Reinforcement Learning

Machine Learning Decal

Hosted by Machine Learning at Berkeley

Agenda

What is Reinforcement Learning?
Useful Concepts in Reinforcement Learning
Techniques for Reinforcement Learning
- Value Iteration
- Policy Iteration
- Approximate Methods
- Q Learning

Deep Reinforcement Learning
Demo
Questions

Game

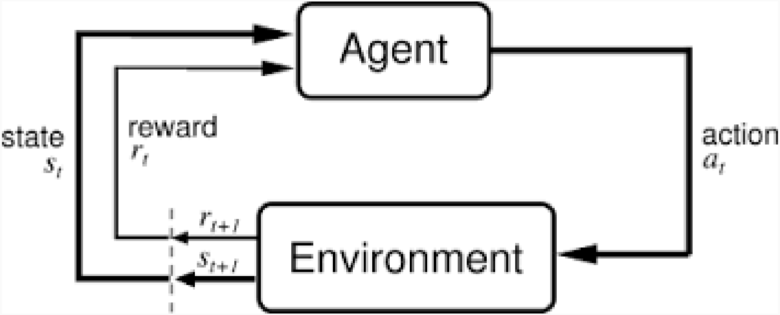
But first a game ....

Did you figure it out?

- It was a game called Frozen Lake.
- S represented start.
- F represented frozen so you could walk on it. (reward 1)
- H represented hole so you would fall in. (reward -10)
- G represented the goal. (reward $+10$)

What is Reinforcement Learning?

Reinforcement Learning Problems



Some Examples: Toy Environments

Acrobot-v1
Swing up a two-link robot.

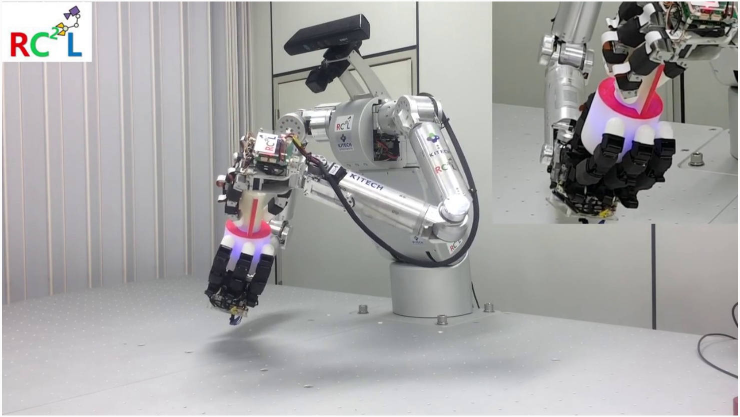
CartPole-v1
Balance a pole on a cart.

MountainCar-v0
Drive up a big hill.

MountainCarContinuous-v0
Drive up a big hill with continuous control.

Pendulum-v0
Swing up a pendulum.

Some Examples: Robotics

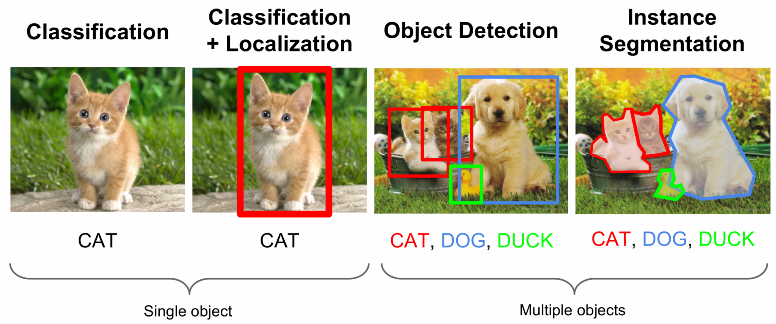


https://www.youtube.com/watch?v=ZtP-I_Bpibs

Some Examples: Video Games

https://www.youtube.com/watch?v=OWEIM2wXyuE

Some Examples: Supervised Learning



Formalization: Markov Decision Process (MDP)

- Reinforcement Learning problems can be modelled as a MDP
- MDP can be thought of as the collection of the following 5 things
- S a set of states
- A a set of actions
- $p(s_{t+1}|s_t, a_t)$, how we move between states
- $p(r_{t+1}|s_t, a_t)$, how we model reward
- γ, how we model the importance of reward in the future.

Solution to an MDP: A Policy

- The "Policy" is the way an agent chooses its actions
- Can be thought of as a function that maps states to actions

$$\pi(s) : S \to A$$

.

