# Bipedal Walker RL Project Presentation

Mohammad Hossein Abbaspour / Sina Zamani

## Introduction

A walker with two moving legs is given and we can pass a random action and receive a bunch of variables indicating agent’s state.

Our goal is to help the agent learn from it’s experiences and perform optimal action.

We have tried four different methods to help our agent accomplish this goal.

## Q-Learning

We store a Q_Value table and use this data to choose the best action in each state.

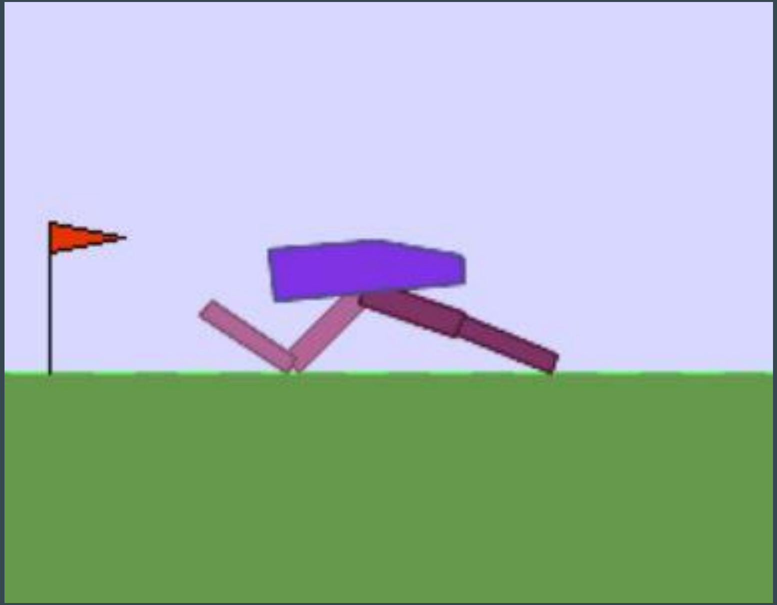
We have implemented two versions of this algorithm.

- At first we thought the observation space is too complicated and our variables are continuous so we can't store our states. Therefore, we built a Q_Table with mere actions.
- The second algorithm is implemented the same as original Q-Learning algorithm. We store tuples of state and action and the reward regarding them.

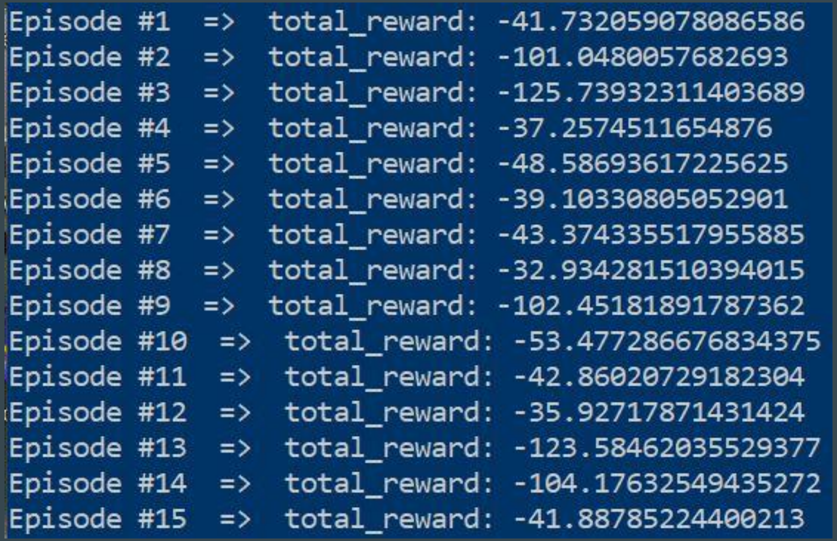
## Results of Q-Learning

**V1**: The first version nearly equals a random algorithm. Because it is not precise and our data is too narrow as we only store actions.

**V2**: The second version needs some time to develop a model that becomes useful in choosing optimal action. After a necessary number of iterations we reach a state that the agent doesn't fall immediately. It kneels on the ground and tries to move forward as it gets positive rewards.



