Artificial Intelligence - Assignment 4

Posted Tuesday, March 23, 2010
Due Tuesday April 6, 2010

Note: This is the last assignment of the term, and it is worth 200 points (compared to 100 for previous assignments)

1. [15 points] **Expected utility and value of information**

   You go to the races. You can decide to:

   - Not place any bets at all
   - Bet on Belle. It costs $1 to place a bet and you get $6 if it wins, for a profit of $5.
   - Bet on Karl. It costs $1 to place a bet and you get $11 if it wins, for a profit of $10.
   - Bet on Sam. It costs $1 to place a bet and you get $3 if it wins, for a profit of $2.

   You believe Belle has probability 0.5 of winning, Karl has probability 0.1 of winning and Sam has probability 0.8 of winning.

   (a) [9 points] What are the expected utilities for all the action, and what is the optimal action?

   (b) [6 points] A shady character comes and tells you that he knows which horse will win, because the race is arranged. But he wants some money for this information. You have reason to believe that his information will be correct. How much should you be willing to pay him?

2. [30 points] **Bandit problems**

   Suppose that you have a bandit with 2 arms; the payoff of the arms average 0.6 and 0.4, and in each case, they are drawn from a uniform distribution in $[0.5, 0.7]$ and $[0.2, 0.6]$ respectively. Implement, in a language of your choice, the bandit and ϵ-greedy exploration. Plot the estimated values of the two actions as a function of the number of plays, going up to 200 plays. Use 3 different values of ϵ: 0, 0.1 and 0.2 (so you should have 2 graphs, with 3 curves each). Briefly explain your results.

   Note that you can re-use the code for the exploration algorithm in the project, if you make it modular enough

3. [15 points] **Problem formulation**

   You are designing a recycling robot whose job is to collect empty soda cans around the building. The robot has a sensor to detect when a can is in front of it, and a gripper to pick up the can. It also senses the level of its battery. The robot can navigate, as well as pick up a can and throw a can it is holding in the trash. There is a battery charger in the building, and the robot should not run out of battery.

   (a) Describe this problem as an MDP. What are the states and actions?

(b) Suppose that you want the robot to collect as many cans as possible, while not running out of battery. Describe what rewards would enable it to achieve this task

(c) Instead of thinking about the actions described above, one could describe the task of the robot as choosing between larger activities: walk randomly to find cans, wait for someone to drop a can, or go dock with battery charger. Describe the advantages and disadvantages of this problem formulation compared to the one you gave before.

4. [15 points] **Maximum likelihood estimation**

For discrete random variables, another distribution of interest is the *Poisson distribution*:

$$p(k) = e^{-\lambda}\frac{\lambda^k}{k!}$$

where $\lambda \in \Re$ is a parameter.

(a) [5 points] Suppose that X is a random variable drawn from a Poisson distribution of unknown parameter λ. You observe n i.i.d. samples $x_1, \ldots x_n$ drawn from the distribution. Write the likelihood of parameter λ w.r.t. this sample.

(b) [5 points] Derive the maximum likelihood estimate for λ. What is the sufficient statistic of the data in this case?

(c) [5 points] Suppose you computed your MLE estimate, λ_n, based on the initial sample. You get a new sample, x_{n+1}. Write a formula that updates the old value, λ_n, to a new value, λ_{n+1}. Your formula for λ_{n+1} should be in terms of λ_n and x_{n+1}.

5. [15 points + 5 points extra credit] **Planning in MDPs**

In class we defined the optimal state value function for an MDP. One can define the optimal action-value function in a very similar way:

$$Q^*(s,a) = E[r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a]$$

(a) [5 points] Express V^* as a function of Q^*

(b) [5 points] Write a Bellman equation that relates $Q^*(s,a)$ to the values of successor states and actions, $Q^*(s',a')$

(c) [5 points] Set up a value iteration algorithm for computing Q^*

(d) [Extra credit: 5 points] Show that the maximum error of this algorithm decreases exponentially at every iteration.

6. [10 points] **Cross-validation**

Consider a data set with 10 examples for a binary classification task. We define a majority classifier as a classifier which returns the class occurring most in the training data.

(a) [5 points] Suppose that you have 7 positive examples and 3 negative examples in your data set. What is the training and testing error obtained after cross-validation?

(b) [5 points] Explain for what mix of examples in the data set the majority classifier would be a bad choice, from the point of view of leave-one-out cross-validation.

7. [5 points] **Overfitting**

You are starting a new company and you want to study the competition first. In particular, you have gathered data about the discounts that they offer to students. Having taken AI, you know a few learning algorithms that can help you analyze the data. You run a few of them and get the following accuracy results:

Algorithm	Training data	Test data
NeuroChamp	70%	70%
TopRegress	65%	75%
Brainy	90%	65%%

Which learner would you prefer for predicting future data, and why?

8. [25 points] **Gradient descent for learning**

(a) [5 points] Suppose you wanted to train a neuron with a single real-valued input x, which had the following activation function:

$$h(x) = \frac{1}{1 + e^{-(w_0 + w_1 x + w_2 x^2)}}$$

Give the learning rules for w_0, w_1, w_2.

(b) [10 points] Suppose that you wanted to train a neuron with two inputs, x_1 and x_2, whose output had the form:

$$h(x_1, x_2) = w_0 + w_1 \log x_1 + w_2 e^{w_3 x_2}$$

For what range of the input variables would this neuron work? Give the learning rules for w_0, w_1, w_2, w_3.

(c) [10 points] Suppose that you were now to put neurons of this type in a feed-forward network with 2 input units, 2 hidden units, one output unit. Explain what changes (if any) would need to be made to the backpropagation algorithm to work in this case.

9. [70 points] **Learning for Philosophers' football**

In this problem, you have to implement a linear function approximator using temporal-difference learning for the game used in the project. Note that you do not have to you this approach in your final project submission, but you can, if you want to.

In order to do this, devise a number of features of the board (at least 4) that you will linearly combine; the weights of this combination will be tuned by learning. Explain what your features are. Start with all the weights at 0.

Game play goes as follows:

- you start at the current position, and look ahead at all your possible moves; evaluate each resulting board position based on your evaluation function
- Use epsilon-greedy strategy to pick the next move; then use the random player to generate the next opponent move
- Perform an update to your evaluation function using temporal-difference (TD) learning for linear function approximators; note that you may need to play around with the learning rate a bit to find a good value; the intermediate reward is 0
- At the end of the game award a +1 or -1 depending on whether your heuristic won or lost

After 1000 games of training, evaluate your heuristic by playing greedily against the random player; report the percentage of wins and losses for your player, and comment on the results. Also comment on any tuning you did during the learning, and overall on your experience in this process.

Turn in your code as well as the write-up