CMSC 818B Homework 3

Krishna Bhatu
UID: 116304764

Siddhesh Rane
UID: 116147286

Explain and justify 1 which kernel function you used in the case of GP or the architecture that you selected for your NN

We used a multi layer perceptron as the function appropriator to predict the Q values for the given set of states. The network used for this homework was a 2 layered perceptron with 128 nodes in the hidden layer.

To incorporate non-linearity in our model we are using ReLu activation function. We choose ReLu for the following 2 reasons:
1. ReLu generates a sparse representation of the input where all the hidden nodes are not activated (where A <= 0). Whereas other activation functions gives a dense representation because all the neurons are activated in every forward pass. This helps us reducing the computation of the network and let us avoid unnecessary gradients to pass.
2. ReLu also avoids the problem of vanishing gradients as it has no upper limit to the activation. Thus it tends to give constant gradients which assists in faster convergence.

We are using only one hidden layer because the input just has 2 dimensions so we don't need more than 1 hidden layer as it will only increase the computation. Selecting 128 nodes was just like a hyper-parameter selection. Theoretically increasing the number of hidden nodes to infinity will give the exact Q value function. But with a good enough number of nodes, we can get a good enough approximation of Q value.

Options Considered:

- *We started the implementation using bigger models with three hidden layers with the number of hidden nodes (128, 48, 48) respectively. Such model took a lot of time to train. Also, the results were not good enough after 10k episodes. Then we realized that the deeper we build our network the better estimate we get only if we have a huge enough*

Krishna Bhatu **Siddhesh Rane**
UID: 116304764 **UID: 116147286**

training data. Thus, our network was facing the problem of the curse of dimensionality. Therefore, we though of using low dimensional network with just one hidden layer and experiment with the number of layers in ascending order. But we good our result with one layer so we kept that network.

- *We considered using the sigmoid function as our activation. But while training we found that the weights were not getting updated. So on looking up the gradients we found that they were getting zeroed out by dead neurons. Thus the network was not learning after some iterations. Then we switched to ReLu which we found avoids such problem.*
- *We tried using the same network to predict the target Q values and the output Q value. But this gave us unstable results. So, we switched to a two networked approach. Where the Q target predictions are predicted from the target network which is being updated at every iteration and the output Q value is found from a frozen network. Thus the weight updates is in the direction of target Q values which will ensure convergence.*

Explain and justify your exploration-exploitation strategy (15 points).

- *As we are getting reward only when we reach the goal state. Thus with the random initialization of network weights, the function approximation will output random Q values, the car will perform random action in this unknown environment. At this point the car doesn't even know where the goal exist and how will it reach the goal. So, we rely on the random exploration to reach the goal for us and hence we can update the weights in the proper direction.*
- *Thus, we start with 100% exploration and let the random action find the goal for us. But we do not want to explore all the time, eventually we will have to exploit the model and converge to the target Q values.*

CMSC 818B Homework 3

Krishna Bhatu
UID: 116304764

Siddhesh Rane
UID: 116147286

So, we start with the exploration factor (epsilon) = 1.0 and decrement it with 0.9% of the current epsilon value after every 35 episodes.

- *Our model is updated after every 7 episodes. We run the process for 10k episodes. Thus, after around 8k episodes, the model was completely exploiting the environment.*
- *We observed that even with 10k episodes, the model was not converged. And as our policy was only exploitation, we couldn't continue the training with the same policy as the model would not converge without exploring more.*
- *So, we used the weights that we got after 10k episodes and used it initialize the training again, starting with the same policy of exploration first and then exploitation.*
- *Thus, the output we got after this new 10k episodes of training was converged to the true target values. Hence, we declared the model to be trained.*

Explain and justify your choice of all other hyperparameters (15 points): in particular, discount
factor, learning rate, loss function, kernel hyperparameters, termination condition

- *Discount Factor that we selected was 0.99, we selected this because as the environment only gives reward when the goal is reached, so in order to learn the process to reach the goal. So due to the rare cases of rewards, the future rewards are also important in order to learn the process to reach goal. Hence, we select a high value for discount.*
- *Learning rate that we select is 0.01, we select a low learning rate because we don't want to learn the randomness in the environment which will govern majority of the episodes. Thus by taking higher value of learning rate, the model would become unstable and would not guarantee convergence because for majority part of training the car*

CMSC 818B Homework 3

Krishna Bhatu **Siddhesh Rane**
UID: 116304764 **UID: 116147286**

would not reach the goal. Thus we select this small value to make small steps towards convergence.

