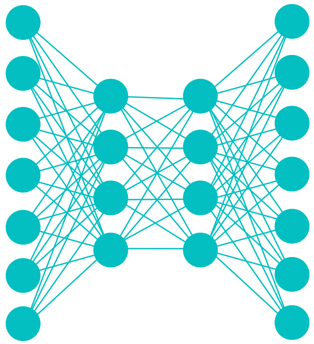
Lecture Notes for

Neural Networks and Machine Learning

Cross Entropy and Value Iteration



Logistics and Agenda

- Logistics
  - Grading Update
  - Class schedule
- Agenda
  - Finish: Cross Entropy Method
  - Value Iteration (and demo)
  - Tabular Q-Learning
  - Deep Q-Learning (next time?)

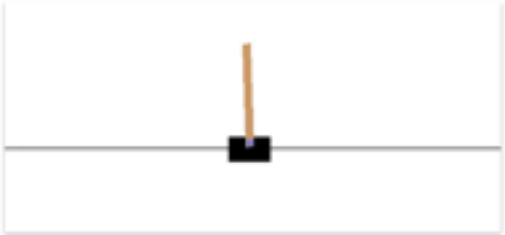
Last Time

```python
import gym


if __name__ == "__main__":
    env = gym.make("CartPole-v0")

    total_reward = 0.0
    total_steps = 0
    obs = env.reset()

    while True:
        action = env.action_space.sample()
        obs, reward, done, _ = env.step(action)
        total_reward += reward
        total_steps += 1
        if done:
            break
```

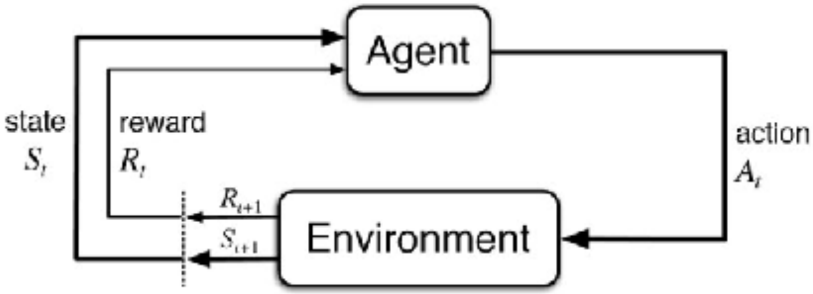


Action Space: One input, [0, 1] pull left or pull right

Obs Space: Dynamic state variables (continuous and four dimensional)

