Learning Discontinuous Functions with Q-Learning, Path/Random Training and NEF Learned Connections

Peter Suma, Travis DeWolf, Xuan Choo and Terry Stewart

Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON, Canada

1 Introduction

Stewart et al 2012 present a functional model of the basal ganglia implemented using a Neural Engineering Framework (NEF) (Eliasmith and Anderson, 2003) based spiking neuron model which solves two and three arm bandit tasks and correlates well with data from rat recordings (Kim et all, 2002). Stewart's model consists of a multi-dimensional ensemble of neurons encoding competing actions connected via a learned connection to a basal ganglia model based on Shouno et al. (2009). The connection strengths to the basal ganglia model are trained using Q-learning. The Q-values serve to strengthen and weaken the connections from the stimulus to the appropriate action neuron ensembles from which the basal ganglia implementation then chooses the appropriate action to perform once training is stopped. The mapping of the current position to the best next move to make (the mapping function) is the fundamental function being learned by the weights on the action selection ensemble.

In this study we extend Stewart et al's basal ganglia and multi-dimensional action selection study of the model (herein called the "simulated rat") by providing it tasks of varying complexity in the form of simulated mazes for the model to navigate and study the effect of varying training algorithms; a-star best move method and q-learning both using random and path-ordered training sequences.

We found that; (1) accuracy was inversely proportional to the number of discontinuities (ie: maze complexity = number of turns in the maze) in the mapping function, (2) the best accuracy was achieved using QLearning training where the reward was primarily based on the A-star algorithms solution of the maze. We hypothesize this to be a result of information as well as recency effects. QLearning embeds in its reward function more and graduated information of the entire solution path into each move's reward, whereas in direct A-Star solution encoding (best move encoding) only information about the very next move in the sequence is captured in the training weights; secondly, by training in a path ordered way as opposed to random training, the recency effects well known in least-squares-based learning algorithms comes into play and aids in the encoding weights which more accurately capture the discontinuities of the mapping function.

2 Implementation

2.1 The Experiment

Stewart et al present the NEF theory and an overview of Q-Learning for action selection in the NEF implementation with NENGO of their basal ganglia model. We refer the reader to their paper for the relevant background, here we will explain the specific tests we performed on our implementation of their model and our extensions of it.

The model's environment is a 10 by 10 cell maze (the yellow, blue and red inset in Figure 1). The rat performs a series of decisions from a starting cell (cell [0,0]) to an end cell where a fictional cheese reward

is located (cell [9,9]). At the end cell the rat has to choose the correct lever (right or left) to press to attain the “cheese” reward and conclude the simulation.

The solution to each maze is a discontinuous function composed of the path to solve the maze for our “rat” composed of basal ganglia, thalamus and learned connection NENGO models which encode the mapping between the various possible actions that the basal ganglia could choose in each given state of the maze being solved. The model has to be trained on each new maze under each training method and then the resulting accuracy of the rat’s ability to solve the maze was measured.

The mazes are presented to the model in increasing complexity modelled by the number of walls. The increasing number of walls force commensurate directional changes and hence discontinuities of the solution mapping of current position to best next move toward the cheese.

The solutions to the mazes are learned into the NEF connection weights matrix over the dimensions of an action selection ensemble connected to the basal ganglia. Various training methods were used to generate the inputs to the learned connection from which the weights are computed by the NEF along the multi-dimensional action representation ensemble’s connection to Stewart et al’s basal ganglia model. The accuracy of the training method was measured by how reliable the encoding of mapping function was compared to the solution trained on.

To analyze the said accuracy of connection weights to learn varying degrees of discontinuous functions using various training methods; path training versus random training, with and without Q-learning and maze complexity;

(1) Training type (random ordering or path ordered), whereby the order of the training sets are presented to the model either randomly (a maze cell is chosen at random and the best move to make is also set on the action selection ensemble and the NEF then computes (“learns”) the weights to solve for that case; or path learning whereby the training cells are chosen in path order starting with the longest paths first and ending with the shortest paths, in all cases from each starting cell the optimal path is trained on only.

(2) Q-learning or without Q-Learning. All mazes were first solved using the A-Star algorithm to produce the training set of best moves for each position in the maze which were computed by solving for the optimal path from each position to the cheese cell and then taking the optional path’s first move from those solutions to form the best next move training set by position. QLearning is used in one case to compute the relative weights to set the action selection ensemble’s dimensions with thereby encoding the relative value of the moves all the way to the end of the maze into the action selection weights. Without QLearning the action selection weights are determined directly from the A-Star algorithm’s best next moves matrix.

(3) Maze complexity, (from simple (3 discontinuities), easy (26 discontinuities), medium (50 discontinuities) and hard (61 discontinuities).

2.2 The Circuit

The model is composed of the following ensembles;

(1) The 6-dimensional ActionValues ensemble holds the relative value of each possible action from which the basal ganglia chooses the one with the maximal value at each time.

(2) The 2-dimensional PlaceEnsemble encodes the (x,y) coordinates of the current position of the rat.
(3) The NextMoveError ensemble is a 6-dimensional ensemble which hosts the learned connection weights modifying the relative values of each action in the PlaceEnsemble, the result "voting" for the action to be chosen.
(4) The LearnSimpleNodeEnsemble implements the training and self-navigation logic which drives the model. There are four major classes which implement the required components;

a. AStarPathFinder which builds the maze of specified size and complexity and then solves the maze using the A-Star path finding algorithm.

b. BestNextMoves which takes an AStarPathFinder class maze and series of paths which solve the maze from all starting cells and then condenses this to the best set of next moves for each cell in the maze and produces a six dimensional vector of 0's and 1's which encodes for each position in the maze the best action to take. There are 6 actions; [Up, Down, Left, Right, LeftLeverPull, RightLeverPull].

c. QLearn_BestNextMoves which implements the QLearn algorithm over the maze solved by AStarPathFinder and then scores each action at each cell with a Q-value which rewards the best next optimal move the most and the least for any move which produces an invalid outcome (moving into a wall or out of the maze); while all other valid moves are scored somewhere in between relative to the path distance improvement or degradation from the current state after the move would have been made. The highest rewards result in Q-values of 1,000 or more and the least with Q-values equal to zero. The Q-values are computed over 1,000's of iterations on each action for each cell (approximately 10,000 or so for each). The action to score are chosen per the Q-value algorithm in a partial random order which results in slightly varying results on each iteration.

