An Investigation of Artificial Neural Network Architectures in Artificial Life Implementations

Serkan Erdur, Tunga Güngör

Department of Computer Engineering
Boğaziçi University
Bebek, İstanbul, 34342, TURKEY

Abstract. In this paper, an artificially created world is defined and simulation results are presented. The proposed world is a complex system consisting of three types of agent: carnivorous, herbivorous, and plants. Agents live on a two-dimensional hypothetical world and have artificial neural network brains, which can learn over a life span and evolve over generations by genetic algorithms. Three senses, vision, smell, and hearing are implemented. A new version of Hebbian rule for short-term learning is defined. Each parameter was tested with numerous simulations and important guidelines were obtained that can direct the design of such artificial worlds.

1 Introduction

Living systems have always attracted the scientific studies on them. These studies have generally dealt with real biological systems. But recently, investigations of biological systems started to be done with artificial systems.

These studies include both the investigations of the real biological systems and also the creation of new artificial life forms which resemble the real living things. These two purposes are both very important. Better understanding of real living systems can increase our ability to overcome the problems that may arise from these systems. And creating life artificially may have many philosophical and psychological consequences. This can lead us to the answer of the question "what is life?"

A good example of an artificial life system is AntFarm [1]. AntFarm is a tool to investigate the evolution of complex behaviours in complex environments. It is used to examine how different designs for the ants' Artificial Neural Network (ANN) brains and different ways of mating them affect foraging ability. Another system is PolyWorld [2], which attempts to bring together all the principle components of real living systems into a single artificial living system. Some of the other studies that use ANN as the decision mechanism of the agents with supervised or unsupervised learning techniques are [3,4,5].

In this study, an artificial world (a computer program) in which an artificial life can emerge was designed and built. Although there is not a consensus on the meaning of life, in this study it is accepted that an artificial life is achieved when the world reaches a stable state that agents keep continuing their existence for approximately infinitely. This artificial world resembles real ecological systems by many aspects. Sensing, learning, moving, mating, eating, fighting and evolving are all included in this world.

A series of experiments were done on this artificial world. One of the biggest aspects of the simulations was to test different artificial neural network structures as the controlling structures of the agents. These experiments have showed the effects of the parameters of the artificial neural networks to the learning process. After giving an overview of the world and describing its internal operation, we will present the results of these experiments and comment on these results.

2 Overview of the Artificial Life World

The Artificial Life World (ALW) designed in this research is a complex system consisting of agents (carnivorous, herbivorous) and plants. Agents and plants live on a two-dimensional hypothetical world. Plants are randomly scattered around the world and emerge at a random point on the world when they are eaten or died because of decay. Agents try to continue their life by eating and mating. They use vision, smell and sound as input to their artificial neural network brains, which utilize Hebbian learning [6] at its connection weights. The output of the brain determines the behavior of the agent. Vision information is produced by constructing a pixel map of the world from the agent's point of view by considering the direction and the view angle of that organism.

Agents have energy levels. This energy decreases with each action they perform (even when doing nothing). They increase their energy level by eating. Carnivorous (from now on they will be called predators) can eat herbivorous (from now on they will be called preys) and also dead predators. Preys eat only plants. When agents are died, they turn into food for predators. So, predators can kill and eat other agents for replenishing their energies.

Each agent has a chromosome and this chromosome determines the structure of that agent's artificial neural network brain. Several distinct ANN encoding schemes are developed and used in this study and some of them store the weights of the ANN in the chromosome and some not. But all of them fully specify the architecture of the neural network.

Agents, which have more than a certain energy level, can mate. If two agents have this ability and they are close enough to each other and also if both of them want to mate, they mate. Reproduction occurs by taking the genetic material from the two parent agents, subjecting them to crossover and mutation, and then expressing the new chromosome (genes) as a child agent.

