Introduction

The method of assessment will consist of a quiz, a short essay, a written report and a large piece of practical coursework. In brief the coursework will ask you to implement the Temporal Difference and Backpropagation learning algorithms that are taught during the course.

You might be interested to know that the coursework topic is regarded as somewhat of a research topic. Hopefully this makes the assignment both challenging and fun.

Robocode

Robocode is an exciting, interactive environment designed by IBM, originally as a teaching aid in Java. However it has since become a popular tool for the exploration of topics in artificial intelligence (AI), including neural networks and reinforcement learning (RL). In this assignment, it is hoped that you will be able to gain hands-on experience of these areas of AI whilst having fun developing your own intelligent robot!

Please read the following problem statement **carefully**.

Problem Statement and Deliverables
You are required to build a robot tank, in software, as instructed in this problem description. The construction will be divided into three stages. Upon completion of each stage, you will be required to submit evidence of your practical work. Each submission will be followed by an assessment.

Only softcopy submissions in Microsoft Word format will be accepted and should be sent via email. A PDF version must accompany each version. The softcopies will be marked and returned via email. Remarks will be inserted using Word's commenting feature or as markups directly on the PDF.

Your work must demonstrate the practical application of both the *backpropagation learning* (BP) and *reinforcement learning* (RL) algorithms. Both topics are covered in the course. You should apply these methods as suggested below.

Application of RL

> Your tank must face a battle with one enemy in the battlefield. Your robot must act fast to defend itself but needs to attack and kill to earn more points. What do you do? Which actions do you chose to make available to your tank? Run? Fire? Chase? Which is the best choice? The goal of the RL agent should be to maximize the rewards associated with winning. It does this by learning those actions in each state, which lead to an optimal long-term reward.

Application of BP

> A key component of RL is the value function, or Q-function in our case. One of the easiest ways to implement this function is as a look-up table. However, in any real world problem the state space is likely to be prohibitively large. E.g. for backgammon, there are 10^{20} different states. In this case, the look up table would need 10^{20} rows! Clearly this is not practical. The key is the ability to approximate the value function. There are many ways. For this assignment you are to implement a feed-forward multi-layer perceptron trained via backpropagation to approximate the value function. There is no "tried-and-tested" application of this technology and as such this is the "research" element of the assignment! (Although I know that to a large degree, it will work. ☺)

Report Guide

As mentioned earlier, function approximation of an RL value or Q-function is in fact a topic of research (for example see http://rlai.cs.ualberta.ca/RLAI/RLFA.html). Meaning that there is no clear or well-defined solution guaranteed to work in all cases. In fact, the successful application of function approximation to RL is a delicate art! There is obviously then, no correct or right answer that I will be looking for. Your understanding and expertise expressed through your report, quiz and essay will be key to attaining a good mark.

Policy note 1: I encourage you to discuss your project work with your fellow students, however in accordance with University policy, your work MUST BE WRITTEN IN YOUR OWN WORDS. In previous years students have been penalized for sections of text which where copied verbatim, from other 592 assignments or published academic works. If you have any concerns or questions, please do not hesitate to ask.

Submission Schedule and Weighting

Part 1a	BP Learning	Due 28th Oct 2015	15%
Part 1b	BP Quiz	Due 04th Nov 2015	5%
Part 2a	RL with LUT	Due 11th Nov 2015	25%
Part 2b	RL CPR Essay (CPR Marking is due 02nd Dec 2014)	Due 18th Nov 2015	15%
Part 3	RL with Backpropagation	Due 09th Dec 2015	40%

Policy note 2: It is understood that sometimes, personal circumstances necessitate more time to complete the work. If you need an extension to these dates, please get a new date agreed with me IN WRITING. Otherwise, late submissions will be subject to a penalty of my choosing. As a rule, no reports will be accepted after 31st Dec 2015.

