drawback is a decrease in the stability of the trained system, which is caused by an increase in the variance of models with an increase in their capacity. Both of these disadvantages are particularly acute when the number of inputs to the fuzzy output system increases.

In most cases, the application area can be fully described with relatively small number of rules. Thus, it is sufficient to encode only a subset of all possible rules, which then evolves to more optimal sets of rules. This encoding method significantly reduces the size of the chromosome to the minimum necessary, which allows the genetic algorithm to find optimal fuzzy inference systems for larger problems in an acceptable time.

In this paper, we consider the adaptation of the genetic algorithm to the problem of training a fuzzy inference system, which involves searching for both a rule base and parameters of membership functions.

To construct a more efficient parallel configuration for fuzzy inference, we used the inference method based on the fuzzy truth value and the decomposition theorem [7, 8].

1. Fuzzy Inference System

The linguistic model is a knowledge base of fuzzy rules $R_k, k = \overline{1, N}$ of the form:

$$R_k : If\ x_1\ is\ A_{1k}, \dots, if\ x_n\ is\ A_{nk}, then\ y\ is\ B_k, \tag{1}$$

where N is a number of fuzzy rules, $A_{ik} \subseteq X_i$, $i = \overline{1, n}, B_k \subseteq Y$ are fuzzy sets that are characterized by membership functions $\mu_{A_{ik}}(x_i)$ and $\mu_{B_k}(y)$ for i = 1,n respectively. $x_1, x_2, \dots, x_n$ are input variables of the linguistic model, and $[x_1, x_2, \dots, x_n]^T = \boldsymbol{x} \in X_1 \times X_2 \times \dots \times X_n$. The symbols X_i and Y represents the spaces of input and output variables respectively. If we enter the notation $\boldsymbol{X} = X_1 \times X_2 \times \dots \times X_n$ and $\boldsymbol{A_k} = A_{1k} \times A_{2k} \times \dots \times A_{nk}$, then rule (1) is represented as fuzzy implication:

$$R_k : \boldsymbol{A_k} \to B_k.$$

The rule R_k can be formalized as a fuzzy relation defined on the set $X \times Y$, i. e. $R_k \subseteq X \times Y$ is a fuzzy set with the membership function:

$$\boldsymbol{\mu_{R_k}}(\boldsymbol{x}, y) = \boldsymbol{\mu_{A_k \to B_k}}(\boldsymbol{x}, y) = I\left(\boldsymbol{\mu_{A_k}}(\boldsymbol{x}), \mu_{B_k}(y)\right),$$

where $I(*)$ is a fuzzy implication.

The task is to determine the fuzzy output $B'_k \subseteq Y$ for a system represented as (1) if the inputs are fuzzy sets $A = A'_1 \times \ldots \times A'_n$, or x_1 is A'_1, and …, and x_n is A'_n.

According to the generalized fuzzy rule modus ponens [2], the fuzzy set B'_k is defined by the combination of the fuzzy set A' and the relation R_k, i. e.:

$$B'_k = \boldsymbol{A'} \circ (\boldsymbol{A_k} \to B_k).) \qquad (2)$$

The complexity of expression (2) is exponential with respect to *n*, i. e. $O(|X|^n \times |Y|)$

To reduce the computational complexity to a polynomial one, the decomposition theorem given below was applied to the original statement of the inference problem. Also, to reduce the amount of calculations, the output is performed using a fuzzy truth value.

After moving from a more general compositional inference rule to a particular generalized modus ponens rule, which for single-input systems is described by the relation [1]:

$$\mu_{B'}(y) = \sup_{x\in X}\left\{\mu_{A'}(x) \overset{T}{*} I(\mu_A(x), \mu_B(y))\right\}, \qquad (3)$$

where $A'(x), A(x), B'(y), B(y)$ are membership functions, $\overset{T}{*}$ is a t-norm representing the intersection of the fuzzy fact A' and the fuzzy implication *I*, the argument of which is the premise *A* and the conclusion *B*. Fuzzy sets are described on the reasoning space *X* for the premise and fact, and on *Y* for the value of *B* and the output result of B'.