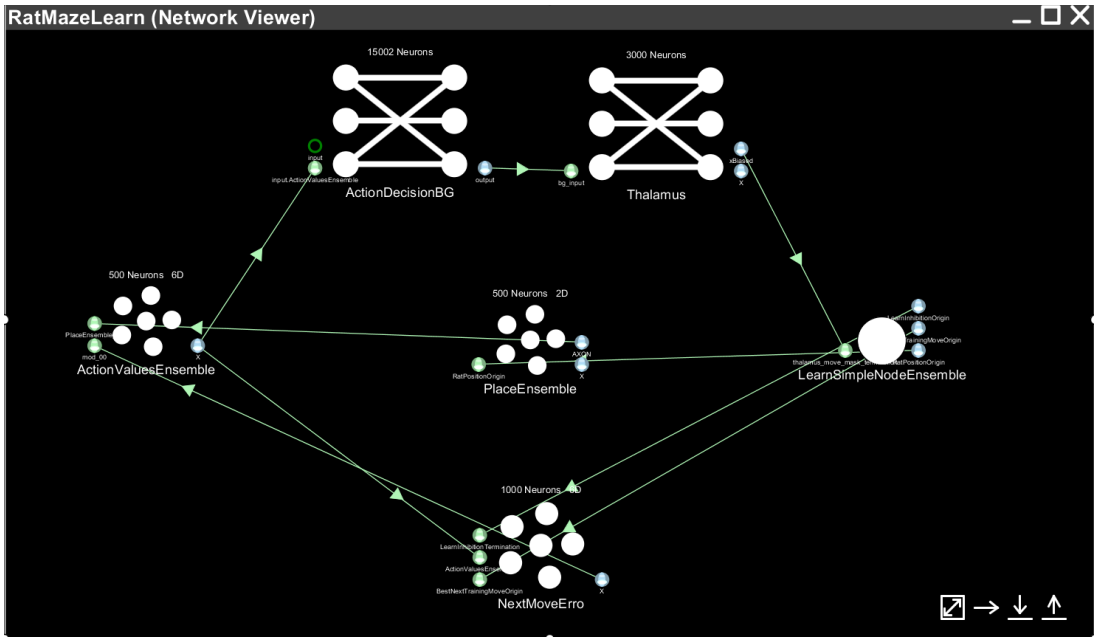


Figure 2: Nengo model showing all ensembles and connections.

The model runs in several stages;

(1) NENGO loads the model and generates all the ensembles with their randomly generated neuron tuning curves and the corresponding NEF encoders and decoders. We have chosen to use 1,000 neuron ensembles for the 6-dimensional NextMoveError ensemble which holds the learned connection and its learned weights and 500 neuron ensembles for the ActionValues ensemble and the PlaceEnsemble.

(2) The TrainSimpleNode ensemble then constructs the requested maze and its solutions preparing for training of the learned connection. Once the run function is called from the interactive mode or in batch mode, the TrainSimpleNode's tick() function is called by Nengo each instant of simulated time and one of three modes are selected per timestep;

    a. Training Mode, wherein the value of the ActionSelection and the PlaceEnsembles are set to the correct pair from the training algorithm chosen for this run (e.g.: QLearning with Path Training) and repeated until all training pairs are trained on. The system then waits a configurable period of time (75 ms in our experiments) for the network to settle before introducing another training pair.

    b. Mapping Mode, where the learned connection's answers as to the best next move for each cell in the maze are polled and the resulting training method is therefore mapped and stored for ranking. The result of the mapping mode is displayed as a series of graphs at the bottom of the maze display, the first is the training result and if Q-learning was used, the bottom one shows how well Q-learning's training set compared to the A-star solution (to make sure the training set was correct, this graph should always have 100% accuracy as it does in the Figure).

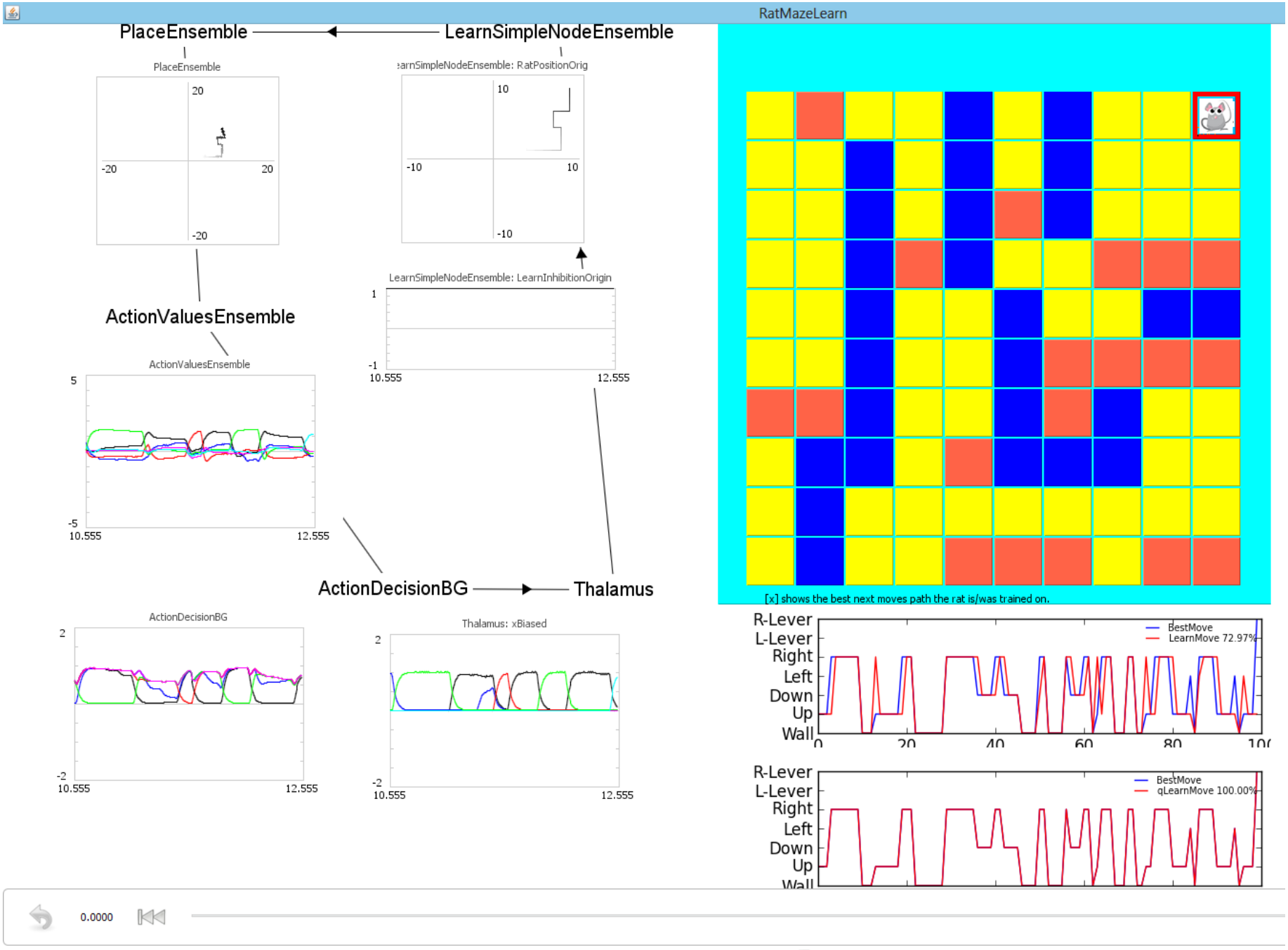

RatMazeLearn
PlaceEnsemble
LearnSimpleNodeEnsemble
PlaceEnsemble
20
-20
20
-20
LearnSimpleNodeEnsemble: RatPositionOrig
10
-10
10
-10
LearnSimpleNodeEnsemble: LearnInhibitionOrigin
1
-1
10.555
12.555
ActionValuesEnsemble
ActionValuesEnsemble
5
-5
10.555
12.555
ActionDecisionBG
Thalamus
ActionDecisionBG
2
-2
10.555
12.555
Thalamus: xBiased
2
-2
10.555
12.555
[x] shows the best next moves path the rat is/was trained on.
R-Lever
L-Lever
Right
Left
Down
Up
Wall
BestMove
LearnMove 72.97%
0
20
40
60
80
R-Lever
L-Lever
Right
Left
Down
Up
Wall
BestMove
qLearnMove 100.00%
0.0000

