Collective Intelligence from a Population of Evolving Neural Networks

Aleksander Byrski
Marek Kisiel-Dorohinicki

Department of Computer Science
University of Mining and Metallurgy, Kraków, Poland
e-mail: {*olekb,doroh*}*@agh.edu.pl*

Abstract. In the paper an agent system of evolving neural networks, being an example of collective intelligence is presented. A concept of an agent-based predicting system is presented and several methods of managing such collective intelligent system are mentioned. Then the problem of evolutionary optimisation of a predicting neural network is introduced. Finally a neural predicting evolutionary multi-agent system (EMAS) is proposed as a means of integration of these two approaches. General considerations are illustrated by the particular system and selected experimental results conclude the work.

1 Introduction

Numerous computationally difficult problems have already been solved utilising analogies to processes observed in nature. In particular this concerns various soft computing techniqes like artificial neural networks or evolutionary algorithms. The former allow for modelling of very complex functions and non-linear structures with large number of variables. The latter are successfully used to solve difficult search and optimisation problems. Yet one may notice that often combinations of different ideas and methods by the effect of synergy exhibit some kind of intelligent behaviour [2], which is sometimes called *computational intelligence* as opposed to rather symbolic artificial intelligence. This is the case with evolutionary neural networks, which combine neural computing with evolutionary computation paradigm.

At the same time intelligent agents and agent-based systems provide concepts and tools for development of intelligent decentralised systems and may be used as a means for analysis and realisation of hybrid systems, in which different techniques also cooperate to fullfill specified demands, and this cooperation, complied with a specific managing algorithm, can be a source of the, so called, collective intelligence.

Neural network can be in a simple way encoded into a chromosome, and undergo the process of evolution, however classical evolutionary computation brings some shortcommings, e.g. to evaluate this encoded chromosome, the network must be trained, so whole population of the networks must be trained before creating of the next population.

A possible serious improvement of this problem comes with the use of a multi-agent system (MAS), which leads to the decentralisation of the evolutionary process. Training of a neural network may be entrusted to an agent which can operate autonomously, particularly performing actions of reproduction and death. This way various agents operate

on their networks (e.g. train them), simultaneously with the process of evolution occuring in the whole population. Such defined *evolutionary multi-agent system* (EMAS) may help in search for the optimal architecture for the neural network for the given problem, or at least help to establish a starting point for the further network structure development.

In a complex, hybrid system being a kind of a complex artificial intelligence (every agent presents some level of intelligence) deciding, how to determine a correct system response for a given task, so it could be really able to act as a collective intelligence is a very crucial task. With help may come some techniques for combining multiple answers to the same problem, e.g. algorithms of managing modular networks.

This paper presents the concept of a hybrid system, in which many different methods co-operate helping one another in attaining its own goals. The main task of the system is searching for the solution of given time series prediction problem.

2 MAS for time-series prediction

A time-series predicting system may be considered as a box with some input sequences, and predictions of successive values of (some of) these sequences as output. Some intelligent (sometimes not so much) mechanism inside that box should be able to discover hidden regularities and relationships in and between the input sequences. Assuming that the characteristics of the signal(s) may change in time, this mechanism should be also able to dynamically adapt to these changes ignoring different kinds of distortion and noise.

2.1 Predicting MAS

When the signal to be predicted is much complicated (e.g. when different trends change over time or temporal variations in relationships between particular sequences are present) the idea of a multi-agent predicting system may be introduced [4]. The predicting MAS may be viewed the box with a group of intelligent agents inside (fig. 1). Subsequent elements of the input sequence(s) are supplied to the environment, where they become available for all agents. Each agent may perform analysis of incoming data and give predictions of (a subset of) the next-to-come elements of input. Specialisation in function or time of particular agents allow for obtaining better results by cooperation or competition in the common environment. On the basis of predictions of all agents, prediction of the whole system may be generated.

