**Artificial neural network training through simulated natural selection**

Kyle Rocha-Brownell, California State University, Chico, CA

**Abstract**

A simulated environment set up to train Artificial Neural Networks (ANNs) through natural selection is presented. Numerous objects created with independent ANNs interact with each other, competing for simulated limited resources. A description of the process and technologies used is presented, followed with a discussion of the results.

**Motivation**

Artificial Neural Networks (ANNs) are an effective implementation of artificial intelligence which can be trained to accomplish specific tasks. The process of training an ANN usually involves providing positive feedback, often manually, when the desired output is achieved. One promising alternative avenue of research in the area of training is simulated natural selection. As long as an environment can be created in which successful ANNs thrive, and unsuccessful ones are destroyed naturally, training can be accomplished with little manual interaction. As an example of the potential of this approach, an environment was created to train basic fight-or-flight intelligence into ANNs.

**Methods**

The Java programming language is used to implement ANNs in an attempt to “evolve” intelligence within a community of AI objects, referred to as individuals, through simulated natural selection. The program output includes a visual simulation of the community as the individuals interact with each other in two dimensions. Each individual is represented as a colorful circle moving on a plane. The program imposes a set of laws on the community that affect their condition, but at the initialization of the simulation, each individual begins only with a randomly weighted ANN.

The rules of the world in which the community operates are basically the rules for simple living creatures. When multiple individuals are in contact, they will attempt to consume each other. The value of the consumption is a function of their relative color. The purpose of this is to give each individual some preference toward what they consume. They must also multiply (create a new individual) once they have consumed a certain amount. The offspring begin with an ANN weighted similarly to its parent, but with each weight having been randomly varied by a small percentage. Their color is also varied slightly from that of the parent. In theory, the individuals who have the most effective weights will survive to reproduce, and others will not. As the generations pass, the ANNs should become increasingly effective at ensuring survival.

The inputs to the ANNs are essentially the “sensory” input of each individual. They include the individual’s color and relevant data on a variable number other nearby individuals - relative position in polar coordinates and color. The output of the ANN is simply the desired direction of movement in radians.

The success of an individual will be gauged based on its ability to survive. Those who have produced the most offspring will be deemed most successful, as that indicates that they have successfully consumed many others, and likely have survived for a long period of time.

**Results**

After running for between \_\_\_ and \_\_\_ generations, the ANNs inside some individuals were consistently weighted in a manner that attempts to keep them alive. That is, they move toward other individuals that they prefer to consume and stay away from those that would consume them.

During some runs of the simulation, many individuals had the tendency to turn white, giving them no consumption value. Many also adopted the strategy of avoiding interaction with all other individuals, both consumable and not. These tendencies were successfully prevented in later runs by imposing an age limit on all individuals. However, in all cases the age limit eventually resulted in total extinction. To prevent extinction, a system was added to create new individuals with randomly weighted ANNs in order to keep the population above a certain value.

**Conclusion**

Though computationally expensive, simulated natural selection is a viable method for training ANNs where applicable. This first implementation in Java was admittedly inefficient, in that the program ran in a largely serial manner. Future implementations should be run on a parallel platform such as NVidia’s CUDA language. This would provide a much faster simulation, resulting in more generations passing in a given time.

The most challenging aspect of simulated natural selection is creating an environment that will train the ANNs in the desired manner. Though a given environment may tend to produce successful ANNs, it may only do so at a certain time. A balance must be achieved to assure that all ANNs remain consistently motivated in a way that results in proper training.