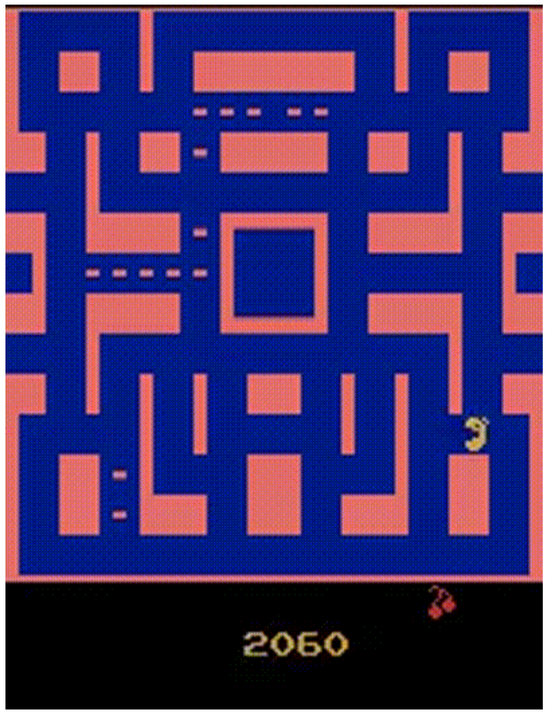
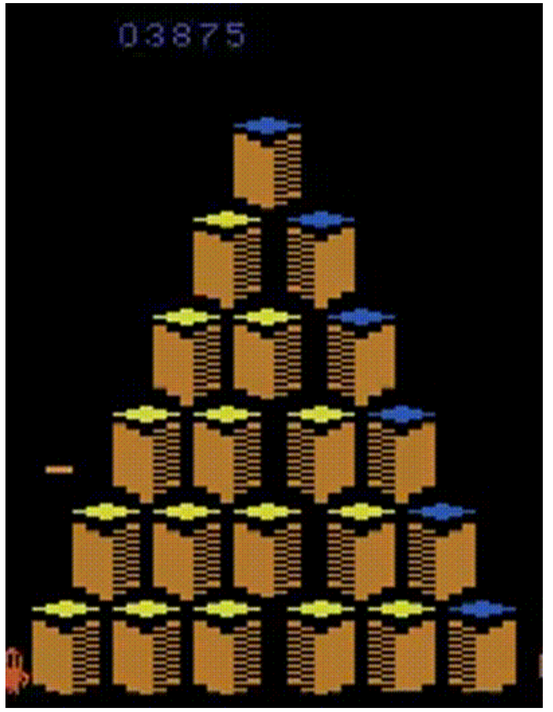
CS 224R Tutorial

Review of Q-Learning





Maximilian Du

Outline of Tutorial

- Review of Markov Decision Processes (MDP)
- Exact MDPs (tabular; simple)
  - Fitted Q Iteration
- Parametric Q-Learning (learned models)
- Practical Details
  - Replay Buffer, Overestimation, TD Gradients
- Algorithm Walkthrough

Many thanks to Anikait Singh, Pieter Abbeel, David Silver, Aviral Kumar, and Justin Fu!

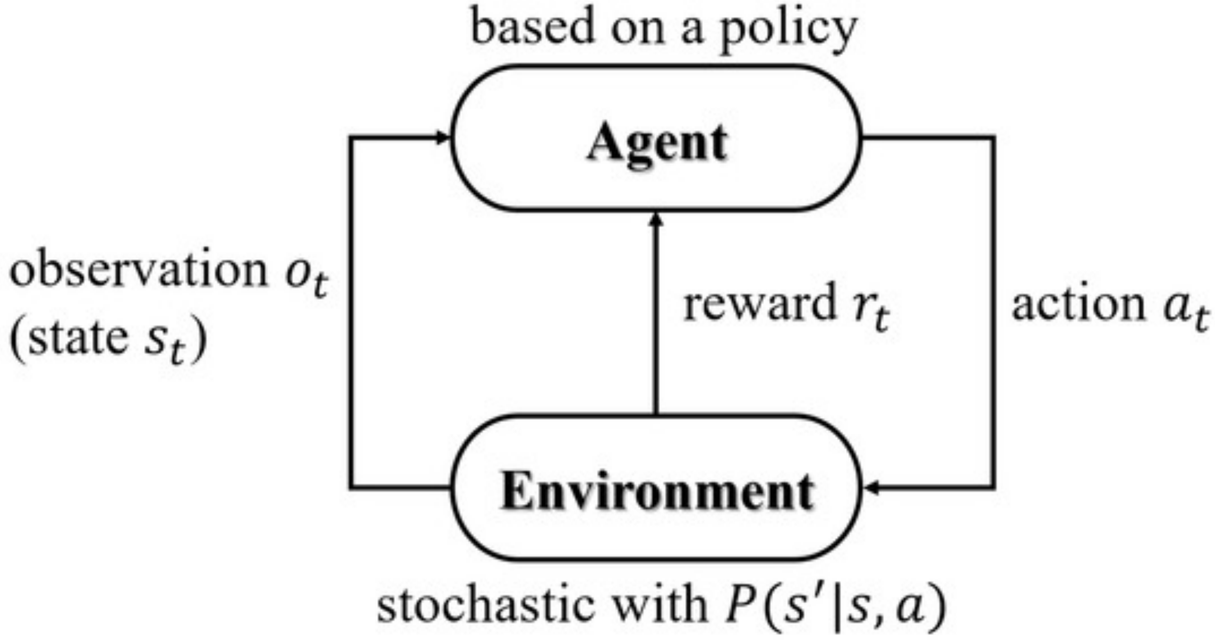
Outline of Tutorial

- **Review of Markov Decision Processes (MDP)**
- Exact MDPs (tabular; simple)
  - Fitted Q Iteration
- Parametric Q-Learning (learned models)
- Practical Details
  - Replay Buffer, Overestimation, TD Gradients
- Algorithm Walkthrough

MDP Formulation

An MDP is defined by:

- Set of states s_t
- Set of actions a_t
- Transition function $p(s_{t+1} \mid s_t, a_t)$
- Reward Function $r(s_t, a_t)$
- Start state s_1
- Discount Factor γ
- Horizon T



What matters: average reward over all trajectories

$$E_\tau \left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \right]$$

Here: $\tau = \{s_1, a_1, s_2, a_2, \ldots, s_{T-1}, a_{T-1}, s_T\}$

Outline of Tutorial

- Review of Markov Decision Processes (MDP)
- **Exact MDPs (tabular; simple)**
  - **Fitted Q Iteration**
- Parametric Q-Learning (learned models)
- Practical Details
  - Replay Buffer, Overestimation, TD Gradients
- Algorithm Walkthrough

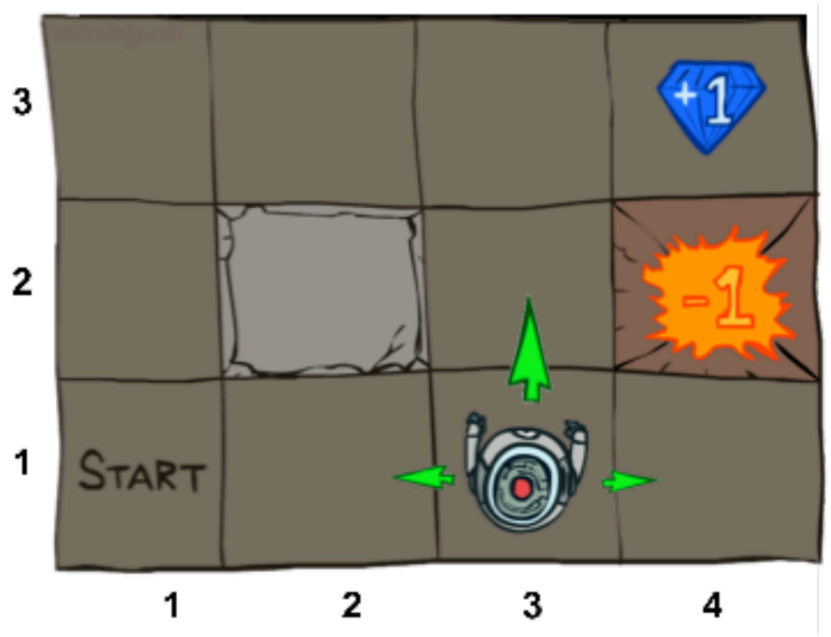
This part is the most important…it also has the most math

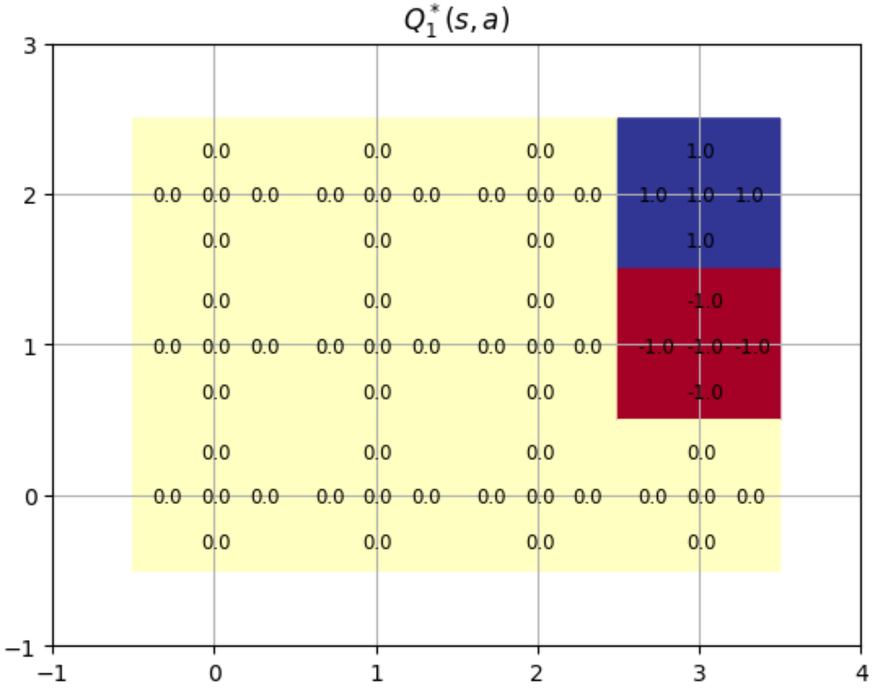
The Tabular Setup

"Tabular" setup means that all states and actions are fully discrete and tractable.