Using the true modification rule [3], we can write:

$$\mu_{A'}(x) = \tau_{A/A'}(\mu_A(x)),$$

where $\tau_{A/A}(*)$ is fuzzy truth value of the fuzzy set *A* with respect to A', which represents the compatibility membership function $CP(A, A')$ A with respect to A', and A' is considered reliable [9, 10].

$$\tau_{A/A'}(t) = \mu_{CP(A,A')}(t) = \sup_{\substack{\mu_A(x)=t \\ x\in X}} [\mu_{A'}(x)], t \in [0,1] \qquad (4)$$

When passing from variable *x* to variable *t*, denoting $t = \mu_A(x)$, we get:

$$\mu_{A'}(x) = \tau_{A/A'}(\mu_A(x)) = \tau_{A/A'}(t)$$

Then (3) is written as follows:

$$\mu_{B'}(y) = \sup_{t\in[0,1]}\left\{\tau_{A/A'}(t) \overset{T}{*} I(t, \mu_B(y))\right\} \qquad (5)$$

If the membership function $A(x)$ and $A'(x)$ are given as Gaussian curves or as a bell-shaped function, then, as follows from [11], the fuzzy truth value (4) is determined analytically. In the case of piecewise linear representation of membership functions, as algorithm with polynomial computational complexity is developed [12].

For a system with multiple inputs (3) has the form:

$$\mu_{B'_k}(y) = \{\mu_{\boldsymbol{A'}}(\boldsymbol{x})\} \qquad (6)$$

If the *t*-norm – *min* is used to model the linguistic conjunction ”AND” in the antecedent of rule (1), the order of calculations in such a system can be transformed using the decomposition theorem of multidimensional fuzzy implication.

Theorem 1. *If the fuzzy implication* $I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y)), i = \overline{1,n}$ *doesn't increase by the argument* $\mu_{A_{ik}}(x_i)$ *then:*

$$I\left(\boldsymbol{\mu_{A_k}}(\boldsymbol{x}), \mu_{B_k}(y)\right) = I\left(\min_{i=\overline{1,n}}\{\mu_{A_{ik}}(x_i)\}, \mu_{B_k}(y)\right) = \\ = \max_{i=\overline{1,n}}\{I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y))\}$$

When the conditions of this non-increasing theorem are met $I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y)), i = \overline{1,n}$ in respect to $\mu_{A_{ik}}(x_i)$ and usage of *min* for an linguistic bundle ”AND” are satisfied, the equation (6) takes the form:

$$\mu_{B'_k}(y) = \max_{i=\overline{1,n}}\left\{\sup_{x_i\in X_i}\left\{\mu_{A'_i}(x_i) \overset{T}{*} I(\mu_{A_{ik}}(x_i), \mu_{B_k}(y))\right\}\right\} \qquad (7)$$

Also, the expression (7) can be written using a fuzzy truth value, as follows from (4), i.e. (7) will have the form:

$$\mu_{B'_k}(y) = \max_{i=\overline{1,n}}\left\{\sup_{t_i\in[0,1]}\left\{\tau_{A_{ik}/A_i}(t_i) \overset{T}{*} I(t_i, \mu_{B_k}(y))\right\}\right\}. \qquad (8)$$

The discrete analogue of the relation (8) has a polynomial computation complexity, i.e. $O(|t_i| \times |Y| \times n)$. It is worth noting that the non-increasing condition $I\left(t_i, \mu_{B_k}(y)\right), i = \overline{1,n}$ with respect to t_i is satisfied for some implications [2].

Applying the method of defuzzification of the center of gravity from the relation (8), we obtain the expression:

$$\bar{y} = \frac{\sum\limits_{k=\overline{1,N}} \bar{y}_k \underset{j=\overline{1,N}}{T}\left\{\max\limits_{i=\overline{1,n}}\left\{\sup\limits_{t_i\in[0,1]}\left\{\tau_{A_{ik}/A'_i}(t_i)\overset{T}{*}I(t_i,\mu_{Bj}(\bar{y}_k))\right\}\right\}\right\}}{\sum\limits_{k=\overline{1,N}} \underset{j=\overline{1,N}}{T}\left\{\max\limits_{i=\overline{1,n}}\left\{\sup\limits_{t_i\in[0,1]}\left\{\tau_{A_{ik}/A'_i}(t_i)\overset{T}{*}I(t_i,\mu_{Bj}(\bar{y}_k))\right\}\right\}\right\}} \quad (9)$$

where $\bar{y}_k$ is the center of the membership function $B_k(y)$.

