Reinforcement Learning

Deep Learning

- Now it is time to learn about Reinforcement Learning!
- We'll also discuss how to use OpenAI's gym library to

Deep Learning

- OpenAI
    - OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence.

Deep Learning

- OpenAI
    - Backed by YCombinator, Elon Musk,

Reinforcement Learning Overview

Deep Learning

- What is Reinforcement Learning?
  - It allows machines and software agents to automatically determine the ideal behaviour within a specific context, in order to maximize its performance.

Deep Learning

- What is Reinforcement Learning?
    - Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal.

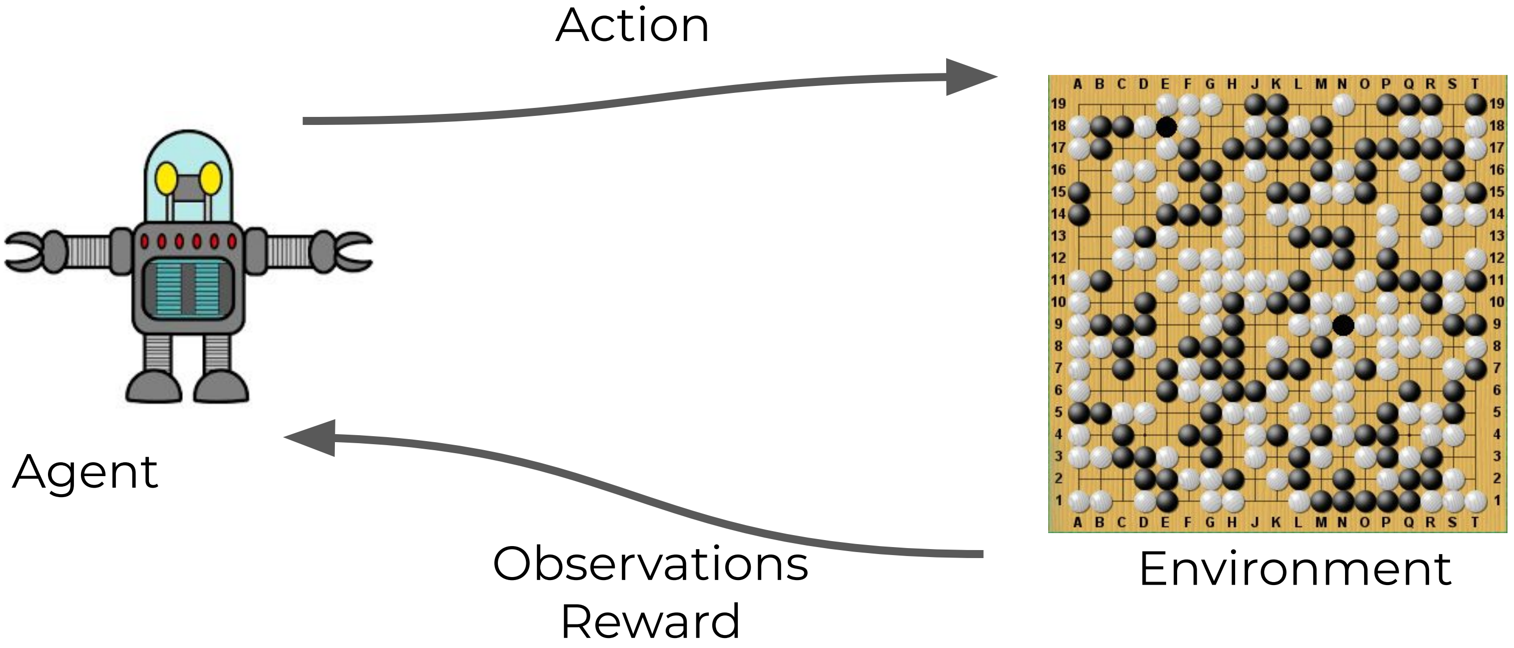
Deep Learning

- What is Reinforcement Learning?
  - There are many different types of algorithms that fall under Reinforcement Learning.
  - Not all of them require a framework such as TensorFlow!

Deep Learning

- What is Reinforcement Learning?
    - Agent
    - Environment
    - Action
    - Reward

Deep Learning



Deep Learning

- Agent
  - Our program or bot
  - Can receive inputs based off the environment
  - Performs actions

- Environment
  - The actual setting the agent is interacting with.
  - Often a game in examples, but it can be any real world or artificial environment.

