STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVIII**, Number 2, 2003

A NEW DYNAMIC EVOLUTIONARY CLUSTERING TECHNIQUE. APPLICATION IN DESIGNING RBF NEURAL NETWORK TOPOLOGIES. I. CLUSTERING ALGORITHM

D. DUMITRESCU AND KÁROLY SIMON

Abstract. Recently a new evolutionary optimization metaheuristics, the Genetic Chromodynamics (GC) has been proposed. Based on this metaheuristics a dynamic clustering algorithm (GCDC) is proposed. This method is used for designing RBF neural network topologies. Complexity of these networks can be reduced by clustering the training data. The GCDC technique is able to solve this problem. In Part I the GCDC technique is presented. It is described, how this method could be used for designing optimal RBF neural network topologies. In Part II some numerical experiments are presented.

Keywords and phrases: Dynamic evolutionary clustering, Genetic Chromodynamics, designing neural networks, RBF neural networks.

1. Introduction

By clustering a data set is divided into regions of high similarity, as defined by a distance metric. In most instances, a prototypical vector (the cluster center) identifies a cluster. Hence, the problem of cluster optimization is twofold: optimization of cluster centers and determination of number of clusters. The latter aspect has often been neglected in standard approaches (static clustering methods), as these typically fix the number of clusters *a priori.*

In case of practical problems the number of natural existing clusters is generally unknown. Opposed to static, dynamic clustering does not require *a priori* specification of the number of clusters. Some tentative to develop dynamic evolutionary clustering algorithms are known [5].

Recently a new evolutionary search and optimization metaheuristics - called Genetic Chromodynamics (GC) (see [4, 13]) - has been proposed. Based on this theory a clustering method is proposed. This GC-based dynamic clustering technique (GCDC) can be successfully used for designing RBF neural network topologies.

Received by the editors: September 25, 2003.

2000 *Mathematics Subject Classification.* 68T05, 68T20, 91C20, 92B20.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Connectionism and neural nets*; I.5.3. [**Pattern Recognition**]: Clustering – *Algorithms.*

Solving a problem with a neural network a primordial task is the determination of the network topology. Generally the determination of the neural network topology is a complex problem and cannot be easily solved. When the number of trainable layers and processor units (neurons) is too low, the network is not able to learn the proposed problem. If the number of layers and neurons is too high then the learning process becomes too slow. The main aim is designing optimal topology.

Radial Basis Function (RBF) neural networks are relatively simple neural networks, used especially for solving interpolation problems (see [1, 9, 10, 11, 12, 14, 15, 18]). Complexity of these networks depends on the number of hidden processor units. In the case of the RBF neural networks is dependence between the number of training samples and the number of hidden neurons. Complexity of networks can be reduced by clustering the training data.

Generally, some static clustering techniques are used in order to reduce the complexity of RBF networks. The most popular static clustering algorithms are the k-means type algorithms (see [16, 17]). These methods require *a priori* specification of the number of existing clusters. Dynamic evolutionary clustering techniques could be more efficient for designing optimal RBF neural network topologies (see [6, 7, 8, 19]).

In the next section the GCDC method is described. Section 3 presents how this method can be used for designing RBF neural networks.

2. GC-based Dynamic Clustering

GC [4] is a new kind of evolutionary search and optimization metaheuristics. GC is a metaheuristics for maintaining population diversity and for detecting multiple optima. The main idea of the strategy is to force the formation and maintenance of stable sub-populations.

GC-based methods use a variable-sized population, a stepping-stone search mechanism, a local interaction principle and a new operator for merging very close individuals.

Corresponding to the stepping-stone technique each individual in the population has the possibility to contribute to the next generation and thus to the search progress. Corresponding to the local interaction principle the recombination mate of a given individual is selected within a determined mating region. Only short range interactions between solutions are allowed. Local mate selection is done according to the values of the fitness function. An adaptation mechanism can be used to control the interaction range, so as to support sub-population stabilization. Within this adaptation mechanism the interaction radius of each individual could be different.

To enhance GC, micropopulation models [13] can be used. Corresponding to these models, for each individual a local interaction domain is considered. Individuals within this domain represent a micropopulation. All solutions from a micropopulation are recombined using local tournament selection. When the local domain of an individual is empty the individual is mutated.