Figure 1: Nengo screen capture of maze model, basal ganglia and action selection output and model performance graphs at the end of mapping mode.

c. Self-Solving Mode, where the model is run and checked to see if it correctly solves the maze from the given trained weights. The NENGO interactive screen of a model running is shown in Figure 3. The simulated rat and the maze are shown in the maze graphic which is updated as the rat learns and then "thinks" about each move. Starting from the top left and proceeding counter clockwise the displays are:

   i. Place Ensemble: the place ensemble is a 2-dimensional ensemble which holds the current x and y coordinates. The value plot shown therefore shows the rat's position in (x,y) coordinates as the rat learns and then as the rat moves itself post-learning to solve the maze.

   ii. The LearnSimpleNode ensemble has 2 display elements;

   -The PositionOriginGraph shows the values of the (x,y) position pairs being fed into the SimpleNode to facilitate learning or generate the learned response.

   - The coloured maze graphic takes the specified maze sizes and paints the maze on the screen with;

   - yellow being a maze cell
   - dark blue representing the walls
   - the mouse icon being the current position of the PlaceEnsemble
   - the cheese icon showing where the cheese reward cell is
   - the light blue "B" cell represents where the best next move is from the maze's trained solution set
   - "TnM" flashes in the cell that the model thinks is the best next move at any instant during self-solving mode. The comparison of this flashing indicator with the stationary light blue "B" cell gives a real time indication during self-solve mode of the training success. The move actually chosen is integrated across 75 ms of time of the Thalamus' output.

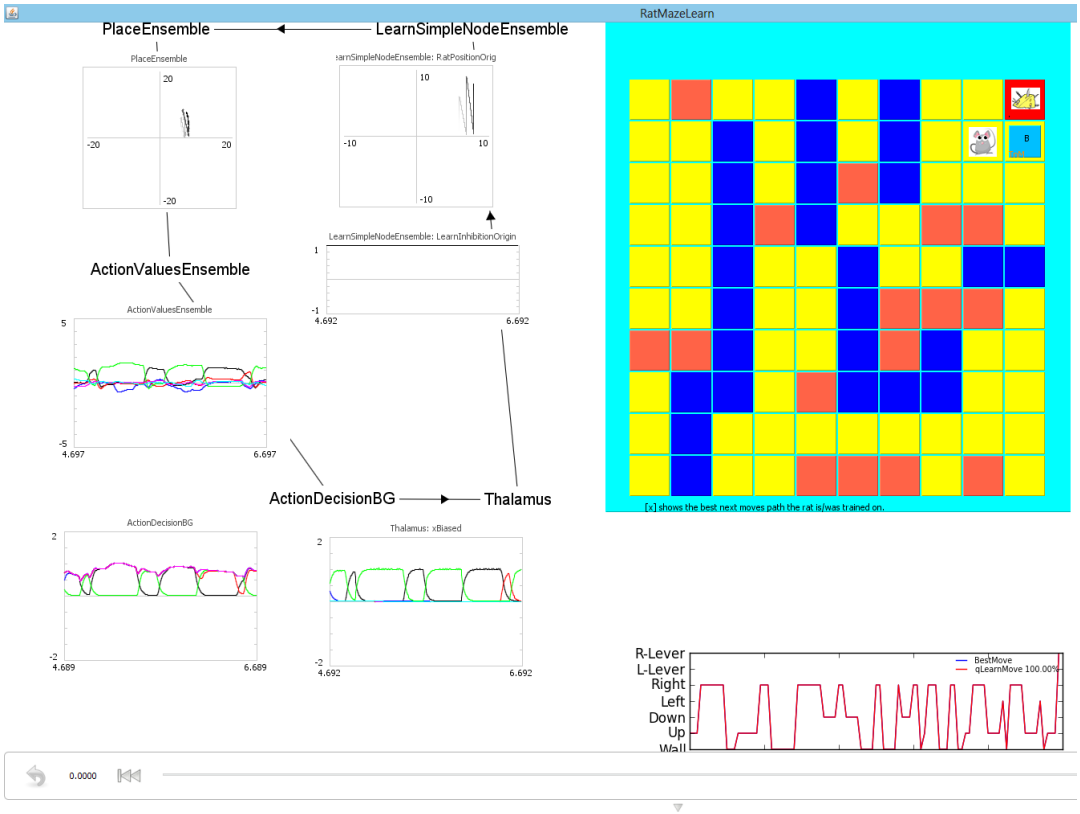


Figure 4: Auto-solve mode running showing the Best Next Move in light blue which at this instant equals the model's current thinking best next move "TnM" in yellow text on the "B" cell.

iii. The ActionDecisionBG ensemble and the Thalamus ensemble graphs show the relative values of the 6 action selection dimensions that these two ensembles code for. The basal ganglia chooses the strongest action from its inputs in a biologically plausible way (Stuart et al, 2003) and the Thalamus takes the inhibitory biased basal ganglia's model output and inverts it to an excitatory output. The choice the model makes is the result of a process of integrating the instantaneous output of the Thalamus over a period of time as follows;

1. The rat is placed at the starting position, (0,0) in our runs and then the network is allowed to settle for 25 ms.
2. Then the values of the Thalamus' six dimensional output are integrated for 75 ms. The action with the highest accumulated value is then "chosen" by the model.
3. The action chosen is applied to the current position and the position ensemble is updated with it and the cycle repeats.

3 Results

3.1 Summary Results

After 4 runs of the 16 models (the averages of which are shown in Figure 6) we observed that;

(1) Accuracy was inversely proportional to the number of discontinuities (ie: maze complexity = number of turns in the maze) in the mapping function.

(2) The best accuracy was achieved using QLearning training where the reward amount used to compute the q-value was primarily based on the A-star algorithms solution of the maze.

We hypothesize these to be a result of information as well as recency effects.

(1) QLearning embeds in its reward function more and graduated information of the entire solution path into each move's reward, whereas in direct A-Star solution encoding (best move encoding) only information about the very next move in the sequence is captured in the training weights;

(2) Secondly, by training in a path ordered way as opposed to random training, the recency effects well known in least-squares-based learning algorithms comes into play and aids in the encoding weights which more accurately capture the discontinuities of the mapping function.

		Random Training			Path Training		
		No Qlearning with Random Training	Qlearning with Random Training		No Qlearning with Path Training	Qlearning with Path Training	
Best Moves #Functional Discontinuities	Maze Difficulty	No Qlearning Random Training	Qlearning with Random Training Learn Connection Accuracy	Qlearning with Random Training Qlearn Accuracy	No Qlearning with Path Training Learn Connection Accuracy	Qlearning with Path Training Learn Connection Accuracy	Qlearn Accuracy
3	No Walls	66.80%	70.80%	100.00%	72.80%	77.40%	100.00%
26	Easy Walls	58.43%	59.57%	100.00%	58.86%	69.01%	100.00%
50	Medium Walls	52.48%	53.18%	100.00%	53.18%	59.53%	100.00%
61	Hard Walls	47.42%	45.31%	100.00%	48.65%	54.74%	100.00%

Figure 6: Summary data from 4 runs of all 16 models.

