Reinforcement Gym

All the raw program output about the cart pole can be found in the *Resut-CartPole.txt* file.

All the raw program output about the mountain car can be found in the *Resut-MountainCar.txt* file.

1 Cart Pole

With the correct implementation of Q-Learning, we can get the total average score as 200. That means we can make the pole stand on the cart every time as long as we want.

Here are the 10 results from policy training and test:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
2	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
3	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
4	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
5	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
6	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
7	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
8	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
9	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
10	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
11	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
12	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
13	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
14	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
15	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
16	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
17	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
18	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
19	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
20	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0

And the average score of the 10 runs are:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Average Score	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0

And the total average score of the run is: 200.0.

2 Mountain Car

2.1 Tune Parameters

In this part we will tune the following parameters to try to get the best result of our learning model:

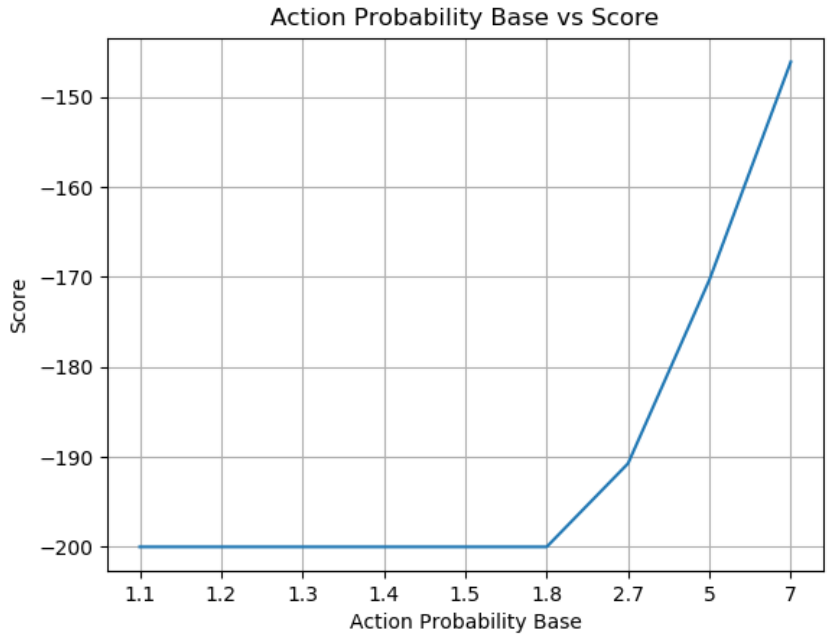
The sweep range for the parameters are:

- Action Probability Base
  - In range [1.1, 1.2, 1.3, 1.4, 1.5, 1.8, 2.7, 5, 7]
- Bins per Dimension
  - In range from 20 to 201 step 10
- Discount Rate
  - In Range [1, 0.99, 0.98, 0.97, 0.96, 0.95, 0.9, 0.8, 0.75]
- Training Iterations
  - In range [20000, 25000, 30000, 35000, 40000, 50000]

2.2 Tuning Results

Here are the results from each parameter tuning.

2.2.1 Action Probability Base



From the results we can find that:

- When the action probability base is small, the learning agent will explore more unseen actions but since there have a limit of 200 timesteps, this might not give us the opportunities to find the correct path.
- When the action probability base is large (for instance: 2.7) the learning agent will try to reuse the known good pass, and this can let us find the correct path within the timesteps limitation.

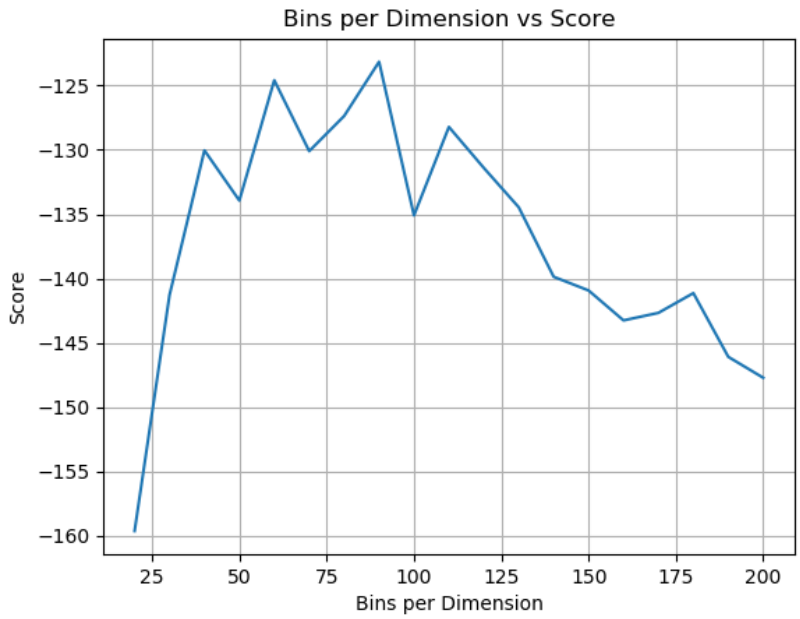
In the following parameter tuning phrase, we will set the action probability base as 7 to boost the learning.