Therefore, all rewards, states, Q functions, value functions can all be represented as a table.

We do this to make algorithms simple at first!





The "Q" Function

$$E_\tau \left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \right]$$

Define function $Q(s_t, a_t) := E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right]$

Average reward will you get from the future *given* your current action

Can we peel out a single time step?

$$E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right] = E_\tau \left[r(s_t, a_t) + \gamma \sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right]$$

$$= E_{s_{t+1}, a_{t+1}} \left[E_{\tau'} \left[r(s_t, a_t) + \gamma \sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right] \right]$$

Here, $\tau = \{s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2} \ldots a_{T-1}, s_T\}$ and the shortened $\tau' = \{s_{t+2}, a_{t+2}, \ldots a_{T-1}, s_T\}$

The "Q" Function

$$E_\tau \left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \right]$$

Define function $Q(s_t, a_t) := E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right]$

Average reward will you get from the future *given* your current action

$$E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right] = E_{s_{t+1}, a_{t+1}} \left[E_{\tau'} \left[r(s_t, a_t) + \gamma \sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right] \right]$$

$$E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right] = r(s_t, a_t) + \gamma E_{s_{t+1}, a_{t+1}} \left[E_{\tau'} \left[\sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right] \right]$$

Current reward

Average *next step* outcome

Look familiar?

The "Q" Function

$$E_\tau \left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \right]$$

Define function $Q(s_t, a_t) := E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right]$

Average reward will you get from the future *given* your current action

$$E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right] = E_{s_{t+1}, a_{t+1}} \left[E_{\tau'} \left[r(s_t, a_t) + \gamma \sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right] \right]$$

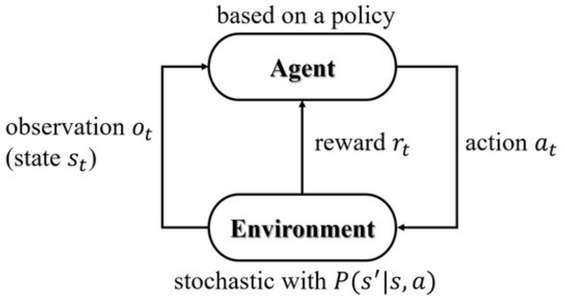
$$E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right] = r(s_t, a_t) + \gamma E_{s_{t+1}, a_{t+1}} \left[E_{\tau'} \left[\sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right] \right]$$

Current reward

Average *next step* outcome

$Q(s_{t+1}, a_{t+1})$

The “Q” Function



Define function $Q(s_t, a_t) := E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right]$

The recursive relationship

$$E_\tau \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{\hat{t}}) \right] = r(s_t, a_t) + \gamma E_{s_{t+1}, a_{t+1}} \left[E_{\tau'} \left[\sum_{t'=t+1}^{T} \gamma^{(t'-(t+1))} r(s_{t'}, a_{t'}) \right] \right]$$

Therefore:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1}, a_{t+1}} \left[Q(s_{t+1}, a_{t+1}) \right]$$

This is very important!! Make sure you understand this :)

Understanding the Bellman equation

The Bellman equation:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1}, a_{t+1}} \left[Q(s_{t+1}, a_{t+1}) \right]$$

Expanding the expectation

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_{t+1} \sim \pi(a_{t+1}|s_{t+1})} \left[Q(s_{t+1}, a_{t+1}) \right] \right]$$

How the environment responds

How we act

Talk to a partner: what future returns (Q) are we estimating? With respect to π? With respect to an optimal π^*? Something else? (30 seconds)

Answer: the Q function is estimating future returns if we follow π because it's in the expectation

Bellman Optimality Equation

Always use a superscript on your Q!!

Expanded Bellman Equation

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_t \sim \pi(a_t|s_t)} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right] \right]$$

What if we are estimating an optimal policy π^*

$$Q^{*}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[\max_a Q^{*}(s_{t+1}, a) \right]$$

Talk with a partner: Why must $\pi^*(a \mid s) = \arg\max_a Q^*(s, a)$?

Contradiction: if $\pi^* \neq \arg\max_a Q^*(s, a)$, then the return of this action isn't the highest we can get -> not an optimal policy

Bellman Optimality Equation

Always use a superscript on your Q!!

Expanded Bellman Equation

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_t \sim \pi(a_t|s_t)} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right] \right]$$

What if we are estimating an optimal policy π^*

$$Q^{*}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[\max_a Q^{*}(s_{t+1}, a) \right]$$

Talk with a partner: Are these equalities always true? For any old function Q? Or does Q have to be a special function?

Answer: This is true if the Q function is already fully accurate. Remember we are not yet describing a way of learning Q! We are just saying truths about this function as it is defined.

Define function $Q(s_t, a_t) := E_{\tau} \left[\sum_{\hat{t}=t}^{T} \gamma^{\hat{t}-t} r(s_{\hat{t}}, a_{\hat{t}}) \right]$

Fitted Q Iteration: Fake it till you make it

This is a truism about a *correct* Q function

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_t \sim \pi(a_t|s_t)} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right] \right]$$

What if we don't have a Q function to begin with?

Recursive leap of faith: what if we assume that $Q^{\pi}(s_{t+1}, a_{t+1})$ is correct?

$$Q^{\pi}(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_t \sim \pi(a_t|s_t)} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right] \right]$$

Exactly the same thing, except you use the right hand side to compute the left hand side; it is no longer an equality!

Q-learning: Variants

With policy-specific Q:

$$Q^{\pi}(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_t \sim \pi(a_t|s_t)} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right] \right]$$

With Q*

$$Q^{*}(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[\max_{a} Q^{*}(s_{t+1}, a) \right]$$

Talk to your partner: after you compute Q*, how do you get the optimal policy?

Answer: Let $\pi(a \mid s) = \max_{a} Q^{*}(s, a)$

Aside: does this actually work?

$$Q^{\pi}(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[E_{a_t \sim \pi(a_t|s_t)} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right] \right]$$

Intuitively:

The right hand side contains more "true" information because of the reward function.

We gradually add this true information into the Q function.

In tabular setting, this reliably works. No guarantees in the parametric setting

Proof (**don't worry about this**):

We can start by taking the sum of the results in (d):

$$||V - V^{\pi}|| + ||V - V^{\star}|| \leq \frac{||V - B^{\pi}V||}{1-\gamma} + \frac{||V - BV||}{1-\gamma}$$

However, we established in part (b) that for the greedy policy, $B = B^{\pi}$. Therefore, we have

$$||V - V^{\pi}|| + ||V - V^{\star}|| \leq \frac{2||V - BV||}{1-\gamma}$$

Now, from the triangle inequality, we know that

$$||V - V^{\pi} - V + V^{\star}|| = ||V^{\pi} - V^{\star}|| \leq ||V - V^{\pi}|| + ||V - V^{\star}||$$

which means that

$$||V^{\pi} - V^{\star}|| \leq \frac{2||V - BV||}{1-\gamma} = \frac{2\varepsilon}{1-\gamma}$$

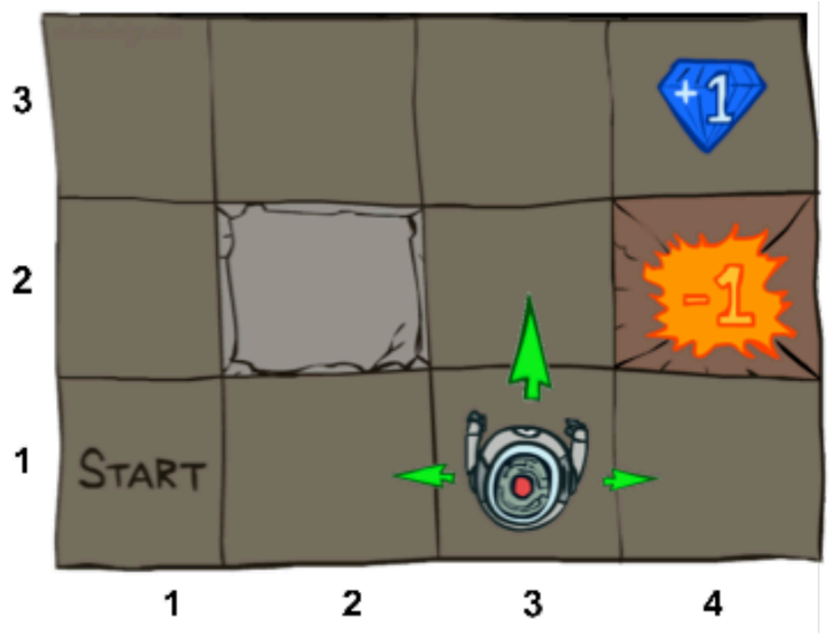
Now, by the property of infinity norms, we know that for all s, we have