Part 1a – Backpropagation Learning: Instructions

For this part of the assignment you are to implement a multi-layer perceptron and train it using the error-backpropagation algorithm. You will not yet need to use Robocode, however you should keep in mind that your neural net will later be used in your robot tank. It will be useful if your neural network software is able to accept as a parameter the number of hidden neurons. To help, I have provided Java interfaces for you to start with:

https://courses.ece.ubc.ca/592/PDFfiles/NeuralNetInterface.java.pdf
https://courses.ece.ubc.ca/592/PDFfiles/CommonInterface.java.pdf

Now follow the instructions below:

Submission instructions

1) Set up your network in a 2-input, 4-hidden and 1-output configuration. Apply the XOR training set. Initialize weights to random values in the range -0.5 to +0.5 and set the learning rate to 0.2 with momentum at 0.0.
   a) Define your XOR problem using a binary representation. Draw a graph of total error against number of epochs. On average, how many epochs does it take to reach a total error of less than 0.05? You should perform many trials to get your results, although you don't need to plot them all.
   b) This time use a bipolar representation. Again, graph your results to show the total error varying against number of epochs. On average, how many epochs to reach a total error of less than 0.05?
   c) Now set the momentum to 0.9. What does the graph look like now and how fast can 0.05 be reached?

   Your submission should be a brief document clearly showing the graphs requested about. Please number your graphs as above and also include in your report an appendix section containing your source code.

Part 1b – Backpropagation Learning Instructions

To complement your practical work, you will be also assessed using a short quiz on the topic of BP learning. More instructions to follow.

Part 2a - Reinforcement Learning using a Look-Up-Table (LUT): Instructions

In this second part you are to implement Reinforcement Learning for your robot tank. For this you will need to use the Robocode environment. Before doing this, you need to decide which actions you would like to support and when & how to generate rewards. Once your learning mechanism is working, you should capture the contents of a trained LUT to file. This will be later helpful when replacing the LUT component with a neural net.

Here are some tips:

- You will find it helpful to attempt to battle only a single enemy. Pick one that comes pre-installed with Robocode.
- It might help if rewards are not just terminal.
- You will have to apply some quantization or dimensionality reduction. If your state space is large you'll, run out of memory when declaring your look up table.
- For now you should implement RL, specifically Q-learning, using a look-up table. Furthermore since this is going to be replaced with the neural net from part one, the class headers defining the LUT should be consistent with the class headers of the neural net class from part 1. Again, to help you, I have provided a java interface definition for you to start with:

  https://courses.ece.ubc.ca/592/PDFfiles/LUTInterface.java.pdf
  https://courses.ece.ubc.ca/592/PDFfiles/CommonInterface.java.pdf

Now follow the instructions below for part 2:

Submission instructions

2) Once you have your robot working, measure its learning performance as follows:
   a) Draw a graph of a parameter that reflects a measure of progress of learning and comment on the convergence of learning of your robot.
   b) Using your robot, show a graph comparing the performance of your robot using on-policy learning vs off-policy learning.
   c) Implement a version of your robot that assumes only terminal rewards and show & compare its behaviour with one having intermediate rewards.

3) This part is about exploration. While training via RL, the next move is selected randomly with probability ε and greedily with probability $1 - \varepsilon$
   a) Compare training performance using different values of ε including no exploration at all. Provide graphs of the measured performance of your tank vs ε.

As for part 1, your submission should be a brief document clearly showing the graphs requested about. Please number your graphs as above and also include in your report an appendix section containing your source code.

Part 2b – Calibrated Peer Reviewed (CPR) Essay

To complement your practical, work you are required to write a short (max. 400 word) essay that demonstrates your understanding of reinforcement learning. More instructions to follow.

Part 3 – Reinforcement Learning with Backpropagation: Instructions

In the third and last part you are to replace the LUT used by the Reinforcement Learning component with a neural net. This will be easy to do in your software if your class headers for the LUT and the neural net match. However this is a little tricky to get working since there are many parameters that can be adjusted. Furthermore, the training data is dynamic as it is generated by the Robocode environment. I.e. it is not an a-priori static set of training data. This means that it is not possible to compute a “total error” during training, which also means that there is no conventional way to judge if learning is happening or not. You’ll have to use your own ingenuity here ☺

Some hints:

- It will help if you capture your look up table from part 2. You now have a static set of data, which you could use to train a neural net. Why is this useful? Well, you will be able to use your neural net software from part 1 and instead of training the XOR problem, apply your robocode tank data. This will allow you to adjust the various learning parameters to get the best training for this “kind of” data upon a static training set. (e.g. learning rate, momentum term, number of hidden neurons). Once you’ve identified a good set of parameters, they should work once your neural net is used on live data.
- It might be helpful for you to be able to load and save the weights of your tank. That way you can save your tank’s behaviour and recall it later as needed.
- In part 2, you should have found that it is absolutely necessary to reduce dimensionality of the state space to prevent having to deal with impractically large look up table. Perhaps with a neural net, this is not necessary?