2.2 Managing collective intelligence of a population of agents

While dealing with a group of autonomous intelligent agents, or any beings which can be perceived as a kind of collective intelligence, trying to solve some kind of problem, an important problem arise: how to determine the answer of the whole system to given task, e.g. for mentioned time series prediction problem, it should be decided how to find an agent which output can be presented as the output of the whole system, or how to combine more than one agent's answers to produce the desired output value.

The simplest approach to determine such answer is to choose the agent with the highest accuracy of prediction (the highest level of energy in the system). However learning is an ill-posed problem, with finite data, each learning algorithm converges to a different solution and fails under different circumstances. To create a more reliable system response several techniques of combining multiple learning individuals were invented [1]:

Voting. The simplest way to combine multiple individuals' responses is to take a linear combination of their answers. Every individual is granted with a weight, which denotes importance level of his answer. To determine weights regression methods can be used. In described system, determining of the weights can be based on the agent's energy.

Mixture of experts. In voting all weights of the individuals are constant. To create more flexible system, mixture of experts can be applied. Every individual is an local expert, and presents its responses to the global expert (or gating expert) which assigns specific weights to these answers.

Stacked Generalisation. It is very similar to the *mixture of experts*, with one difference - gating expert can be designed as any nonlinear model (e.g. Multi-Layer Perceptron).

Cascading. A sequence of individuals is considered, each having a level of confidence preassigned (e.g. when one uses a costlier method or uses features that are costlier to extract, it may be more confident than others). Individual d_j can be used if every previous individuals in the sequence were not confident enough.

2.3 Choosing the resultant prediction

While applying kind of voting technique to time series prediction problem, assigning weights to the individual's answer can be based on probabilistic analysis [10]. Every prediction of the given time series has a probability assigned (or a credit function), which can be used to determine the answer of the whole group of predicting individuals. After every prediction step, every individual basing on its predictions and errors:

$$y_t^k = f_K(y_{t-1}, y_{t-2}, \ldots, y_{t-M})$$

$$e_t^k = y_t - \hat{y}_t^k$$

computes its credit function:

$$p_t^k = \frac{p_{t-1}^k \cdot e^{-\frac{|e_t^k|^2}{2\sigma^2}}}{\sum_{n=1}^{K} p_{t-1}^n \cdot e^{-\frac{|e_t^k|^2}{2\sigma^2}}}$$

basing on this function the response of the group of individuals can be weighted combination of the answers (as in voting algorithm) or can be the result of the winner-take-all combination:

1. Weighted combination: $\hat{y}_t = \sum_{k=1}^{K} p_{t-1}^k y_t^k$.
2. Winner-take-all combination: $\hat{y}_t = y^{\hat{z}_t}, where \hat{z}_t = argmax_{k=1,2,\ldots,K} p_{t-1}^k$.

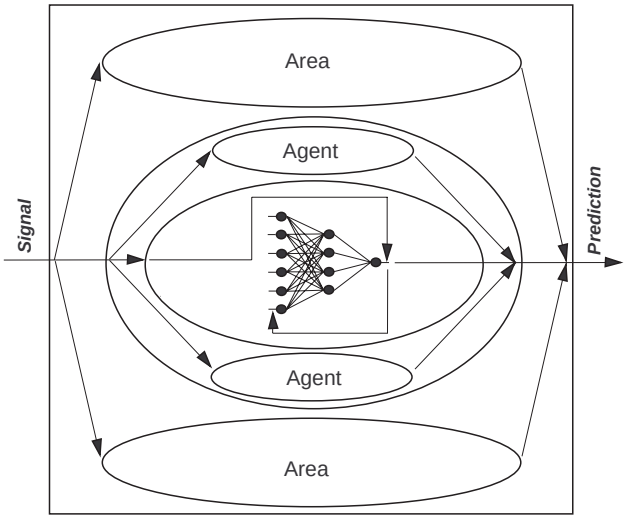

Fig. 1. Predicting neural MAS

3 Evolutionary neural networks for time-series prediction

The main advantage of artificial neural networks is probably their ability to learn from examples and generalise acquired knowledge to new cases in such a way that no explicit problem-dependent knowledge is needed. That is why they are often used in control problems, i.e. management of some process or device, as well as approximation problems, e.g. classification and prediction. The last is of our interest and will be discussed further.