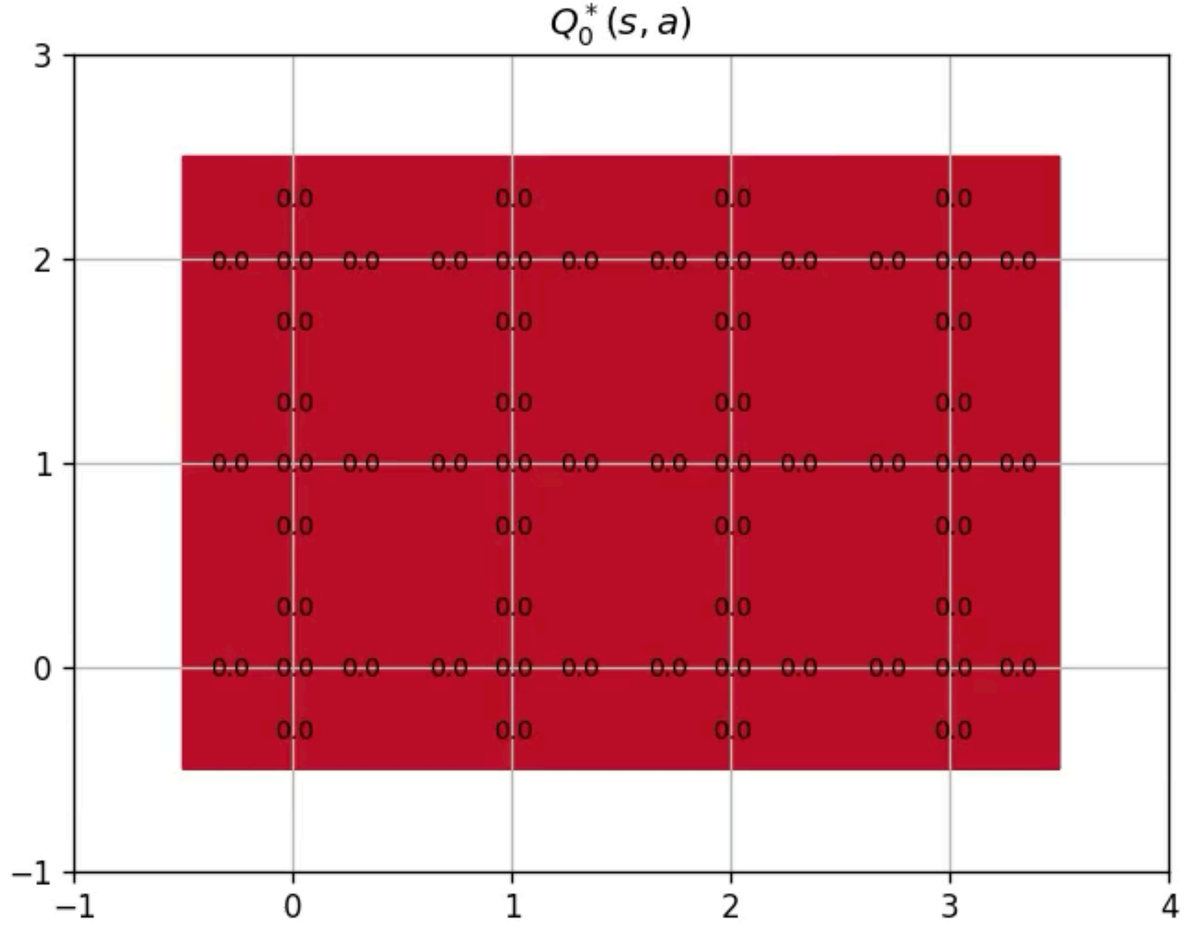
$$|V^{\pi}(s) - V^{\star}(s)| \leq \frac{2\varepsilon}{1-\gamma}$$

Proof result: each operation gets you exponentially closer to the true solution. Like this? Take CS234!

What does this look like in a tabular environment?

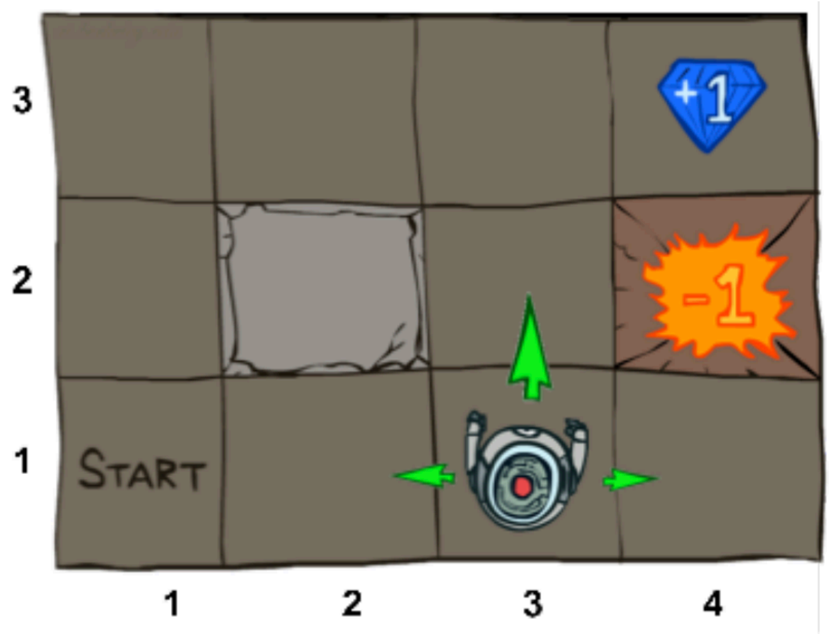
Notice how the rewards propagate backwards from the sources!