The ratio (9) corresponds to the network structure (Fig. 1). On this figure, each network-layer corresponds to the sub-expression of the related expression.

2. Genetic Algorithm for Training a Fuzzy Model

By specifying a learning algorithm, a fuzzy model is formed from a fuzzy system. In this case, the fuzzy system is defined by a parameter vector consisting of a set of fuzzy term sets and a rule base. According to the description given earlier, a Gaussian membership function with two parameters and is selected for specifying a fuzzy term set, and each rule is defined as a tuple of the numbers of the corresponding term sets in the set. The work of the genetic algorithm consists in setting the parameter vector that defines the fuzzy inference

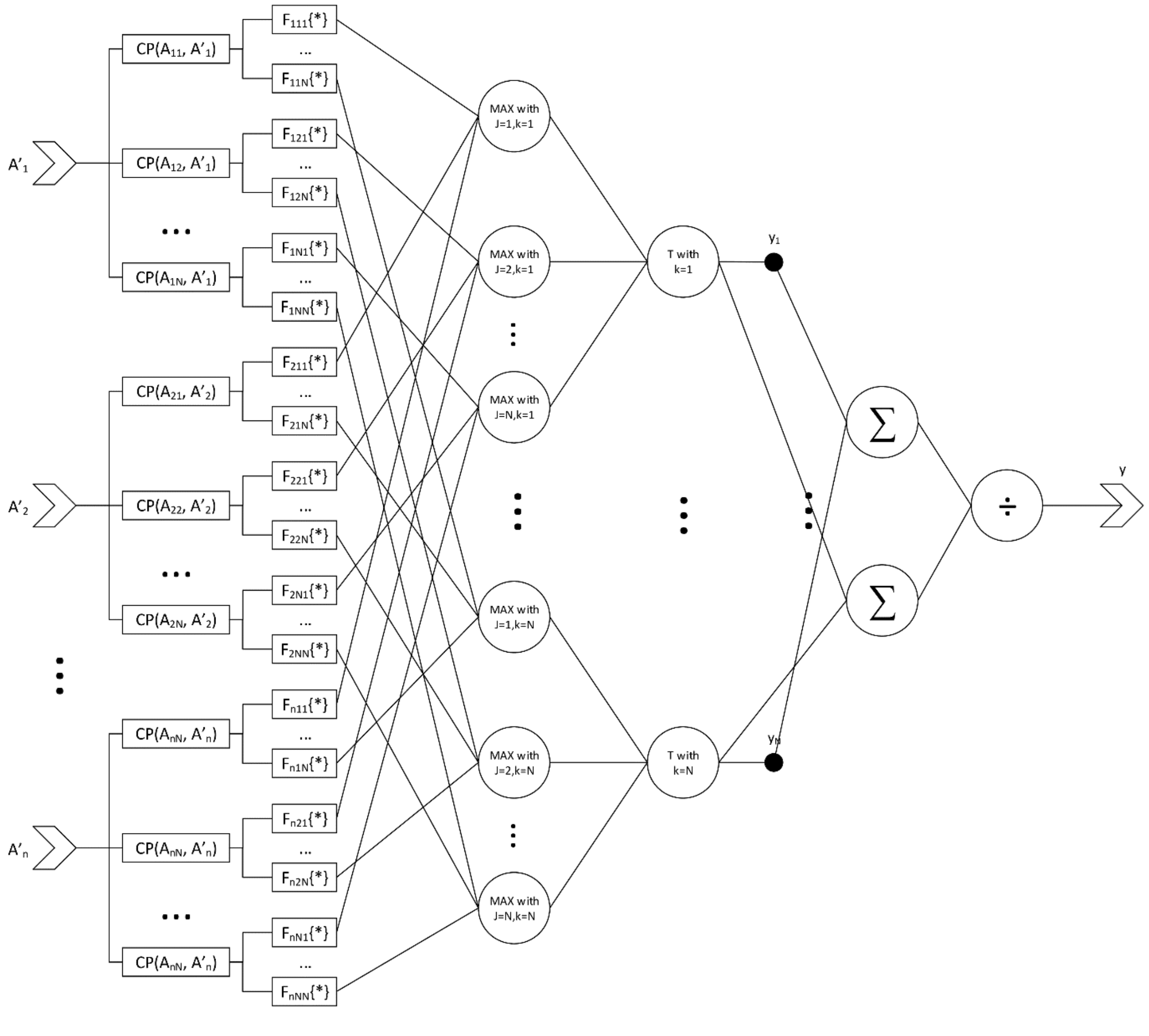

Fig. 1. Network structure for the rule base

system [11]. In this case, the size of the rule base, the power of term sets, and the types of membership functions are considered fixed before the training algorithm is launched and are not subject to adjustment during its operation. The block diagram of the genetic algorithm adapted for configuring the fuzzy inference system is shown in the Fig. 2. In this flowchart, the parameters of the genetic algorithm are: N - the number of iterations of the algorithm, PP - the population size, n-the number of system inputs, FP_i-the power of the term set of the linguistic variable for the i-th input, RP - the number of rules in the database, p_c and p_m-the probability of crossing and mutation, respectively.

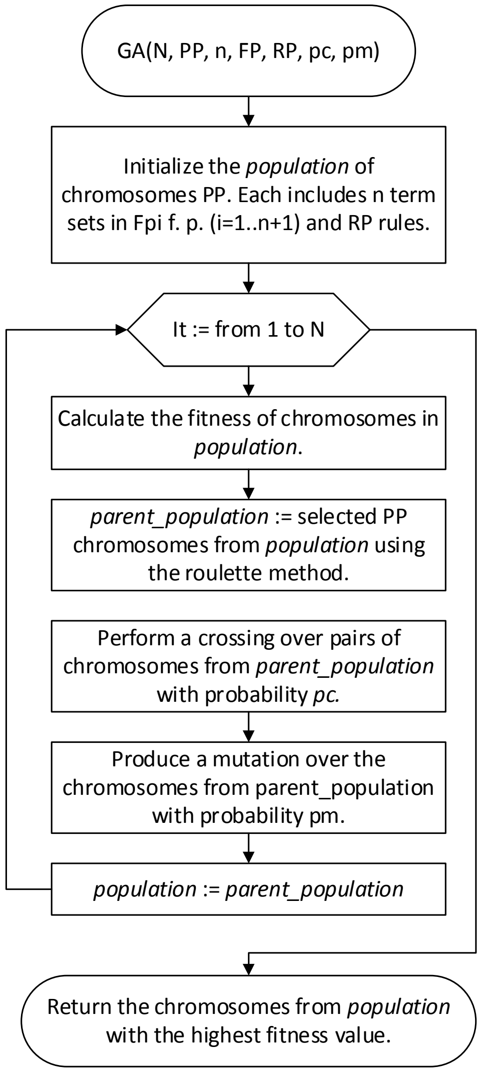
GA(N, PP, n, FP, RP, pc, pm)
Initialize the population of chromosomes PP. Each includes n term sets in Fpi f. p. (i=1..n+1) and RP rules.
It := from 1 to N
Calculate the fitness of chromosomes in population.