Deep Learning

- Environment
  - Keep in mind you need to be able to represent this environment in a way an agent can understand (probably needs to be an array in the end)

Deep Learning

- Environment
  - Previously it was very difficult to create environments that were easy to use and shareable.
  - Later on we'll discover how OpenAI's gym library solves this!

Deep Learning

- Action
  - The actual interaction your agent will perform on the environment.
  - Moving in an environment, choosing the next move in a game, etc...

Deep Learning

- Reward
  - The metric that allows your agent to understand whether or not the previous sets of actions helped or hurt in its overall goal.

Deep Learning

- These four aspects are fundamental to Reinforcement Learning.
- The OpenAI Gym works with variables designed to fit within this framework, allowing us to focus on model building.

Deep Learning

- Keep in mind, Reinforcement Learning isn't just for games!
- Games are just an easy way to clearly show all the major aspects of reinforcement learning.

OpenAI Gym

Deep Learning

- OpenAI
    - OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence.

Deep Learning

- OpenAI Gym
  - Toolkit that aids in developing and comparing reinforcement learning algorithms.

Deep Learning

- OpenAI Gym Library
  - Python library with a collection of environments that you can use with your reinforcement learning algorithms

Deep Learning

- OpenAI Gym Service
    - A site and API where you can compare algorithm performance.

Deep Learning

- Let's explore the official documentation before diving into working with OpenAI Gym with Python!
- **gym.openai.com**

OpenAI Gym Working with the Environment

Deep Learning

- OpenAI
  - OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence.

Deep Learning

The environment's `step` function returns exactly what we need. In fact, `step` returns four values. These are:

- `observation` (object): an environment-specific object representing your observation of the environment. For example, pixel data from a camera, joint angles and joint velocities of a robot, or the board state in a board game.
- `reward` (float): amount of reward achieved by the previous action. The scale varies between environments, but the goal is always to increase your total reward.
- `done` (boolean): whether it's time to `reset` the environment again. Most (but not all) tasks are divided up into well-defined episodes, and `done` being `True`indicates the episode has terminated. (For example, perhaps the pole tipped too far, or you lost your last life.)
- `info` (dict): diagnostic information useful for debugging. It can sometimes be useful for learning (for example, it might contain the raw probabilities behind the environment's last state change). However, official evaluations of your agent are not allowed to use this for learning.

OpenAI Gym Set-Up

Deep Learning

- Let's review best practices when using OpenAI with Python.

Deep Learning

- Recommended Environment
  - A text editor (Sublime , Atom)
  - IDE - PyCharm
- Code out .py scripts in editor
- Execute code at command line.
- Many possible plugins for Python!

Deep Learning

- OpenAI does technically work in Jupyter Notebook, but running multiple renderings can sometimes lead to freezing.

Deep Learning

- Let's quickly check that OpenAI Gym is installed and how to run a .py script
- NOTE: Please feel free to use any IDE configuration you prefer!

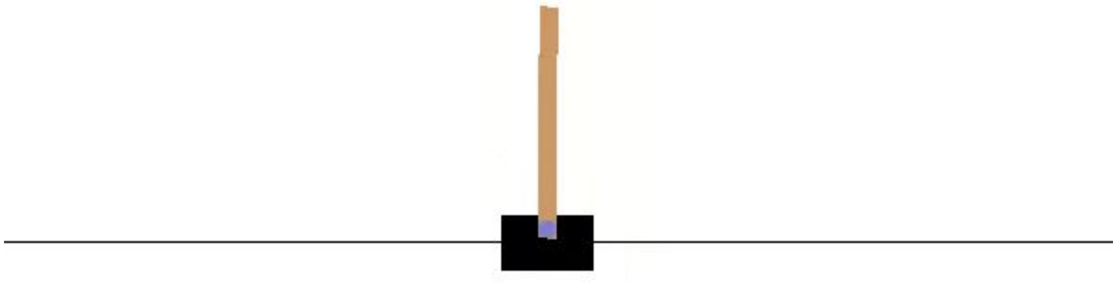
Deep Learning

- We'll use Atom for this series of lectures, remember that you have many options, including just sticking with Jupyter Notebooks!
- Go to: **atom.io**

