10-301/601: Introduction to Machine Learning Lecture 20: Markov Decision Processes

Henry Chai & Matt Gormley

11/8/23

Front Matter

- Announcements
  - Exam 2 on 11/9 (tomorrow!)
  - HW7 released 11/10, due 11/20 at 11:59 PM
    - Please be mindful of your grace day usage (see the course syllabus for the policy)

Recall: Mini-batch Stochastic Gradient Descent just the beginning!

- Input: training dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, step size γ, and batch size B

1. *Pre-train* the parameters $\boldsymbol{\theta}^{(0)}$ and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample B data points from $\mathcal{D}$, $\left\{\left(\boldsymbol{x}^{(b)}, y^{(b)}\right)\right\}_{b=1}^{B}$
   b. Compute the gradient of the *fine-tuning* loss
   $$\nabla J^{(B)}\left(\boldsymbol{\theta}^{(t)}\right)$$
   c. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \boxed{\gamma \nabla J^{(B)}\left(\boldsymbol{\theta}^{(t)}\right)}$
   d. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

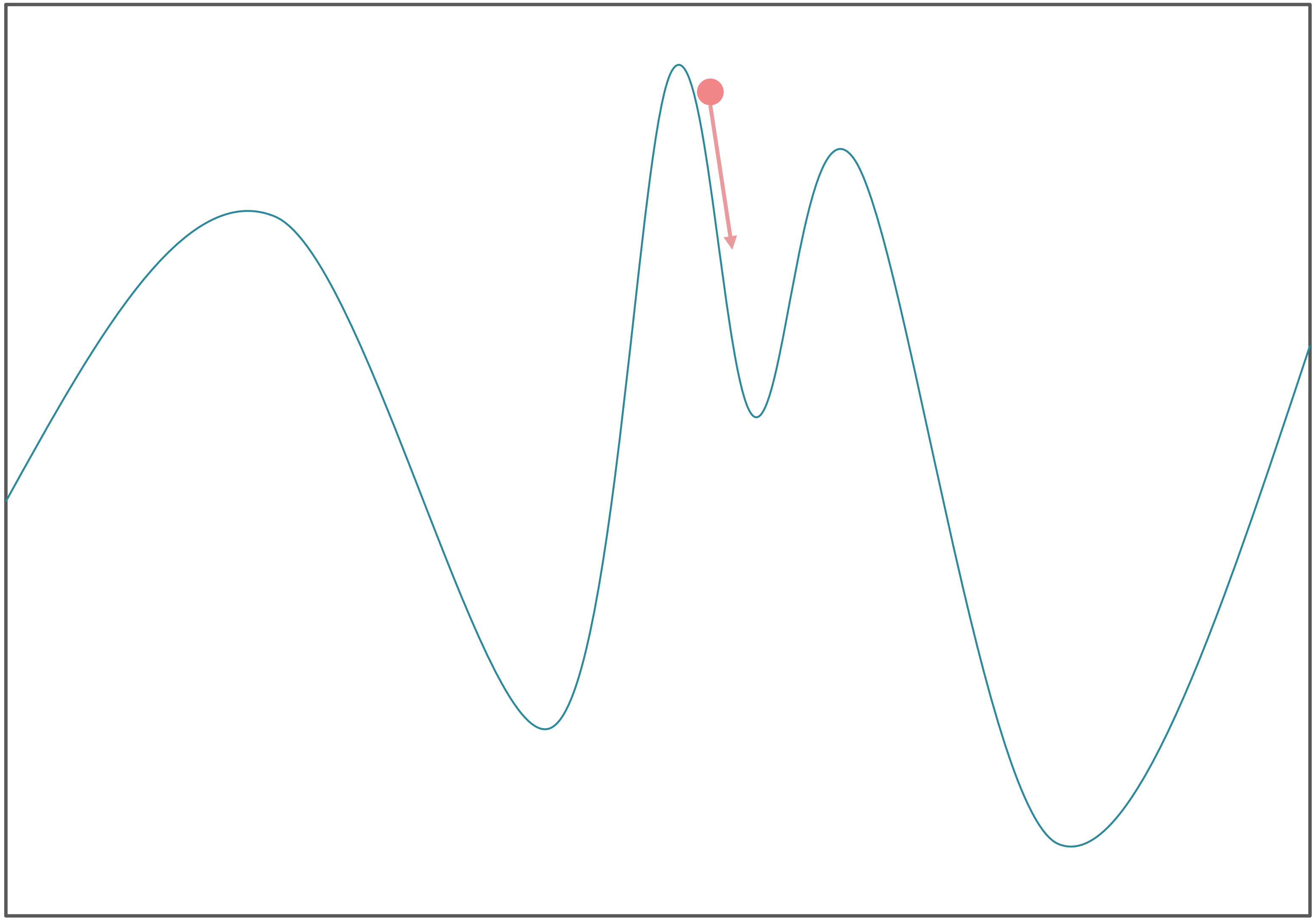
Mini-batch Stochastic Gradient Descent with Momentum

- Input: training dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, step size γ, and batch size B, decay parameter β