End: When more than 15 degrees off or too far from center

Reward: +1 for each time step




Edward Thorndike


B.F. Skinner


Ivan Pavlov


Bernard Widrow


Marvin Minsky


Ted Hoff


Claude Shannon

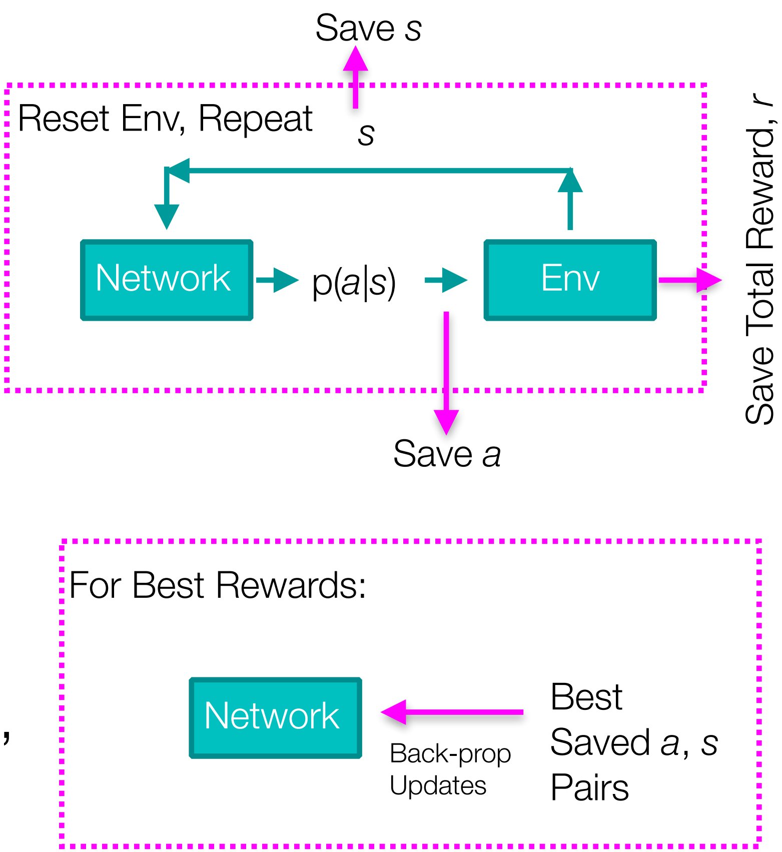
Cross Entropy Method

Direct Policy Exploration and Optimization

- Instead of defining what is optimal, just setup a comparison of different actions we might take (**policy**)
- A **policy** is defined as $\pi(a, s) = P(a_t = a \mid s_t = s)$
  - Given the current state, we have a certain probability of selecting each action
  - Action selection is **probabilistic**, but easy to discover **deterministic** actions (*set one action to 1.0, all others to 0.0*)
- Try different policies, select one with best average reward
- First try: Cross Entropy Method

Cross Entropy Method

- Create a random neural network, with output $p(a|s)$
- Let it interact with the environment (randomly)
  - For some set of episodes (*e.g.*, 20)
    - Use network output to sample from possible actions
    - Run episode to completion
    - Repeat
- Calculate reward for each episode
- Keep best episodes (some percentile, e.g., best five)
- For the given best episodes, develop loss function incentivizing the actions taken based upon the input observations



Repeat until desired performance!

Cross Entropy Method

- Model based or Model Free?
  - Model Free (no assumptions of problem)
- Value or Policy Based?
  - Policy Based (randomly sample actions based on policy)
- On-policy or Off-Policy?
  - On-Policy (need to interact with environment to get better)

Mathematical Motivation

- If we have the optimal policy p(x) and a reward function H(x), then maximize

$$\mathbf{E}_{x \leftarrow p(x)}[H(x)] = \mathbf{E}_{x \leftarrow q(x)}[\frac{p(x)}{q(x)} H(x)]$$

- We can approximate the distribution by: $\frac{1}{N} \sum_i \frac{p(x_i)}{q(x_i)} H(x_i)$
- Proven that this is optimized when $\mathbf{KL}(\ q(x)\ ||\ p(x)H(x)\ )$ is minimized. But its intractable, so we can only optimize upper bound … minimizing (neg) cross entropy of samples

Performance Measure

$$\pi_{k+1}(a \mid s) = \arg\max_{\pi_k} \mathbf{E}_{z \leftarrow \pi_k}[\mathbf{1}_{R(z) > \psi} \log \pi_k(a \mid s)]$$

min CrossEntropy(*neural_net_actions*, *best_actions*)

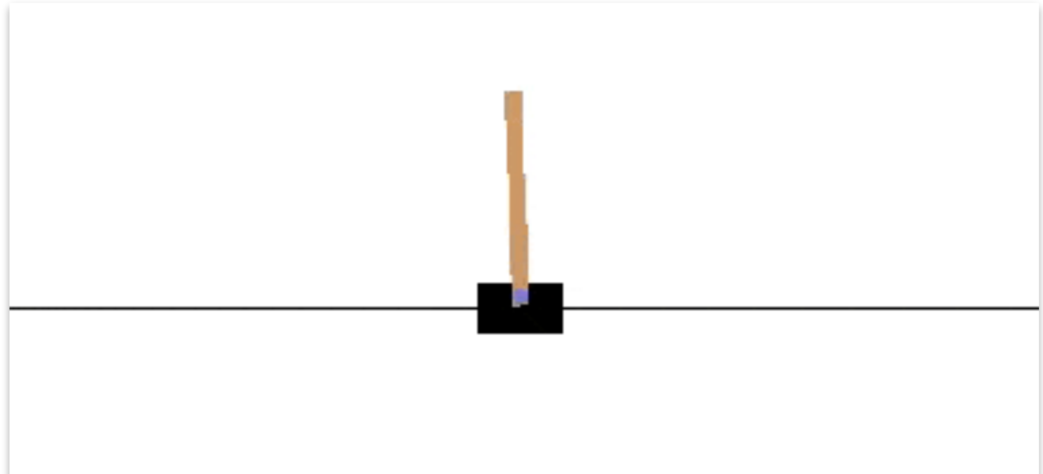
Review: Basics of Cartpole

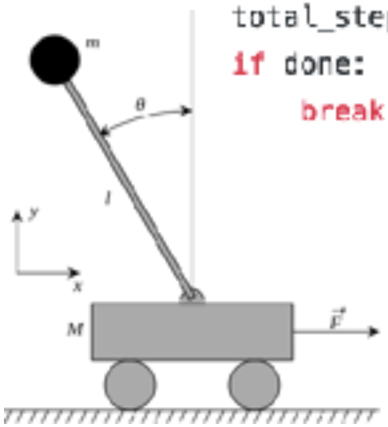
```python
import gym


if __name__ == "__main__":
    env = gym.make("CartPole-v0")

    total_reward = 0.0
    total_steps = 0
    obs = env.reset()

    while True:
        action = env.action_space.sample()
        obs, reward, done, _ = env.step(action)
        total_reward += reward
        total_steps += 1
        if done:
            break
```





