Brief Article

Cai Crumley

October 5, 2013

1 Introduction

This project conducted research in the field of neuro-evolution (NE), in which evolutionary algorithms (EAs) are applied to generate artificial neural networks (ANNs). In particular, we attempted to improve upon the NE method called Enforced Sub-Populations (ESP) by using behavioural and genotype metrics. We used a simulation of a simple predator/prey scenario as the testbed to compare ESP and its modifications with varying parameters.

An ANN is a processing structure that simulates a neural network. An ANN consists of a set of nodes or neurons with weighted, directed connections between them. ANNs can be classified as either feedforward or recurrant depending on the connections between the nodes. An ANN is classified as feedforward if there exists a way of numbering the nodes such that there are no connections from a node to another node with a lower number. If no such numbering exists then the ANN is classified as recurrant [3]. Thus a feedforward ANN can be modelled by a directed acyclic graph, whereas a recurrant ANN cannot. The architecture of feedforward networks typically consists of layers of nodes in which each node is connected to all nodes in adjacent layers, so that the outputs of a layer of nodes become the inputs to the next layer. The first layer of such an ANN corresponds to external inputs and the last layer corresponds to the outputs of the ANN. The output of a node is a function of the sum of its inputs minus the bias, or threshhold, of the node. Formally, we can express this using the following equation [3]:

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) \tag{1}$$

where y_i is the output of node i, n is the number of incoming connections, x_j is the jth input to node i, w_{ij} is the weight of the connection between node i and the node with the output correponding to x_j, and θ_i is the bias or threshhold of node i. f_i is called the transfer function and is typically a nonlinear function such as a Gaussian or sigmoid function [3].

Evolutionary algorithms are population-based stochastic search algorithms that mimic evolution in nature in order to generate a solution to a problem [3][1]. A population of genotypes that represent candidate solutions to the problem is gradually improved by repeated application of genetic operators, including crossover, mutation and selection. The evolution process is guided by a fitness function which defines the goal of the algorithm and is used to rate the fitness of the genotypes. In each generation genotypes are evaluated using the fitness function, and then genotypes that have a poor fitness are replaced with offspring created by applying the crossover and/or mutation operators to fitter genotypes. This process is repeated until some termination condition is met, such as a maximum number of generations being reached or a desired fitness level being achieved. EAs are well suited to solving high-dimensional optimisation problems that have complex fitness landscapes containing many local optima [3].

EAs can been used in conjunction with ANNs in a few different ways. They can be used to optimise the connection weights between nodes (which is known as training the ANN), and they can also be used to optimise the overall architecture of ANNs [3]. EAs can also be used together with other algorithms as part of the traning process [3]. In this project we used a predefined architecture and evolved only the connection weights.

In this project we used a predator/prey simulation similar to the one used by Yong and Miikkulainen in [4] in which 3 predators attempt to capture a single prey on a toroidal grid-based world. This project attempts to expand on the work done by Gomez and Miikkulainen in [2] and Yong and Miikkulainen in [4] by modifying the evolution process by applying crossover between populations of genotypes of different predators based on their similarity, as measured by behavioural and genotype metrics. The hypothesis of this research is that, by using inter-agent crossover appropriately, we can reduce the number of generations taken until a predetermined level of fitness is achieved.

2 Background

More text.

References

[1] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, January 2008.

[2] Faustino Gomez and Risto Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342, 1997.

[3] Xin Yao. Evolving artificial neural networks. In *Proceedings of the IEEE*, volume 87, pages 1423–1447. IEEE, 1999.

[4] Chern Han Yong and Risto Miikkulainen. Cooperative coevolution of multi-agent systems. *University of Texas at Austin, Austin, TX*, 2001.