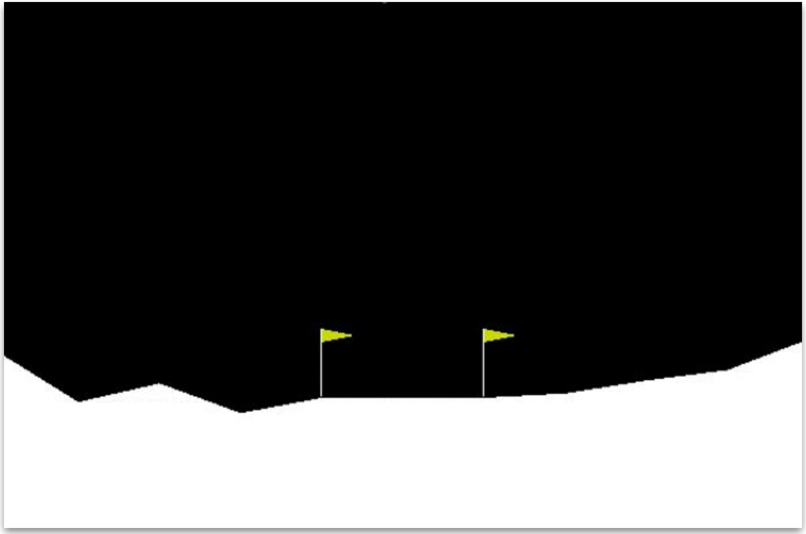
Lunar Lander

Sarthak Khillon & Tim Stoddard

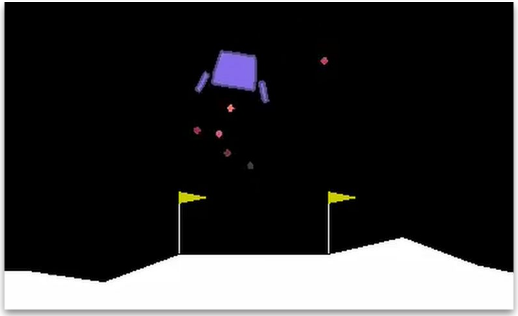
Introduction

- Lunar Lander v2 from the OpenAI Gym
- Land a simulated spaceship 🚀 on the moon!
- The terrain is randomly generated each round, but the designated landing spot is always flat
- Train lander to land itself, while maximizing score
- Many challenges!

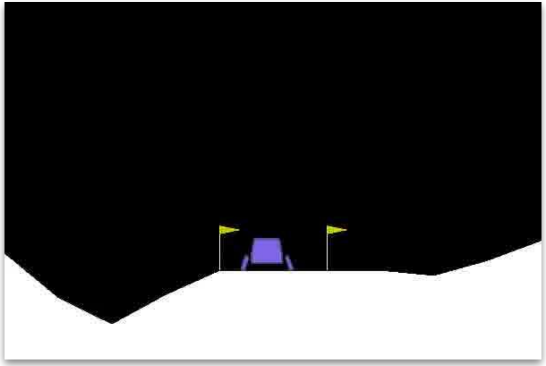


Overview

- Each frame, lander can perform 4 actions
  - Do nothing
  - Fire left engine
  - Fire main (bottom) engine, -0.3 points per frame
  - Fire right engine
- Lander has infinite fuel
- Game is over when lander crashes (-100 points), or lands on the ground (+100 points)
  - Each leg on ground is +10 points
- Need at least 200 points to consider the challenge solved



Firing the main engine



Successful landing!

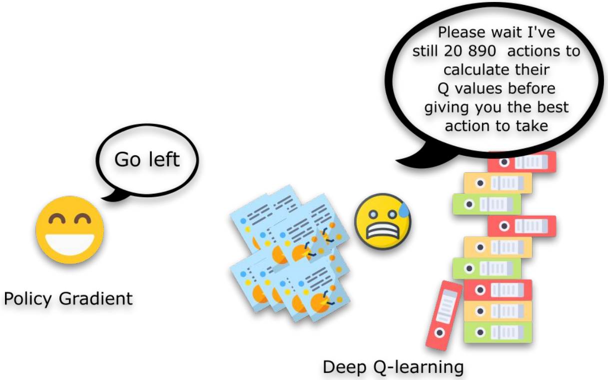
Value-based Learning (Q-learning)

- Learn a value function $V(s)$
  - Best sum of rewards up to that state.
  - “How good is this state?”
- Creates a policy π from V and evaluates best set of coefficients Θ from multiple tests.
- Pros:
  - Independent of environment
- Cons:
  - Must discover policy by trial & error
  - Can have a large oscillation while training

```
valueIteration(S, A)
  V <- best value for each S

  for each state in S
    bestQ <- random value

    for each action in A:
      currQ <- Q(state, action)
      update bestQ to currQ if better

    V[s] <- bestQ

  return V
```

