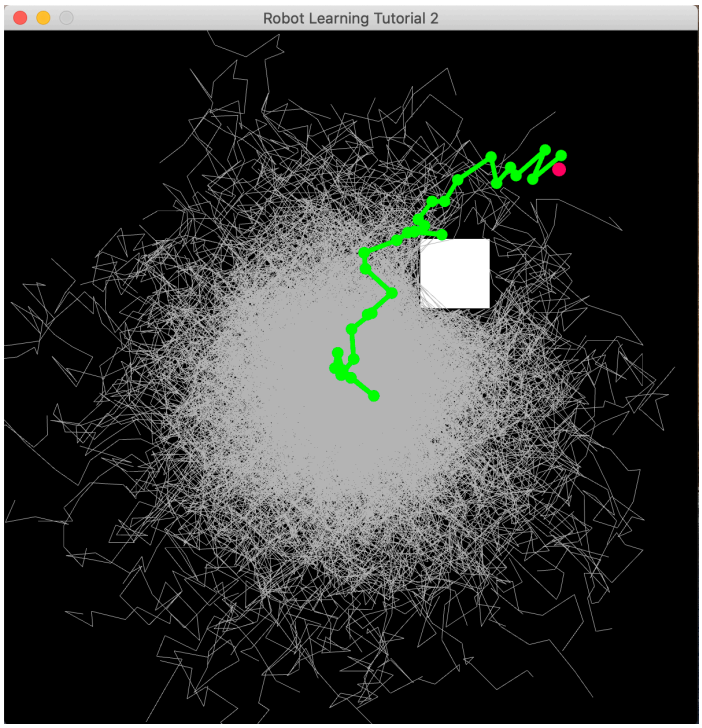
Benjamin Leroy

02107259

Coursework 1

Section 1: Random Shooting

Figure 1



With:

The red point: the goal

The white square: the obstacle

The grey lines: the sampled paths

The green line: the best path

Parameters:

1000 samples of 30 actions each

Question 1a: As the number of sampled action sequences increases, is it more likely or less likely that the agent will be able to plan a path to the goal? Explain your answer.

If we fix the number of actions per sequence and increase the number of sampled sequences, then we increase the probability of the agent to reach to goal. This is due to the fact that for a certain number of actions there's a finite number of ways to reach the goal. Therefore if we increase the number of sequence we give the agent more chance to find one of those sequence of actions that allows the agent to reach the goal. However, we need to suppose that the settings such as the number of actions and the length of the moves enable the agent to reach the goal.

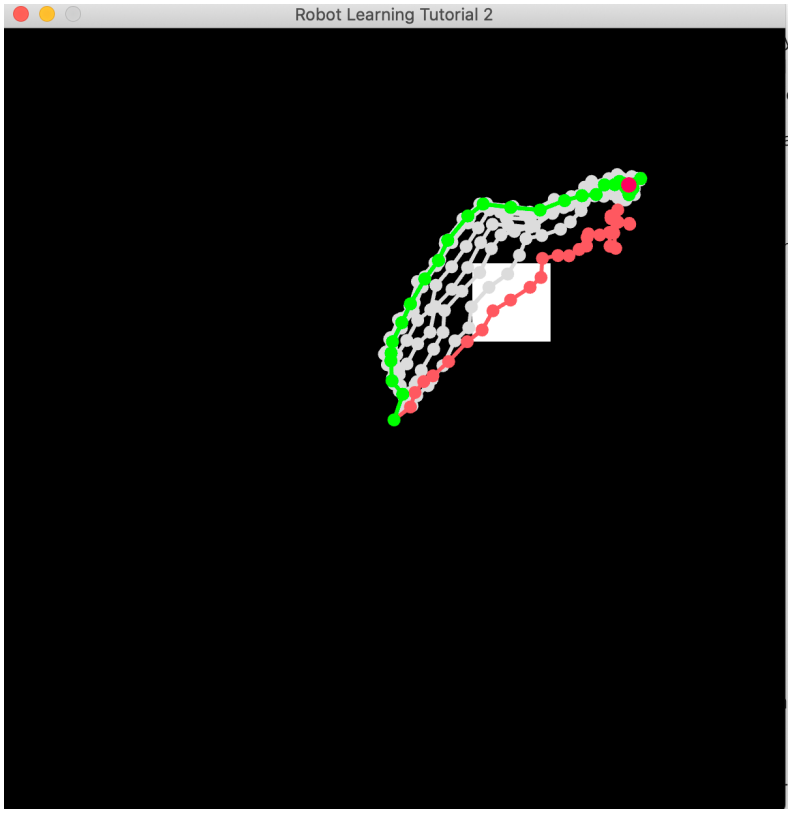
Question 1b: As the number of actions in each sampled sequence increases, is it more likely or less likely that the agent will be able to plan a path to the goal? Explain your answer.

