Machine Learning - Assignment 1

4th of September 2013 Michael Ferguson (FRGMIC005)

# Learning Method

Reinforcement Learning: Neuro-Evolution

# Description

The chosen learning method implements reinforcement learning through the use of neuro-evolution. An evolutionary algorithm (EA) is used to evolve Artificial Neural Networks (ANNs) according to their fitness value which is calculated through reinforcement learning.

The evolution of neural networks is accomplished by first creating a population of genotypes. At each iteration of evolution the genotypes are decoded into an ANN which is evaluated in terms of success in the task, resulting in a fitness score for each genotype. Elitism and Survival of the Fittest genomes (Best two genomes are crossed over into to offspring's and mutated) are used to generate the next generation of genotypes.

# Motivation

Reinforcement learning seems like an ideal approach to deal with the environment of the minesweepers. The environment is initially unknown through random allocation of mines, rocks and super-mines. The minesweeper then interacts with this environment and is able to improve its policy.

The framework has been set up in a way which supports the implementation of reinforcement learning. On each iteration the environment is randomly allocated, thus the minesweepers policy can improve on each iteration.

Neuro-evolution is used to accomplish this policy improvement by evolving the ANN with an EA. Neuro-evolution in this instance can be seen as a policy search method to reinforce desired behaviours from the minesweepers.

The primary motivation for neuro-evolution is the ability to train ANNs in sequential decision tasks whereby the reinforcement information is sparse. Thus it allows the possibility to find an ANN that is designed to optimize behaviour given such sparse reinforcement information without the need to explicitly know what they should be doing.

Using a supervised learning algorithm would have been difficult to successfully create input-output pairs. Neuro-evolution allows an easy measure of a networks performance at a task.

Neuro-evolution allows for easy behavioural adaptations, by customising the conditions whereby fitness is incremented or decremented. Thus this approach can easily be adapted for mine gathering.

# Pseudocode

1. For each Minesweeper and for iNumTick iterations.

- Call the Update function.

- Add inputs to ANN (Vector to closest mine and lookAt Vector).

- CalculateOutput by giving the ANN the input Vectors.

- Sum weights of neuron multiplied by weights from input.

- Filter the new weights using sigmoid activation function and add them to the output.

- Assign the output to the left and right Tracks of the Minesweeper.

- Assign the output to the Minesweeper's steering force.

- Assign the output to the Minesweeper's speed (acceleration).

- Increment or decrement the Minesweeper’s fitness score accordingly.

- Increment fitness score if Minesweeper does not turn too drastically.

- Decrement fitness score if Minesweeper hits a mine or a rock

2. Get the weights from the minesweeper’s ANN. (vector)

3. Evolve the minesweeper's ANN using neuro-evolution to produce a new population vector of weights.

- Elitism and Survival of the fittest are used to produce an evolved population from the chromosomes fitness scores.

- Elitism - The four strongest chromosomes (highest fitness's) are added to the new population.

- Survival of the fittest - Two strongest chromosomes are crossed-over to form 2 offspring's which are then mutated. This is repeated until the chromosome population is the same length as the old one.

4. Replace the minesweepers ANN with the evolved ANN.

- Replace weights.

5. Repeat cycle - go back to #1.

- Increment iteration counter.

- Undergo another evolution cycle.