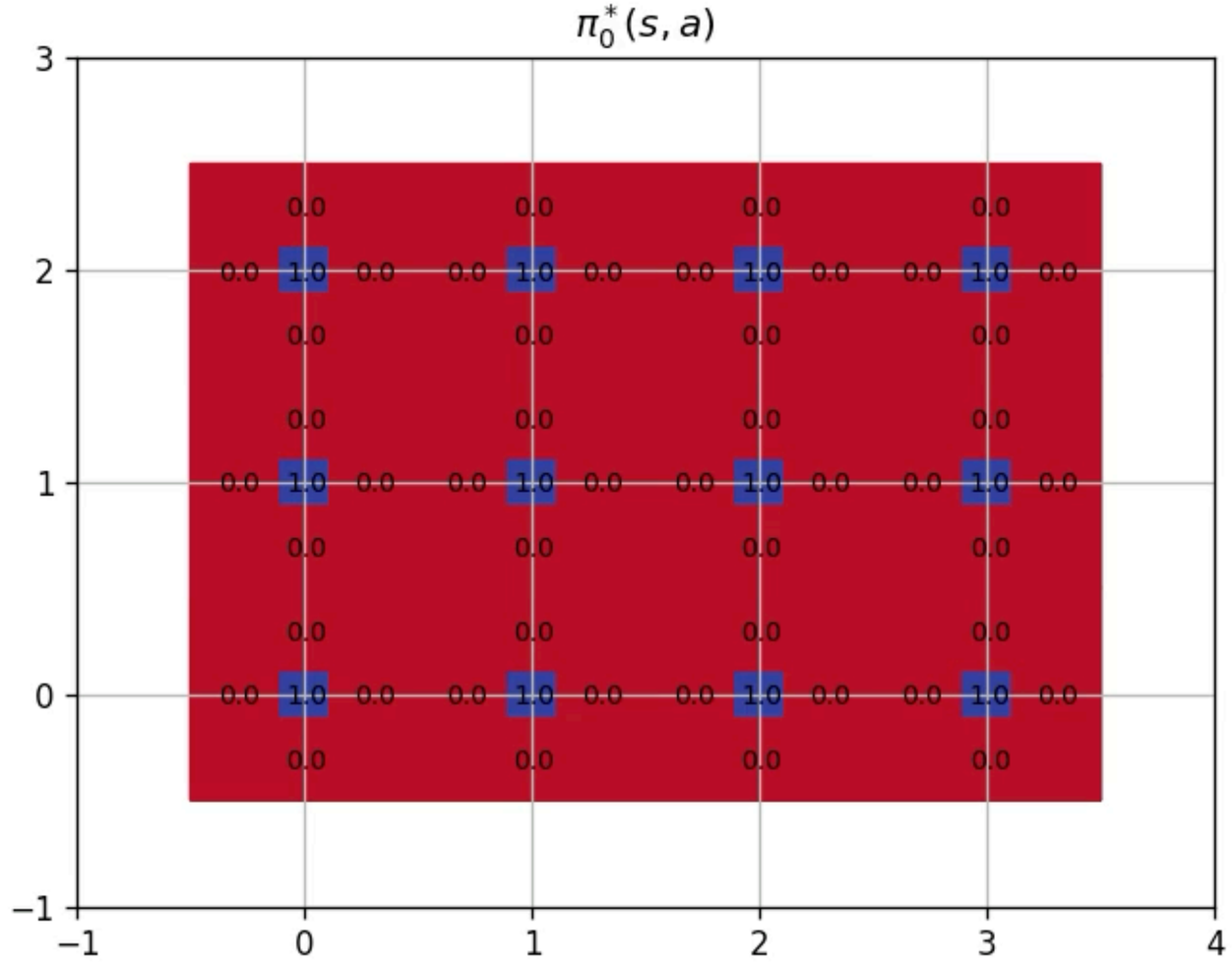




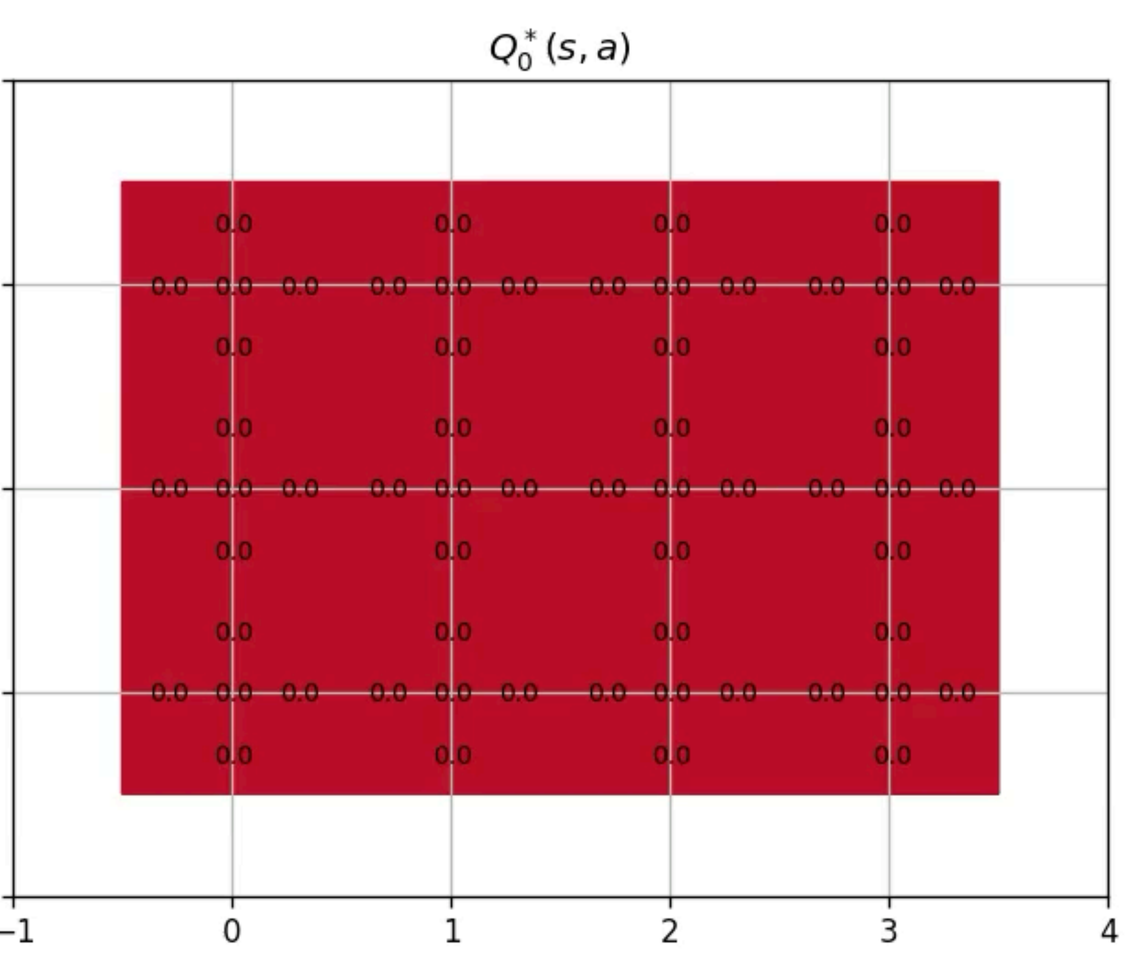
What does this look like in a tabular environment?

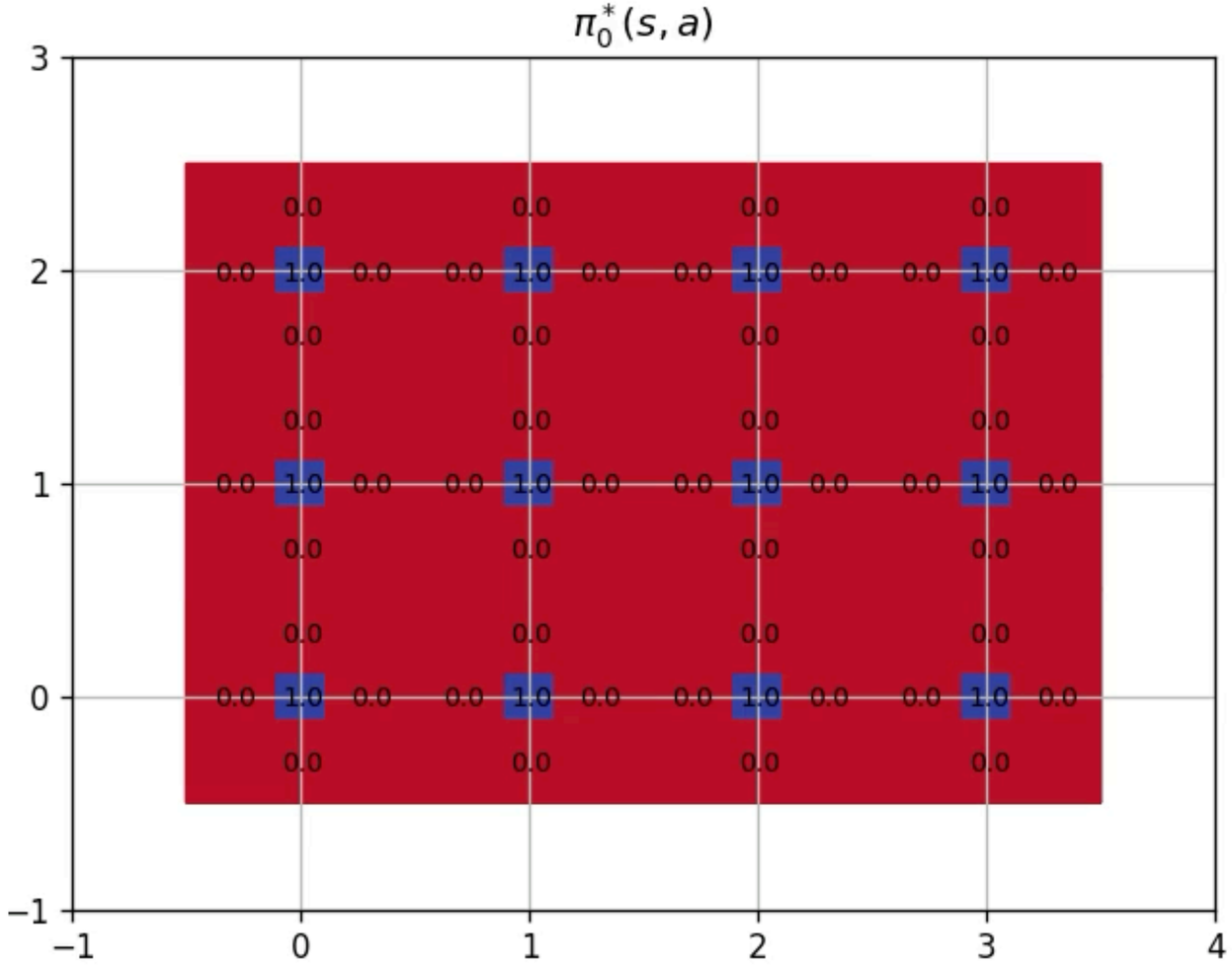
Notice how the actions point to high reward!





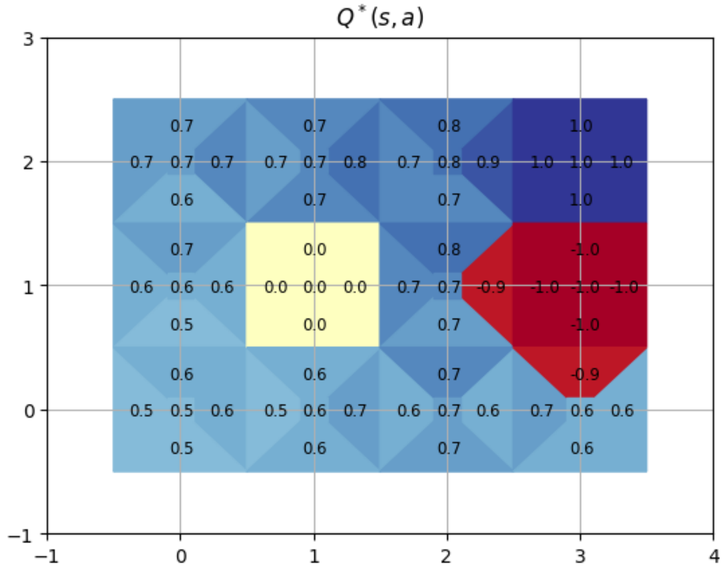
What does this look like in a tabular environment?