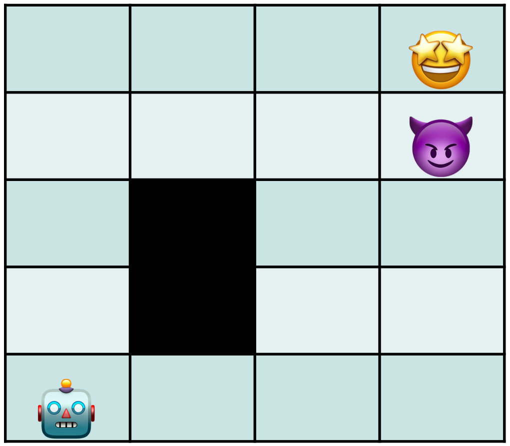
Action Space: One input, [0, 1] pull left or pull right

Obs Space: Dynamic state variables (continuous and four dimensional)

End: When more than 15 degrees off or too far from center

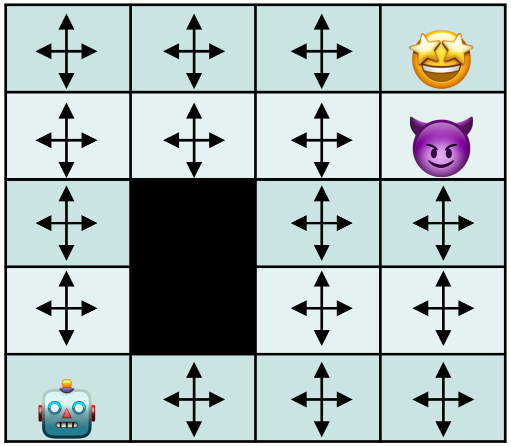
Reward: +1 for each time step

Another Example: Frozen Lake

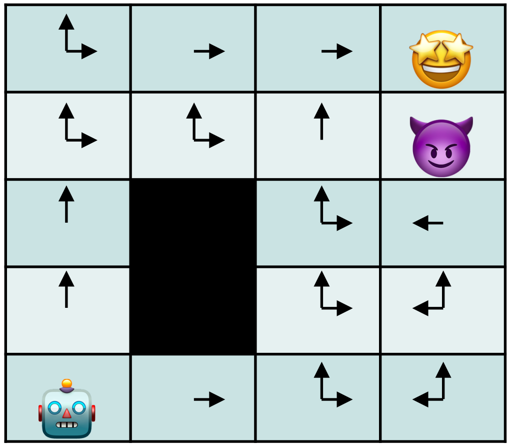


- **State**: Every square in grid
- **Action**: Move to make (l,r,u,d), *with probability*
- **Reward**: Goal, Death
- **Policy**: Given state, where should we move?
- **Optimal Policy**:

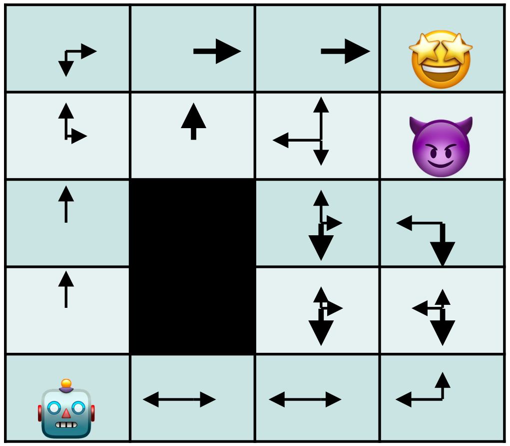
$$\pi^* = \arg\max_{\pi} \mathbf{E}\left[\sum_k \gamma^k R_{t+k+1} \mid \pi\right]$$