parent_population := selected PP chromosomes from population using the roulette method.
Perform a crossing over pairs of chromosomes from parent_population with probability pc.
Produce a mutation over the chromosomes from parent_population with probability pm.
population := parent_population
Return the chromosomes from population with the highest fitness value.

Fig.2. Adaptation of a genetic algorithm for setting up a fuzzy inference system

Encoding

In this algorithm, the sets of parameters for training are both parameters of Gaussian accessory functions and term numbers in the rule base.

The set of parameters of the term sets of the j-th chromosome is represented as:

$$G_j = \big((\mu_1, \sigma_1), \dots, (\mu_{FTP}, \sigma_{FTP})\big),$$

where $FTP = \sum_{i=1}^{n+1} FP_i$, n is the number of inputs, and FP_i is the power of the ith term.

The part of the jth chromosome corresponding to kth rule corresponding to the following sequence of parameters:

$$R_{j,k} = (r_1, \dots, r_n, r_{n+1}).$$

Since the idea of the genetic algorithm is primarily aimed at selecting discrete values, it is necessary to choose a method for discrete encoding of the real components of the parameter vector. For each such component of the parameter vector, some basic (mean) value can be selected, in the vicinity of which the desired value of this component is located. Then the neighborhood of the selected value can be divided into $N - 1$ sections, resulting in a coordinate grid consisting of N points. Now the real value can be defined using a discrete parameter s equal to the point number in the coordinate grid. Then the work of the genetic algorithm will consist in finding the optimal position in the grid of N neighboring points for each real parameter.