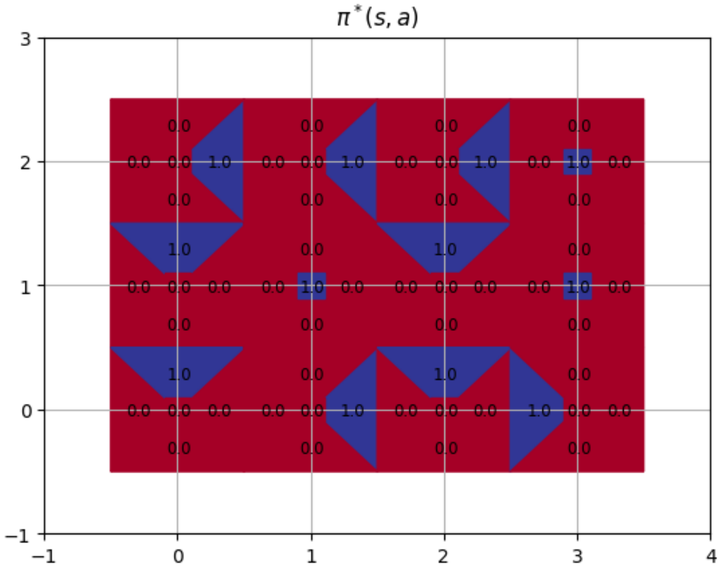




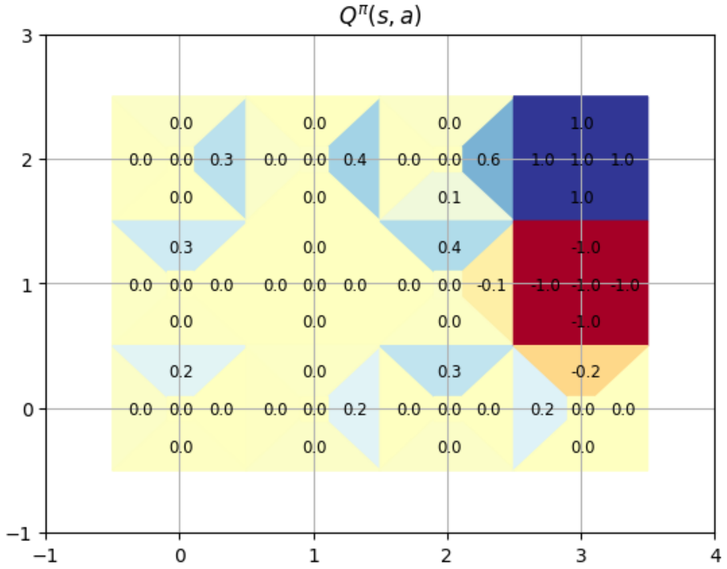
Q^* vs Q^π

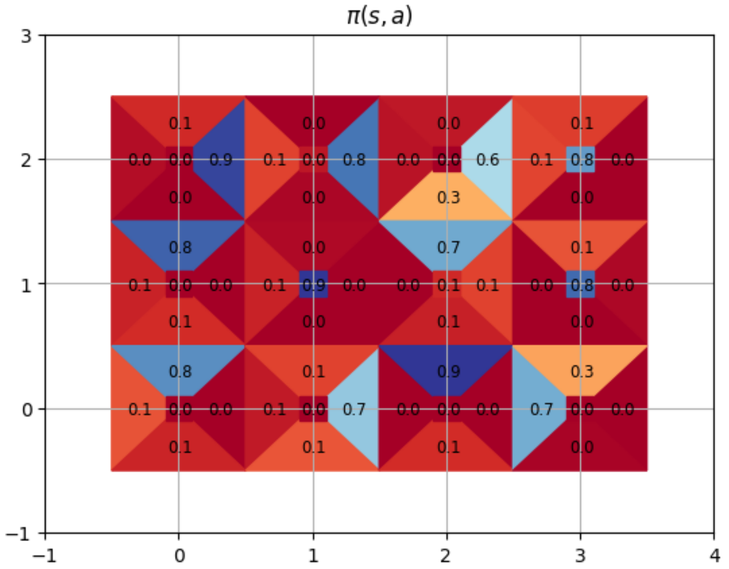
Optimal Policy
(Deterministic for MDP)





Suboptimal Policy

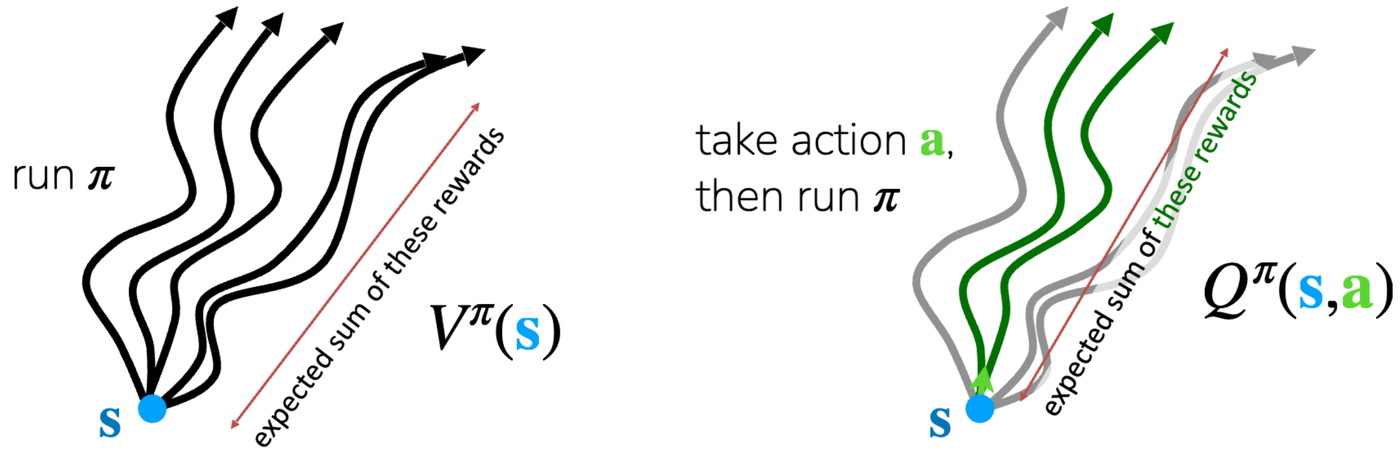




Value functions and Advantages

value function $V^{\pi}(\mathbf{s})$ - future expected rewards starting at $\mathbf{s}$ and following π

Q-function $Q^{\pi}(\mathbf{s}, \mathbf{a})$ - future expected rewards starting at $\mathbf{s}$, taking $\mathbf{a}$, then following π

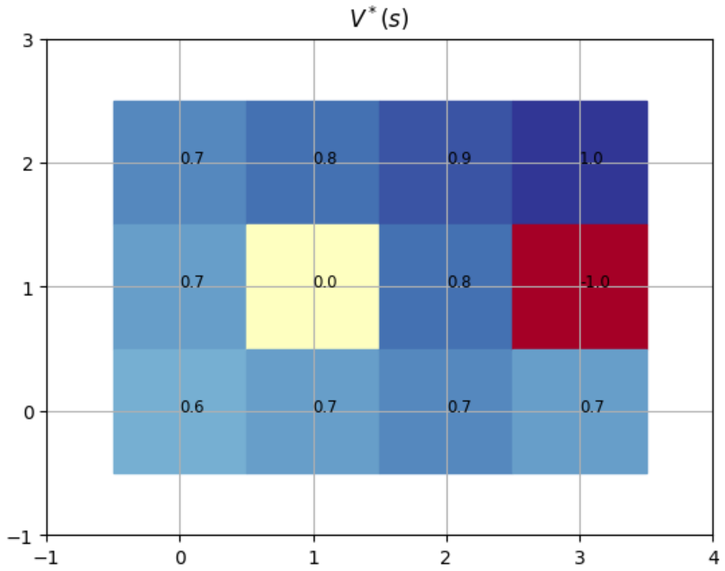


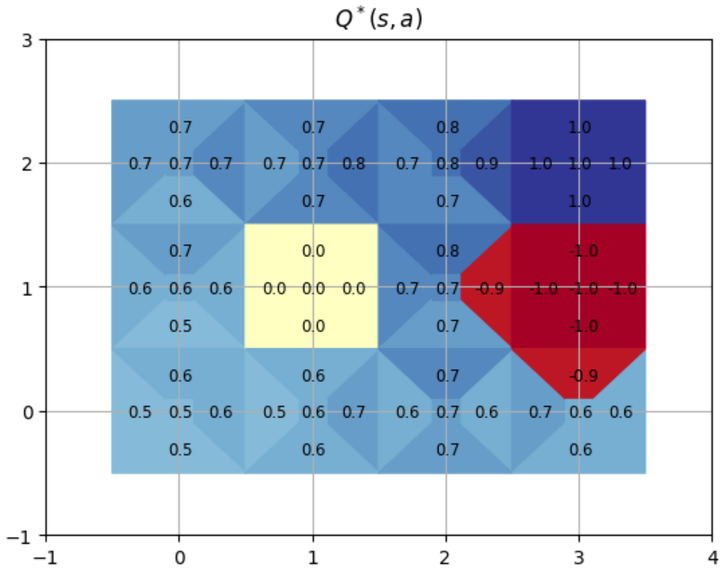
Useful relation: $V^{\pi}(\mathbf{s}) = \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[Q^{\pi}(\mathbf{s}, \mathbf{a})\right]$