2.2.2 Bins per Dimension

From the chart we have below, we can know that:

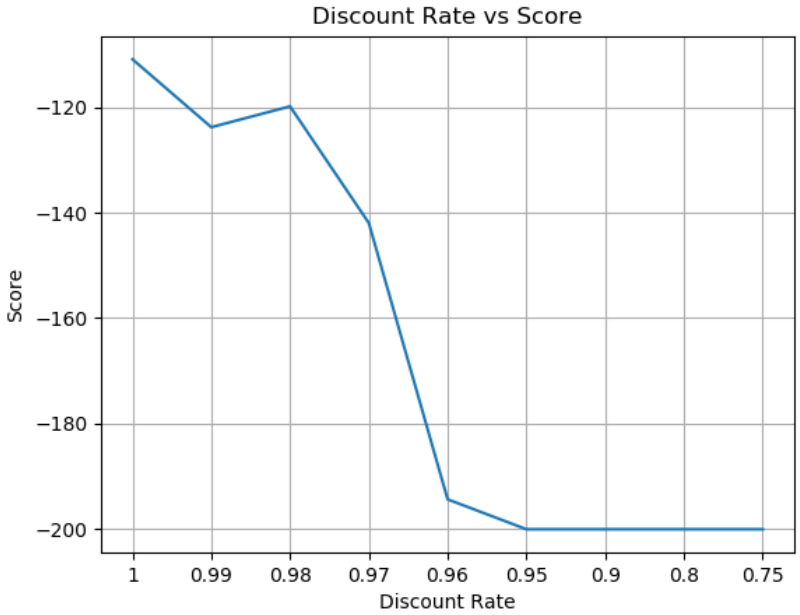
- Small bins will provide less information about the current state, and this can let the learning agent hard to find the correct path to the end.

- Large bins will provide more bias to the learning agent because it provides more information, and since we have the fix training iteration for now, we cannot train the model very well.



In the following parameter tuning phrase, we will set the bins per dimension as 90 to boost the learning. And of cause once we finished this iteration of the parameter sweep, we can understand more about these parameters, and will sweep the range of this parameter which have similar outcomes.

2.2.3 Discount Rate



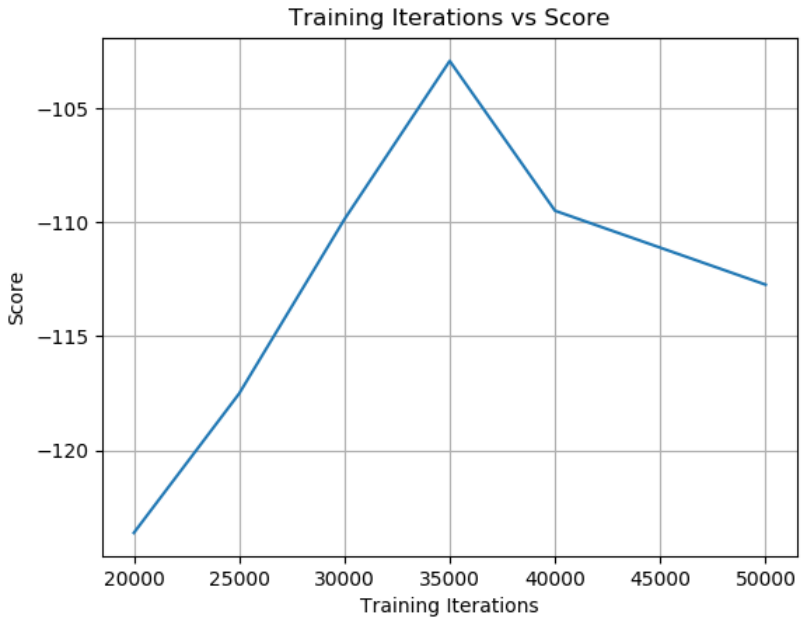
From the curve, we can know that:

- When there has no discount rate, all action performance good will be keep using, and can let us find the correct path within the timesteps limitation.

- With discount rate increase, the final reward might not pass to the previous state and action combination. But this might not give us the opportunities to find the correct path.
- Certain mount of the discount rate will be useful for us to find the final rewards.

In the following parameter tuning phrase, we will set the discount rate as 1 to boost the learning. But for other iteration of the parameter sweep, we can still use the range of this parameter which have similar outcomes.

2.2.4 Training Iterations



From this we can understand that:

- Certain amount of the increase of training iteration will help the learning agent have better performance.
- Not enough iteration or too many iterations will cause the learning agent has underfitting and overfitting situation.