The value of such real parameters for the Gaussian membership function is calculated using the following formulas:

$$\mu = \mu_0 + step \times \frac{s_\mu + \frac{N}{2}}{N}$$

$$\sigma = \frac{s_\sigma}{N},$$

where *step* expresses the width of the section corresponding to the membership function when dividing the entire acceptable range by a number equal to the power of the term set of this linguistic variable, and $\mu_0 = i \times step$, ith number of the membership function in the term set.

Initialization

Let the input of the training procedure be a sequence of instances $\big(x(t), d(t)\big)$.

The input values $x_i, i = \overline{1, n}$ of each instance of a pair of parameters of the Gaussian membership function (μ_i, σ_i). As a rule, the range of definition of the membership function is single [0,1], because for a system that accepts only fuzzy values, it is only important to preserve the semantic proportions between concepts, and the scale of the range for displaying can be chosen arbitrarily.

For each such range, fuzzy partitioning is performed. The number of partitions for each range coincides with the power of the corresponding term set. For each such section, the membership function for the term corresponding to the specified section takes the maximum value. Thus, for input variables, the split width is $step_i = \frac{1}{FP_i}$, $i = \overline{1,n}$ for the output variable, we have $step_{n+1} = \frac{1}{FP_{n+1}}$.

Since the center of each Gaussian membership function must fall on the allocated area, the value of the parameter s is selected randomly from the range $[1, N]$. Discrete initial values corresponding to the root-mean-square deviations of The Gaussian membership function are selected using a similar principle.

The values of the membership function numbers in the rule base for the *i*th input are selected randomly from the possible FP_i values. The values of the accessory function numbers in the output rule base are selected in the same way from FP_{n+1}.

Selection Operator

The selection procedure is applied after evaluating the quality level of fuzzy systems encoded in each of the chromosomes using the fitness function. In this paper, the roulette method was chosen as a selection procedure, which, when randomly playing a number, provides a high probability of hitting the sector corresponding to the fuzzy system that most accurately describes the input data set. As a result of playing numbers and random selection of chromosomes, the parent population is formed.

Crossingover Operator

After selection, pairs are formed from the set of chromosomes of the parent population. During the crossing procedure, the parameter vector of both chromosomes of the pair is divided into two parts in the same ratio. After that, one of the parts is exchanged between the two chromosomes with a certain probability of p_c. As a result of this operation, two chromosomes should be formed, one of which inherited the best components from the two parents and the corresponding fuzzy system has a higher rating of the quality of work.

So, as a result of crossing between two databases of rules of different chromosomes, the set of rules included in the final database of rules can more fully reflect the necessary set of expert knowledge to solve the problem.

Mutation Operator

The mutation procedure is aimed at finding new parameter vectors, usually slightly different from the vectors in the original chromosomes. A change in a small number of parameters allows you to show the degree of proximity of these parameters to the optimal ones at the stage of calculating the system quality function, and, in the end, the fuzzy inference system set by the selected value of a parameter is more likely to be in the final population.

For mutation of parameters of membership functions of term sets of linguistic variables, a random selection is made in the range from 1 to *N* parameters *s*, one of which corresponds to the median, and the other to the standard deviation. When the rule base is mutated, the term number of the corresponding linguistic variable is randomly selected. Mutation of all components of the parameter vector is performed with a certain probability p_m. The value of this parameter affects the ability of this instance (for a given p_m) of the genetic algorithm to search space.

Fitness Function

Let's assume that a set of N data instances is used to train the system. Each instance of the set forms a pair $\left(x(i), d(i)\right), i = \overline{1,N}$. to evaluate the accuracy of a fuzzy system formed from a set of parameters encoded in each chromosome, the fitness function is used. The choice of fitness function for this training method is much wider than in methods using gradient descent. However, in this case, it is proposed to use the inverted average square of the error as the fitness function.

$$fitness = -\sqrt{\frac{1}{N}\sum_{i=1}^{N}(f\left(x(i)\right) - d(i)^2}$$

The Balance Between Examination and Exploitation

For a clearer illustration of the concepts of survey and operation, the concept of selective pressure is introduced in [1]. This concept expresses the ability of a given instance of a genetic algorithm to improve the average fitness level of the parent population. The algorithm is characterized by a high selective pressure when the expected number of copies of the best individual exceeds the expected number of copies of the worst individuals [12].