Useful Concepts in Reinforcement Learning

Idea of Value of a “Policy”

- We get a reward at each timestep
- But we need to encode future reward as well
- I.e. taking the action that will give us the most immediate reward is not always the best in the long run (HINT: this can be true in life as well)

Discounted Expected Future Reward

- Define value of a policy as the expectation over the total reward we will receive if we use that policy
- We need an expectation because our environment might not be deterministic.
- In math terms we can define value as

$$V^{\pi}(s) = \mathbb{E}[\sum_{i=1}^{T} \gamma^{i-1} r_i]$$

.

Q Function

- Q Function is very similar to expected future reward.
- Just also considers the reward at your current time step.
- In math,
$$Q(s, a) = R(s, a) + \mathbb{E}_{s'}[V(s')]$$
.

Idea of Optimal

- There is an "optimal" value function.
- The optimal value function that has the highest value for all states.
- In math $V^*(s) = \max_\pi V^\pi(s)$.
- The optimal policy is then the policy that corresponds to this optimal value function.
- In math $\pi^*(s) = \arg\max_\pi V^\pi(s)$.

Optimal Value vs optimal Q

- Natural correspondence between optimal value function and optimal Q function
- In math, $V^*(s) = \max_a Q^*(s, a)$.
- In the same way as with value the optimal policy can be related to the optimal Q function.

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

Techniques for Reinforcement Learning

Do we know the MDP?

- The first techniques we will look at assumes we know how are actions affect the state.
- In other words, these techniques assume we know exactly how the environment works.
- This is not often the case.
- For example, how would you go about explicitly stating the transistion function of Doom?

Value Iteration

- Basic Idea: compute the optimal value and Q functions.
- If we have the optimal Q function we can design a policy vary easily.
- Our policy is simply to take the action at a given state that leads to the highest Q function value.
- Recall this equation:

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

How do we compute the optimal values

- Store current estimates of values and iteratively update them.

```
Initialize V(s) to arbitrary values
Repeat
    For all s ∈ S
        For all a ∈ A
            Q(s,a) ← E[r|s,a] + γ Σ_{s'∈S} P(s'|s,a)V(s')
        V(s) ← max_a Q(s,a)
Until V(s) converge
```

Policy Iteration

- Basic Idea: Instead of computing optimal value, compute optimal policy directly.
- We iteratively update our policy.
- Each time we recompute our value function for our current policy, or $V^{\pi}(s)$
- Then improve our policy based on this value function.

Algorithm

Initialize a policy π' arbitrarily

Repeat

$\pi \leftarrow \pi'$

Compute the values using π by

solving the linear equations

$$V^\pi(s) = E[r|s,\pi(s)] + \gamma \sum_{s' \in S} P(s'|s,\pi(s))V^\pi(s')$$

Improve the policy at each state

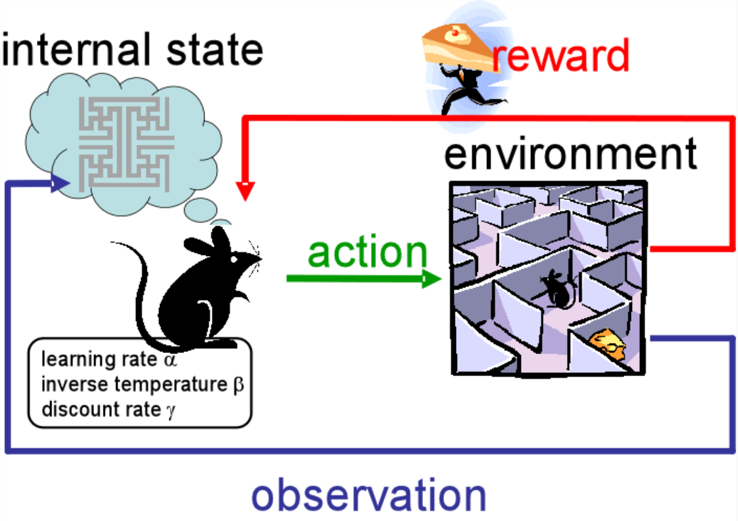
$$\pi'(s) \leftarrow \arg\max_a (E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V^\pi(s'))$$

Until $\pi = \pi'$

What is wrong with these methods?

- These methods rely on being able to see the results of each action in a given state
- Usually we only get to see the result of the action we take.

Model Based vs Model Free Methods



Model Based vs Model Free Methods

Model Based RL

- Learns a model of the environment
- In technical language: models the transistion function of the MDP.
- Using the model of the environment compute optimal policy under that model
- Could use policy iteration or other methods.

Model Free RL

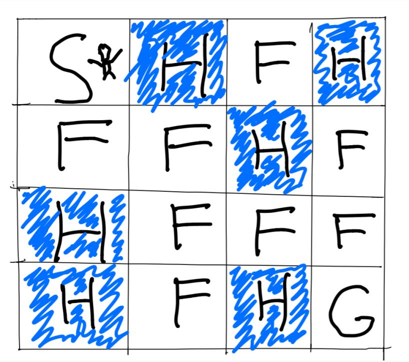
- No model of the system
- Directly learns a policy based off the reward signal.
- We are going to talk about model-free methods.

