**Help the reader identify the project**

* Name the file: README.md
* Project’s name is the first heading: Landing on the moon
* Add a URL for the project
* Identify the owner or author of the project:
  + [Coursera] Reinforcement Learning Specialization by “University of Alberta” & “Alberta Machine Intelligence Institute”
  + Steve Vu …….

**Help the reader evaluate the project**

* What the project does – Focus on Why not What

***Introduction***

The project aims at building a simulator that trains a lunar module, thereafter known as an agent, to land on the moon. We frame it as a reinforcement learning problem that the agent learns simply through interaction with the dynamics of the world. The ultimate goal is to find a robust and efficient landing policy for the agent.

As the project simplifies the complexities of outer space, it is a great starting point to gain experience in converting word descriptions of problems into a concrete solution.

***Implementation***

*Create an environment*

The project starts with creating a lunar lander environment that is well developed by [OpenAI](https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py). The goal is to land the lunar module in the landing zone located between the two yellow flags. The landing zone is in the same location, but the shape of the ground around it may change. The agent can fire the main thruster or either of the side thrusters to orient the module and slow its descent.

The state is composed of eight variables. It includes the XY position, the XY velocity, the angle, the angular velocity and the signals from each leg’s sensor that determines if the module is touching the ground.

There are four possible actions as the agent can fire the main thruster, fire the left thruster, fire the right thruster, or do nothing on this step.

About the reward function, we encourage the agent to move towards the goal, pile to the surface in a safe way, use fuel efficiently, and not fly off into outer space.

**Video**

*Choose the learning algorithm*

While the RL textbook, [Reinforcement Learning: An Introduction by Sutton and Barto](https://d18ky98rnyall9.cloudfront.net/H684ot2tTAKvOKLdrTwCaA_0c14658bf3114aa0bb1566195b706212_RLbook2018.pdf?Expires=1627862400&Signature=OiF1YHSFq5lJeoXUMcin6WotE7d9SqCTTie~mG-dh~9Lk8ded7JcGGCd5Q-5wYeh1aFTEtvp-jFC~xgriRCa5BGfZ8L4KmFhCrZXkpZ5gcY31Fi~nhs9Odl29D2zYUEALBunkRH8jLscQQE4XyCiljI~DSv16~31jdsSs1yHvwI_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A), and the RL Specialization provides informative surveys of RL algorithms, we need to find a well-fitted algorithm for the problem. By the end, the Expected SARSA is able to outperform other algorithms in the specific problem.

The Expected SARSA is to deal with continuous state variables and episodic learning process of the problem. It is also beneficial as we are able to update the policy and value function on every time step before the end of the episode with the algorithm. Finally, the Expected SARSA allows us to learn an optimal epsilon soft policy that is more robust than learning a deterministic policy.

**Intuitive pictures**

*Make design choices*

Which function approximation will be employed is one of the most important decision as the number of features grows exponentially with the input dimension. Using neural network is the best choice for the problem. Besides setting the neural network’s parameters, picking an effective activation function is critical. By the end, we go ahead with ReLUs function as it avoids the issue of saturation in neural networks.

**ReLUs**

Training the neural network is not easy as stochastic gradient descent is too slow. We will learn the advantage of the ADAM optimizer which combines the curvature information and momentum to speed up learning.

**ADAM**

The exploration – exploitation tradeoff is a fundamental problem in Reinforcement Learning. While it is difficult to maintain the effect of optimistic initial values on exploration when using neural networks, the epsilon greedy approach has a downside of ignoring the information about the action values. Softmax policy is the best choice as the probability of selecting an action is proportional to the value of the action.

Learning the idea of planning from Dyna-Q that uses simulated experience to improve the value estimates, we would like to employ the experience replay method to make the agent more sample efficient when using function approximation. We simply save a buffer of experience and let the data be the model before using several samples from the buffer, called a mini batch, and updating the value function with those samples.

*Study impacts of meta-parameters*

To create a successful learning agent, it is critical to choose good values for key meta parameters such as the step size, the temperature parameter for the Softmax policy, the capacity of the replay buffer, so on. How to pick reasonable values for meta parameters is an open question. In the project, we will have a chance of understanding the impact of the step size on the performance of the agent while other parameters are fixed.

**Help the reader use the project**

* List the project’s prerequisites

It is highly recommended that you complete [Course 1: Fundamentals of Reinforcement Learning](https://www.coursera.org/learn/fundamentals-of-reinforcement-learning), [Course 2: Sample-Based Learning Methods](https://www.coursera.org/learn/sample-based-learning-methods) and [Course 3: Prediction and Control with Function Approximation](https://www.coursera.org/learn/prediction-control-function-approximation) before starting the project.

You need some basic math knowledge including, basic knowledge and comfort with expected values, distributions, and sampling, as well as basic calculus (derivatives), linear algebra (vectors, matrices, inner products), functions.

* List the steps to install and use the project one time
  + Install libraries