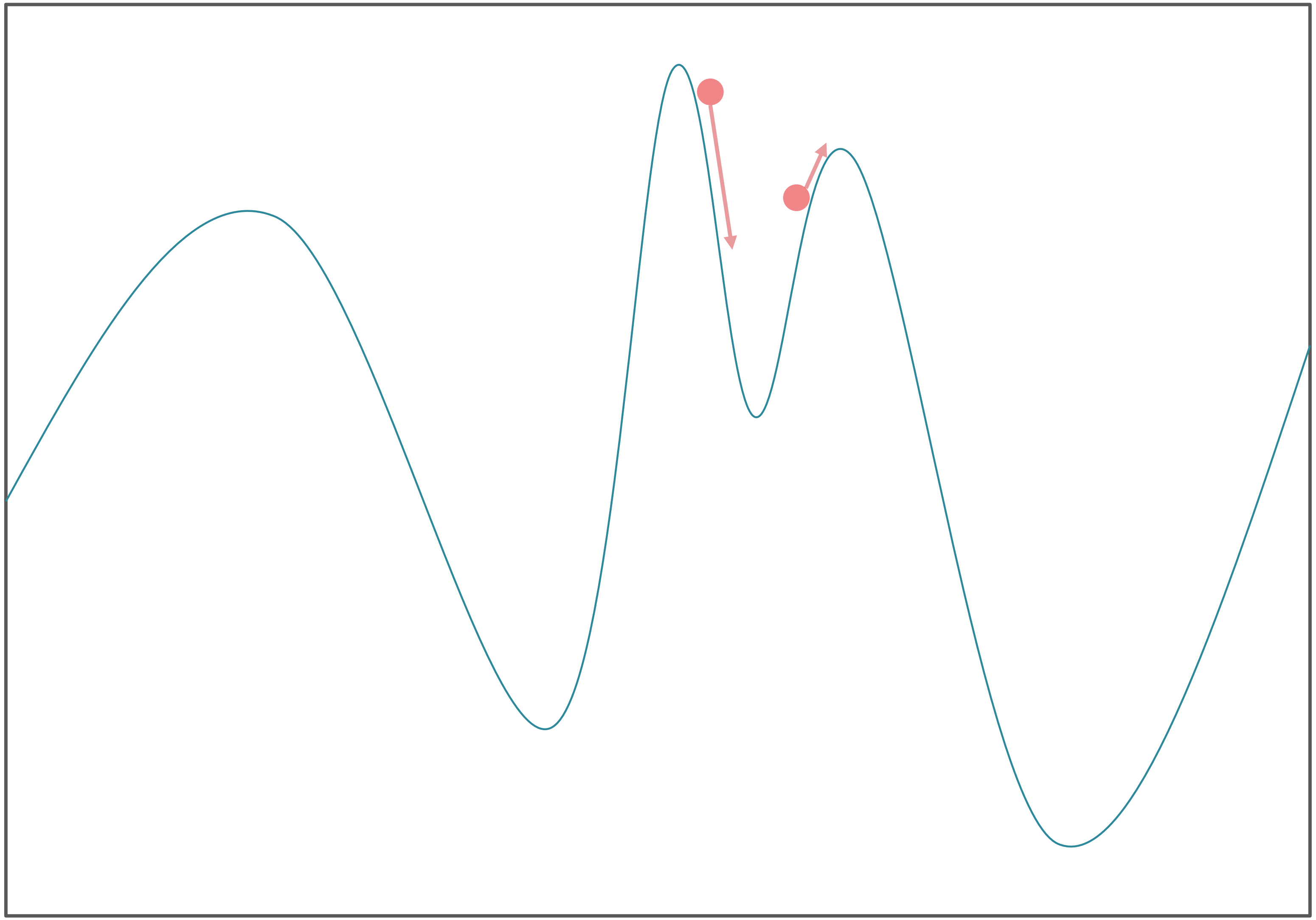
1. *Pre-train* the parameters $\boldsymbol{\theta}^{(0)}$ and set $t = 0$, $G_{-1} = 0 \odot \boldsymbol{\theta}^{(0)}$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample B data points from $\mathcal{D}$, $\left\{\left(\boldsymbol{x}^{(b)}, y^{(b)}\right)\right\}_{b=1}^{B}$
   b. Compute the gradient of the *fine-tuning* loss
   $$G_t = \nabla J^{(B)}\left(\boldsymbol{\theta}^{(t)}\right)$$
   c. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma\underbrace{(\beta G_{t-1} + G_t)}$
   d. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