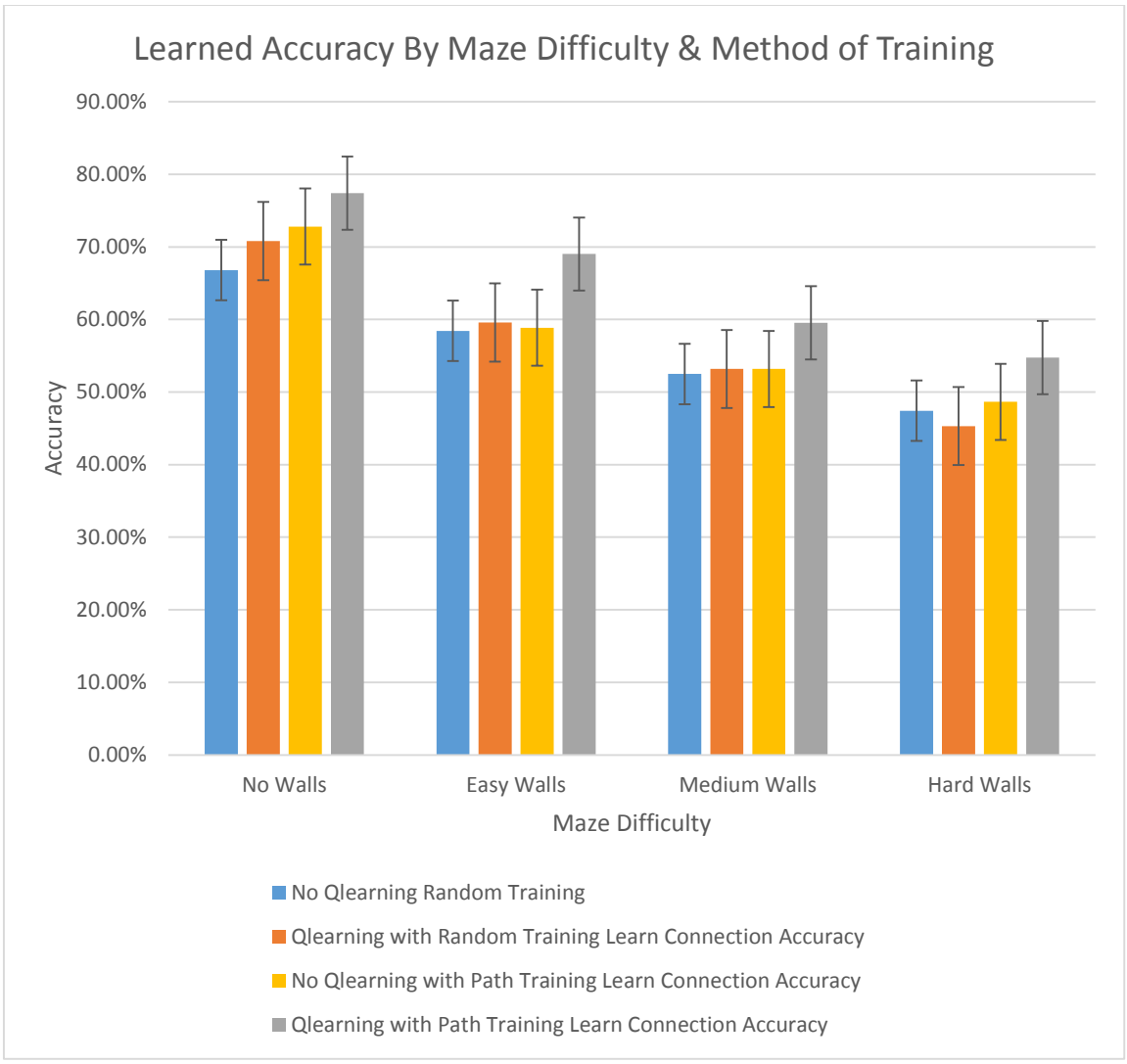
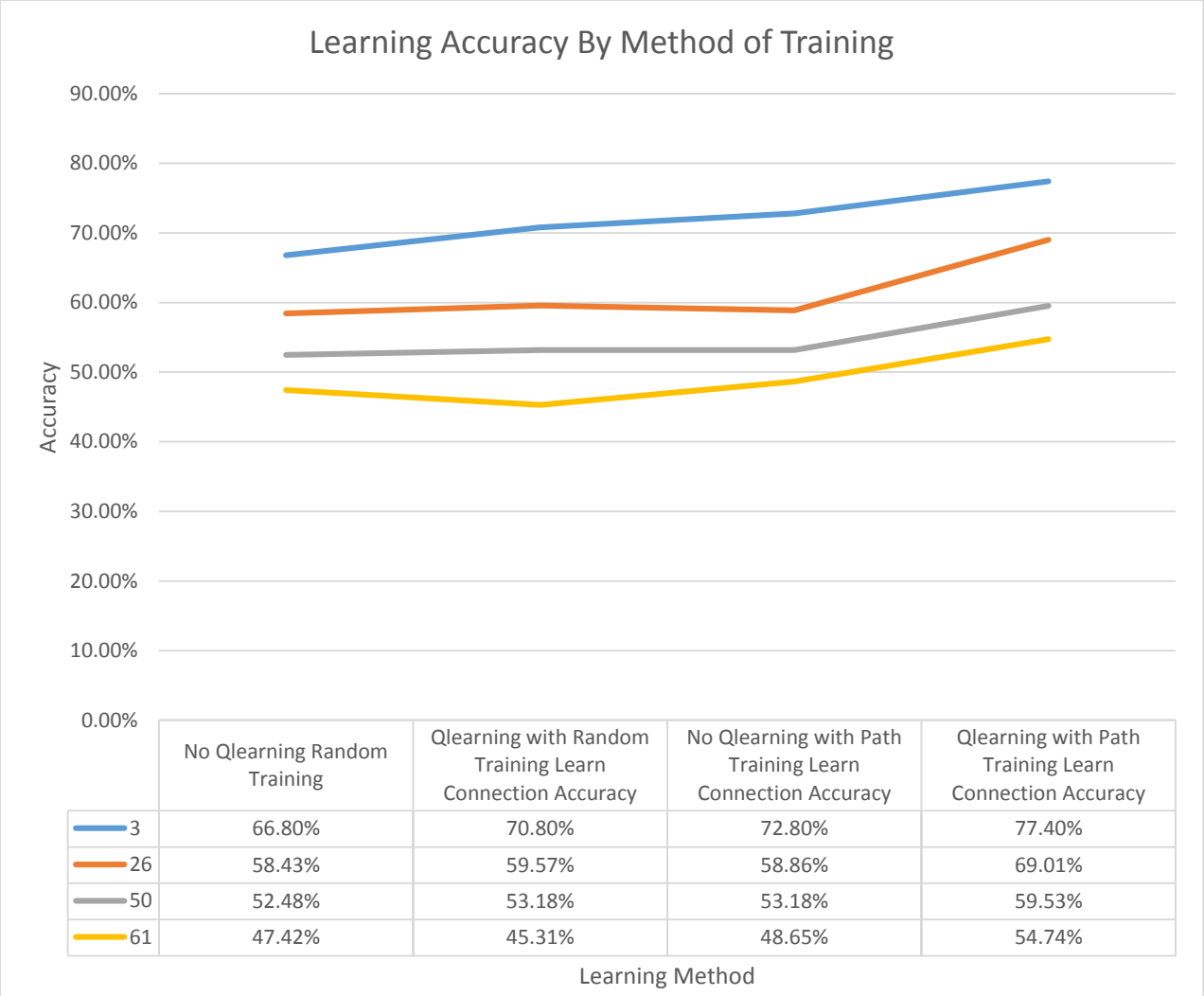


Figure 7: Summary data of all runs graphed by maze complexity.