When the probability of crossingover p_c and mutation p_m increases, the selective pressure of the algorithm increases, and when it decreases, it decreases.

The concepts of survey and exploitation denote two extreme cases of the genetic algorithm. In one case, the genetic algorithm can conduct a survey of the model search space in order to approach the point corresponding to the global optimum of the problem. This occurs when the selective pressure decreases. In another case, when the selective pressure increases, the search space points that are close to optimal are exploited.

While maintaining the balance of these two extreme processes, the genetic algorithm tends to increase the expectation of fitness in the entire population. To maintain the balance, it makes sense to organize a dynamic adjustment of the level of selective pressure during the operation of the genetic algorithm. In this paper, this is achieved by modifying the p_c and p_m parameters at each iteration through a linear transition from a certain initial value to the final one. In the case of p_c, there is a transition from 0.4 to 0.9, and in the case of pm-from 0.005 to 0.1. There are also other approaches for adjusting p_c and p_m parameters based on the use of a fuzzy output system [11].

3. Parallel Implementation

The implementation of a parallel approach to the organization of calculations is provided using the CUDA software technology. The computing model offered by this technology implies simultaneous execution of program code on the Central processor and on the graphics processor [15-17]. In addition, asynchronous code execution is usually organized on the host and on the device.

A genetic learning algorithm consists of certain sequential computational steps, such as: fitness assessment, selection, interbreeding, and mutation. Each of these steps, in turn, can also be broken down into several smaller steps. This partitioning allows you to provide a greater level of flexibility when building the launch configuration of a particular CUDA core and achieve significantly more full employment of streaming multiprocessors.

According to the performance estimates of CUDA cores obtained using profiling utilities, the largest amount of calculations is performed on the core corresponding to the calculation of the fitness function values. This is because there is a need to process operations in a grid of large-sized blocks. In the simplest case, the grid of blocks is formed in such a way that each block processes one chromosome and one instance of data.

However, for GPUs with version 3.5 of the functionality, there are restrictions on the number of resident blocks for each streaming multiprocessor equal to 16 and on the number of resident threads equal to 2048. In this case, with a small number of input parameters and a small number of rules that do not exceed 128 in the product, there will inevitably be a downtime of computing resources. As a solution to this problem, you can ensure that multiple instances of data are processed in a single block. For this, we introduce the notion of a “thread group” or “warp group”. Within this abstraction, each thread group is responsible for processing a single instance of data. This approach allows you to circumvent the restriction on the number of chromosomes and data instances that can be efficiently processed simultaneously by a single streaming multiprocessor. Moreover, it makes sense to minimize the number of resident blocks by achieving the maximum size of the thread group. This will give the hardware scheduler more flexibility when replacing idle warps, for example, due to delays in operations, with warps of other blocks.

Since the prototype of the genetic algorithm was a natural mechanism, the learning process implies a stochastic modification of the learning context. In this regard, for each next generation, the values of the chromosome genes are randomly selected. As part of the implementation, it makes sense to use cuRAND, a pseudo-random number generation library included in the CUDA technology software package. This library provides a set of overloaded functions for each pseudo-random number generation algorithm. The input point for using the functionality of this library is the function for initializing the context of the selected algorithm - *curand init*. The initialized context can then be used to generate non-negative integer values using the curand function and to generate real numbers using the *curand uniform* or *curand normal* functions. The last two functions generate a sequence of numbers distributed according to a uniform or normal law, respectively. As a result, each of these functions finds its application in the implementation of a particular operation of the evolutionary learning algorithm. It is also worth mentioning that the context of the pseudorandom number generation algorithm must be provided for each thread using the cuRAND library functions separately.

Data organization in memory is a sequence of grouped n-dimensional tensors of the 1st, 2nd, and 3rd order. Some of them are located in memory in such a way that the elements of the first dimension are aligned for more efficient access to them from separate blocks. The first group of data is represented by a set of information about dimensions and linear offsets in an array of parameters of membership functions of fuzzy term sets. The next group consists of two pairs of sets of directly trainable parameters of the evolutionary algorithm corresponding to the parameters of the Gaussian membership function and a set of rules. Each pair represents a parent and child population. Next is a group of input and output data from the training sample. The data structures of the latter group are used to a greater extent in assessing the fitness function and subsequent selection of chromosomes.