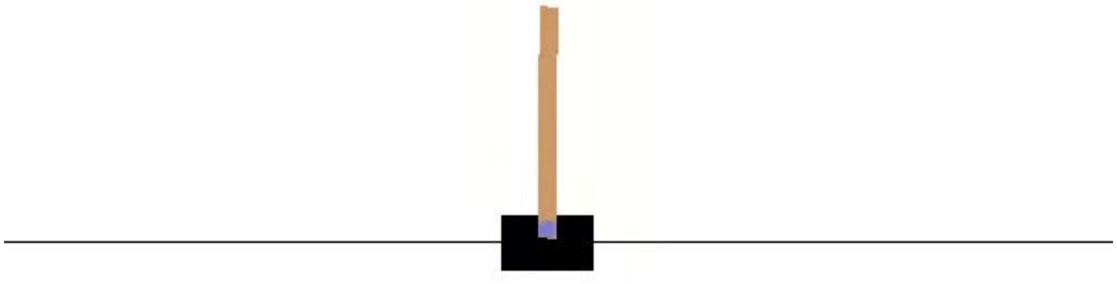
Gym Environments

Deep Learning

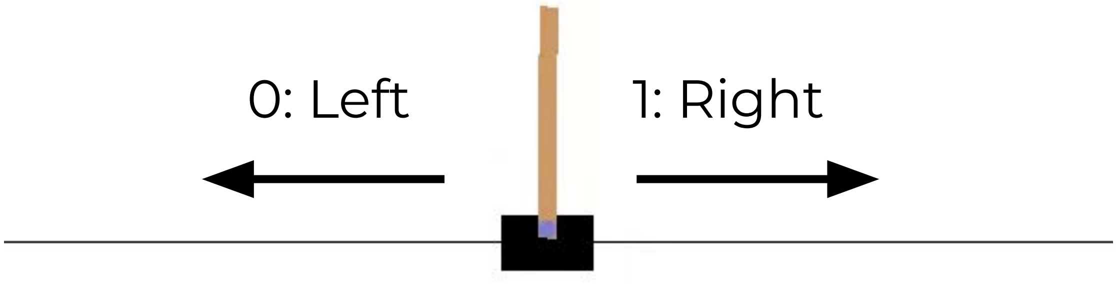
- Let’s explore how to create an example environment with OpenAI Gym!



- Goal of CartPole environment is to balance the pole on the cart.

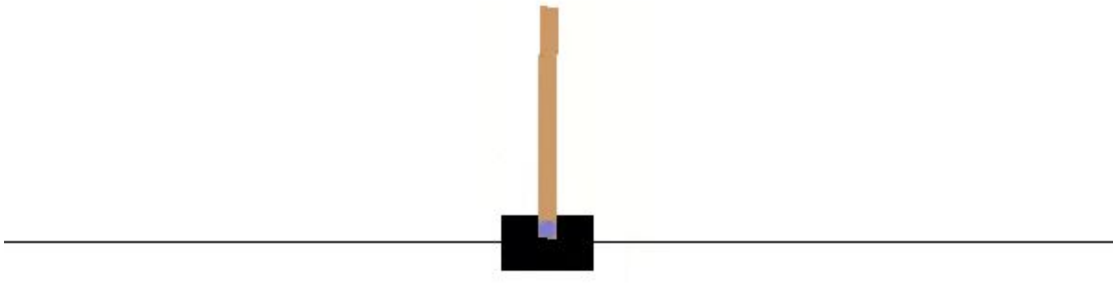


- Actions allow us to move the cart left and right to attempt to balance the pole.



