CMPUT 655 Assignment 7

Seth Akins

October 2024

1 Theory Questions

1.1 Differences Between Reinforcement Learning and Supervised Learning Regression

In reinforcement learning, supervised learning regression appears when trying to fit a function approximation to our dataset. In supervised learning, we have a set of $\{x_i, y_i\}_{i=1...N}$ samples with our x_i's being the input and the y_i's being the output, and we try to fit a function to the data so that given an input x_j we can predict an accurate estimate $\hat{y}_j$. In RL, we try to fit a function to our data, but our data is of the form:

$$\{(s_i, a_i), r_i + \max_{a'} Q_i(s'_i, a')\}$$

In RL, we try to repeatedly fit a function to this data, but we run into several issues. One of the problems is not having IID data. IID means that our data is identically and independently distributed, or that:

- $Pr(X, Y) \sim x_i, y_i, \forall i$
- x_i is independent from $x_j, \forall i \neq j$
- All of our data is drawn from the same distribution (even if we have separate testing and training data).

This is often not the case in an RL environment, because we sample data from our environment based on our current policy. As such, the distribution of data we see often changes with time.

This lack of stationary data is another issue. In supervised learning, we have a stationary (unchanging) dataset used to fit a function. In RL, as mentioned above, the dataset changes as the agent interacts with the environment, and changes the policy it follows. As such, the set of data we train on is constantly changing. There is a similar problem with the loss function because, by definition, our loss depends on the value of our y_i's, but this changes as the approximation for our Q function changes.

As already mentioned, reinforcement learning often violates these assumptions for several reasons. In RL, our data is typically correlated because the policy we follow will be more likely to produce certain sequences of states. This means our data is not independent, and the distribution we draw data from changes based on the current policy. As a result of this, the data is not stationary, as we typically constantly interact with the environment to gather more data.

1.2 Comparison of FQI with Tabular Methods

One of the main advantages of FQI over tabular Q-Learning is that FQI can generalize between states, and to states it has never seen. This means we can still use RL to solve problems in large state spaces where tabular methods become computationally intractable. A disadvantage of FQI is we have no guarantee of convergence, as we do in tabular methods (still under some assumptions like a small enough learning rate) and we also are unable to produce the exact optimal value for each state.

Formalizing our RL problem as a SL regression implies that we need our data to be IID and both our loss and data are stationary. This takes away some of the power of RL, such as being able to learn online, as our data and loss must be stationary to satisfy these conditions, and online learning will change the distribution of our data. Subsequent samples in RL are also correlated, which means that when we learn online, our data is also not IID.

2 Coding Implementation

All the code and graphs for this assignment can be found here.

3 Discussion of FQI Algorithm

In FQI, fixing our target guarantees that our loss is stationary. This is because we use the value of Q as a portion of the value of our y_i's in RL, so the loss would change if we did not fix the target.

Doing FQI as an offline batch algorithm gets us the guarantee that our data is fixed. This is because our distribution of data would change if we fit our function online, whereas the data is drawn from the same distribution if we do a batch update instead.

We could use importance sampling to correct our estimates. The Q function we are using to compute Q values is different than the Q function that we used to generate the ϵ-greedy policy we used to generate the data, and importance sampling allows us to reconcile this difference.

In our case, this does not matter because we are doing linear function approximation with a quadratic loss. As such, we have one local optima which as also the global optima, so we are guaranteed to converge to it (with the right assumptions on learning rate and such).

This special case is the Q-Learning algorithm, just done with function approximation instead of with a tabular method. We get one sample, compute our target by taking the max of the next state-action pair, and take one step with the gradient the update our estimate of Q.

4 Discussion of Coding Results

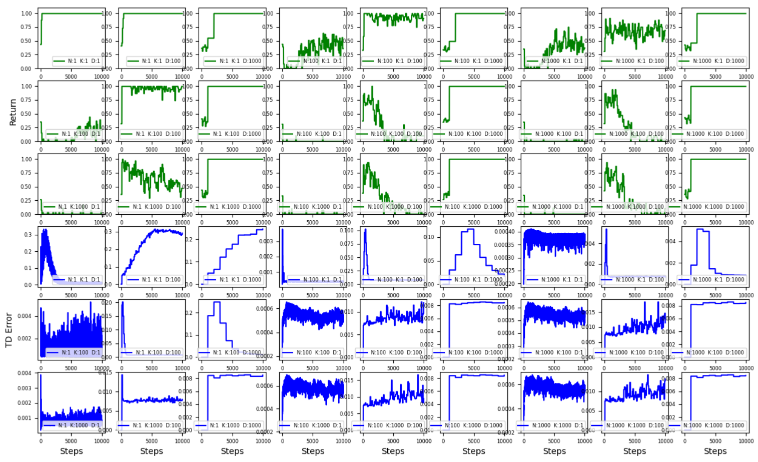

Figure 1: FQI Return and TD Error for Various Hyperparameters

The hyperparameter that matters most for this environment is D. This is because our environment has some stochasticity, so whenever an action is taken, we have a chance to take a random action instead. As such, with a higher number of data samples, we effectively average out this stochasticity in our environment.

Similarly, if out environment is now deterministic, we should require less samples to learn, because every time our agent takes an action, that is the action that actually happens in the environment. We need a larger amount of data to average out the stocahsticity, but that is not required now.

The problem happening is overfitting. If we have a very large number of fits and gradient steps on a small set of data, we will overfit to that specific set of data, and fail to generalize to other cases. This could be especially bad in this environment due to the chance of taking a random action.

It is more important for us to have fewer fits of the function than to have fewer updates per fit. This is because with fewer fits, we move our weights closer to the TD target with a larger value of N, and as such, the difference in the values of our TD Target between fits will be smaller. As a result, each fit changes the function less and less. Less fits means that our function will not be taken as far in a sub-optimal direction, and more of the underlying behaviour from previous fits will be preserved.

The results could be much different with a non-linear function, because we are no longer have one local optima which is also the global optima. Depending on our initial weights, we may become trapped in a bad local optima, or become stuck somewhere where the gradient is near zero.