	No Qlearning Random Training	Qlearning with Random Training Learn Connection Accuracy	No Qlearning with Path Training Learn Connection Accuracy	Qlearning with Path Training Learn Connection Accuracy
3	66.80%	70.80%	72.80%	77.40%
26	58.43%	59.57%	58.86%	69.01%
50	52.48%	53.18%	53.18%	59.53%
61	47.42%	45.31%	48.65%	54.74%

Figure 8: Summary data ordered by maze complexity and then method of training.

4 Improvements Suggested

The fundamental question of the study asking whether there are ways to encode with q-learning or not, path or random training and a learned connection using least-squares minimization (LSM) would be best expanded to look for other alternative encoding methods for capturing discontinuous functions. Within the context of LSM with and without Q-Learning, the study could be improved in many ways by collecting data over many more runs and by exploring the encoding accuracy using different network settling times and network integration methods.

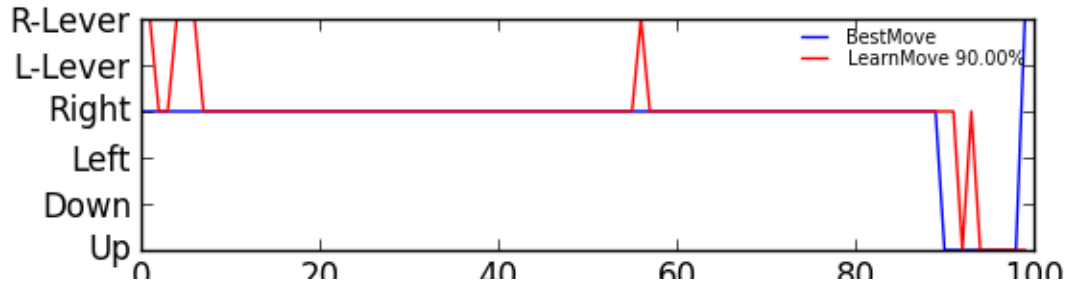
5 Appendices

5.1 Sample Results Graphs

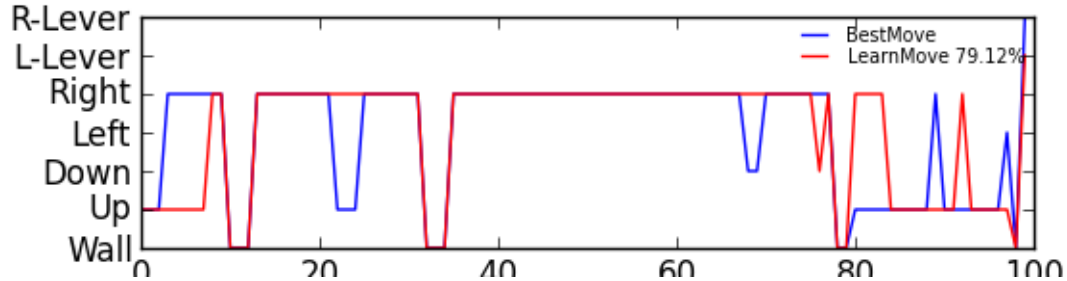
The resulting learned mapping function of current position to the best next action to take (one of [Up, Down, Left, Right, Left Lever Pull, Right Lever Pull]) are shown below for each learning method run. The Q=Learning mapping to show that the Q-Learning training signal was in fact accurate before training are also shown for those cases where Q-Learning was used. The graphs give a quick visualization of the discontinuous nature of the mapping function as well as the correlation from training set to learned responses.