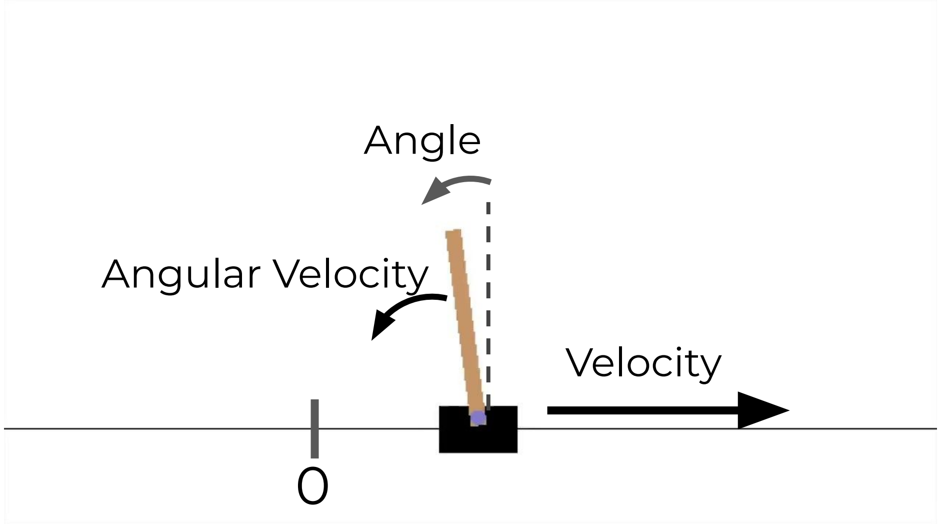
Deep Learning

- Environment is a numpy array with 4 floating point numbers



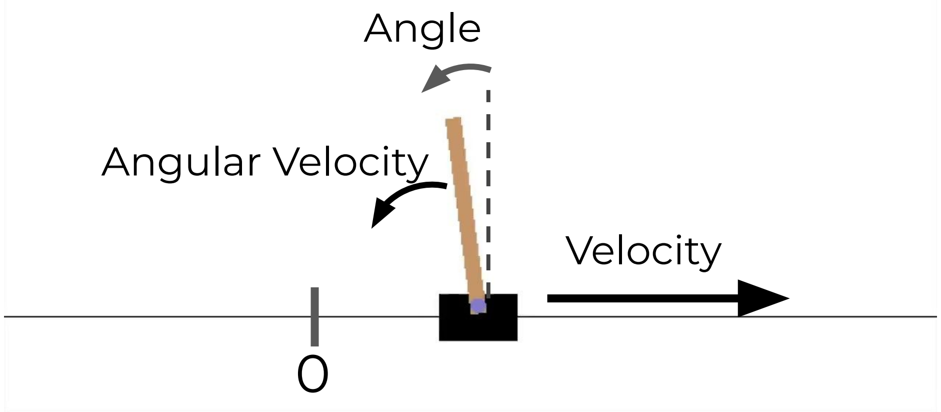
Deep Learning

- [Horizontal Position,Horizontal Velocity, Angle of Pole, Angular Velocity]



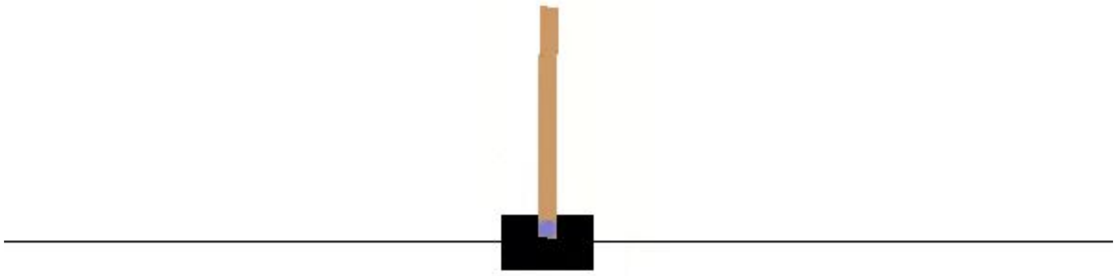
Deep Learning

- You can grab these values from the environment and use them for your agent



Deep Learning

- Let's create the environment!



Gym Observations

Deep Learning

- The environment step() function we saw earlier returns back useful objects for our agent.

Deep Learning

- observation
  - Environment specific information representing environment observations
  - Examples:
    - Angles, Velocities, Game States, etc…

Deep Learning

- reward
  - Amount of reward achieved by previous action
  - Scale varies based off environment, but agent should always want to increase reward level

Deep Learning

- done
  - Boolean indicating whether environment needs to be reset.
  - Example:
    - Game is lost, Pole Tipped Over, etc...

Deep Learning

- info
  - Dictionary object with diagnostic information, usually used for debugging.