The genes pass to the further generations by reproduction. And this is done in a similar way to the nature. Agents have to find their mates themselves. And for an agent to find a mate successfully, it must travel and search for a mate. To do this, it must have energy and to gain energy, it must eat something. To eat something, it must search for food. And if that agent performs these actions, then it can be considered as an enough fit agent to be selected as a parent. This also describes the nature's way of selecting the parents to produce children. No fitness function is used to manually mate the fittest agents. But only for some specified period of time, a minimum number of agents are preserved according to a fitness function. Until this period ends, if the number of agents decreases too much that a lower limit is exceeded, a new agent is selected from living agents of the world and a clone of it is inserted to the world at a random location. This is because when a world starts, all of its agents are

created from scratch with random neural network architectures and random weights. And most of these agents are not capable of surviving and even approximately half of them cannot move until they die. So choosing agents, which are more capable of surviving, at the beginning of the simulation increases the chance of reaching successful breeds. This fitness function rewards agents for eating, mating and moving, with more points to eating and mating.

3 Genetics

The brains of the agents are feed-forward artificial neural networks. The architecture uses an encoding schema that encodes both the existence and the weights of the connections between neurons. Weights are stored as real numbers. If a connection exists between two neurons, then its weight is different than zero. A zero value in the matrix and chromosome means that a connection does not exist between two neurons.

The construction of a chromosome is done by taking the columns of the connection matrix and appending them one after another. The reason of adding columns together instead of rows is to keep all the connections to a neuron intact. This gives us an advantage when new chromosomes are produced from this chromosome by using crossover. With this structure, we can group these connections as functional units and can preserve their structure while performing crossover operation on them. Crossover operator does not divide the chromosome at these units, so the connections of a neuron stay intact after a crossover.

With this structure, successful parents can transfer their connection weights as well as connection architecture to their children. An example ANN and its corresponding matrix representation are given in Figures 1 and 2, respectively.

4 Neural Systems and Learning

Each agent has a neural network brain that gets its main inputs from the agent's vision, smell and hearing information. Another input is a normalized value of agent's current energy level. This input may help the agent to plan its movements. If it has plenty of energy, it may move to far points or mate with another agent. Some simulations used the outputs of the network at the previous step as the inputs at the current step to utilize short-term memory.

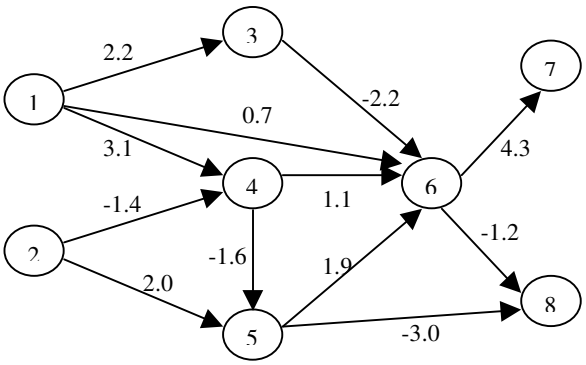

Figure 1. A sample feed-forward ANN with real valued weights

Neurons		3	4	5	6	7	8
Input	**1**	2.2	3.1	0	0.7	0	0
	2	0	-1.4	2.0	0	0	0
3		0	0	0	-2.2	0	0
4		0	0	-1.6	1.1	0	0
5		0	0	0	1.9	0	-3.0
6		0	0	0	0	4.3	1.2

Figure 2. Connection matrix representing the ANN of Figure 1

There are five outputs of the brain that determine the primitive behaviors of the agent. These behaviors are eating, mating, fighting, moving and waiting. The outputs are real numbers changing between zero and one.

There are 32 input neurons; 15 of them are the visual information, 12 are smell information, 4 of them are hearing information and 1 is the current energy level of the agent as stated before. Optionally, there can be six more inputs that come from the outputs of the brain calculated at one step before the current step. The connections of the internal and output neurons to the input neurons are determined at the genes of the agent as specified in Section 3. The connections and their weights are determined randomly when the agent is created. There is no predefined functionality of internal neurons and connections. Their functionality is determined by the evolution.