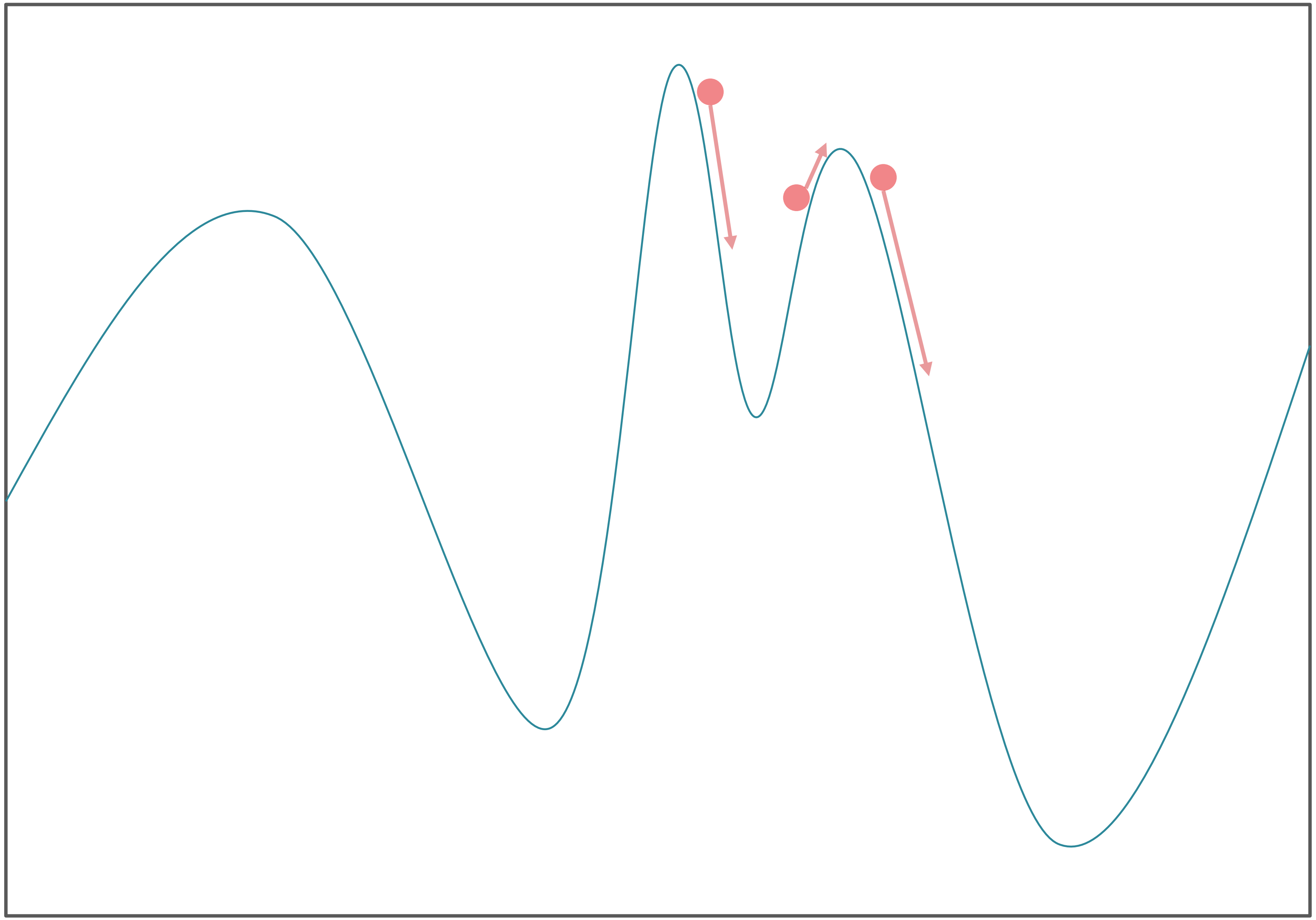
Mini-batch Stochastic Gradient Descent with Momentum



Mini-batch Stochastic Gradient Descent with Momentum



Mini-batch Stochastic Gradient Descent with Momentum



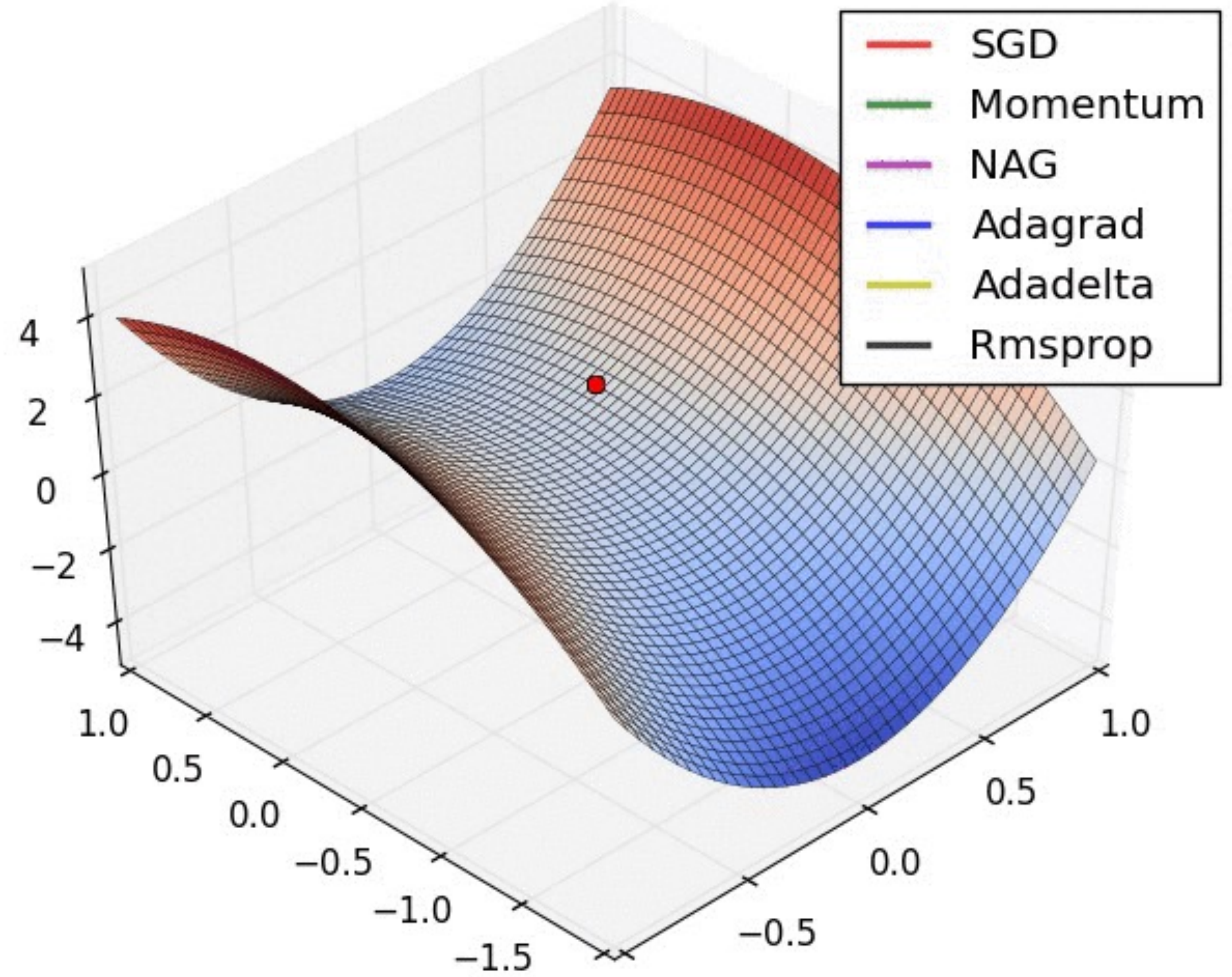
Mini-batch Stochastic Gradient Descent with Root Mean Square Propagation (RMSProp)