Within GC sub-populations co-evolve and eventually converge towards several optima. The number of individuals in the current population usually changes with the generation. A merging operator is used for merging very close individuals. At convergence, the number of sub-populations equals the number of optima. Each final sub-population hopefully contains a single individual representing an optimum, a solution of the problem.

GC allows any data structure suitable for the problem together with any set of meaningful variation/search operators. For instance solutions may be represented as real-component vectors. Moreover the proposed approach is independent of the solution representation.

Based on the GC metaheuristics a new dynamic clustering algorithm - called GCDC - is developed. This technique is described below.

2.1. **Solution Representation.** Let

$$X = \{x_1, ..., x_m\}, x_i \in \mathbf{R}^s, s \geq 1,$$

be the data set for clustering. The cluster structure of X is given by a fuzzy partition $P = \{A_1, ..., A_n\}$ of X. Every class A_i is represented by a prototype $L_i \in \mathbf{R}^s$. $L = \{L_1, ..., L_n\}$ is the representation of the partition P.

In the proposed clustering technique each prototype is encoded into a chromosome. Totality of these chromosomes represents a generation.

The idea of the method is to determine formations of evolving chromosomes converging towards prototypes of real clusters.

The initial population is randomly generated and it contains a large number of individuals. Operations involved in the searching process are selection, crossover, mutation and merging.

2.2. **Fitness Function.** The fitness value of the chromosome L is calculated using the following fitness function:

$$f(L) = \sum_{i=1}^{m} \frac{1}{d^{\alpha}(x_i, L) + C},$$

where $\alpha \geq 1$ and $C > 0$.

The role of the constant C is to prevent infinite or too great values for the fitness function, and together with α controls the granularity of the clusters.

2.3. **Interaction Range.** For each individual in the population (a chromosome representing a prototype) a mating region is considered as the closed ball with radius d^*, where the *interaction radius* d^* depends on the chromosome.

Initially we consider the neighborhood distance for each chromosome as the standard deviation of the all points. For a chromosome L the mean distance $\overline{\delta}$ is calculated between the points in $V\left(L, d^{*}\right)$ and L:

$$\overline{\delta}=\sum_{i=1}^{n_{d^{*}}} \frac{d\left(x_{i}, L\right)}{n_{d^{*}}},$$

where $x_{1}, \ldots, x_{n_{d^{*}}}$ are the points in the neighborhood with radius d^{*} of L.

When the points in $V\left(L, d^{*}\right)$ are uniformly distributed, the value of $\overline{\delta}$ is $\frac{d^{*}}{\beta}$, where $\beta \in(1,2]$ is a fixed number, which depends on the dimension s of the search space (generally the best value for β is $\sqrt[s]{2}$). d^{*} is adjusted such that $\overline{\delta}$ to be equal with $\frac{d^{*}}{\beta}$, so if $\overline{\delta} \leq \frac{d^{*}}{\beta}$ then the next value for d^{*} will be $\beta \overline{\delta}$, else $\overline{\delta}$. If there are not at least two points in the neighborhood of the chromosome, then the previous distance value will be not modified.

2.4. **Genetic Operators.** A micropopulation model is used. At each step of the generation process every chromosome is selected to produce an offspring through crossover or mutation. An individual can be involved into a crossover operation only with individuals that are at smaller distance than d^{*}. The crossover operation is a convex combination of the codes of the genes. The coefficient of the combination is a randomly generated number for each gene.

The mate for the crossover operation for an individual is selected among the chromosomes in its neighborhood with a proportional selection. Later the mate will be selected as first parent to produce its offspring. For this reason at crossover only one new chromosome is generated. If there is no mate for the crossover operation in the neighborhood of radius d^{*} of an individual, then the mutation operator will be applied.

Mutation is an additive perturbation of the genes with a randomly chosen value from a N(0,σ) normal distribution, where σ is a control parameter called *mutation step size.*

At each generation every chromosome is involved in crossover or mutation. An offspring can replace only its parent. When an offspring is produced, it is compared with the parent and the best (with better fitness) is introduced in the new generation.