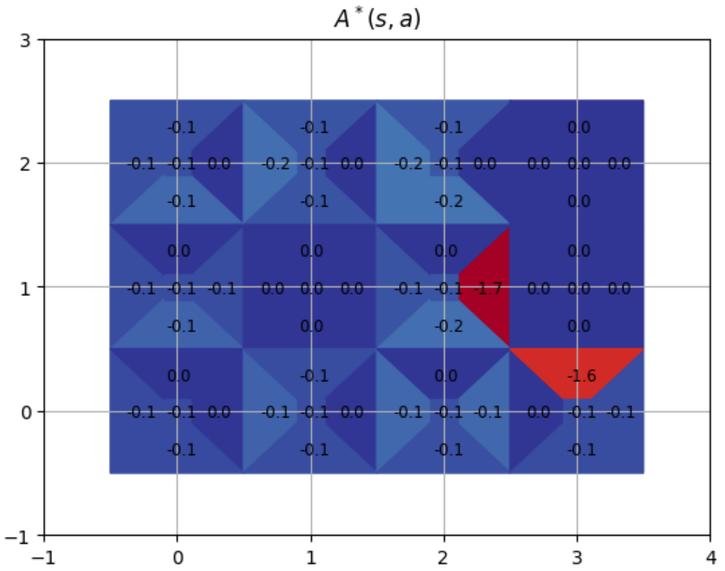
advantage $A^{\pi}(\mathbf{s}, \mathbf{a})$ - how much better it is to take $\mathbf{a}$ than to follow policy π at state $\mathbf{s}$

$$A^{\pi}(\mathbf{s}, \mathbf{a}) = Q^{\pi}(\mathbf{s}, \mathbf{a}) - V^{\pi}(\mathbf{s})$$

Advantage (Optimal Policy)







$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t)\text{: how much better } \mathbf{a}_t \text{ is}$$

Questions?

Another idea: What if we just computed the Q function directly? (Monte Carlo)

Value function $V(s_t) := E_\tau \left[\sum_{\hat{t}=t}^{T} \gamma^{\hat{t}-t} r(s_{\hat{t}}, a_{\hat{t}}) \right]$

We use value function for a cleaner formulation

Bellman update $V(s_t) \leftarrow E_{a_t} \left[r(s_t, a_t) + \gamma E_{s_{t+1}} \left[V(s_{t+1}) \right] \right]$

Monte Carlo Approximation $V(s_t) \leftarrow \sum_{\hat{t}=t}^{T} \gamma^{\hat{t}-t} r(s_{\hat{t}}, a_{\hat{t}})$

Monte Carlo: approximate expectation with a single rollout

Talk to your partner: what are some pros and cons of Monte Carlo Approximations?

Bias Variance Tradeoff

Bellman update $V(s_t) \leftarrow E_{a_t}\left[r(s_t, a_t) + \gamma E_{s_{t+1}}\left[V(s_{t+1})\right]\right]$

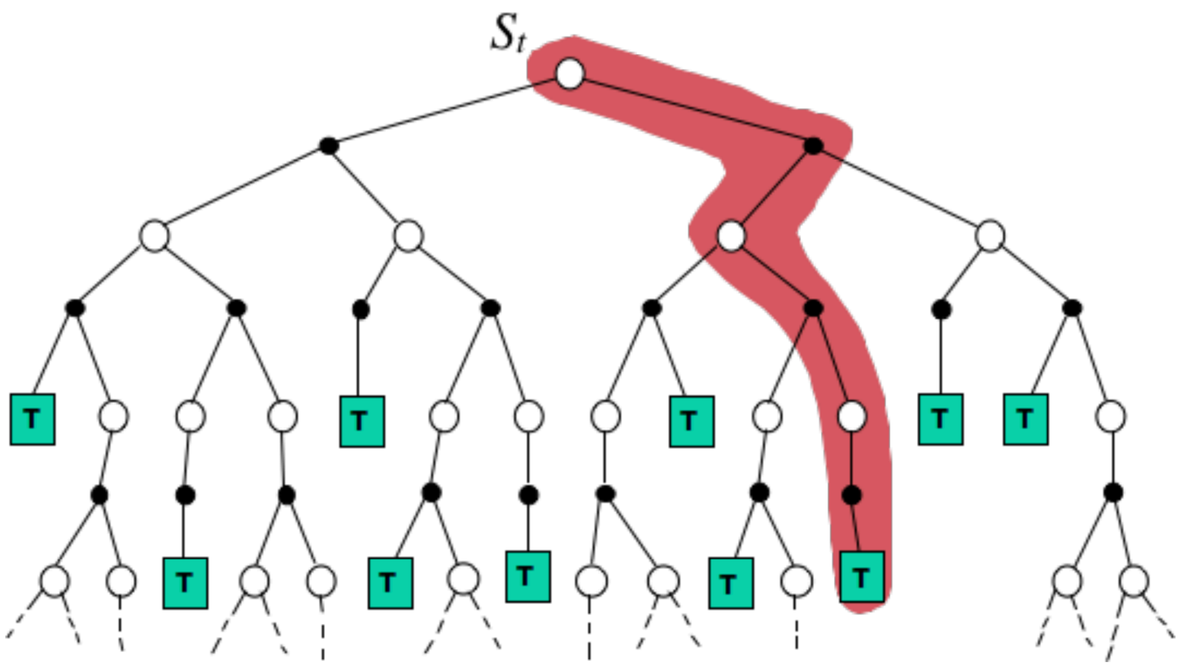
The updated value may not be correct: BIAS

Monte Carlo Approximation $V(s_t) \leftarrow \sum_{\hat{t}=t}^{T} \gamma^{\hat{t}-t} r(s_{\hat{t}}, a_{\hat{t}})$

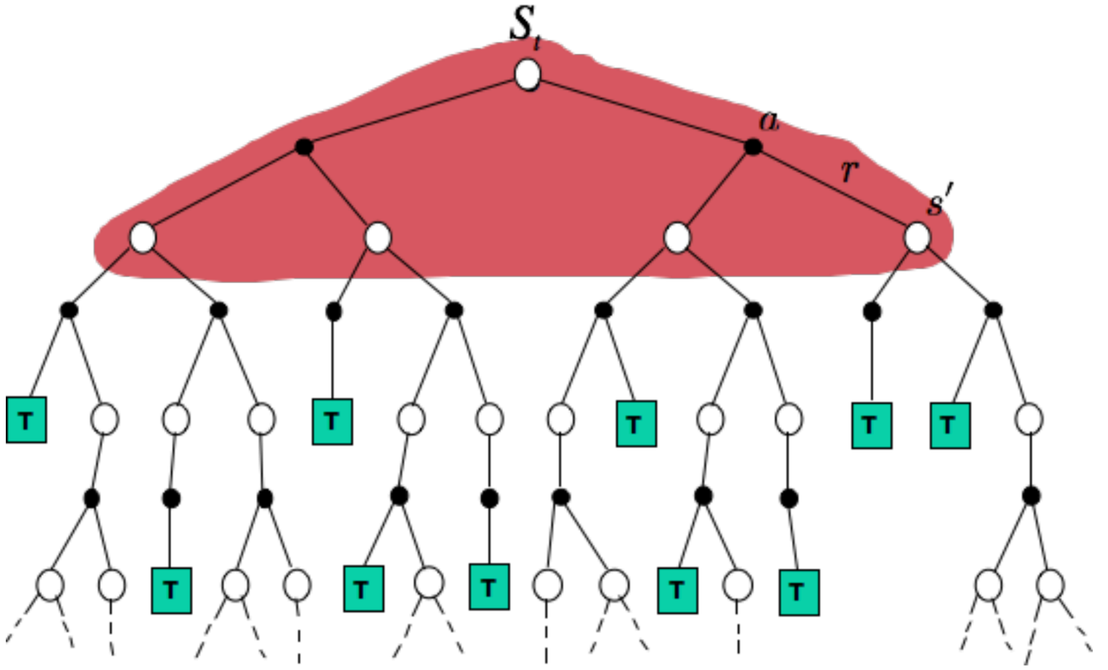
The updated value can change wildly based on current trajectory: VARIANCE

Bias-variance tradeoff: either you get a low-noise but potentially inaccurate estimate, or you get a noisy but accurate (on expectation) estimate.

Monte Carlo vs. Bellman Backup Visual



Monte-Carlo Rollouts
(Regression)



TD / Bellman Backup
(Dynamic Programming)

N-step Returns: A Compromise

Bellman update $V(s_t) \leftarrow E_{a_t}\left[r(s_t, a_t) + \gamma E_{s_{t+1}}\left[V(s_{t+1})\right]\right]$

The updated value may not be correct: BIAS

Monte Carlo Approximation $V(s_t) \leftarrow \sum_{\hat{t}=t}^{T} \gamma^{\hat{t}-t} r(s_{\hat{t}}, a_{\hat{t}})$

The updated value can change wildly based on current trajectory: VARIANCE

N-step Returns $V(s_t) \leftarrow \left(\sum_{\hat{t}=t}^{k} \gamma^{\hat{t}-t} r(s_{\hat{t}}, a_{\hat{t}})\right) + \gamma^k V(s_{t+k})$