- *Loss Function that we use is mean squared loss. We did experiment with this parameter. We knew that we had to bring the predicted Q value as close as the target Q value, so we used the difference error. And we used the squared term to ensure that the gradient updation in the direction of minima.*
- *Termination Condition. We stopped training after 20k episodes, this was based on our observation that the the car was reaching the goal for every iteration and the reward was not improving further.*

In the pdf report, report how well did the function approximator approximate the Q-values. You can
do this by reporting, for example, the mean square error for training data

The mean square error for the training data ran over 1000 episodes with 0 exploration was found to be 0.1054

Result Plots

The following results are performed while training of the network, where the initial training of 10k episodes was already done and this is the 2nd half of training where we load the weights learn in the first half. This training was also done for 10k episodes with the same hyperparameters as mentioned above.

CMSC 818B Homework 3

Krishna Bhatu
UID: 116304764

Siddhesh Rane
UID: 116147286

CMSC 818B Homework 3

Krishna Bhatu
UID: 116304764

Siddhesh Rane
UID: 116147286

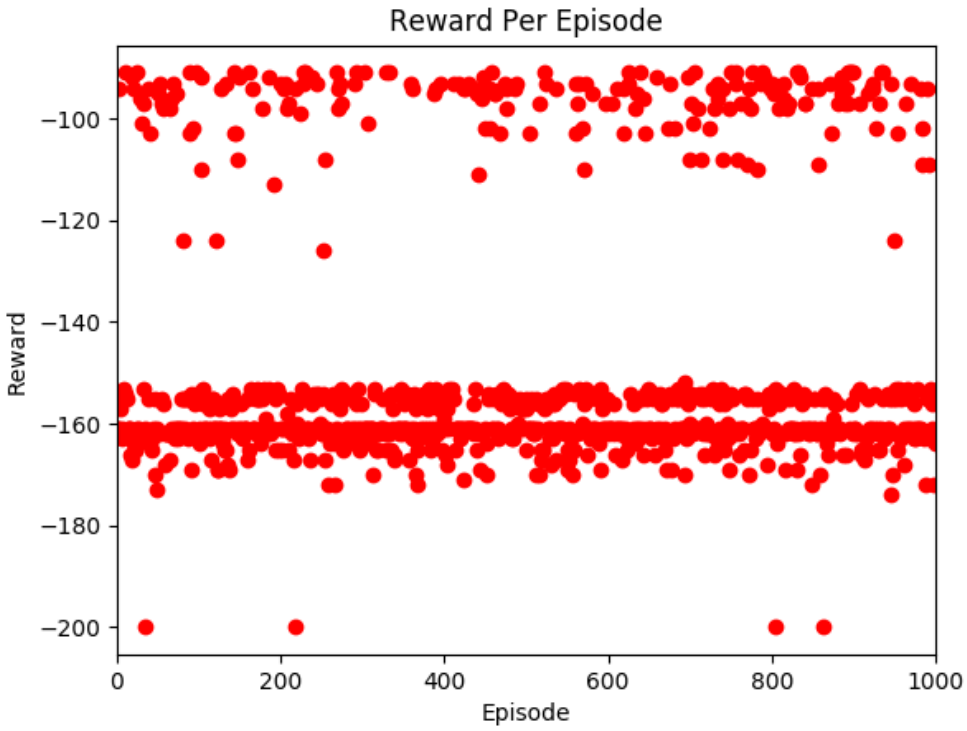
From the above graph we can see that the running average of reward drops below -175 at the 6639 th episode.

CMSC 818B Homework 3

Krishna Bhatu
UID: 116304764

Siddhesh Rane
UID: 116147286

The following results are running the learnt policy with epsilon set to 0, so there is no exploration. And this was running for 1000 episodes.



CMSC 818B Homework 3

Krishna Bhatu
UID: 116304764

Siddhesh Rane
UID: 116147286