Now answer the following questions:

Questions

1) The use of a neural network to replace the look-up table and approximate the Q-function has some disadvantages and advantages.
   a) Describe the architecture of your neural network and how the training set captured from Part 2 was used to “offline” train it mentioning any input representations that you may have considered. Note that you have 3 different options for the high level architecture. A net with a single Q output, a net with a Q output for each action, separate nets each with a single output for each action. Draw a diagram for your neural net labeling the inputs and outputs.
   b) Show (as a graph) the results of training your neural network using the contents of the LUT from Part 2. Include how you arrived at the parameters that best learned your LUT data. Compute the RMS error for your best results.

c) Try mitigating or even removing any quantization or dimensionality reduction (henceforth referred to as *state space reduction*) that you may have used in part 2. A side-by-side comparison of the input representation used in Part 2 with that used by the neural net in Part 3 should be provided. (Provide an example of a sample input/output vector). Compare using graphs, the results of your robot from Part 2 (LUT with state space reduction) and your neural net based robot using less or no state space reduction. Show your results and offer an explanation.
d) Comment on why theoretically a neural network (or any other approach to Q-function approximation) would not necessarily need the same level of state space reduction as a look up table.

2) Hopefully you were able to train your robot to find at least one movement pattern that results in defeat of your chosen enemy tank most of the time.
   a) What was the best win rate observed for your tank? Describe clearly how your results were obtained? Measure and plot *e*(s) (compute as Q(s',a')-Q(s,a)) for some selected state-action pairs.
   b) Plot the win rate against number of battles. As training proceeds, does the win rate improve asymptotically?
   c) Theory question: With a look-up table, the TD learning algorithm is proven to converge – i.e. will arrive at a stable set of Q-values for all visited states. This is not so when the Q-function is approximated. Explain this convergence in terms of the Bellman equation and also why when using approximation, convergence is no longer guaranteed.
   d) When using a neural network for supervised learning, performance of training is typically measured by computing a total error over the training set. E.g. as you did for question 6(b). When using the NN for online learning of the Q-function in robocode this is not possible since there is no a-priori training set to work with. Suggest how you might monitor learning performance of the neural net now.

3) Overall Conclusions
   a) This question is open-ended and offers you an opportunity to reflect on what you have learned overall through this project. For example, what insights are you able to offer with regard to the practical issues surrounding the application of RL & BP to your problem? E.g. What could you do to improve the performance of your robot? How would you suggest convergence problems be addressed? What advice would you give when applying RL with neural network based function approximation to other practical applications?
   b) Theory question: Imagine a closed-loop control system for automatically delivering anesthetic to a patient under going surgery. You intend to train the controller using the approach used in this project. Discuss any concerns with this and identify one potential variation that could alleviate those concerns.

For this last and final part, you should submit a written report that includes answers for each of the questions above.

Your report should be well structured, written clearly and demonstrate an understanding of the backpropagation and reinforcement learning paradigms. For each of these, it should describe the problem being addressed and provide an analysis of how that learning mechanism was applied. It is important that you describe how your solution was evaluated and offer a conclusion. Pay attention to your results and be scientific in your approach.

Try to be as thorough and clear as possible with your answers, which may not be unique. When providing explanations, supporting your statements with practical results helps. You'll be judged based on what you can deduce from your experiments and how well you understand the theory. Try to make your report as neat and presentable as possible. It takes me many hours (days in fact) to read and mark all material and there is no opportunity for me to re-read the work. It really helps me if the work is presented neatly and is easy to read. Don’t be surprised if you lose marks for poorly organized material. You should use tables, graphs and diagrams to help in this respect.

Please also format your report such that you precede each answer with a copy of the question in *italics*.

References:

1 Fausett, L. (1994). *Fundamentals of Neural Networks. Architectures, Algorithms and Applications*. Prentice Hall.
2 Sutton R.S., and Barto A.G. (1998) *Reinforcement Learning*. The MIT Press.
3 Li, S. (2002) *Rock 'em, Sock 'em Robocode.* http://www-128.ibm.com/developerworks/java/library/j-robocode/
4 Reinforcement Learning & Function Approximation group at the University of Calgary. http://rlai.cs.ualberta.ca/RLAI/RLFA.html

Acknowledgments:

My friend Julian Rendell, for bringing Robocode to my attention and suggesting its use as a course project.