5.1.1 Without Q-Learned Training, Random Training

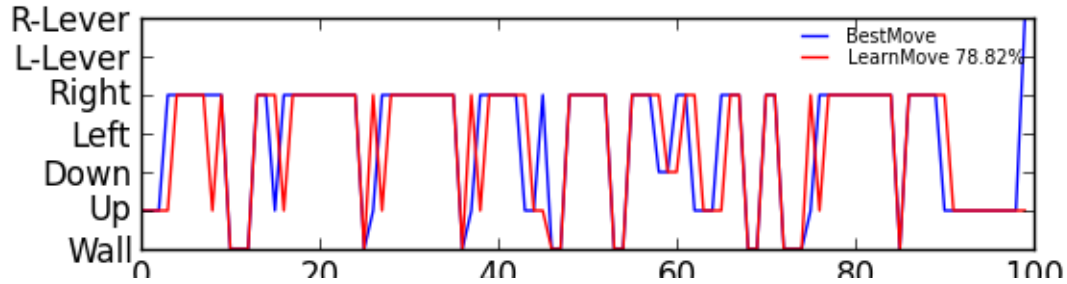
5.1.1.1 No Walls



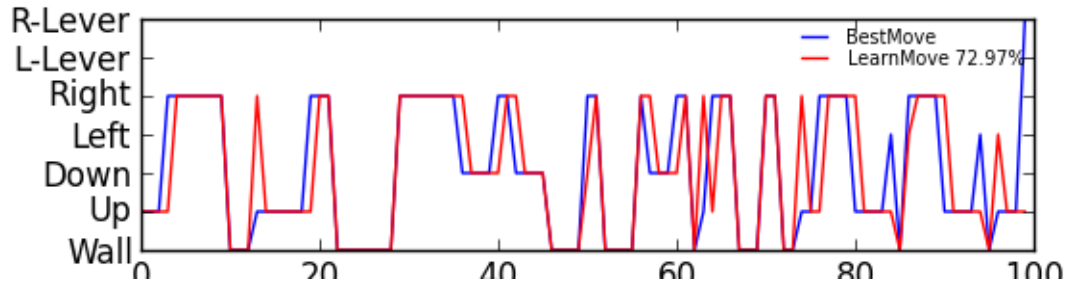
5.1.1.2 Easy Walls



5.1.1.3 Medium Walls

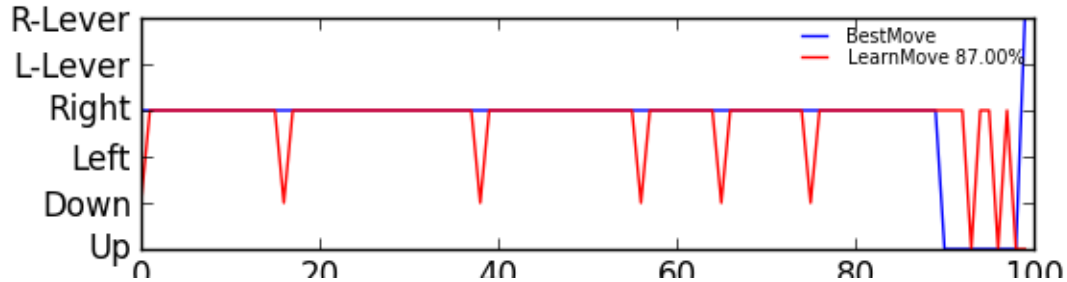


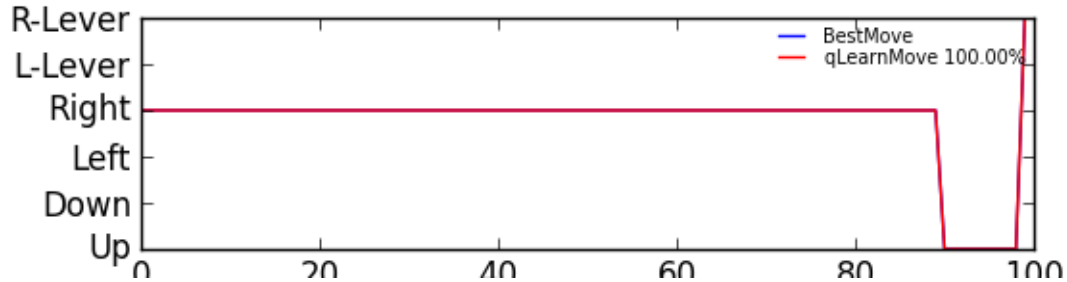
5.1.1.4 Hard Walls



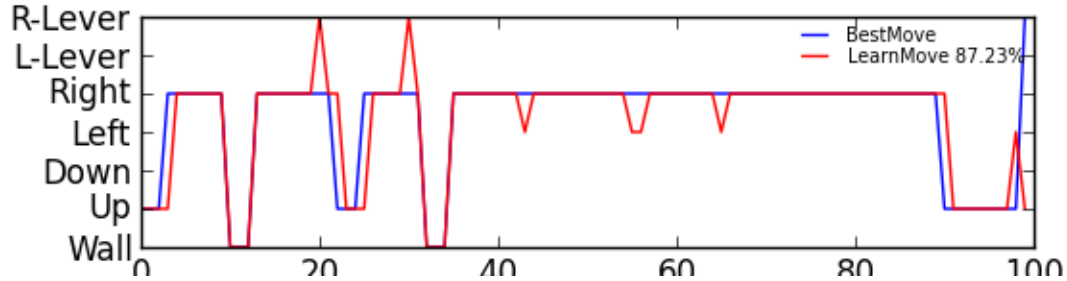
5.1.2 With Q-Learned Training and Random Training