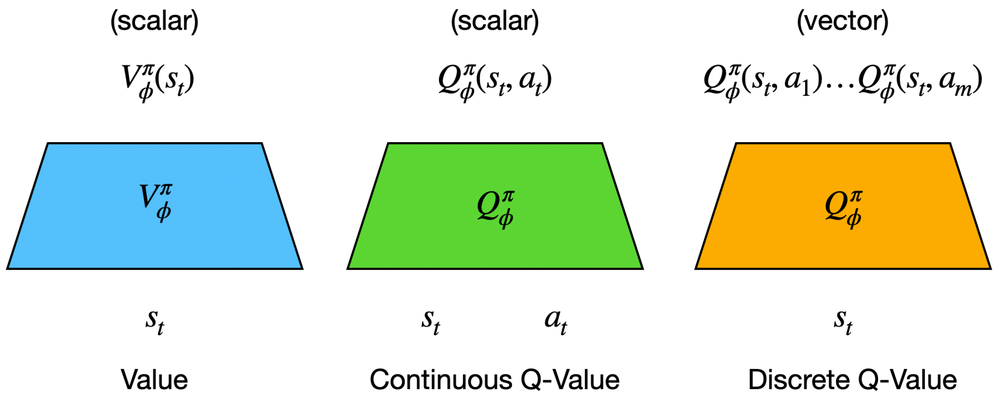
The compromise: use a monte carlo approximation for a little bit, then finish it off with a value function approximation. The “k” is your tuning knob for your tradeoff

Outline of Tutorial

- Review of Markov Decision Processes (MDP)
- Exact MDPs (tabular; simple)
  - Fitted Q Iteration
- **Parametric Q-Learning (learned models)**
- Practical Details
  - Replay Buffer, Overestimation, TD Gradients
- Algorithm Walkthrough

Moving to Parametric Functions

- A tabular value function intractable as state/action space grows
- Use neural network to learn V, Q, and π!
  - Related states have similar estimations (smoothness)
- Use regression to train



Tabular:

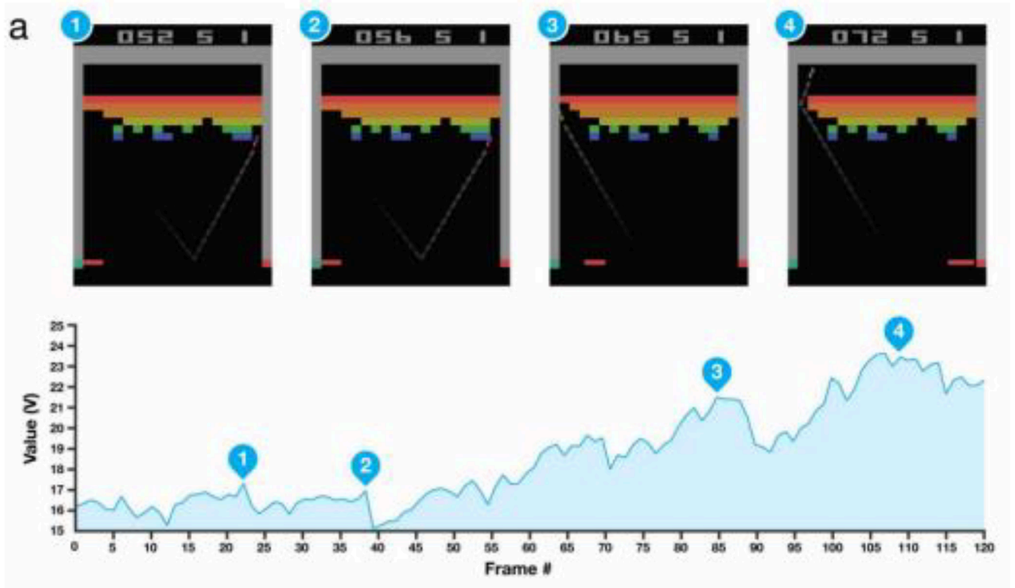
$$Q^*(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[\max_a Q^*(s_{t+1}, a) \right]$$

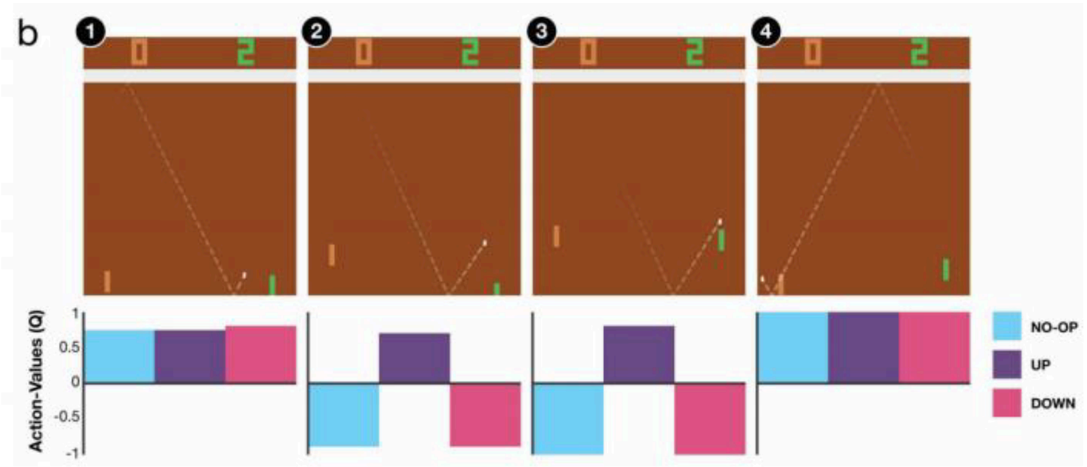
Parametric:

$$\mathscr{L}\left(Q_\phi^*(s_t, a_t), r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[\max_a Q_\phi^*(s_{t+1}, a) \right] \right)$$

Discrete vs Continuous Actions for Q Functions

- Different formulations for continuous and discrete action spaces. Example for $Q^{\pi}(s, a)$
  - In **continuous (a)**, condition on action, state. Output a scalar $Q^{\pi}(s, a)$
  - In **discrete (b)**, condition on state, output a vector $\mathbf{Q}$ such that $\mathbf{Q}_i = Q^{\pi}(s, a_i)$





Outline of Tutorial

- Review of Markov Decision Processes (MDP)
- Exact MDPs (tabular; simple)
  - Fitted Q Iteration
- Parametric Q-Learning (learned models)
- **Practical Details**
  - **Replay Buffer, Overestimation, TD Gradients**
- Algorithm Walkthrough (if extra time)

Semi-Gradient and Target Network

We have the model on both sides of the optimization!

$$Q_\theta^\pi(s,a) \leftarrow r(s,a) + \gamma \max_{\hat{a}} Q_\theta^\pi(s', \hat{a})$$

Can lead to “tail chasing” problem

How you might approach optimization (Semi-gradient)

$$Q_\theta^\pi(s,a) \leftarrow r(s,a) + \gamma \max_{\hat{a}} \textit{stopgrad}\left(Q_\theta^\pi(s', \hat{a})\right)$$

Use a target network:

$$Q_\theta^\pi(s,a) \leftarrow r(s,a) + \gamma \max_{\hat{a}} \textit{stopgrad}\left(Q_{\textit{target}}^\pi(s', \hat{a})\right)$$

Then, update the target network slowly

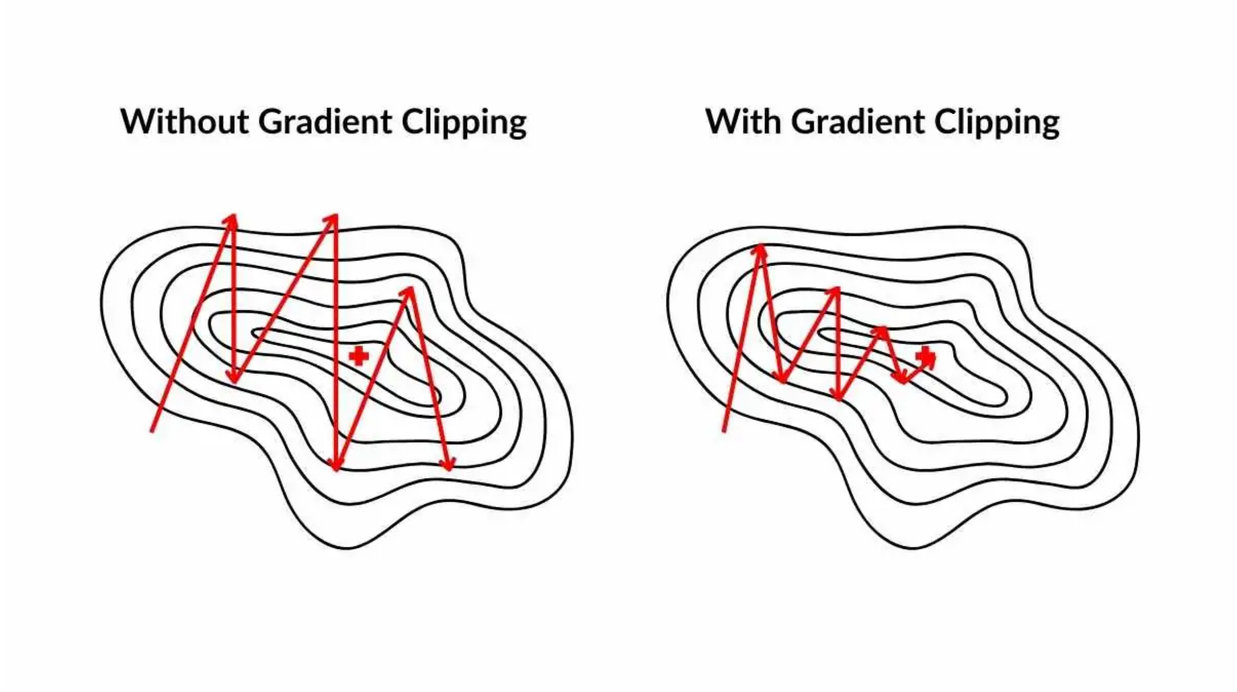
$$w' \leftarrow w \quad \textit{when} \quad \mod(n, N) = 0$$

