16.412J PS 1

Tom Temple

February 15, 2005

Part A

1. **Reinforcement Learning**
   I am interested in reinforcement learning primarily because I think it is one of the better models for how humans act. Take for instance the problem of the inverse kinematics of a robot. With very good measurements of the system, they can be painstakingly computed. But if you let a person play with the controls for a while, they can make the robot do what they want, without measurement or calculation. Furthermore, the human control is more robust.

2. **Exploration vs Exploitation**
   I think this is a fascinating problem. I am regularly faced with the decision of how to drive from Cambridge to Lexington. Let me tell you that this is a difficult problem. I have some estimates of the means and variances of the time certain routes will take. But I don't really have much data on the true distributions especially when conditioned on the other information I have at my disposal in the morning. Luckily, topology and speed limits bound the number of possible paths to a manageble number. Even so, when rt 2 is a parking lot, it brings into play new potential routes for which I naturally have less information.

3. **Dynamic Baysian Networks**
   I find DBN fascinating because they are a powerful tool for representing temporal information. They are what I would consider the usable version of the HMM. As fascinating and useful as the Kalman filter is, its restrictions to gaussian distributions is potentially over-limiting. Lets say, for instance, that I have this sensor that is perfectly correct except for once in a while just it says 7 regardless. An astute person will quickly learn to trust the 7's far less than the 8's. I think that a computer should be able to do so also. The DBN is able to capture a much larger swath of the real

world. I am excited to learn more about making optimal decisions with under more general probability distributions.

Part D

Inverted Helicopter

Inverted autonomous helicopter flight via reinforcement learning, Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger and Eric Liang. In *International Symposium on Experimental Robotics*, 2004
I selected this paper since there was a picture of a helicopter in AIMA whose controller was developed by Andrew Ng. On his webpage he had a number of interesting articles. This was among the most relevant.

The important feature of this paper was that reinforcement learning was able to hover an autonomous helicopter inverted. The methodology consisted of having a human pilot fly the craft inverted and collecting data. Then they used this data to construct a model. They ran simulations using this model in order to find a policy that could fly the helicopter inverted. Finally, they successfully implemented the controller on the real helicopter.

A feature of the simulations was that each potential policy was tested on the same set of pre-determined random trials. That prevented variation in the trials to unduly affect which policies were considered the best. He described this previously in (Pegasus. Ng and Jordan 2000). The important idea is that one can eliminate the non-determinism of an MDP by generating a set of random trial outcomes and evaluating each policy on the same trials, treating the transitions as deterministic. As long as the trials are representative, the results will be representative of the true, non-deterministic case.

While this was a very exciting result, the methodology was nothing new. Furthermore, the reward function had a number of magic parameters that are not described and I can only suppose they had to be hand tuned. I would also like to limit the amount of learning that is done in simulation. I think that if the controller starts off very slowly (and with a little guidance), it could do all of the learning online without any crashes. People manage to do precisely that.

Relational Reinforcement Learning

Relational Reinforcement Learning. Sašo Džeroski. 2003
I found this paper in a citation by Forbes who was cited by AIMA.

This paper presents the RRL algorithm in which the Q-function is approximately represented by a regression tree. This regression tree is maintained so that similar examples are combined (or one is rejected) and sufficiently different examples yield an additional branching of the tree. Thus, a size constraint can be maintained. In cases where states are best described relationally, this data structure is intuitive and compact. For instance, it was effective in finding optimal solutions to the block stacking problem. The tree is populated by generalizing a set of manually provided optimal solutions.

A problem with this algorithm is that it scales poorly. The number of potential relationships grows very quickly. The algorithm over-generalized when it received too many training cases and was pretty bad at tetris. More importantly to the task in question, since the relationships were propositional, it doesnt generalize into the continuous case very well.

Effective Reinforcement Learning

William D. Smart and Leslie Pack Kaelbling, "Effective Reinforcement Learning for Mobile Robots," International Conference on Robotics and Automation, May 11-15, 2002.
I found this one on LPK's webpage because I had her as a professor and knew she was into this stuff.

This paper deals with the problem of Q-learning with sparse rewards. Since the initial distribution of rewards is so flat, the agent ends up having to pick actions arbitrarily which means it might take a very long time for the agent to find a good policy. She fixes the problem manually much like in Dzeroski, with manual information. In one case, she provides a simple starting learning policy that ensures that the goal is found. This allows the Q-function to start being filled in. Then she drives the agent manually towards the goal. In this case, unlike in Dzeroski, it doesnt matter if this initial input is optimal or not, it is only to aid fleshing out the Q-values. In all three papers, a human operator is used to hone in on the region of the state space that is of interest.

I thought this paper was fantastic. It was interesting and well written. I guess if I had to complain about something it would be that there is no discussion of choosing the discount factor or learning rate. Also, some details of how the Q-values are updated are not specified.

Part E

I propose implementing a controller for the Quanser helicopter that learns how to fly on its own and does not rely on prior knowledge of the system model. I plan on using Q-learning or policy search. I would start by providing a dense reward surface for hovering and let the controller determine the control. Then I will try to extend that to general trajectory following. If that goes well, I will try a much sparser set of rewards. If possible, I would like to have it learn from a human operator like in the Kaebling paper. I estimate the probability of persuing this project at 3/4.