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, step size γ, and batch size B, decay parameter β

1. *Pre-train* the parameters $\boldsymbol{\theta}^{(0)}$ and set $t = 0$, $S_{-1} = 0 \odot \boldsymbol{\theta}^{(0)}$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample B data points from $\mathcal{D}$, $\{(\boldsymbol{x}^{(b)}, y^{(b)})\}_{b=1}^{B}$
   b. Compute the gradient of the *fine-tuning* loss
   $$G_t = \nabla J^{(B)}(\boldsymbol{\theta}^{(t)})$$
   c. Update the scaling factor: $S_t = \beta S_{t-1} + (1-\beta)(G_t \odot G_t)$
   d. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \frac{\gamma}{\sqrt{S_t}} \odot G_t$
   e. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Mini-batch Stochastic Gradient Descent with Root Mean Square Propagation (RMSProp)



Source: https://imgur.com/a/Hqolp#2dKCQHh

Adam (Adaptive Moment Estimation) = SGD + Momentum + RMSProp

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, step size γ, and batch size B, decay parameters β_1 and β_2

1. *Pre-train* the parameters $\boldsymbol{\theta}^{(0)}$, $t = 0$, $M_{-1} = S_{-1} = 0 \odot \boldsymbol{\theta}^{(0)}$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample B data points from $\mathcal{D}$, $\{(\boldsymbol{x}^{(b)}, y^{(b)})\}_{b=1}^{B}$
   b. Compute the gradient, momentum and scaling factor

$$G_t = \nabla J^{(B)}(\boldsymbol{\theta}^{(t)})$$

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) G_t \text{ and } S_t = \beta_2 S_{t-1} + (1 - \beta_2)(G_t \odot G_t)$$

   c. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \frac{\gamma}{\sqrt{S_t/(1-\beta_2^t)}} \odot (M_t/(1 - \beta_1^t))$
   d. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Recall: Reinforcement Learning from Human Feedback (RLHF)

- Insight: for many machine learning tasks, there is no universal ground truth, e.g., there are lots of possible ways to respond to a question or prompt.
- Idea: use human feedback to determine how good or bad some prediction/response is!
- Issue: if the input space is huge (e.g., all possible chat prompts), to train a good model, we might need tons and tons of (potentially expensive) human annotation…
- Idea: use a small number of annotations to learn a "reward" function!

What the heck is Reinforcement Learning?

- Insight: for many machine learning tasks, there is no universal ground truth, e.g., there are lots of possible ways to respond to a question or prompt.
- Idea: use human feedback to determine how good or bad some prediction/response is!
- Issue: if the input space is huge (e.g., all possible chat prompts), to train a good model, we might need tons and tons of (potentially expensive) human annotation…
- Idea: use a small number of annotations to learn a "reward" function!

Learning Paradigms

- Supervised learning - $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \dots, C\}$
- Reinforcement learning - $\mathcal{D} = \{(\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}, r^{(n)})\}_{n=1}^{N}$

Source: https://techobserver.net/2019/06/argo-ai-self-driving-car-research-center/

Source: https://www.wired.com/2012/02/high-speed-trading/

Reinforcement Learning: Examples

Source: https://www.cnet.com/news/boston-dynamics-robot-dog-spot-finally-goes-on-sale-for-74500/

Source: https://twitter.com/alphagomovie

AlphaGo

Source: https://www.youtube.com/watch?v=WXuK6gekU1Y&ab_channel=DeepMind

Outline

- Problem formulation
  - Time discounted cumulative reward
  - Markov decision processes (MDPs)
- Algorithms:
  - Value & policy iteration (dynamic programming)
  - (Deep) Q-learning (temporal difference learning)

Reinforcement Learning: Problem Formulation

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition function
  - Stochastic, $p(s' \mid s, a)$
  - Deterministic, $\delta: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