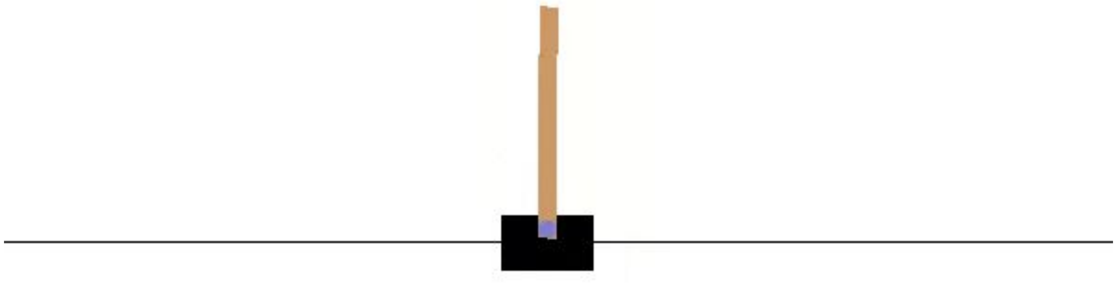
Deep Learning

* Let’s explore what this looks like in practice!

Gym Actions

Deep Learning

- Let's create a very simple policy. Move the cart to the right if the pole falls to the right and vice versa.



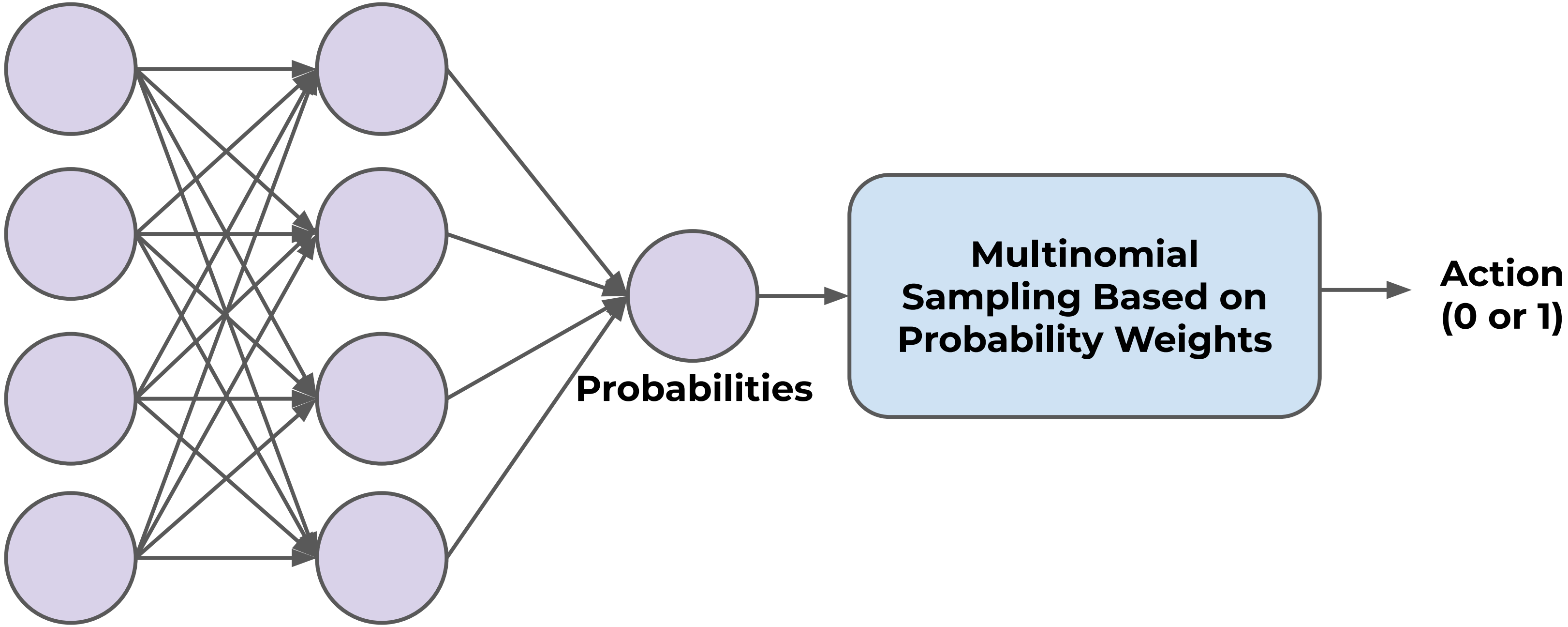
Simple Neural Network

* Let's design a simple Neural Network that takes in the observation array passes it through a hidden layer and outputs 2 probabilities, one for left and another for right.