2.3 Result

We can get the best performance so far of the average score of the 10 runs are:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Average Score	-99.85	-97.2	-100.2	-102.8	-106.35	-110.35	-99.0	-100.0	-100.05	-113.5

The total average score of the run is: -102.929999. And in here we have the average score for best single policy training as -97.2.

Here are the parameters to reach this.:

- Training Iterations 35000
- Action Probability Base 7
- Discount Rate 1

- Bins per Dimension 90

3 Improve the Model

3.1 More Parameter Sweep

After understanding more about our parameter, we can have another round of parameter sweep iteration with a denser parameter matrix.

Here is the sweeping range:

- Action Probability Base
  - In range [5, 7, 11, 13]
- Bins per Dimension
  - In range from 20 to 101 step 10
- Discount Rate
  - In Range [1, 0.99, 0.98]
- Training Iterations
  - In range [30000, 35000, 40000]
- Random Action Rate
  - In range [0.01, 0.02, 0.03, 0.05, 0.07]
- Learning Rate Scale
  - In range [0.01, 0.02, 0.03, 0.05, 0.07]

We can get a better performance, the average score of the 10 runs isare

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Average Score	-96.55	-104.1	-104.4	-95.9	-96.0	-104.4	-99.9	-109.45	-109.5	-98.5

The total average score of the run is: -101.869999. And in here we get the average score for best single policy training as -95.9.

Here are the parameters to reach this.:

- Training Iterations 30000
- Action Probability Base 7
- Discount Rate 1
- Bins per Dimension 50
- Random Action Rate 0.01
- Learning Rate Scale 0.01

Also, from this iteration of parameter sweep, we can understand more about the relationship between the parameters, here are some fun facts:

- The change of Learning Rate Scale won't affect the outcome very much.
  - The reason of this I think is because the values we choose aren't very large and therefore change of this value won't change the learning rate too much.
- The change of Random Action Rate will affect the outcome.

    - It adds the randomness to choose not the best route for exploration, during the start of training this will be helpful but will also increase the chance failed to find the right move.
- With larger Action Probability Base, have Discount Rate will performance better than without it.
    - Larger Action Probability Base will increase the probability to choose known good move. But if there have no discount rate, the move might actually take longer steps to reach the goal, that's will give us the bad outcome.

3.2 Update the Model - Memory

We can also add the memory for our Q Learning Agent, this can give the learning agent easier to find the way to the goal while save us lots of the training time.

With the memory, we can have another round of parameter sweep iteration with the range:

- Training Iterations
    - In range [30000, 35000]
- Bins per Dimension
    - In range from 20 to 91 step 10
- Action Probability Base
    - In range [2, 2.7, 5, 7, 11]
- Discount Rate
    - In Range [1, 0.99, 0.98]

We can get the best average score of the 10 runs from this iteration are:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Average Score	-96.65	-102.15	-98.35	-96.5	-113.15	-112.55	-98.2	-99.75	-108.8	-113.55

The total average score of the run is: -103.965. And in here we get the average score for best single policy training as -96.5.

Here are the parameters to reach this.:

- Training Iterations 30000
- Action Probability Base 5
- Discount Rate 0.99
- Bins per Dimension 50
- Random Action Rate 0.01
- Learning Rate Scale 0.01

When we put Action Probability Base as 7, we can get a similar average score result:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Average Score	-105.2	-104.8	-107.55	-100.5	-103.05	-107.1	-100.55	-105.4	-105.6	-108.6

The total average score of the run is: -104.835. And in here we get the average score for best single policy training as -100.5.

Some fun facts about the Learning Agent with Memory:

- For small Action Probability Base, even previously don't have very good performance, with memory, it can still performance very well.
    - Such as, when Action Probability Base is 2. We don't have the score of this but form the chart we can say it should below -190. Now can get 10 runs average as -110.14, and highest single run can even reach -98.35. And for Action Probability Base is 2.7, previously the average score is -190, but with memory, it can reach -108.51 as the average now.
    - The reason should be: once the learning agent find a way to get the final goal, the memory and replay logic will enhance the value inside of Q table.
- After we add the memory to our learning agent, the agent with discount rate will give better performance than the one without.
    - Such as, when Action Probability Base is 5, we can get the average score as -109.55 without the discount rate while we can get the average score as -103.965 when we have the discount rate as 0.99. This also happened for other situations, for example: when Action Probability Base is 7, we can get -113.3 vs -104.835.

4 Other Approach

Just a thought, since the game's rule is already predefined and quite simple, maybe we can directly write a program to complete this game without any Machine Learning model. Here are the 3 simple rules to for the program.

- If the velocity is negative (move to left), then push left.
- If the velocity is positive (move to right), then push right.
- If is stop now, push right.