Reinforcement Learning: Problem Formulation

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - Specifies an action to take in *every* state
- Value function, $V^{\pi}: \mathcal{S} \rightarrow \mathbb{R}$
  - Measures the expected total payoff of starting in some state s and executing policy π, i.e., in every state, taking the action that π returns

Toy Example

- $\mathcal{S}$ = all empty squares in the grid
- $\mathcal{A}$ = {up, down, left, right}
- Deterministic transitions
- Rewards of +1 and -1 for entering the labelled squares
- Terminate after receiving either reward

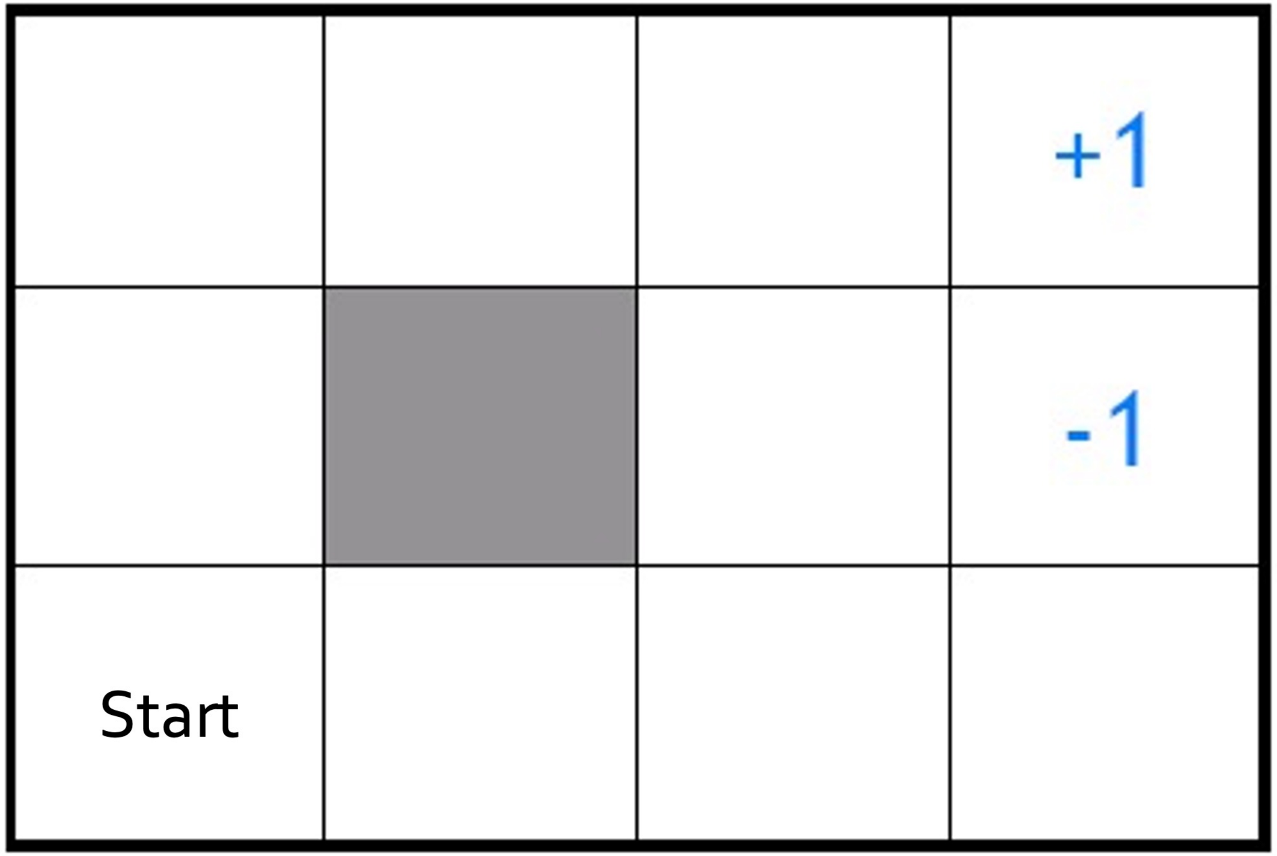


Figure courtesy of Eric Xing

Toy Example

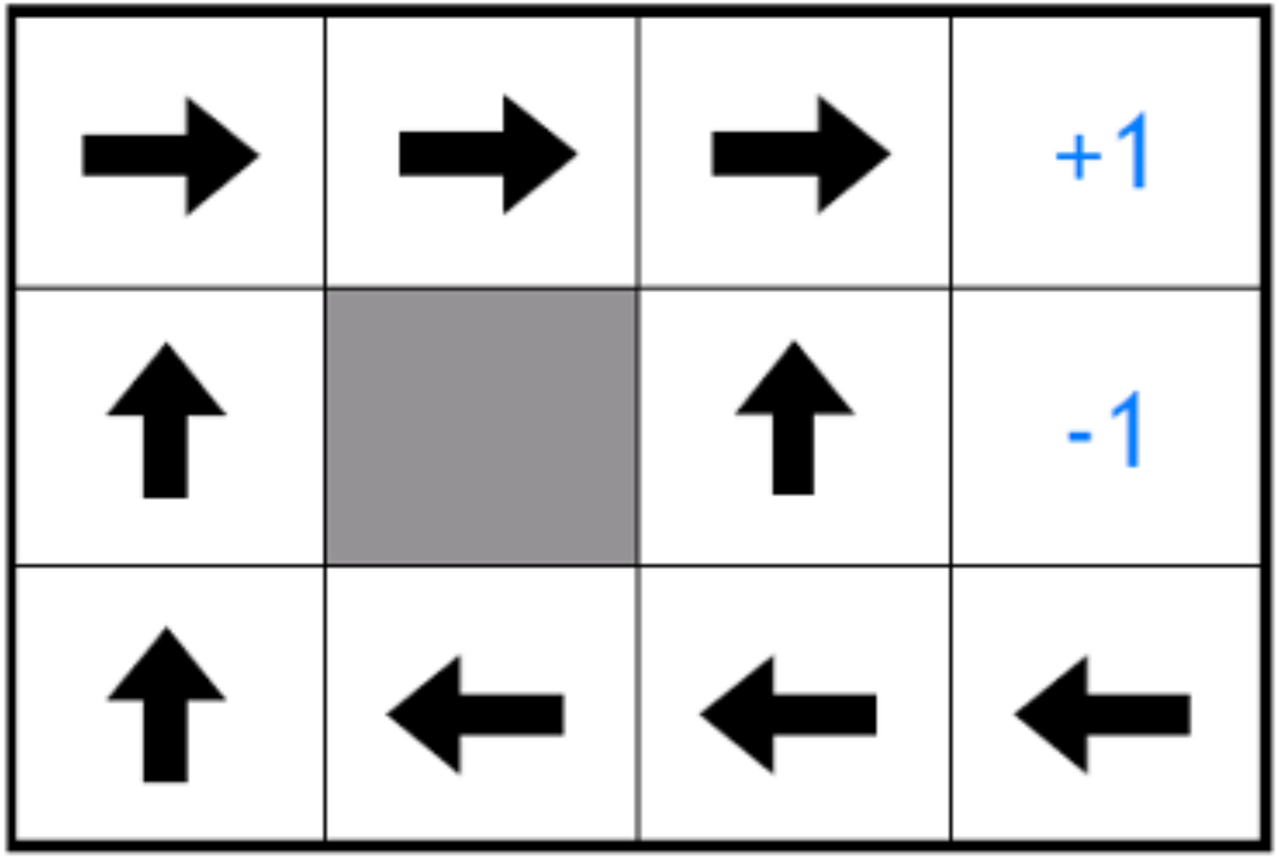


Figure courtesy of Eric Xing

Poll Question 1:

Is this policy optimal?

A. Yes

B. TOXIC

C. No

Poll Question 2:

Justify your answer to the previous question

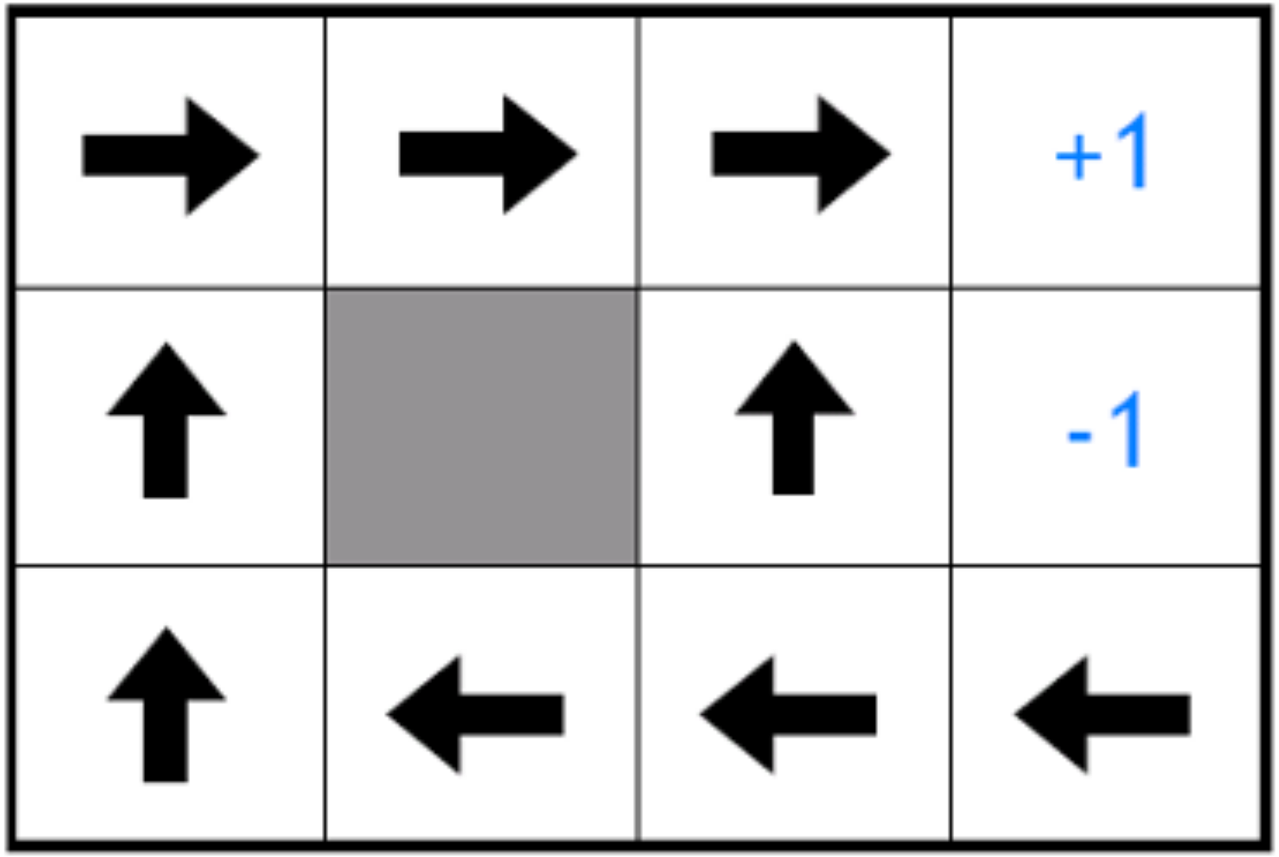


Figure courtesy of Eric Xing

Toy Example

Optimal policy given a reward of -2 per step

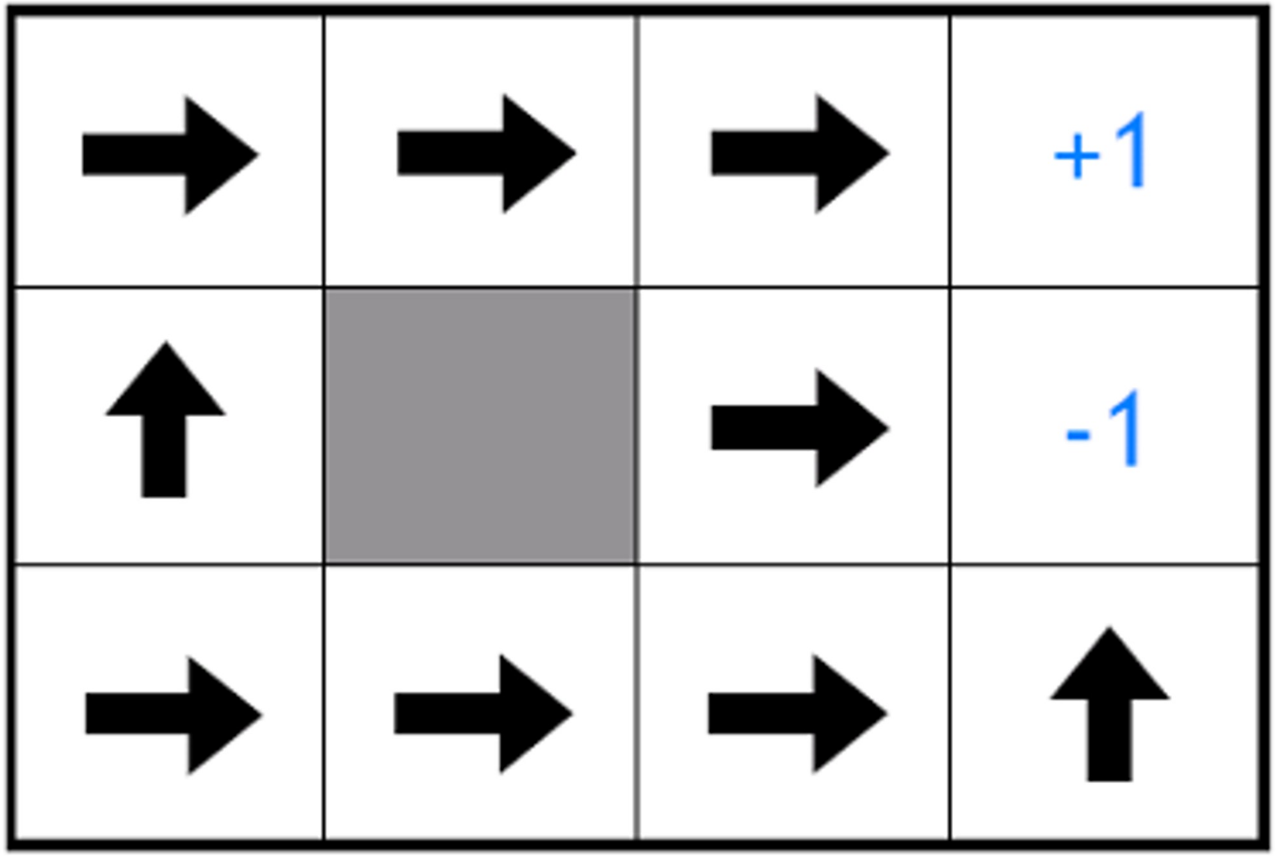


Figure courtesy of Eric Xing

Toy Example

Optimal policy given a reward of -0.1 per step

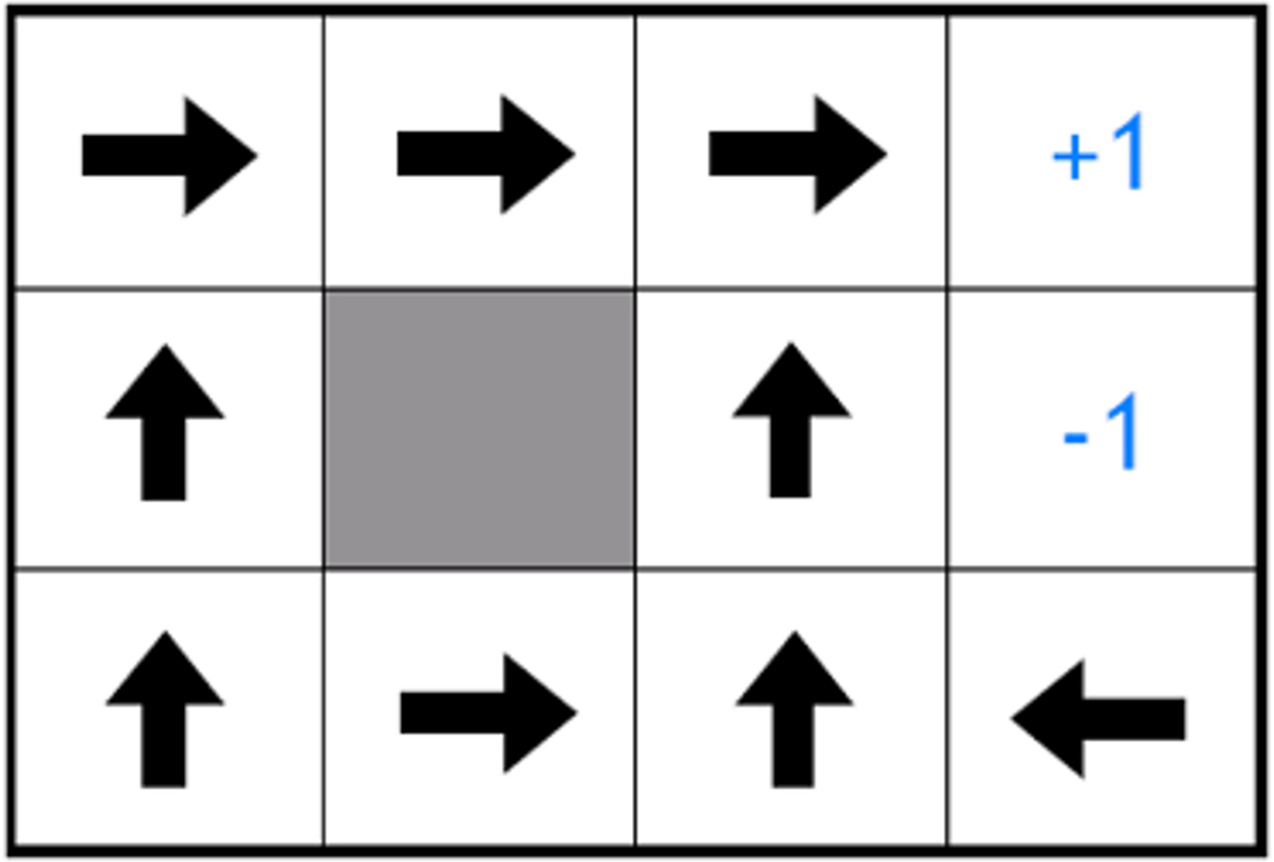


Figure courtesy of Eric Xing

Markov Decision Process (MDP)

- Assume the following model for our data:

1. Start in some initial state s_0
2. For time step t:
   1. Agent observes state s_t
   2. Agent takes action $a_t = \pi(s_t)$
   3. Agent receives reward $r_t \sim p(r \mid s_t, a_t)$
   4. Agent transitions to state $s_{t+1} \sim p(s' \mid s_t, a_t)$
3. Total reward is $\sum_{t=0}^{\infty} \gamma^t r_t$ discount factor $0 \leq \gamma < 1$

