Problem Set 4

Lars, Runar and Årne

March 2024

1 Function Approximation

1. (*) What is the difference between using a linear function for feature engineering and function approximation?

When doing feature engineering we are looking for a linear mapping from a state to a set of features which can be used to perform computations. This allows us to find useful features of states or find similarities or differences between states. Function approximation however tries to approximate a linear mapping from input features to state or action values.

3. (*) Consider the policy improvement theorem that we discussed in the context of dynamic programming. Why do the guarantees of this theorem fail to apply in case of function approximation? What step of the proof does not hold anymore?

When performing evaluation with function approximation we update our estimated weights at each time step to minimize our MSVE for state s. As such for a single time step at state s with weights w when we update w to w' it is guaranteed that MSVE(s,w') $\leq$ MSVE(s,w) however we have no guarantee that for all s' in S that MSVE(s',w') $\leq$ MSVE(s',w).

4. (*) Suppose you have been training the Mars Rover 2.0 robot using the tabular TD(λ) algorithm and relying on off-policy data gathered by the Mars Rover 1.0. A colleague suggests that to increase the performance of the Mars Rover 2.0 you should use a neural network to approximate its action-value function. What sort of challenge would you expect to face if you were to follow the suggestion of your colleague?

Neural network generally require large amounts of data in order to reduce the variance of the fitted model. So we would need sufficient data in order to get a good approximation. Deep neural networks can also be time consuming to train and maintain. If we can overcome those constraints however neural networks generally have high capacity for generalization which would improve the rovers abillity to adapt to new situations far better than a tabular implementation. It also requires us to make sure our value function is differentiable as that is a requirement for train neural networks. Lastly neural networks have many challenges regardless of the overall problem that might cause difficulty. Architecture and hyperparameter tuning, overfitting/underfitting and vanishing/exploding gradients to name a few.

2 Function Approximation

For all tasks $\gamma = 1$

1. What would be v_π(uni) and v_π(home) computed using MC?

$v_\pi(uni) = 6/8$
$v_\pi(home) = 0$

2. What would be v_π(uni) and v_π(home) computed using TD?

$v_\pi(uni) = 6/8$
$v_\pi(home) = 6/7$

3. (*) Are the state values computed by MC and TD identical? If not, can you explain why not?

The values computed are not identical. This is due to the way the TD target is computed compared to the return in MC. In MC we consider the reward for the current state and future states in the same trajectory, however in TD we consider the reward for the current state as well as the state value of the next state. So in this case since uni the state value of uni is 6/7 at τ_8 and the reward is 0, the home state has state value 0 in MC and state value 6/7 with TD.

12. (*) How do the state-values you have learned with DP compare to the ones learned by MC and TD? Are they identical or not? If not, can you explain why not?

Dynamic programming gave $v_\pi(uni) = 6/8$ and $v_\pi(home) = 6/8$. This is slightly different from TD due to the fact that TD runs through the collected trajectory once so when the home state is encountered, not all of the uni states have been encountered so the state value of home is set before the state value of uni is finalized. With DP however we repeat until the state values do not change so the final state values are identical as the only reward for home is by transitiong to uni.

13. Compute again v_π(uni) and v_π(home) using MC from real and simulated data.

$v_\pi(uni) = 6/8$
$v_\pi(home) = 1/2$

14. Compute again v_π(uni) and v_π(home) using TD from real and simulated data.

$v_\pi(uni) = 6/8$
$v_\pi(home) = 667/840$

15. 15. (*) Did the values computed by MC and TD change? If so, how did they change and why?

For Both MC and TD the values for uni stayed the same while the values for home increased. This is due to the original data only having one trajectory with state home leading to uni with reward zero. The simulated data contained more cases of state home leading to state uni both with reward 0 and 1. For MC this means that we actually see the home state leading to reward giving higher state value. For TD when we saw home only once the state value for home became exactly the state value of uni at the time step where we evaluated home. However now that we home multiple times the state value of uni varies each time we see home which changes our estimate.

3 Monte Carlo Tree Search

1. (*) Suppose you are training an agent to drive a self-driving car. A colleague, who has recently read about Monte Carlo Tree Search, suggests that the implementation of MCTS in your problem could improve the quality of the driving. What do you think of this suggestion? What considerations would you need to make to decide whether the use of MCTS is justified or not?

Generally MCTSs are generally employed in problems with discreet state spaces such as board games and other turn based competitions. The computational cost for each action is often high which translates poorly to a traffic situation where quick decisions are required. Such a situation is generally difficult to split into discreet timesteps. Furthermore MCTSs simulate possible futures in order to estimate which course of action is optimal. This means that we need our model to markovian in order to accurately estimate future states. However in a traffic situation with a lot moving parts it might be infeasible in practice to have state representations that are complex enough to make the model markovian. If we would like to implement MCTS we would need to consider whether we can make an accurate markovian model of the world and the computational feasibility of simulating a large amount of possible futures within the short window of time one has to make decision when driving. On the other hand MCTS have been shown to be quite good at adapting to non-deterministic environments due their simulating of future trajectories.