CS 4700 Final Review

1. Policy iteration
   1. There are number of policies in an environment, where is the number of actions and is the number of states.

We know that the optimal policy converges here with and , so solve and , which gives us .

1. Q-Learning
   1. Then the update formula would be . Since , then the updated Q-value will **increase**.
   2. No, it will never be higher or lower, but equal to to cancel out with it because it converged already. This is only in the deterministic setting. In the stochastic setting, higher or lower than is still possible.
2. Multi-Armed Bandits
   1. You add that term to consider how much you’ve explored a machine. The less you’ve pulled a machine, the greater its is and the more optimistic you are about its reward. The more you’ve pulled it, the smaller its is and the closer it is to its actual average value.
   2. An optimistic upper bound on how much reward Machine returns. It is roughly our current estimate of plus one deviation
   3. So the numerator and thus the grows more slowly the more you pull the machines. Ok that is completely wrong. It’s because so guarantees a regret bound that grows logarithmic with , and that it allows our reward to be “one deviation” above the mean.
   4. WTF is the Gittins index. Because the Gittins index is difficult or impossible to calculate sometimes.
3. Monte Carlo Tree Search

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | 2 | 2 | 0 |
|  | 2 | 3 | 1 |

* 1. ^;
  2. Because it makes the higher. It means its games won to games played ratio is very high.
  3. Because it makes the UCB higher. It means the ratio of number of all games played to number of games played on that machine is high, so is relatively low, so we would explore that machine; it encourages exploration.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | 1 | 0 | 0 |
|  |  |  |  |

* 1. ^

1. Perceptron
   1. False. An that’s too large may overstep (more misclassifications in each iteration by a decision boundary that rotates too much), causing slower converge.
   2. True. Perceptron converges within steps, where all the data is within a ball of radius and the data has a margin .
   3. The bias term stays the same. Negate the weight vector. Add some small value to the bias to shift the points on the boundary back to the correct labelling.
   4. Running Perceptron on the dataset:

error

; error

; error

1. Logistic Regression
   1. Multiply by : .
   2. Naïve Bayes returns the label of a data point, logistic regression returns the likelihood of a certain label to appear on a data point. On top of that, in Naïve Bayes we have a procedure for computing based on the data, while in Logistic Regression we simply find the best by optimization.
   3. A heck lot of math:

Hint 1 says our original , since , so we get rid of the in the denominator by integrating it into the summation.

Rewrite this as

Hint 2 says to multiply both the numerator and denominator by

Therefore,

* 1. Because Logistic Regression finds ’s using optimization by gradient descent, so there’s still room for error, i.e. they are calculated differently. There are infinitely many equivalent assignments which yield the same probability if you scale the weights classes differently.

1. Neural Nets and Backpropagation
   1. Later
2. Naïve Bayes
   1. Later
3. Logic