An effect of the crossover operation is that the chromosomes in the same subpopulation are overlapping after a number of iterations. When the distance between two chromosomes is smaller than a considered value ε (*merging radius*) they are merged. In this way the size of the population decreases during the process until n individuals remain, where n is the optimal cluster number.

2.5. **Termination and Fuzzy Class Detection.** If no more changes occur in the population through a fixed number of iterations then the process will stop. Individuals constituting the last population are considered as prototypes of the detected clusters. For all data points the fuzzy membership degrees to the clusters determined by the prototypes are calculated.

3. Designing RBF Neural Networks Using the GCDC Technique

Complexity of RBF neural networks depends on the number of hidden processor units. There is dependence between this number and the number of training samples. Complexity of networks can be reduced by clustering the training data.

3.1. **Designing and Training RBF Neural Networks.** RBF is a feed-forward neural network with an input layer (made up of source nodes: sensory units), a single hidden layer and an output layer. The network is designed to perform a nonlinear mapping from the input space to the hidden space, followed by a linear mapping from the hidden space to the output space.

The activation functions for the processor units in the hidden layer are radial basis functions (for example Gaussian functions). These functions generally have two parameters: the center and the width. The argument of the activation function of each hidden unit computes a distance between the input vector and the center of that unit.

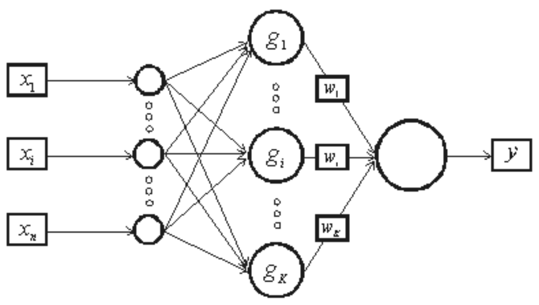

Figure 1. RBF neural network topology

If **x**$= (x_1, ..., x_n)$ is the input vector, $g_i()$ is the activation function and w_i is the synaptic weight corresponding to the i^{th} hidden neuron, then the output created by the network will be:

$$y = \sum_{i=1}^{K} w_i g_i(\mathbf{x}).$$

Usually Gaussian functions are used as RBF. In this case we may consider:

$$g_i(\mathbf{x}) = \mathbf{e}^{-\frac{\|\mathbf{x}-\mathbf{c_i}\|^2}{2\sigma_\mathbf{i}}},$$

where c_i is the center parameter and σ_i is the variance (width parameter) for the function corresponding to neuron i.

The hidden layer of the RBF neural networks may be trained with a supervised learning algorithm. The aim is to establish the synaptic weights of the network. A descendent gradient-based algorithm can be considered.

Let us to note:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, i = 1, ..., K,$$

where η is the *learning rate* and E is the *global learning error.*

At the l^{th} step of the learning process the global learning error is calculated according to the formula:

$$E_l = \frac{1}{N} \sum_{i=1}^{N} (z_i - y_i)^2$$

where N is the number of points in the training data set, z_i is the desired output and y_i is the network output.

Network weights are modified according to the following correction rule:

$$w_i = w_i + \Delta w_i, i = 1, ..., K.$$

3.2. **Using GCDC for Designing Optimal RBF Neural Network Topologies.** Complexity of RBF neural networks depends on the number of hidden neurons. This is the number of radial basis functions with different center parameters. It should be favorable the use of training samples as RBF centers, but in some cases this is impossible. If few training points are present then all of them should be used as centers. In this case the number of processor units in the hidden layer is equal with the number of training samples. If the number of training samples is high, then not all of them might be used (the number of hidden processor units must be reduced). In this situation the locations of the centers may be chosen randomly from the training data set. In practical situations this solution is not very efficient. A better idea is to consider a single neuron for a group of similar training points. These groups of similar training points can be identified using clustering methods.

Generally, some static clustering techniques are used in order to solve the RBF center detection problem. GCDC does not require *a priori* specification of the number of clusters. The algorithm is able to determinate this number, so it can be used for designing optimal RBF neural network topologies.

Let

$$T = \{(x_i, z_i) | x_i \in R^n, z_i \in R, i = 1, ..., N\},$$

be the set of training samples.