Hard Target Update

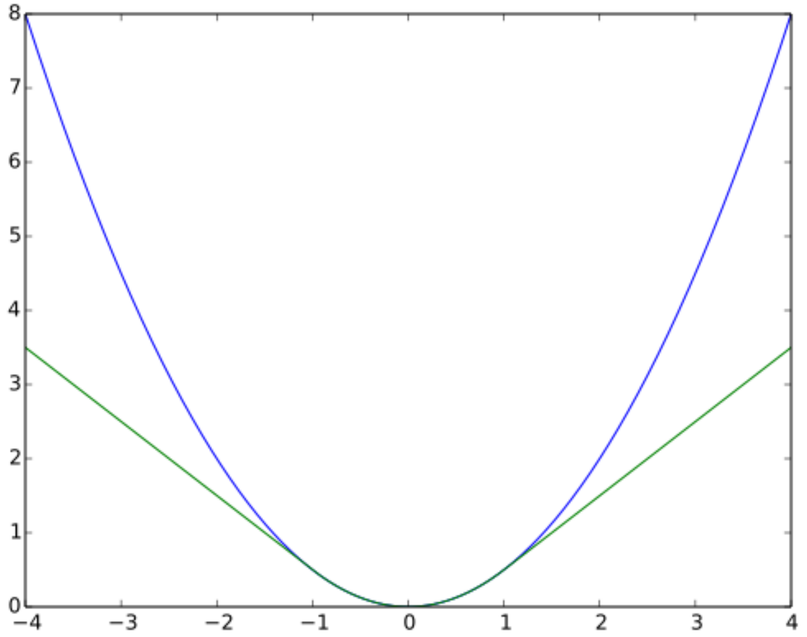
$$w' \leftarrow \tau \cdot w + (1-\tau) \cdot w'$$

Soft Target Update (Polyak)

Dealing with TD Gradients



Gradient Clipping: Prevent disruptive noise



Huber Loss: Reduce Outlier Effects

Q-Learning Overestimation

Q-Learning is typically formulated as follows:

$$Q^*(s,a) \leftarrow r(s,a) + \gamma \max_{\hat{a}} Q^*(s',\hat{a})$$

During learning, the Q function will not be correct (contain some error $Y_{a,s}$)

$$Q^*_{\theta,approx}(s',a) = Q^*(s',a) + Y_{a,s}$$

Here, Q^* is a magical, perfectly correct function

Talk to your partner: if $Y_{a,s}$ is a zero-centered error, will the error from $\max_{\hat{a}} Q^*_{\theta,approx}(s',a)$ also be zero-centered?

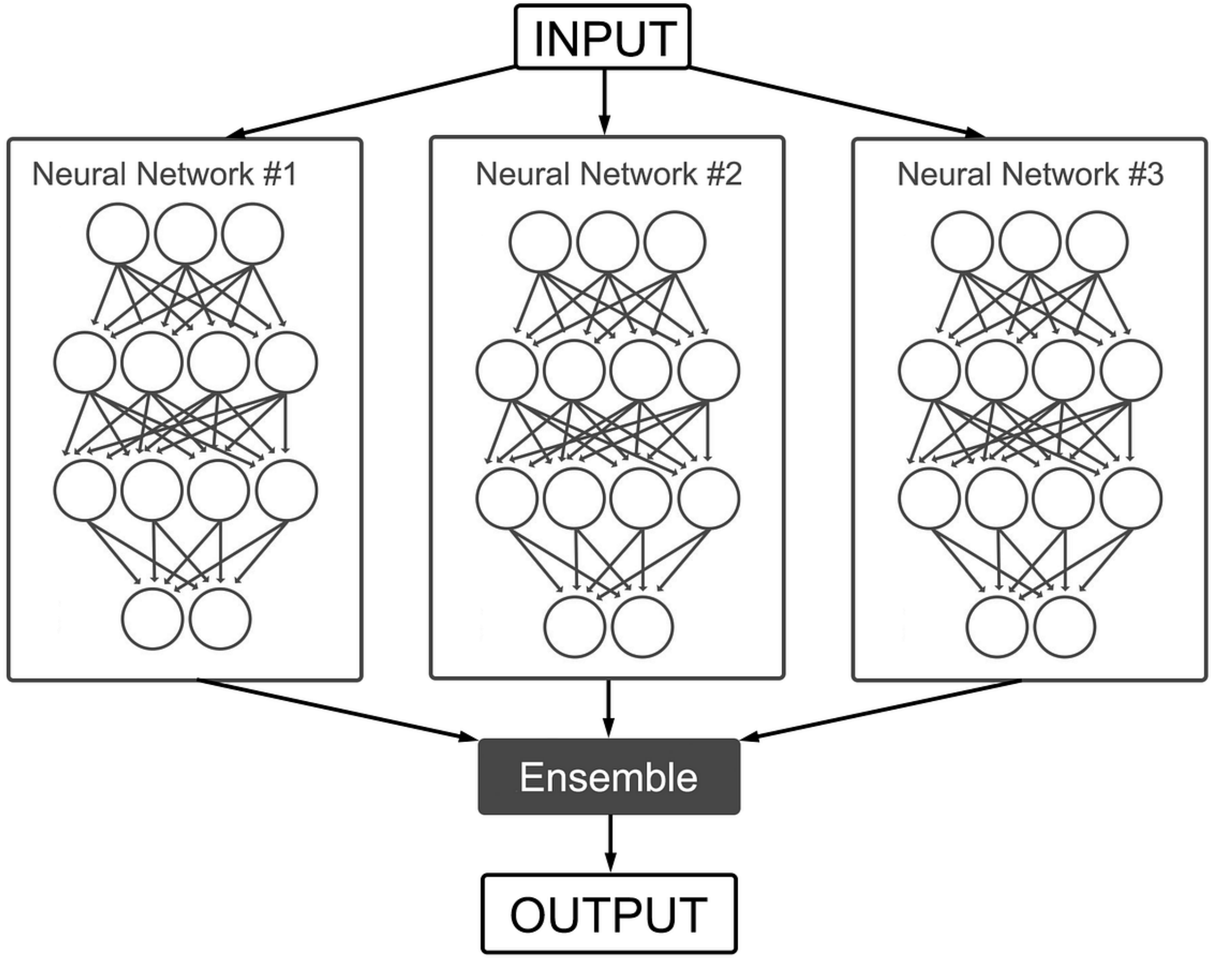
No, because you are also taking the maximum of the noise! Maximizing an unbiased noise distribution yields a biased one!

Dealing with Q overestimation: double Q learning

- If you maximize and evaluate on the same Q function, you exploit noise
- If you maximize over one Q function and evaluate on another, you reduce the risk of exploiting noise
- Old: $Q_\theta^\pi(s,a) \leftarrow r(s,a) + \gamma \max_{\hat{a}} Q_\theta^\pi(s', \hat{a})$
- New: $Q_1^*(s,a) \leftarrow r(s,a) + \gamma Q_1^*(s', \arg\max_{\hat{a}} Q_2^*(s', \hat{a}))$

Dealing with Q overestimation: critic ensembling

- Double Q learning used two Q functions to reduce noise exploitation
- Can expand to an ensemble of critics



Questions?

Outline of Tutorial

- Review of Markov Decision Processes (MDP)
- Exact MDPs (tabular; simple)
  - Fitted Q Iteration
- Parametric Q-Learning (learned models)
- Practical Details
  - Replay Buffer, Overestimation, TD Gradients
- **Algorithm Walkthrough (if extra time)**

DQN Walkthrough

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Soft Actor Critic

Algorithm 1: Soft Actor-Critic (no entropy term)

Input: Replay buffer $\mathcal{D}$, critic nets $Q_{\theta_1}, Q_{\theta_2}$, actor net π_ϕ, target critics $\theta'_i \leftarrow \theta_i$, discount γ, smoothing τ, batch size N

for *each interaction step* $t = 1, 2, \ldots$ **do**

Observe state s_t

Sample action $a_t \sim \pi_\phi(\cdot \mid s_t)$

Execute a_t, observe (r_t, s_{t+1}) and store in $\mathcal{D}$

`// Sample a mini-batch`

Sample $\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{N} \sim \mathcal{D}$

`// Compute targets (no entropy term)`

for $i = 1, \ldots, N$ **do**

$a'_{i+1} \leftarrow \pi_\phi(s_{i+1})$;

$y_i \leftarrow r_i \; + \; \gamma \, \min_{j=1,2} Q_{\theta'_j}\left(s_{i+1}, a'_{i+1}\right)$

`// Critic update`

for $j = 1, 2$ **do**

$$\theta_j \leftarrow \theta_j \; - \; \lambda_Q \, \nabla_{\theta_j} \frac{1}{N} \sum_i \left[Q_{\theta_j}(s_i, a_i) - y_i \right]^2$$

`// Actor update (maximize Q only)`

$$\phi \leftarrow \phi \; + \; \lambda_\pi \, \nabla_\phi \frac{1}{N} \sum_i Q_{\theta_1}\left(s_i, \pi_\phi(s_i)\right)$$

`// Target networks soft-update`

for $j = 1, 2$ **do**

$\theta'_j \leftarrow \tau \, \theta_j + (1 - \tau) \, \theta'_j$

Questions?