Bias values of each neuron change during the simulation with the change of weights. Biases are connected to the neuron as if they are a synaptic connection from another neuron. This way, bias values can be updated using the same learning technique of updating the weights.

When simulation runs, at each time step, input neurons are set to the values of the sensing information and agent's current energy level. Activation values of the neurons are calculated according to the following formula:

$$y(i) = \sum_j a_j{}^t * w_{ji} \tag{4.1}$$

$$a_i{}^{t+1} = 1/\left(1 + e^{-\beta * y(i)}\right) \tag{4.2}$$

where a_j^t is the activation value of neuron j at time t, w_{ji} is the weight of the connection from neuron j to neuron i at time t, a_i^{t+1} is the activation value of neuron i at time t+1, and β is the logistic slope. The second formula is the well-known sigmoid function.

Brain can learn in two forms: short-term learning and long-term learning. Long-term learning is the learning by evolution. It is developed by natural selection as the genes of more fit agents are carried to the next generations. Short-term learning is the learning of an

individual agent for the duration of its life. Hebbian learning [6] is adopted for short-term learning in this work. Weights (and bias) are updated by a Hebbian learning rule as in the PolyWorld [2] and Linsker's work [7,8,9]. Linsker in his work demonstrated that Hebbian learning can self-organize important types of neural response patterns observed in early visual systems of real organisms. Hebbian learning is an unsupervised learning technique. This means that no external judgment is done for the outputs of the ANN and no external help is provided to the ANN to change its outputs. In this work, it must be in this way because in real life most of the times organisms must learn without a supervisor.

Hebbian learning adjusts the network' s weights (synaptic efficacies) such that its output reflects its familiarity with an input. The more probable an input, the larger the output will become (on the average). It makes the network respond to the same inputs with the same outputs. And this makes the network more stable.

We can define the Hebbian learning rule as follows: If a_j is the output of the presynaptic neuron, a_i the output of the postsynaptic neuron, and w_{ij} the strength of the connection between them, and LR learning rate, then according to the Hebbian learning rule:

$$w_{ij}(t+1) = w_{ij}(t) + LR * a_j * a_i \tag{4.3}$$

This weight update occurs for each neuron at each time step that network calculates its outputs. Here a_i and a_j can be positive or negative. If they carry different multiplicative signs then weight will be decreased so that the effect of presynaptic neuron on the postsynaptic neuron will be decreased. If they carry the same multiplicative sign, then the weight of the connection will be increased so that the effect of presynaptic neuron on the postsynaptic neuron will be increased.

In this paper, we define a new version of Hebbian rule:

$$w_{ij}{}^{t+1} = w_{ij}{}^{t} + LR * (a_j{}^{t+1} - a_i{}^{t}) \tag{4.4}$$

where w_{ij}^{t+1} stands for the weight of the connection from neuron i to neuron j at time step t+1. a_j^{t+1} is the activation value of the neuron j at time step t+1 and a_i^t is the activation value of the neuron i at time step t. LR is the learning rate constant defined globally for the simulation. Weights of the network are updated according to this rule at every step.

When an agent is born, its chromosome is constructed by using its parent's chromosomes. If the connection weights, which are updated during the parent's lifetime, are used to construct the child's weights, then the child can learn what its parents have learnt up to that time. This kind of learning is called Lamarckian learning and it is used as long-term learning in this work.

5 Results

The simulations first run for a definite number of steps (time steps) in a controlled period of time. During this controlled period, when a type of agent population size decreases below a predetermined value, a new agent of that type is inserted into the world. This agent is a mutated copy of another agent chosen among the living agents of the world. An elitist strategy is used here such that the fittest agent according to the fitness function is chosen.