Q Learning: A model-free method

- Very similar to Value Iteration
- Iteratively update $Q(s, a)$, then use $Q(s, a)$ to create our policy.
- We can store the values of Q in a table, and then iteratively update them.

Back to the game of Frozen Lake we played before.

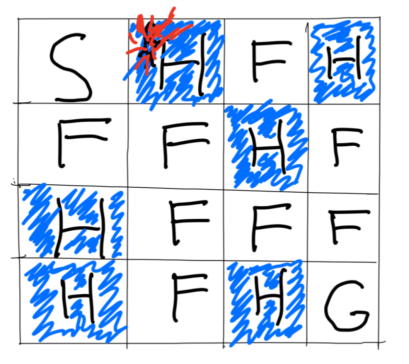
Frozen Lake: Tabular Q Learning



0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Think of each square as storing q value for each possible action in that square.

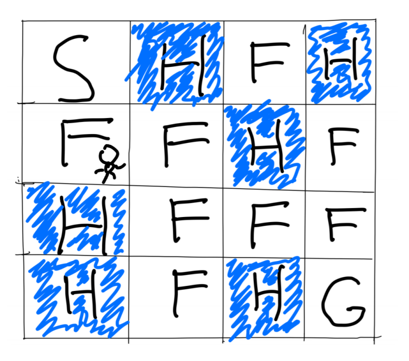
Frozen Lake



$[0, -10]$	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

We get -10 reward so we update the Q-value for the square we were in. We use the equation $Q(s,a) = R(s,a) + max_a' Q(s', a')$

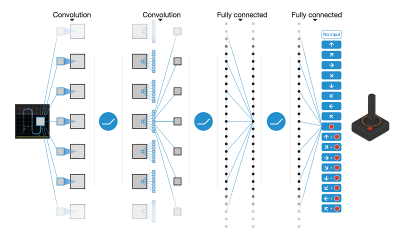
Frozen Lake



[1, -10]	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Deep Reinforcement Learning

Deep Q Learning (DQN)

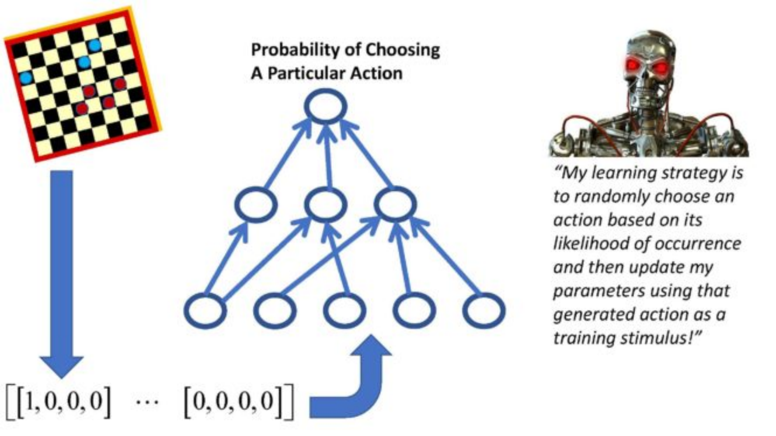


- Approximate the Q-function with a Neural Network
- Each output neuron corresponds to one possible action
- Policy is to take the action corresponding to the neuron with the highest value.
- Network can often be convolutional.

DQN playing Atari Games

https://www.youtube.com/watch?v=LJ4oCb6u7kk

Policy Gradient Methods



Policy Gradient Methods

- Let a neural network directly represent our policy.
- Number of output neurons is the number of possible actions.
- The output neurons represent probability of taking a certain action.
- Update our network using gradient ascent over the value of our policy.

List of Policy Gradient Methods

- REINFORCE

List of Policy Gradient Methods

- REINFORCE
- Deep Deterministic Policy Gradients

List of Policy Gradient Methods

- REINFORCE
- Deep Deterministic Policy Gradients
- Trust Region Policy Optimization

List of Policy Gradient Methods

- REINFORCE
- Deep Deterministic Policy Gradients
- Trust Region Policy Optimization
- Natural Policy Gradients

List of Policy Gradient Methods

- REINFORCE
- Deep Deterministic Policy Gradients
- Trust Region Policy Optimization
- Natural Policy Gradients
- Proximal Policy Gradients

List of Policy Gradient Methods

- REINFORCE
- Deep Deterministic Policy Gradients
- Trust Region Policy Optimization
- Natural Policy Gradients
- Proximal Policy Gradients
- And many more...

Demo

Questions

questions?