Random Policy



Another Policy



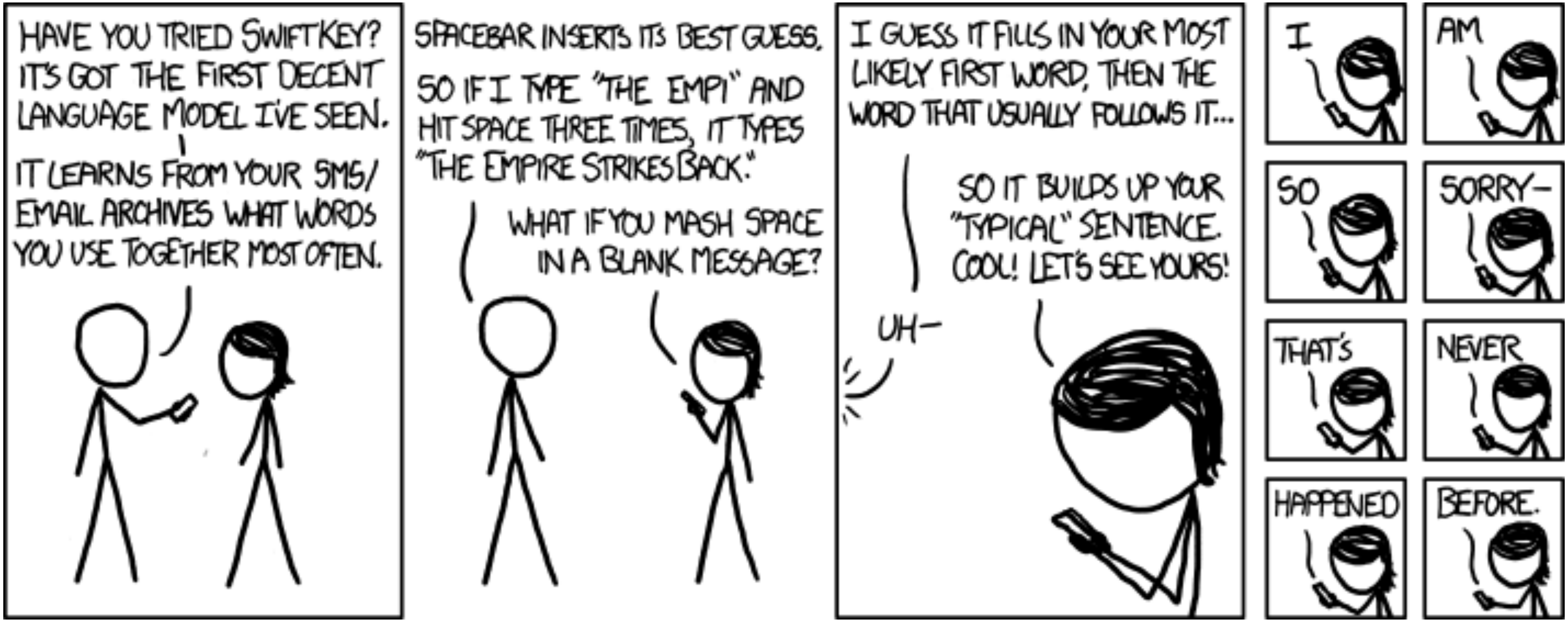
Another Policy

Cross Entropy Reinforcement Learning

M. Lapan Implementation for CartPole and Frozen Lake

```
Follow Along:
08a_Basics_Of_Reinforcement_Learning.ipynb
```

Markov Building Blocks



Markov Processes (MP)

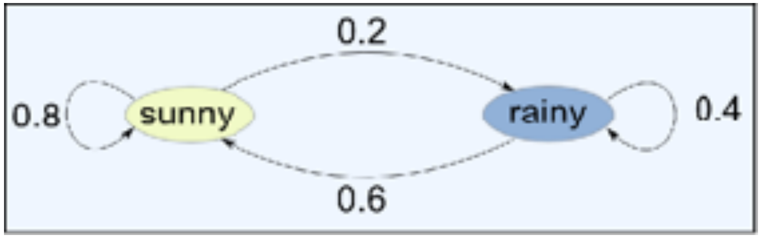
- **Definition**: Any process that can be explained (or simplified) through a sequential set of states that depend only on the previous state
- **Practical Meaning**: For N states, there will be the probability of transition to any other state, encoded through an NxN transition matrix of discrete probabilities
- State sequences are not deterministic, they are sampled from these distributions
- Despite **simplicity**, MP can model a number of real processes with **good enough** precision

Next State, s_{t+1}

Current State, s_t					
	0.1	0.2	0.1	0.6	0.0
	0.9	0.0	0.1	0.0	0.0
	0.0	0.4	0.0	0.4	0.2
	0.0	0.4	0.2	0.0	0.4
	0.0	0.0	0.6	0.0	0.4

MP Example from Maxim Lapan

	Sunny'	Rainy'
Sunny	0.8	0.2
Rainy	0.6	0.4



				...	
Sun+Summer					
Rainy+Summer					
Sun+Fall					
Rainy+Fall					
Sun+Else					
Rainy+Else					

Adding One Variable Can Have Drastic Effect on State Space Size