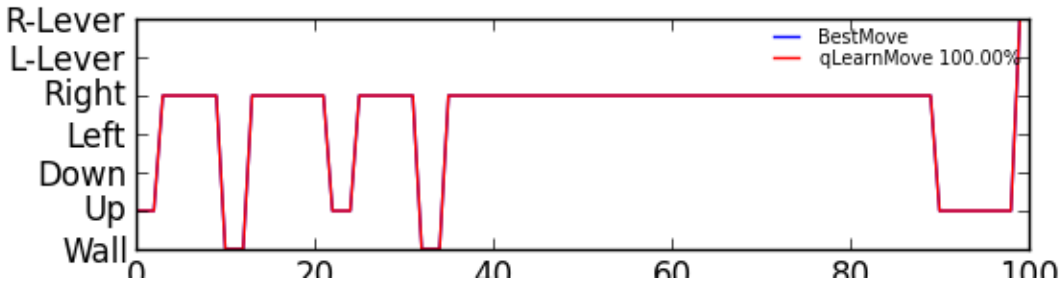
5.1.2.1 No Walls



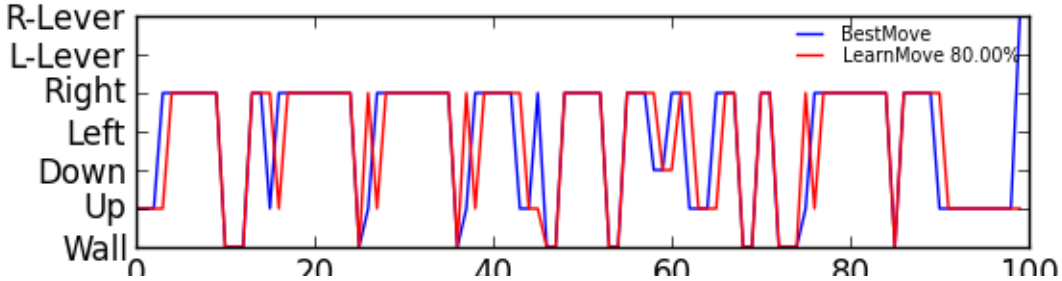


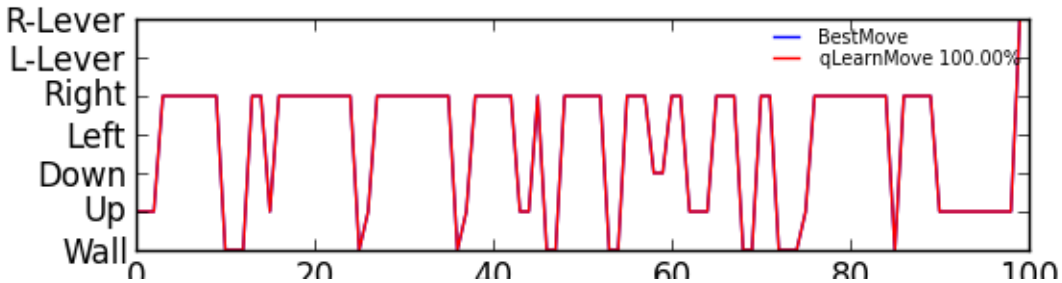
5.1.2.2 Easy Walls



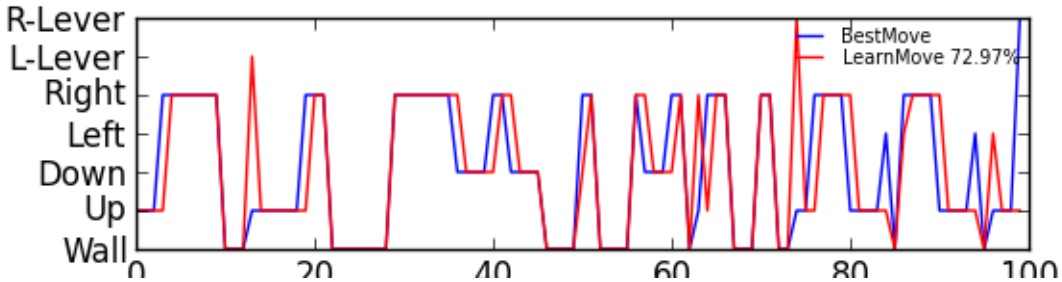


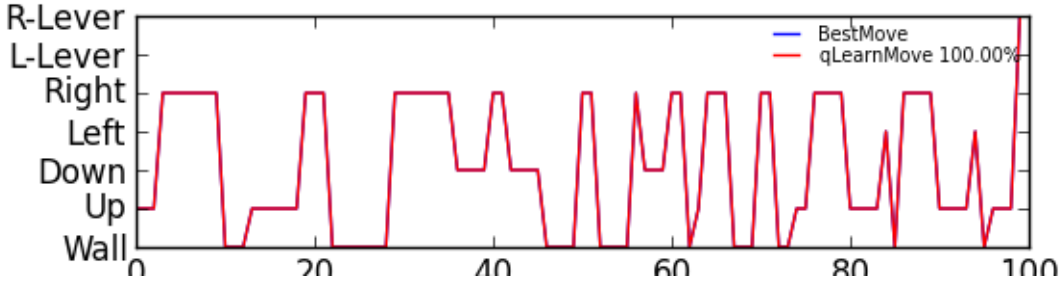
5.1.2.3 Medium Walls



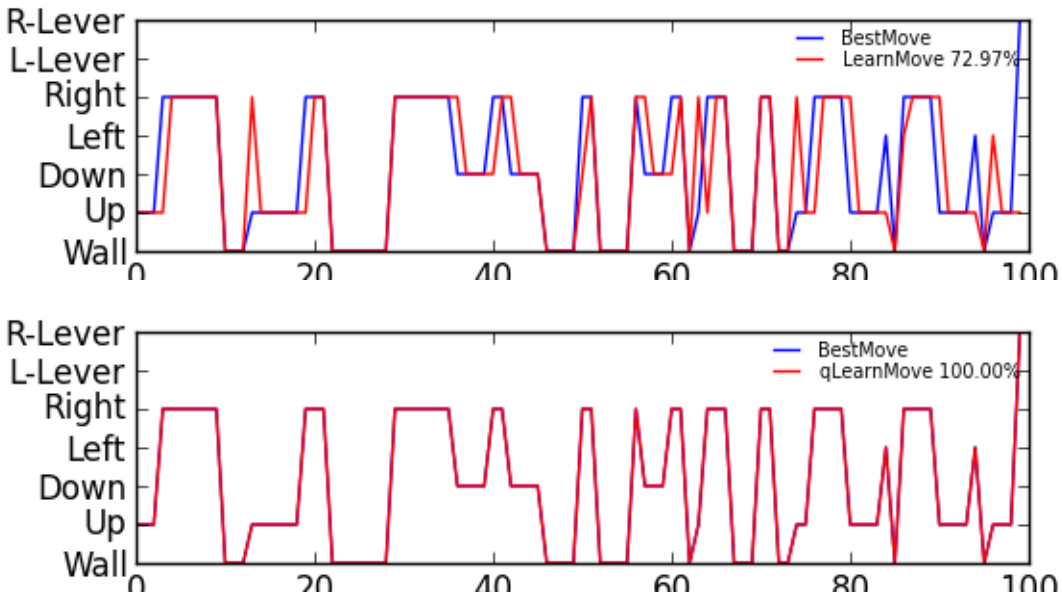


5.1.2.4 Hard Walls



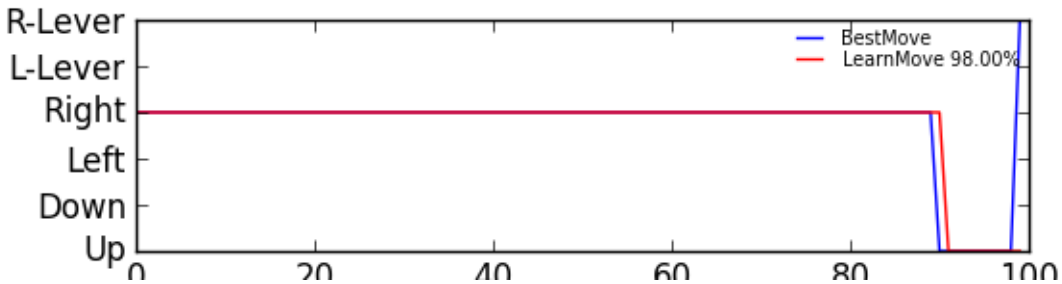


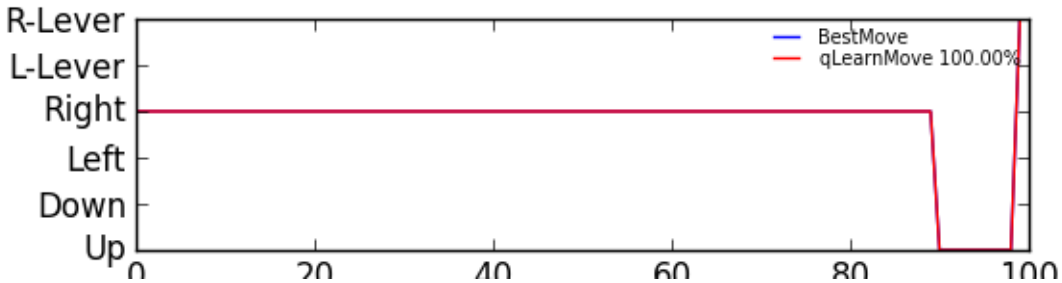
5.1.2.5 Hard Walls with Path Training and 1000 Neurons in Learning Population



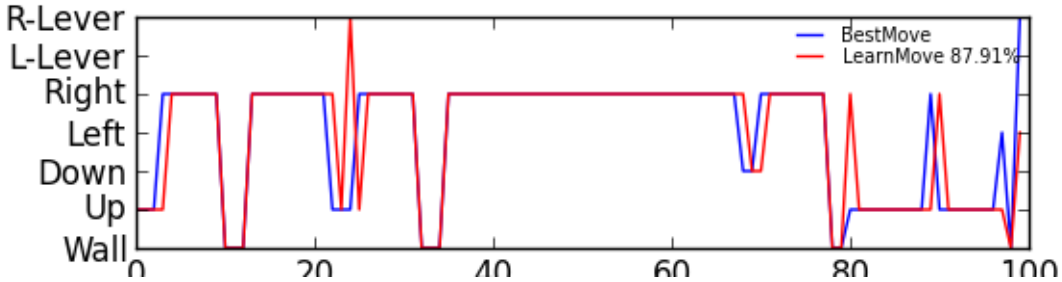
5.1.3 With Path Training and Q-Learned Training