Deep Learning

- We will then choose a random action, weighted by the probabilities.
- Let's diagram this network

Deep Learning



- Notice how we don’t just automatically choose the highest probability for our decision.
- This is to balance trying out new actions versus constantly choosing well-known actions.

Deep Learning

- Once we understand this network and code it out, we'll explore how to take into account historic actions by learning about Policy Gradients.

Policy Gradients

Deep Learning

- Our previous Network didn't perform very well.
- This may be because we aren't considering the history of our actions, we are only considering a single previous action.

Deep Learning

- This is often called an **assignment of credit** problem.
- Which actions should be credited when the agent gets rewarded at time t, only actions a t-1, or the series of historical actions?

Deep Learning

- We solve this problem by applying a **discount rate.**
- We evaluate an action based off all the rewards that come after the action, not just the first immediate reward.

Deep Learning

- We choose a discount rate, typically around 0.95-0.99.
- Then we use this to apply a score to the action with a formula:
  - R is Reward , D is discount Rate
  - $R_{t=0} + R_{t=1}D + R_{t=2}D^2 + R_{t=3}D^3 + \ldots\ldots + R_{t=n}D^n$

Deep Learning

- Closer D is to 1, the more weight future rewards have. Closer to 0, future rewards don't count as much as immediate rewards
- Score:
  - R is Reward , D is discount Rate
  - $R_{t=0} + R_{t=1}D + R_{t=2}D^2 + R_{t=3}D^3 + ... + R_{t=n}D^n$

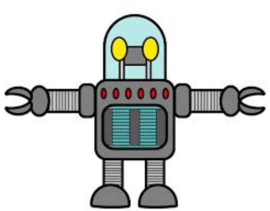
Deep Learning

- Choosing a discount rate often depends on the specific environment and whether actions have short or long term effects.
- Score:
  - R is Reward , D is discount Rate
  - $R_{t=0} + R_{t=1}D + R_{t=2}D^2 + R_{t=3}D^3 + \ldots\ldots + R_{t=n}D^n$

PIERIAN DATA

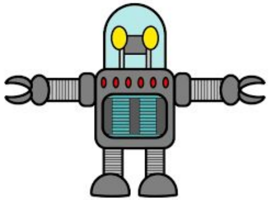
- Let’s quickly diagram this formula!
- Score:
    - R is Reward , D is discount Rate
    - $R_{t=0} + R_{t=1}D + R_{t=2}D^2 + R_{t=3}D^3 + \ldots\ldots + R_{t=n}D^n$

Deep Learning



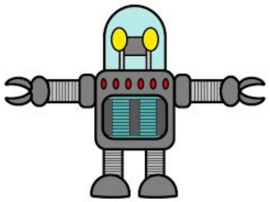
	Pole Goes Up →	Pole Goes Up →	Pole Falls Over →
Actions	Right 1	Left 0	Right 1
Rewards	+10	+10	-100

Deep Learning



	Pole Goes Up	Pole Goes Up	Pole Falls Over
Actions	$\boxed{\text{Right 1}}$	Left 0	Right 1
Rewards	+10	+10	-100

Deep Learning



	Pole Goes Up	Pole Goes Up	Pole Falls Over
Actions	Right 1	Left 0	Right 1
Rewards	+10	+10	-100

Sum of Discount Rewards

$$\boxed{-70.75} = 10 + (0.95)*10 + (0.95)^2(-100)$$

- Because of this delayed effect sometimes good actions may receive bad scores due to bad actions that follow, unrelated to their initial action.
- To counter this, we train over many episodes.

Deep Learning

- We also then must normalize the action scores by subtracting the mean and dividing by the standard deviation.
- These extra steps can significantly increase training time for complex environments.

Deep Learning

- Implementing this Gradient Policy with Python and TensorFlow can be complex, so let's go over the steps that we will perform!

Deep Learning

- Neural Network plays several episodes.
- The optimizer will calculate the gradients (instead of calling minimize)
- Compute each action's discounted and normalized score.

Deep Learning

- Then multiply the gradient vector by the action's score.
- Negative scores will create opposite gradients when multiplied.
- Calculate mean of the resulting gradient vector for Gradient Descent.

Deep Learning

- Because of the complexity of manually implementing Policy Gradient techniques with TensorFlow, we encourage you to check out the extra resources for additional examples and explanations!
- Let’s get started!