Among the features of a more subtle implementation of the algorithm, it is worth noting that when initializing the parameters of chromosomes, it makes sense to set the parameter h to a value of a fairly small order. This will lead to a temporary fixation of the parameters of membership functions in all chromosomes at the first iterations of the algorithm and will allow the algorithm to actually select chromosomes with the best rule bases, which in the future will form a more stable population.

4. Efficiency evaluation

This section provides an assessment of the results of choosing a training method and its parameters when implementing a neuro-fuzzy model. The training was performed for the Balance Scale dataset [18], which contains 4 categorical input attributes and a categorical output attribute. The primary criterion to be evaluated is the accuracy of the system obtained as a result of the training algorithm. The following practically significant criteria are the complexity of selecting the parameters of the selected evolutionary algorithm and the model itself, as well as the resource intensity of the learning algorithm.

The accuracy of the obtained model is evaluated in a cross-section of the solution domain with fixed parameters of the learning algorithm: the number of iterations is 1000, the size of the rule block is 10, and the size of the chromosome population, respectively, in the parent and time population of 100. For the most reliable measurement of the model's accuracy, the data sample was mixed and divided in a ratio of 8: 2 into training and test subsamples. Thus, the accuracy of the neuro-fuzzy model was evaluated at each iteration of the evolutionary algorithm for the entire training subsample, since in this algorithm, the conditions of evolutionary selection must be preserved throughout the entire learning process. The accuracy of the predictions of the resulting fuzzy system was evaluated on previously unknown instances of the model, which also allows us to judge the achieved level of generalization of the model by comparing its quality estimates in the process and as a result of training.

The growth dynamics of the average accuracy for a population depends directly on the capacity of the population representatives (in this case, this value is identified with the number of rows in the rule database) and on the size of the population itself. This is due to both an exponential increase in the number of possible combinations of algorithm values, and a decrease in the step of evolutionary adjustment of parameters, which also depends on the number of genes in the chromosome.

As a result of training the accuracy of the trained model measured on the test sample is: 0.78.

In order to simplify the implementation of the genetic algorithm, the size of the rule block is fixed when this algorithm is run. However, this parameter can be selected experimentally. To do this, the learning process can be started for a small number of generations and with a different number of rules in the rule database. After that, the accuracy of each of the fuzzy systems obtained as a result of training is evaluated for each value of the size of the rule base. Then the value corresponding to the maximum accuracy should be used when starting the full training process.

The values of the parameters μ and λ increase with the number of possible combinations of values forming a block of rules.

The training time for the above configuration of the genetic algorithm running on the CPU (using OpenMP technology) was 186 minutes. At the same time, the training process using the GPU was completed in 2 minutes. An Intel(R) Core(TM) i7-3930K CPU and an Nvidia Tesla K20c GPU were installed on the target machine.

In the process of searching for the optimum point, it is possible that chromosomes containing contradictory rules in the rule base may occur. Let's consider the situation of a contradiction of a pair of rules for some one input. Then the membership functions of the corresponding fuzzy sets do not intersect. For the system considered in this article, when the rules are connected by a bundle AND, such a situation causes a decrease in the value of the fitness function, which increases the likelihood of excluding this chromosome from the population. If the rules are connected by a bundle OR, then such a contradiction is equivalent to excluding this input for this pair of rules.

If such a fitness function is chosen as in this paper, then the selected data set for training will directly affect the generalizing ability of the resulting fuzzy system. To increase the generalizing ability, regularization can be included in the fitness function, for example, requiring a uniform total level of operation of each rule in the rule database on the entire data set [14].

At the same time, the genetic algorithm does not guarantee finding the global optimum, but can stop at the point of the local optimum. In the case of the method of encoding the parameters of a fuzzy system used in this article, the encoded rule base contains contradictory rules in the corresponding found optimum point of the chromosome, or the found rule base does not generalize enough.