3.1 Prediction with neural networks

A neural network may be thus used as a particular mechanism to model the characteristics of a signal in a system for time-series prediction [8]. Usually the next value of a series is predicted based on a fixed number of previous ones (fig. 2). Thus the number of input neurons correspond to the number of values the prediction is based on, and the output neuron(s) give prediction(s) of the next-to-come value(s) of the series.

The choice of a particular architecture of the network is in great measure determined by the particular problem. The feed-forward neural network on 2 should predict t_{n+1} value of the series, basing on some previous values, which are given on the inputs of the first layer. When t_{n+1} value is predicted, the inputs are shifted, and the value t_{n+1} is given as the input to the last neuron of the first layer.

As a predictors, radial basis function networks (RBF) were used. Way of predicting with use of RBF network is very similar to predicting with multi-layer perceptron networks (MLP) [3]. And the RBF networks being universal approximators [6], seem to be good to use and compare with the multi layer perceptrons.

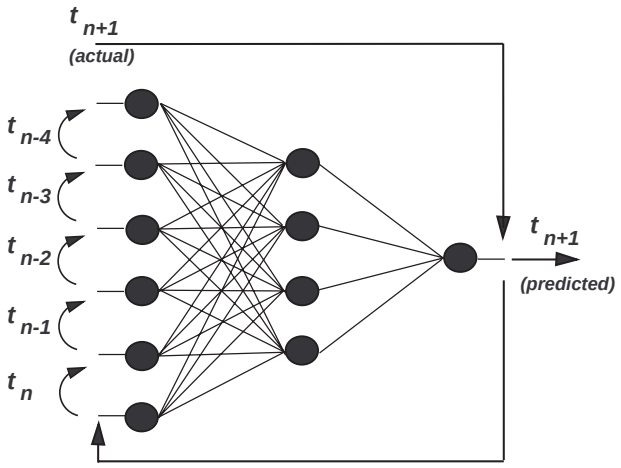

Fig. 2. Predicting neural network

3.2 Training of radial basis function networks

Radial basis function networks consist usually of three layers - first one is the input layer and is used only for normalisation purposes, second one is the hidden layer, consist of neurons with activation function:

$$F(X) = -\frac{||X - T||^2}{\sigma^2}$$

where $||X - T||$ denotes an euclidean norm, σ is deviation of the radial function, X is an input vector and T is a center of the radial function. Third layer consist of one neuron used to separate hidden layer's output vectors, and can be e.g. single neuron with linear activation function [6].

Training of RBF network can consist of two phases. First, the centers of the radial functions are set as random, or are inherited from agent's parents and the deviations of the functions are calculated [6]. Second, the output layer of the network may be supervisory trained (during agent's life), using simple steepest gradient descent method, basing on the comparison between values predicted and received as an error measure (compare with [3]).

This way of training RBF networks was considered as a very natural to implement in evolutionary systems, because the task which is very important for the training process, i.e. initialisation of radial function centers, can be performed in the evolutionary process.

Other variations of RBF and training algorithms are considered, e.g. three-layer RBF with the sigmoid-activated neuron in the output layer, or four-layer RBF with the layer of sigmoid-activated neurons in the third and forth layer, trained using backpropagation algorithm.

3.3 Evolution of neural networks

Even though the use of neural networks allows to perform many difficult tasks with a little or even without problem-dependent knowledge, one still needs to define the suitable network — its topology (number of layers, number and type of neurons in each layer, structure of connections), initial state and training algorithm. The constructed neural network should have enough complex structure to be able to solve the problem, at the same time this structure should not be too complex to avoid overfitting the data.