But that agent's genes are changed slightly with the mutation operator. The mutation operator is the one used for the reproduction process. This controlled period is usually set as 4,000 time steps. It is experimented that generally this much time is sufficient for a simulation to produce fit enough agents to continue the simulation successfully. Successfully here means that agents are able to sustain the population's existence. After that controlled period, the agents are expected to keep the population size at a certain level by sustaining their lifes and producing new agents by mating for the place of dead agents.

The fitness function mentioned above is a function that rates the agents for their actions. In each step of the simulation, agents are assigned their fitness scores. This score is updated at each step according to the action done by the agent.

For observing the effects of ANN parameters to the success of the simulations, we performed a series of simulations for each parameter. At each simulation series, all the parameters were kept constant except the one, which is being observed. A range of values was selected for each parameter. These ranges were chosen to span all the possible values that the parameter can take. Each parameter value was tested at least 10 times. This number was increased over 40 for successful series to observe the effects more accurately. Simulations were allowed to run a maximum of 20,000 time steps. As stated above, 4,000 steps of this 20,000 time steps formed the controlled simulation time. In the following sections, graphs are presented for showing the effects of the parameters to the success of the simulation. A simulation is accepted as successful if the agents are able to sustain the population's size at a constant level.

5.1. Number of Neurons

When the number of neurons is too low, the ANN will be incapable of learning patterns that make it survive. If the number is too high, ANN will be able to memorize all the examples by forming a large database, but will not generalize well to inputs that have not been seen before. The results that we have obtained from the simulations are fully complying with this knowledge. Figure 3 shows the results of the experiments on the number of neurons. It presents the success percentage of the simulations with the change of number of neurons.

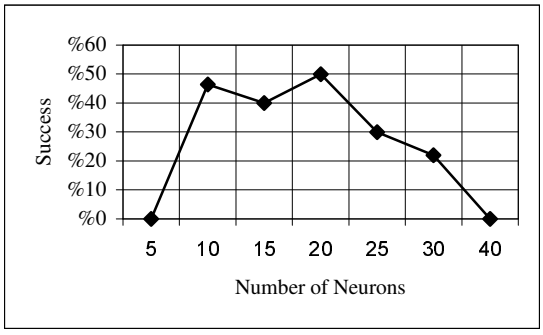

Figure 3. "Number of neurons" vs. "Success"

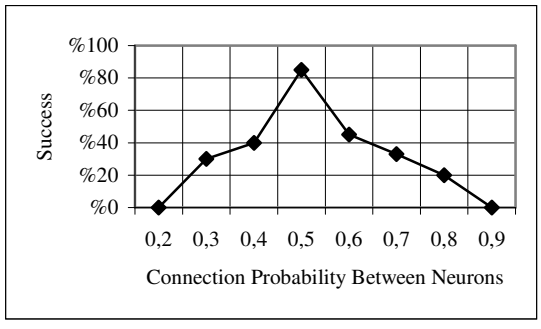

Figure 4. "Connection probability between neurons" vs. "Success"

5.2. Connection Probability Between Neurons

This parameter is used when the ANN brains of the agents are constructed. With the probability dictated by this parameter, existence of a connection between two neurons is determined. If the probability is defined high, then the neurons of the ANN will be more connected.

Results of the experiments on this parameter are shown in Figure 4. This parameter's effect is like the effect of the number of internal neurons. When connection probability is too low, then ANN cannot learn the patterns; if it is too high, then it may memorize the input-output pairs instead of learning patterns.

5.3. Hebbian Learning Rate

Learning rate parameter is used in the learning function as described in Section 4. Learning function is used to strengthen the weights of the connections in the appropriate direction. If the learning rate is high, then the network learns the input pattern faster. If the rate is too high, the weights may be updated beyond the optimal value. If it is too small, the network cannot learn input patterns because of the differentiation of the input at each time step.

Results of experiments on this parameter are shown in Figure 5.