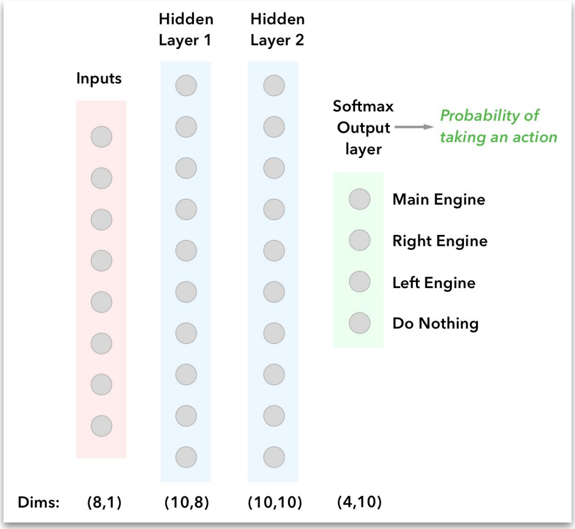
Policy-based Learning ★

- Goal is to find a policy $\pi(S_t \in S, A_t \in A)$ that maximizes reward
  - S is the state space, A is the action space
- Learn directly the policy function that maps a state to an action
  - Select actions without using a value function
- Converges faster than value-based (Q learning), but the policy evaluation step is more expensive
- How it applies to Lunar Lander
  - In each frame, the reward is a combination of how close the lander is to the landing pad and how close it is to zero speed
  - Closer to landing successfully = higher reward



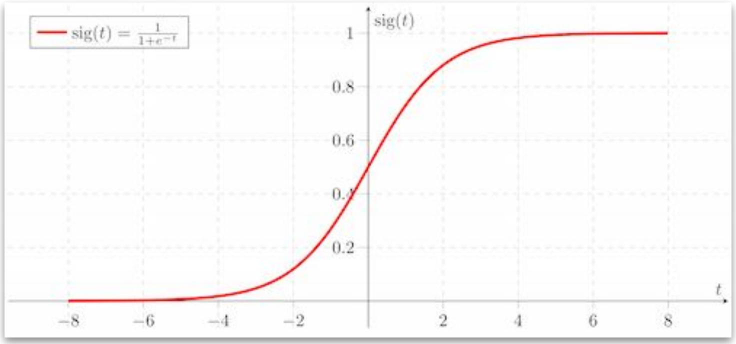
Deep Deterministic Policy Gradient Network

- Approximates argmax(a) on Q
- Advantages over a value-based approach
  - Better at convergence than value-based methods
    - Simply follow the gradient to find the best parameters
  - More effective in high dimensional action spaces, or when using continuous actions
- For loss, regular version uses softmax cross entropy, continuous version uses mean-squared error (MSE)
  - Adam Optimizer provides stochastic gradient descent



Xavier Initialization

- Initializing the network with the right weights is important for deep neural networks
  - If too small, input eventually drops to a really low value and can no longer be useful
  - If too large, it becomes so large that it becomes useless
- Need to make sure that the weights are in a reasonable range before we start training the network
- One good way is to assign the weights from a Gaussian distribution
  - This distribution would have zero mean and some finite variance



Algorithm Pseudocode

```
BEGIN
Continuous loop:
      Reset environment

      While not terminal condition:
            Choose and perform action.
            Environment returns next state + reward
            Update total reward for this episode

      Train: Fit θ values for policy

END
*Terminal state = crash, tilt, or landing.
```

Libraries used

- numpy - fancy math things
- scipy - scientific computing
- tensorflow - dataflow programming
- gym - develop reinforcement learning algorithms

Demo!


MEOW IT'S TIME
FOR A DEMO
memegenerator.net


WAKE UP!
IT'S DEMO TIME
memegenerator.net

Conclusion

- We created a Policy Gradient Network to solve the Lunar Lander v2 from the OpenAI Gym
- Policy gradients converge faster than value-based approaches, but are more computationally expensive
- Xavier initialization assigns network weights from Gaussian distribution
- Improvements - if we had more time
  - Hyperparameter tuning
    - Learning rates
    - Loss functions
    - Optimizer methods
- Where to go (future possibilities)
  - Based on problem, pick more general or more constrained network
  - Explore results from Actor-Critic and Q-learning approaches

References

1. Deep Reinforcement Learning with Policy Gradients
2. Policy Gradients in a Nutshell
3. Tensorflow Deep Learning Projects
4. Deep Reinforcement Learning Demystified (Episode 2)
5. An introduction to Policy Gradients with Cartpole and Doom
6. Deep Q Network vs Policy Gradients
7. Reinforcement Learning Lecture, University of Washington
8. Understanding Xavier Initialization In Deep Neural Networks

Images
- Slide 2
- Slide 5
- Slide 6
- Slide 7
- Slide 9
- Slide 9