Due to a huge number of possible network configurations, manual process of refining the neural network architecture requires many experiments and may not necessarily lead to the best solution. That is why many techniques were invented that allow for automatic construction of a neural network that fulfils the requirements of the problem. The alternative approach are evolutionary neural networks (ENN), in which the search for a desirable neural network is made by an evolutionary algorithm [9, 13].

In order to encode the structure of the network into a chromosome, several questions should be answered. Is the exact image of structure to be encoded in this genotype? Or maybe it will be enough if a general outline is encoded? Answering these questions leads to two main kinds of genotypic representation of a neural network:

- direct encoding (strong specification scheme) - a chromosome contains complete information about the structure of NN, it requires little effort do decode it, i.e. transformation of genotype into a phenotype is trivial (an example of such an encoding is a connection matrix that precisely and directly specifies the architecture of corresponding neural network),
- indirect encoding (weak specification scheme) - there are some rules of creating a neural network encoded in the chromosome, it requires a considerable decoding effort in the construction of a phenotype (an example of such an encoding is one that uses rewrite rules to specify a set of construction rules that are recursively applied to yield the phenotype [7]).

Thus during the evolution process not only connections of NN can be evolved. The same process can operate on weights between neurons, but it is much easier to evolve the structure of connections, leaving the search for the values of weights to the network training algorithm. In direct encoding longer chromosomes are needed, and indirect encoding suffers from noisy fitness evaluation [13].

4 Neural EMAS for time-series prediction

The configuration of the agents in a predicting MAS (kind of specialisation or method of co-operation) is often difficult to specify. What is more, when dynamic changes of the characteristics of the signal are possible, the configuration of the agents should reflect these changes, automatically adapting to the new characteristics. The mechanisms of evolution may help to transform the whole population of agents (by means of mutation and/or recombination) so as it fits best current profile of the input signal (proper selection/reproduction) – this evolutionary development of predicting MAS meets the general idea of an evolutionary agent system (EMAS).

4.1 Evolutionary Multi-Agent Systems

The key idea of EMAS is the incorporation of evolutionary processes into a multi-agent system (MAS) at a population level. It means that besides interaction mechanisms typical for MAS (such as communication) the agents are able to reproduce (generate new agents) and may die (be eliminated from the system). A decisive factor of agent's activity is its fitness, expressed by amount of the possessed non-renewable resource called life energy. Selection is realised in such a way that agents with high energy are more likely to reproduce, while low energy increases possibility of death.

In EMAS training of a neural network may be entrusted to an agent which can operate autonomously – this way various agents manage their networks (e.g. train them), simultaneously with the process of evolution occuring in the whole population. Such system performs not only search for the optimal neural network structure, but also exhibits collective intelligence at agent population level since agents are able to cooperate providing even better solutions to the given problem.

In fact, each agent simply possesses some vector of parameters, which describes its behaviour in the system. This vector plays role of agent's genotype, and as such may be modified by genetic operators when inherited by its offspring. The evaluation of the agents is based on the quality of prediction by means of gained/lost life energy.

4.2 A population of agents as a dynamic modular neural network

Predictive Modular Neural Network (PREMONN) is a group of the neural networks, which solve the same problem, and their responses are combined together to yield the final result [10]. In a neural multi-agent system every agent contains a neural network, which acts as a computational model for the given problem. Entrusting the task of solving specific problem to the complex system, to determine the most accurate answer for the problem, it seems natural to assign to every of the agents the probability of the correct answer, using the credit function.

Especially in the predicting neural multi agent system every agent containing the neural network is granted a probability of succesful prediction, based on the PREMONN model [10]. Thus a group of agents can produce the answer, which will be more accurate than the prediction of one arbitrarily chosen agent, as it comes from the group of agents using at least simple voting, or a more sophisticated bayesian stochastic scheme.