Conclusion

As a result of the research conducted in this paper, we can draw the following conclusions about the feasibility of using the approach proposed in this article for the automated construction of a fuzzy inference system. The disadvantages of the training method developed in the article are both the need for a large amount of computing resources and memory, and the need to manually set a number of parameters of the training algorithm. The advantages of the adapted genetic algorithm include the possibility of flexible parallelization of the computational process, as well as the accuracy of the fuzzy output models obtained at the output of the genetic algorithm.

References

1. L. Rutkowski. Methods and techniques of artificial intelligence. Hot Line Telecom, Moscow, 2010. ISBN 978-5-9912-0105-6. doi: 10.1049/piee.1974. 0328.
2. L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning—i. 8(3):199–249, 1975. doi: 10.1016/0020-0255(75) 90036-5.
3. N. Borisov, A. V. Alekseev, O. A. Kromberg, and etc. Decision-making models based on a linguistic variable. Senate, Riga, 1982.
4. Wen-Ruey Hwang and W. E. Thompson. Design of intelligent fuzzy logic controllers using genetic algorithms, 1994.
5. C. L. Karr and E. J. Gentry. Fuzzy control of ph using genetic algorithms. IEEE Transactions on Fuzzy Systems, 1(1):46–, 1993. doi: 10.1109/TFUZZ. 1993.390283.
6. Michael A. Lee and Hideyuki Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In Proceedings of the 5th International Conference on Genetic Algorithms, page 76–83, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1558602992. doi: 10.5555/645513.657425.
7. D. Dubois and A. Prad. Theory of possibilities. Applications to the representation of knowledge in computer science. Radio and communications, Moscow, 1990. ISBN 5-256-00184-1. doi: 10.1007/3-540-45493-4 24.
8. V G Sinuk and M V Panchenko. Method of fuzzy inference for one class of MISO-structure systems with non-singleton inputs. IOP Conference Series: Materials Science and Engineering, 327:042074, 3 2018. doi: 10.1088/1757-899x/327/4/042074.
9. V. G. Sinuk and E. V. Pivnenko. About an analytic calculation of fuzzy truth value. pages 129–133, 2006.
10. D. A. Kutsenko and V. G. Sinuk. Algorithms for finding cp under a piecewise linear representation of membership functions. pages 87–92, 2008.
11. Yuhui Shi, R. Eberhart, and Yaobin Chen. Implementation of evolutionary fuzzy systems. IEEE Transactions on Fuzzy Systems, 7(2):109–119, 1999. doi: 10.1109/91.755393.

12. Komartsova L.G., Kureychik V.V., Sorokin S.N., Tsoi Y.R., Yankovskaya A.E., Yarushkina N.G. Bionic information systems and their practical applications. Fizmatlib, 2011. ISBN: 978-5-9221-1302-1.
13. Gladkov L. A., Kureychik V. V., Kureychik V. M. Genetic algorithms. Fizmatlit, 2nd ed, 2006. ISBN: 978-5-9221-0510-1.
14. Jason Sanders and Edward Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional, 1st edition, 2010. ISBN 0131387685.
15. NVIDIA Developer Zone. Cuda programming guide. https://docs.nvidia. com/cuda/cuda-c-programming-guide/ index.html, 2020. [Online; accessed 1-December-2020].
16. NVIDIA Developer Zone. Cuda best practices guide. https://docs.nvidia. com/cuda/cuda-c-best-practices-guide/ index.html, 2020. [Online; accessed 1-December-2020].
17. UCI Machine Learning Repository. Balance scale data set. https:// archive.ics.uci.edu/ml/datasets/Balance+Scale, 2020. [Online; accessed 1December-2020].
18. Yuqi Cui and Dongrui Wu and Jian Huang. Optimize TSK Fuzzy Systems for Classification Problems: Minibatch Gradient Descent With Uniform Regularization and Batch Normalization. IEEE Transactions on Fuzzy Systems. Institute of Electrical and Electronics Engineers (IEEE), 12 (28), pages 3065-3075, 2020. doi: 10.1109/tfuzz.2020.2967282.

Sinuk V. G. Professor, Belgorod state technological university named after V. G. Shukhov, 46 Kostukova str., Belgorod, 308012, Russia, email: vgsinuk@mail.ru

Karatach S. A. Student, Belgorod state technological university named after V. G. Shukhov, 46 Kostukova str., Belgorod, 308012, Russia, email: karatach1998@yandex.ru