If we fix the number of sequences but we increase the number of actions per sequence then it's more likely that the agent will be able to plan a path to the goal. From one state there's a finite number of ways to find the goal in less than x actions. Increasing the number of actions gives more "time" to the agent to find the goal because it can execute more actions than

before. Before he could have been too far from the goal to be able to reach it and now he might be close enough and will be able to reach it if he executes the good actions.

Section 2: Cross Entropy Method

Figure 2



With:
The red point: the goal

The white square: the obstacle

The red line: the path for the first iteration

The green line: the path for the last iteration

The grey lines: the other paths

Parameters:
Refitted 10 times to the 5% best paths

1000 samples of 30 actions each

Question 2a: Is the Cross Entropy Method usually more computationally efficient, or less computationally efficient, than Random Shooting, when used for robot planning? Explain your answer.

To execute the random shooting method, we create sequences of actions by taking a random action from all the possibilities. That is done for num_actions_sequences x num_actions_per_sequence times.

On the contrary for the cross entropy method we create sequences of actions using gaussian distribution but we furthermore refit the distribution to the top few action sequences. That is done for number_times_refit x num_actions_sequences x num_actions_per_sequence times.

Therefore cross entropy method is less computationally efficient than random shooting even if it gives better solutions.

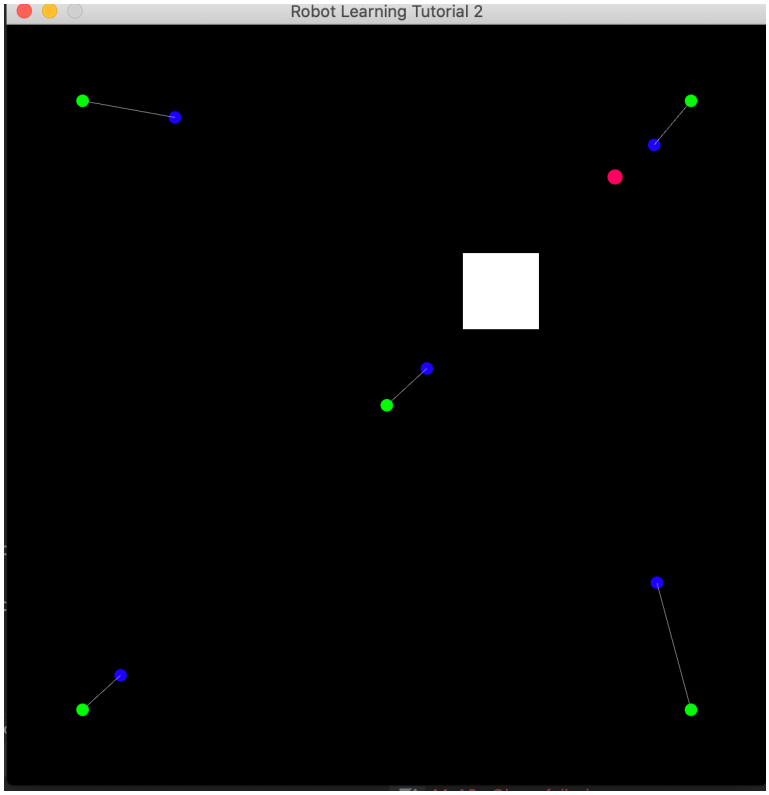
Question 2b: The covariance used to sample action sequences could be a diagonal covariance matrix (all elements are zero apart from the diagonals) or it could be a dense covariance matrix (all elements can be non-zero). When planning with the Cross Entropy Method, are there scenarios when using a dense covariance matrix would be

better than using a diagonal covariance matrix, or would there be no benefit to this? Explain your answer.

Having a dense density matrix allows to see the correlation between the movements on x and on y. Therefore, when we need to avoid an obstacle it's a good idea to use a dense covariance. By using a dense matrix, he will learn a more efficient path faster than without a diagonal matrix. However, a diagonal matrix can be useful when we want to keep exploring the environment. Nevertheless, we should not forget that computing a full covariance matrix is more computationally expensive than computing a diagonal matrix.