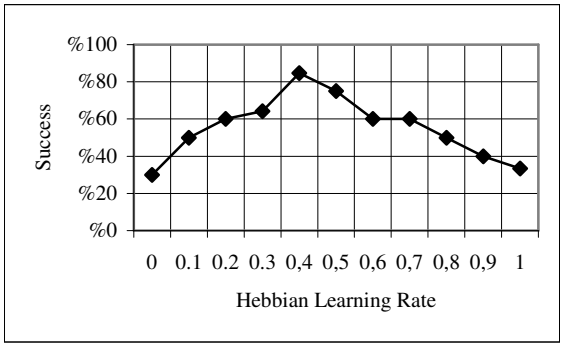

Figure 5. "Hebbian learning rate" vs. "Success"

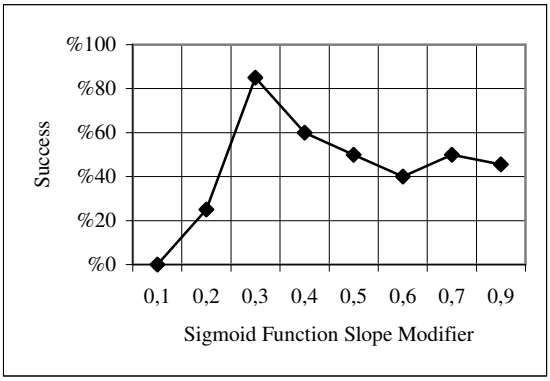

Figure 6. "Sigmoid function slope modifier" vs. "Success"

5.4. Sigmoid Function Slope Modifier

This parameter directly affects the output of the activation functions and consecutively the output of the ANN. The activation function used in the ANN is sigmoid function as stated before and the output of this function is between zero and one. If the slope modifier is small, then the slope of the sigmoid function becomes small. If the slope is small, then activation values of the neurons come closer to 0.5.

Figure 6 depicts the results of the experiments. As the results show, simulations with slope modifier values smaller than 0.2 have given worst results. Reason of this is, at such

small values, outputs of the neurons come close to 0.5 regardless of the inputs. This situation directly affects the outputs of the ANN and its behaviors become very restricted.

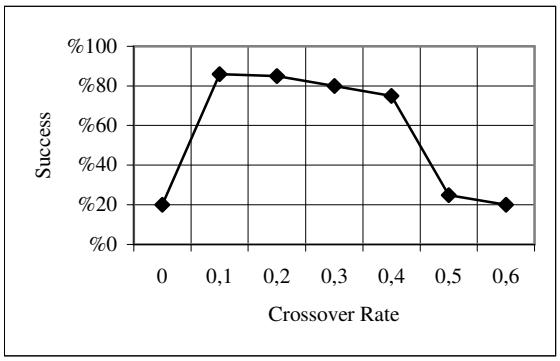

Figure 7. "Crossover rate" vs. "Success"

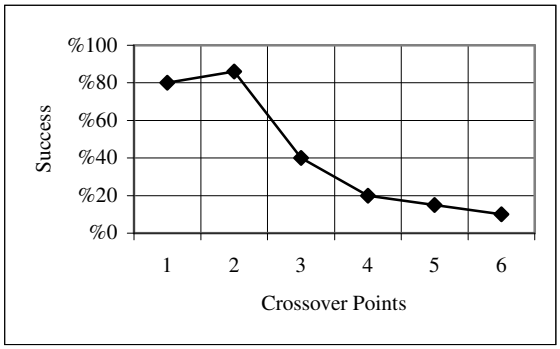

Figure 8. "Crossover points" vs. "Success"

5.5. Crossover Rate and Points

Crossover rate defines the probability of a crossover happening during a reproduction. Crossover points defines the maximum number of crossovers that can happen during a reproduction. Use of crossover can sometimes destruct the functional blocks of the ANNs.