## Rewards of Q-Learning Version_2



```
Episode #1  =>  total_reward: -41.732059078086586
Episode #2  =>  total_reward: -101.0480057682693
Episode #3  =>  total_reward: -125.73932311403689
Episode #4  =>  total_reward: -37.2574511654876
Episode #5  =>  total_reward: -48.58693617225625
Episode #6  =>  total_reward: -39.10330805052901
Episode #7  =>  total_reward: -43.374335517955885
Episode #8  =>  total_reward: -32.934281510394015
Episode #9  =>  total_reward: -102.45181891787362
Episode #10  =>  total_reward: -53.477286676834375
Episode #11  =>  total_reward: -42.86020729182304
Episode #12  =>  total_reward: -35.92717871431424
Episode #13  =>  total_reward: -123.58462035529377
Episode #14  =>  total_reward: -104.17632549435272
Episode #15  =>  total_reward: -41.88785224400213
```

# Approximate Q-Learning

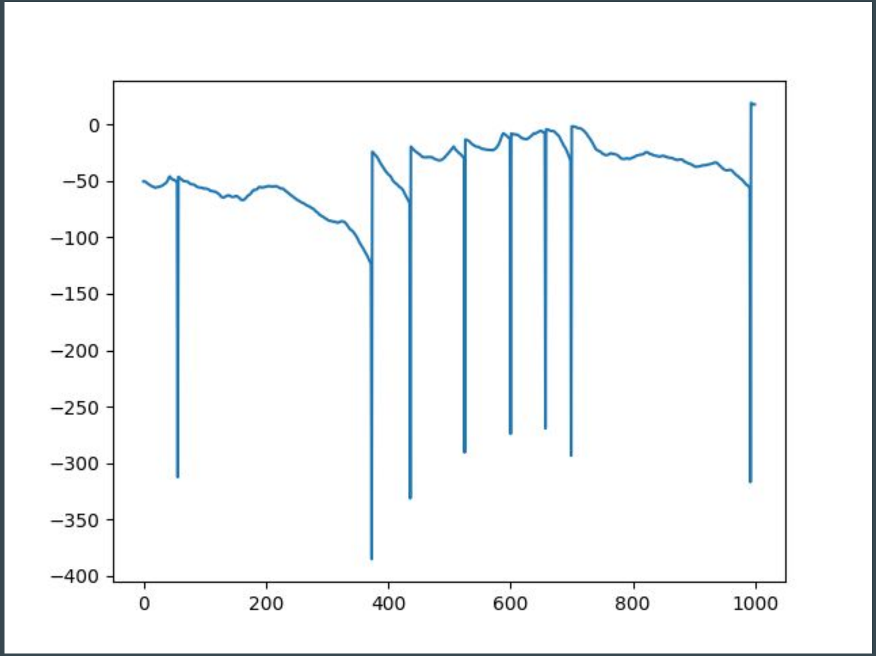
In this approach we don't save the history of states and actions; but, we store some coefficients for our features. About 12 features are considered for this problem which impact our QValue function.

- 1, 2 features are x and y speed of hull, which can be calculated from <hull_angle> and <hull_angularvVelocity>.
- 3, 4, 5, 6 features are x and y speed of hip$_1$ and hip$_2$, that can be calculated from <hip_joint_j_speed>, <hip_joint_j_angle> and speed of hip□ in action.
- 7, 8, 9, 10 features are x and y speed of knee$_1$ and knee$_2$, which can be computed from <knee_joint_j_speed>, <knee_joint_j_angle> and speed of knee□ in action.
- The last two features are x and y velocity of agent.

## Results of Approximate Q-Learning

Our learning model has follow parameters:

- Vector of features
- Vector of numbers (coefficients)
- Learning rate
- Gamma
- Action space
- Epsilon



## Deep Q-network

We tried to use the code available on github to check the performance of this algorithm, but all the codes made errors and due to our limited knowledge in necessary topics, we couldn't resolve them.

## Scikit-learn

We intended to use regression rules to train a model for predicting rewards of actions. But because of shortage of time we didn’t succeed.

Thanks for your attention :)