Section 3: Model Learning

Figure 3



With:
The red point: the goal

The white square: the obstacle

The green points: the initial states

The blue points: the predicted states

The grey lines: the link between the initial and the predicted states

Parameters:
1000 samples of 30 actions each

Trained on 750 iterations

Learning rate: 0.01

Batch size: 300

Question 3a: Rather than randomly exploring the environment to collect data, what might be a better strategy to ensure that the data is better aligned with the optimal policy?

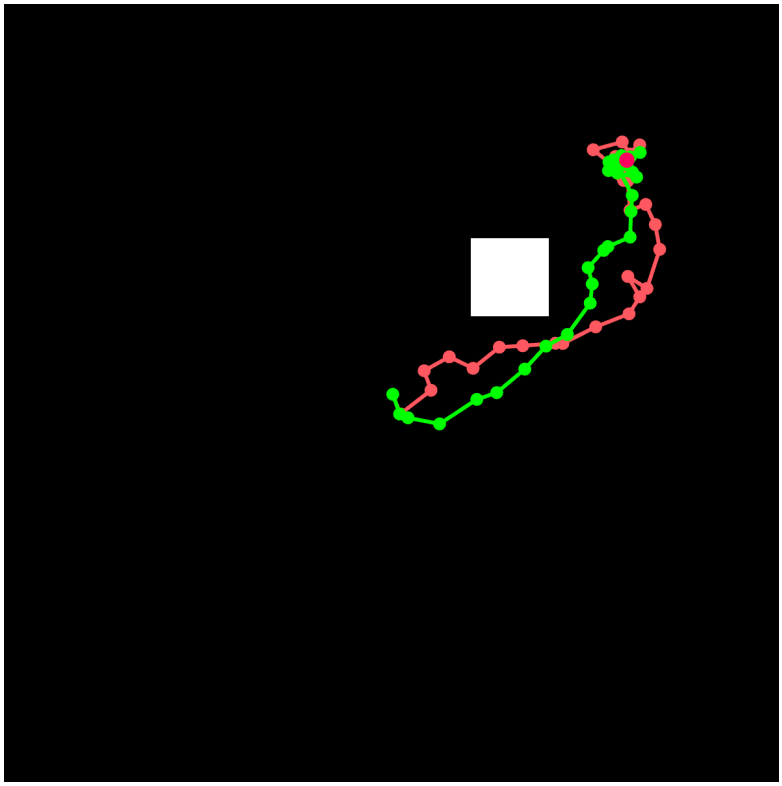
A better idea than randomly exploring the environment to collect data would be to use the cross-entropy method. This method will allow to explore the environment at the beginning and then use the best solutions to get closer to the goal and keep exploring the ways to reach it action after action by sampling from a gaussian distribution. Therefore, the collected data will be better aligned with the optimal policy.

Question 3b: For each of the above 5 robot states (0.1, 0.1), (0.1, 0.9), (0.9, 0.1), (0.9, 0.9), (0.5, 0.5), in which state is the learned model likely to be most accurate? In which state is the learned model likely to be least accurate? Explain your answer.

When we create the dataset for training the neural network, we create many sequences of actions and for every sequence the robot starts at the state (0.5,0.5). Therefore, we could say that the dataset is a bit unbalanced because many inputs (which are the concatenation of the states and the actions) have the same state (0.5,0.5) as part of their definition. The neural network will have many examples of actions for the state (0.5,0.5) so it's for that state that the prediction will be most accurate.

Section 4: Model Predictive Control

Figure 4



With:
The red point: the goal

The white square: the obstacle

The green line: the closed loop path

The red line: the open loop path

Parameters:
1000 samples of 30 actions each

Trained on 750 iterations

Learning rate: 0.01

Batch size: 300

Refitted 10 times to the 5% best paths

Question 4a: Why is it helpful for the agent to use Model Predictive Control now, compared to your implementation in Section 2?

It's now useful for the agent to use model predictive control because the agent is now executing each action using a neural network. We know that the prediction aren't perfect so the actual path and the predicted path aren't the same. Therefore after each action we need replanning to mitigate that problem and to help the agent to reach the goal.

Question 4b: Is it important that the neural network used to learn the model, can learn a non-linear function? Or is it not important for this environment? Explain your answer.

For this environment it's important to have a neural network learning a non-linear function because when we plot the vector from the initial state to the goal state it goes through the obstacle. Therefore the curve of the path linking those two states isn't representing a linear function. If we can go to the goal with a straight line then the model doesn't have to learn a non-linear function.