The GCDC algorithm is used for clustering this training set. A RBF neural network is considered. The number of neurons in the hidden layer of the network is K, where K is the number of clusters determined by the GCDC method. Cluster centers identified by the GCDC algorithm are used as center parameters for the activation functions. Width parameters can be determined corresponding to the diameter of the clusters. In this way optimal RBF neural network topology can be obtained.

4. Conclusions

Within clustering problems a primordial task is the determination of the number of natural existing clusters. Dynamic clustering techniques are able to solve this problem.

Based on the GC metaheuristics, GCDC is a new evolutionary technique for dynamic clustering. The method can be used for designing optimal RBF neural network topologies.

References

[1] Broomhead D. S., Lowe D.; *Multivariable Functional Interpolation and Adaptive Networks*, Complex Systems, 2 (1988), pp. 321-355.

[2] Dumitrescu D.; *Algoritmi Genetici şi Strategii Evolutive - Aplicaţii în Inteligenţa Artificială şi în Domenii Conexe*, Editura Albastra, Cluj Napoca, 2000.

[3] Dumitrescu D., Lazzerini B., Jain L. C., Dumitrescu A.; *Evolutionary Computation*, CRC Press, Boca Raton, 2000.

[4] Dumitrescu D.; *Genetic Chromodynamics*, Studia Univ. Babes-Bolyai, Ser. Informatica, 35 (2000), pp. 39-50.

[5] Dumitrescu D.; *A New Evolutionary Method and its Applications in Clustering*, Babeş-Bolyai University, Seminar on Computer Science,2 (1998), pp. 127-134.

[6] Dumitrscu D., Simon K.; *Evolutionary Clustering Techniques for Designing RBF Neural Networks*, Babeş-Bolyai University, Seminar on Computer Science, (2003).

[7] Dumitrscu D., Simon K.; *Reducing Complexity of RBF Neural Networks by Dynamic Evolutionary Clustering Techniques*, Proceedings of the 11^th Conference on Applied and Industrial Mathematics, (2003).

[8] Dumitrescu D., Simon K.; *Genetic Chromodynamics for Designing RBF Neural Networks*, Proceedings of SYNASC, (2003).

[9] Enăchescu C.; *Caracterizarea Reţelelor Neuronale ca şi Metode de Aproximare- Interpolare*, Petru Maior University, Buletinul Stiintific, 7 (1994).

[10] Enăchescu C.; *Elemente de Inteligenţă Artificială*, Petru Maior University, Tg. Mureş, 1997.

[11] Haykin S.; *Neural Networks*, Macmillan College Publishing Company, New York, 1994.

[12] Moody J., Darken C.; *Fast Learning in Networks of Locally Tuned Processing Units*, Neural Computation, 1 (1989), pp. 281-294.

[13] Oltean M., Groşan C.; *Genetic Chromodynamics Evolving Micropopulations*, Studia Univ. Babes-Bolyai, Ser. Informatica, (2000).

[14] Poggio T., Girosi F.; *Networs for Approximation and Learning*, Proceedings of IEEE, 78 (1990), pp. 1481-1497.

[15] Powell M. J. D.; *Radial Basis Functions for Multivariable Interpolation: A review*, in Algorithms for Approximation, J. C. Mason and M. G. Cox, ed., Clarendon Press, Oxford, 1987, pp. 143-167.

[16] Schreiber T.; *A Voronoi Diagram Based Adaptive k-means Type Clustering Algorithm for Multidimensional Weighted Data*, Universitat Kaiserslautern, Technical Report, (1989).

[17] Selim S. Z., Ismail M. A.; *k-means Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality*, IEEE Tran. Pattern Anal. Mach. Intelligence, PAMI-6, 1 (1986), pp. 81-87.

[18] Simon K.; *OOP Pentru Calculul Neuronal*, Petru Maior University, Dipl. Thesis, 2002.

[19] Simon K.; *Evolutionary Clustering for Designing RBF Neural Networks*, Babeş-Bolyai University, MSc. Thesis, 2003.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, Cluj-Napoca, Romania
E-mail address: `ddumitr@cs.ubbcluj.ro`

E-mail address: `ksimon9@yahoo.com`