COMP-424: Artificial intelligence

Homework 4

Due in class Tuesday March 24.

In this assignment, you will train a neural network to learn to predict the outcome (WIN or LOOSE) of a game of Breakthrough. Most of the code has been provided. You do not need to re-use any code from assignment 1 to do this. You will need the initial Breakthrough and Boardgame directories provided. You can re-download them from the course website if necessary.

In addition, you will also need the following files:
BTNeuralNetPlayer.java -- A Neural Net player
NeuralNet.java -- Generic code implementing the backpropagation class (the implementation is not complete – see Question 1 below.)
NetTrainer.java -- A training program to automatically run the network through many training steps. To train the network, it runs a game between two heuristic players and keeps all the boards seen in the game. It then uses the outcome of the games with each of the visited board positions to train the network.
BTHeuristicPlayer.java -- A heuristic player, against which your neural net will learn (the BTFixedPlayer used in Homework 1 is not good enough to learn anything from.)

You can also download these files from: www.cs.mcgill.ca/~jpineau/comp424/schedule.html (see lecture 18).

Copy these files to your project's main directory (not in breakthrough/ or in boardgame/).
Compile them:
```
% javac *.java
```

To train the neural network:
(1) Launch the training program.
```
% java NetTrainer <numGames>
```

The default value for <numGames> is 10^5, in which case training takes ~30 sec.
When you run NetTrainer, the following two output files are written.
The network is saved in 'network.txt'.
The weights after every 100 training examples are stored in the file 'train.csv'.

To test your neural network player:
(1) Launch a server:
```
% java boardgame.Server &
```

(2) Launch the NeuralNet player:
```
% java boardgame.Client BTNeuralNetPlayer
```

(3) Launch the Heuristic player:
```
% java boardgame.Client BTHeuristicPlayer
```

Observe the outcome of the game.

Note that BTNeuralNetPlayer evaluates all next-step board configurations, and picks the one with the highest activation output (= strongest prediction of a win); it does no search.

Question 1 [20 points]:

Write the code for doing backpropagation in the Neural Network. Most of the support code (including data structures) has been written. You only need to edit a section of NeuralNet.java, marked:

```
// ========= BEGIN SOLUTION ========= //
```

We will assume a neural network without hidden layers, so your code only needs to update weights for the output layer. Your code must compile and run correctly.

Submit your file NeuralNet.java with the implemented backpropagation via the Handin system.

Question 2 [20 points]:

Using your code from Question 1, train a neural network using 10^5 games.
Plot the weights after every 100 training examples. The necessary data is stored in the file 'train.csv'.

Test BTNeuralNetPlayer by playing 20 games against BTHeuristicPlayer, report on how many games are won by the neural net player.

Now train your neural network using 1000 games and again test BTNeuralNetPlayer by playing 20 games against BTHeuristicPlayer, report on how many games are won by the neural net player.

Comment (in words) on the results. Submit this part in writing (in class).

Question 3 [10 points]:

Currently, BTNeuralNetPlayer uses nine input features:

```
 *  0 - A random number, just for the fun of it.
 *      The net should learn to ignore this.
 *  1 - 1 if this is a winning board, 0 otherwise.
 *      This is a perfect indicator for final boards.
 *  2 - 1 if our foremost piece moved further than
 *      the opponent, 0 otherwise
 *  3 - Sum squared distances moved by my pieces,
 *      divided by 200
 *  4 - Same as 3, for the opponent
 *  5 - Number of my pieces left
 *  6 - Number of opponent pieces left
 *  7 - Maximum distance moved by any of my pieces, divided by 7
 *  8 - Maximum distance moved by any opponent piece, divided by 7
```

Can you suggest another useful feature to consider? Describe this new feature, and explain why you think it would improve performance of the neural net predictor. Submit this part in writing (in class).

Question 4 [30 points]:

Implement the new feature you suggested in Question 3 as an input to the neural network. In BTNeuralNetPlayer.java, simply replace feature 0 (the random number), by your new feature. Analyze performance of your new feature by presenting results as in Question 1 above (plot of weights, #games won after 10^5 training examples, #games won after 10^3 training examples). Comment (in words) on the results.

Submit your file BTNeuralNetPlayer.java with the implemented new feature via the Handin system.

Question 5 [20 points]:

Are the learning techniques explored in this assignment a good way to train a Breakthrough player? If yes, explain why. If no, propose a better approach. Discuss the pros and cons. Submit this part in writing (in class).