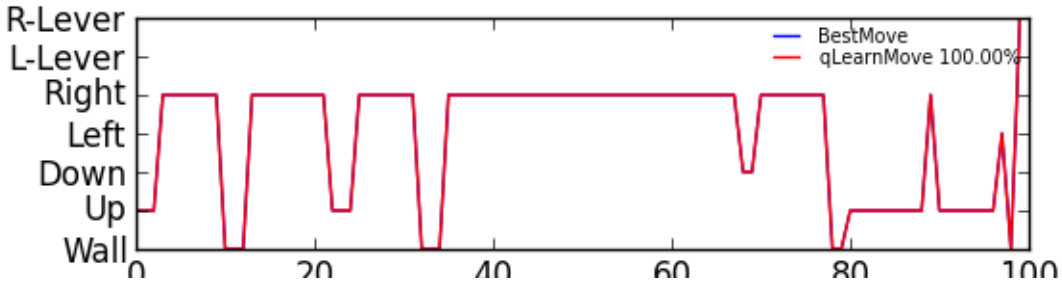
5.1.3.1 No Walls



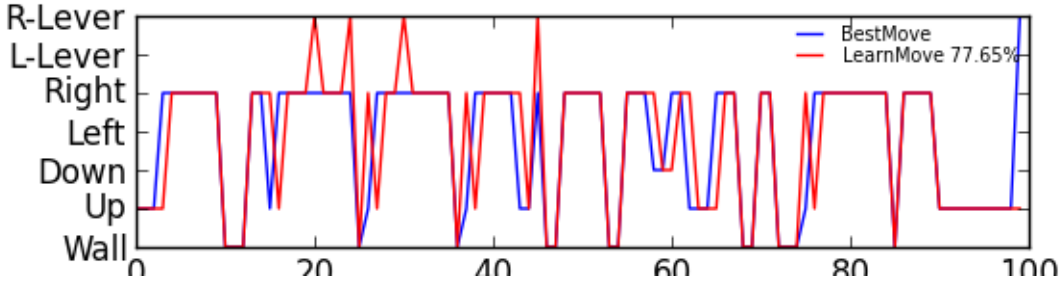


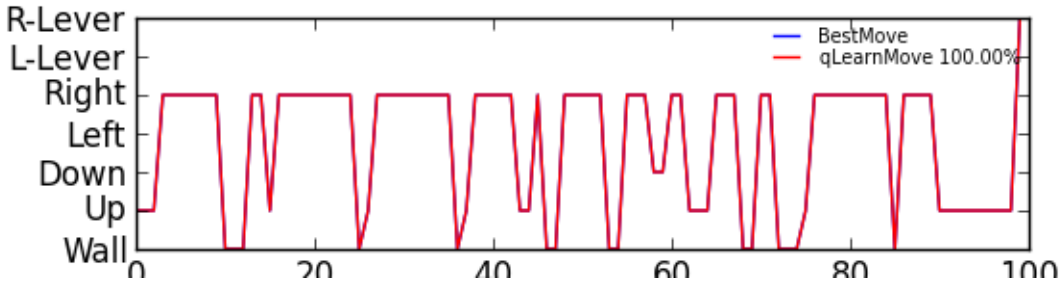
5.1.3.2 Easy Walls



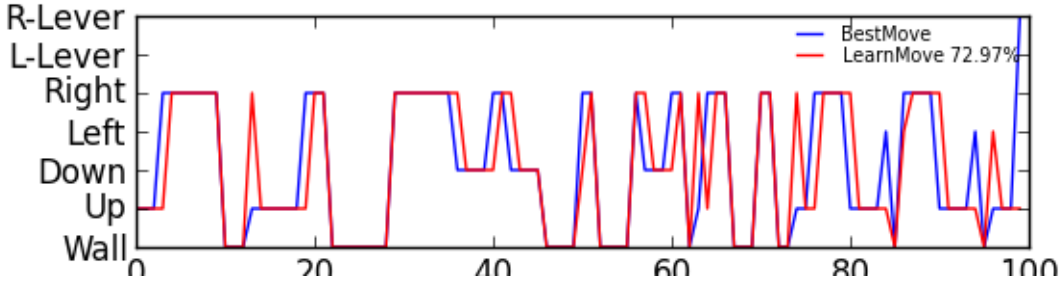


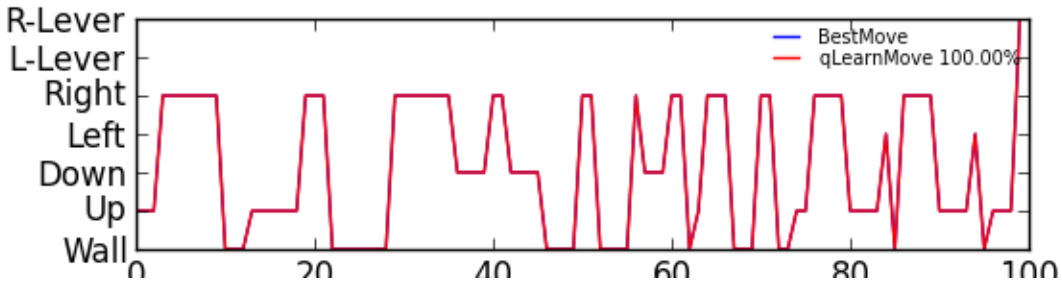
5.1.3.3 Medium Walls





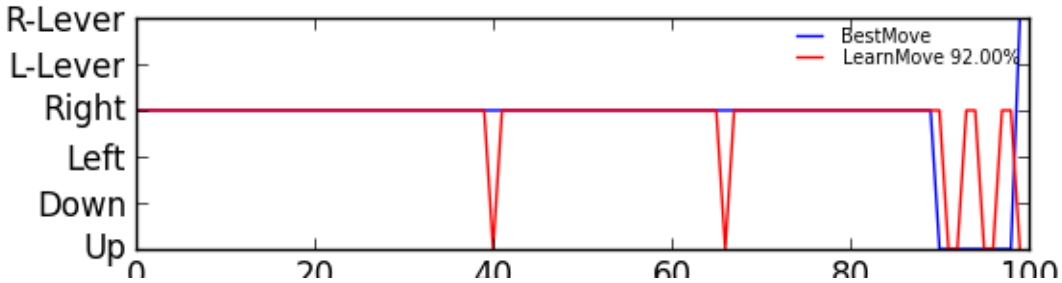
5.1.3.4 Hard Walls



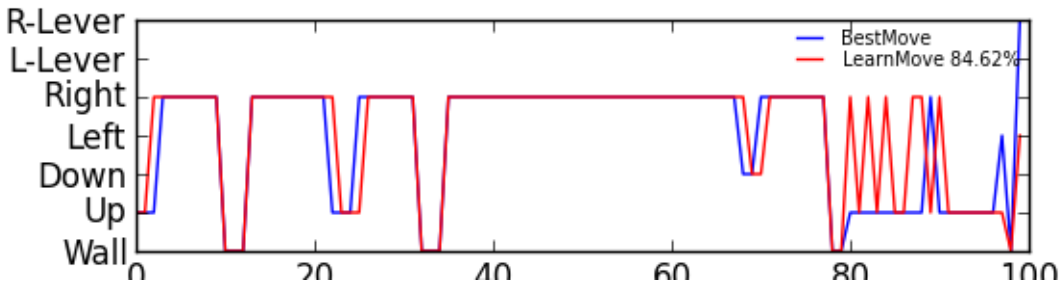


5.1.4 Without Q-Learned Training and Path Training

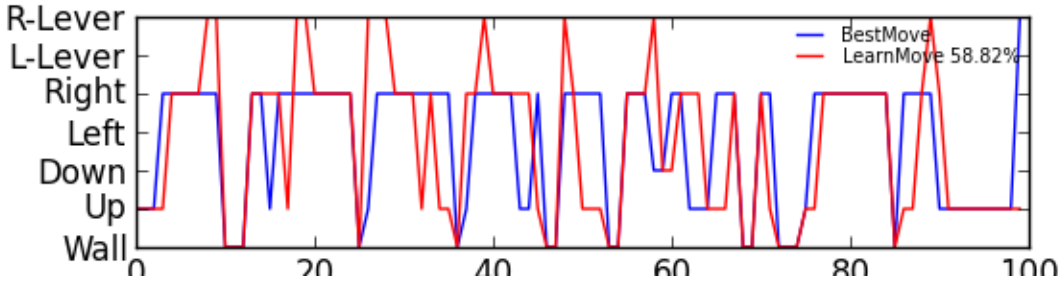
5.1.4.1 No Walls



5.1.4.2 Easy Walls



5.1.4.3 Medium Walls



5.1.4.4 Hard Walls