The system which is constructed in this way is also adaptive, its adaptation abilities base on the stochastic features. The probabilities of the correct answer of the agents are dynamically changed by the gating expert, so in the group of the agents, the answer of the whole group as the (somehow) weighted answer of every agent, is reliable, and the agents which produce worse answers should be replaced with new agents, in this way, globally, the system can adapt to the new features of the environment.

4.3 Evolving collective intelligence of agent populations

In a complex agents' population, acting as predictive modular neural network, it will be difficult to determine correct values of specific parameters, on which depends way

of managing this population. With help may come as usual in situations concerning multiple criteria optimisation, evolutionary computation methods.

In an EMAS, evolution usually is performed at the level of individual agents [3] (as every of them contains chromosome – encoded neural network structure, thus can be reproduced, mutated, inherited etc.), but, perceiving the whole system as a group of modular neural networks (groups of agents, every group can be perceived as a single being), it seems natural to propose the way of evolving such ,,complex beings". Every modular network must have its own parameters, characterising its very behaviour, such as parameters of credit function, and the parameters defining the way of evolving individual agents (amount of rewards and punishments [3]), which can be subject of the evolution process.

Such two-level evolution can lead to automatic determination of the system parameters, making it more reliable and adaptive to the changes of the work conditions (e.g. in time series prediction, to the changes of the predicting time series).

5 Experimental results

The neural networks used in the system were radial basis function networks with three layers, the centers of the radial function were selected randomly at the beginning, then the synaptic weights of the output neurons were changed with use of the simple steepest descent algorithm.

The chromosome consisted of the number of neurons in the input and hidden layers and the centers of the RBF activation functions. During the process of reproduction, children derive mentioned parameters from its parents. Child's genotype is a result of the crossing over and mutating the genotypes of its parents.

The agents acted individually, and everyone of them was rewarded for his prediction results (his life energy was changed appropriate to his work effects).

In conducted experiments as the prediction data to the system were shipped simple sinusoidal time series, which period was 10 steps of system's work and the signal range was from $0,1$ to $0,9$.

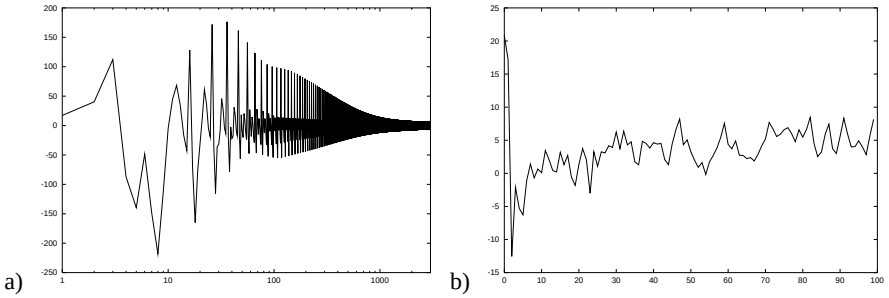

Fig. 3. Prediction error of a single agent (a) and average prediction error of the population (b)

In the first graph (fig. 3)a a percent error of the prediction can be seen. To expose the process of the network learning, in the x-axis of the plot was applied logarithmic scale. It can be seen that the neural network which belongs to a specific agents learns to predict specific time series, because the percent error of the prediction decreases. This process is naturally faster at the beginning of the process, and slows down after few hundreds of prediction steps. Thus it can be said that the networks used to predict time series were properly constructed.

In the second graph (fig. 3)b an average percent prediction error of the whole population of agents can be seen. This error changes very fast at the beginning of the evolution process, then, after several hundreds of steps it begins to stabilise, because many agents with better abilities than its parents were created. The population of the agents seem also to be stable, as it can be seen in the third graph (fig. 4)a, where count of agents in system is presented. It is to notify that (similar to average prediction error) count of agents at the beginning of the evolution changes very fast, then begins to stabilise.