Although in this work neurons with their input connections are considered as functional units and not destructed by the crossover operator, exact functional units of the ANN cannot be known. ANNs are a kind of black box and the internal workings of them are not known so it is best to keep crossover rate and crossover points low. Figures 7 and 8 show the results of the experiments done on crossover rate and crossover points.

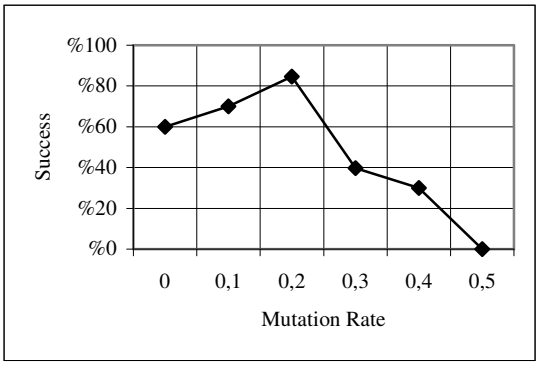

Figure 9. "Mutation rate" vs. "Success"

5.6. Mutation Rate

Mutation rate parameter is used when reproduction takes place and during the controlled simulation period when a copy of the fittest agent is inserted to the world as stated before. It determines the probability of mutating a gene. If this parameter is too high, then the child's

genes are mutated so much that it loses the characteristics of its parents. This may lead to the loss of valuable information, which has been gathered for generations. If mutation rate is so small, then the genetic discrepancy will be minimum among the individuals of the population and evolution will be too slow. Figure 9 shows the results of the experiments done on mutation rate.

6 Conclusion

In this paper, we have presented the design of an artificial life world and the results of the simulations on this world. The particular simulation results presented here show the effects of some important artificial neural network parameters on the success of the artificial life simulations. These results will hopefully aid to the researchers who work with artificial neural networks, particularly on the artificial life field.

An important extension to the work in this paper may be encoding more than just the neural architecture into the genes of the agents. For example, physical attributes of the agents can be encoded into the genes and they may evolve with the current evolution mechanism. Another important information that can be encoded into the genes is the parameters of the ANN brains. Some learning algorithms may be encoded into the genes and allow the agents evolve the most effective learning algorithm rather than assuming it to be Hebbian.

References

1. L. Yaeger, *Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or PolyWorld: Life in a New Context*, In Artificial Life III, Ed. C. Langton, Addison-Wesley. (1994)
2. R. Linsker, *An Application of the Principle of Maximum Information Preservation to Linear Systems*, In Advances in Neural Information Processing Systems 1, Ed. D.S. Touretzky, Morgan Kaufmann Pub., CA. (1989)
3. R.J. Collins and D.R. Jefferson, *AntFarm: Towards Simulated Evolution*, In Artificial Life II, Eds. C. Langton, C. Taylor, J. Farmer and S. Rasmussen, Vol. X, Addison-Wesley, CA. (1992)
4. R. Linsker, *Towards an Organizing Principle for a Layered Perceptual Network*, In Neural Information Processing Systems, Ed. D.Z. Anderson, American Institute of Physics, New York. (1988)
5. R. Linsker, Self-Organization in a Perceptual Network, *Computer* 21(3), pp. 105-117. (1988)
6. S. Nolfi, J.L. Elman and D. Parisi, *Learning and Evolution in Neural Networks*, CRL Tech. Rep. 9019, Center for Research in Language, UCSD, CA. (1990)
7. D. Parisi, S. Nolfi and F. Cecconi, *Learning, Behavior, and Evolution*, Tech. Rep. PCIA-91-14, Dept. of Cognitive Processes and Artificial Intelligence, Institute of Psychology, C.N.R.–Rome. (1991)
8. D. Chalmers, The evolution of learning: an experiment in genetic connectionism, In *Connectionist Models, Proceedings of the 1990 Summer School*, San Mateo, CA. (1991)
9. D.O. Hebb, *The Organization of Behavior*, John Wiley and Sons Inc., New York. (1949)