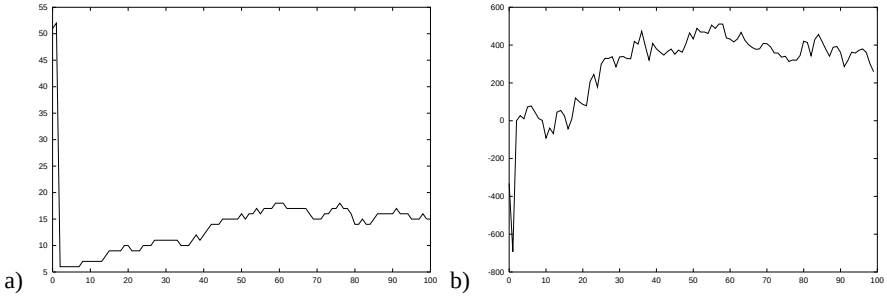

Fig. 4. Number of agents (a) and average agent energy (b) in the population

As the processes of evolution and death are based on the life energy of agents, it can be seen in the fourth graph 4b, that amount of this resource is also stable during work of the system, what proves that mechanisms of giving agents prices and penalties was satisfactory.

6 Conclusions

Evolutionary search for parameters of neural networks can lead to obtaining fast and better results than these designed by a man from scratch. Automatic processes of neural network optimisation can lead to finding the best, or at least better than others solutions for given tasks. Obtaining results from the evolutionary multi-agent system perceived as a collective intelligence can be formalised and conducted in such manner that these responses can be much more reliable than taking only the best (as it may seem) response at a time. Two-level evolution conducted in EMAS can lead to new methods of solving complex problems, changing not only characteristics of the individuals, but also the features of the environment, where evolution is performed.

References

1. E. Alpaydin. Techniques for combining multiple learners. In *International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*. ICSC Academic Press, 1998.
2. P. Bonissone. Soft computing: the convergence of emerging reasoning technologies. *Soft Computing*, 1(1):6–18, 1997.
3. A. Byrski, M. Kisiel-Dorohinicki, and E. Nawarecki. Agent-based evolution of neural network architecture. In M. Hamza, editor, *Proc. of the IASTED Int. Symp.: Applied Informatics*. IASTED/ACTA Press, 2002.
4. K. Cetnarowicz, M. Kisiel-Dorohinicki, and E. Nawarecki. The application of evolution process in multi-agent world (MAW) to the prediction system. In Tokoro [12].
5. J. Gonzalez, I. Rojas, H. Pomares, and J. Ortega. Rbf neural networks, multiobjective optimization and time series forecasting. In J. Mira and A. Prieto, editors, *Connectionist Models of Neurons, Learning Processes and Artificial Intelligence*, volume 2084 of *Lecture Notes in Computer Science*, 2001.
6. S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall, 1999.
7. H. Kitano. Designing neural network using genetic algorithm with graph generation system. *Complex Systems*, pages 461–476, 1990.
8. T. Masters. *Neural, Novel and Hybrid Algorithms for Time Series Prediction*. John Wiley and Sons, 1995.
9. M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.
10. V. Petridis and A. Kehagias. *Predictive Modular Neural Networks – Application to Time Series*. Kluwer Academic Publishers, 1998.
11. J. Sjoberg, H. Hjalmarsson, and L. Ljung. Neural networks in system identification. In M. Blanke and T. Soderstrom, editors, *Proc. of 10th IFAC Symposium on System Identification (SYSID'94)*, volume 2, 1994.
12. M. Tokoro, editor. *Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96)*. AAAI Press, 1996.
13. X. Yao and Y. Liu. Evolving artificial neural networks through evolutionary programming. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Evolutionary Programming V: Proc. of the 5th Annual Conf. on Evolutionary Programming*. MIT Press, 1996.