- MDPs make the *Markov assumption*: the reward and next state only depend on the current state and action.

Reinforcement Learning: Key Challenges

- The algorithm has to gather its own training data
- The outcome of taking some action is often stochastic or unknown until after the fact
- Decisions can have a delayed effect on future outcomes (exploration-exploitation tradeoff)

MDP Example: Multi-armed bandit

- Single state: $|\mathcal{S}| = 1$
- Three actions: $\mathcal{A} = \{1, 2, 3\}$
- Deterministic transitions
- Rewards are stochastic

???

???

???

???

???

MDP Example: Multi-armed bandit

???

???

???

???

???

???

???

???

???

Bandit 1	Bandit 2	Bandit 3
1	2	1
1	0	3
1	0	0
1	2	2
0	???	4
1	???	1
0	???	0
???	???	4
???	???	???
???	???	???
???	???	???
???	???	???

Reinforcement Learning: Objective Function

- Find a policy $\pi^* = \underset{\pi}{\operatorname{argmax}}\ V^{\pi}(s)\ \forall\, s \in \mathcal{S}$
- Assume deterministic transitions and deterministic rewards
- $V^{\pi}(s) =$ *discounted* total reward of starting in state s and executing policy π forever

$$= R(s_0 = s, \pi(s_0)) + \gamma R(s_1 = \delta(s_0, \pi(s_0)), \pi(s_1)) + \gamma^2 R(s_2 = \delta(s_1, \pi(s_1)), \pi(s_2)) + \ldots$$

$$= \sum_{t=0}^{\infty} \gamma^t R(s_t = \delta(s_{t-1}, \pi(s_{t-1})), \pi(s_t))$$

$$0 \le \gamma < 1$$

Reinforcement Learning: Objective Function

- Find a policy $\pi^* = \underset{\pi}{\operatorname{argmax}}\ V^{\pi}(s)\ \forall\, s \in \mathcal{S}$
- Assume stochastic transitions and deterministic rewards
- $V^{\pi}(s) = \mathbb{E}[$*discounted* total reward of starting in state s and executing policy π forever$]$

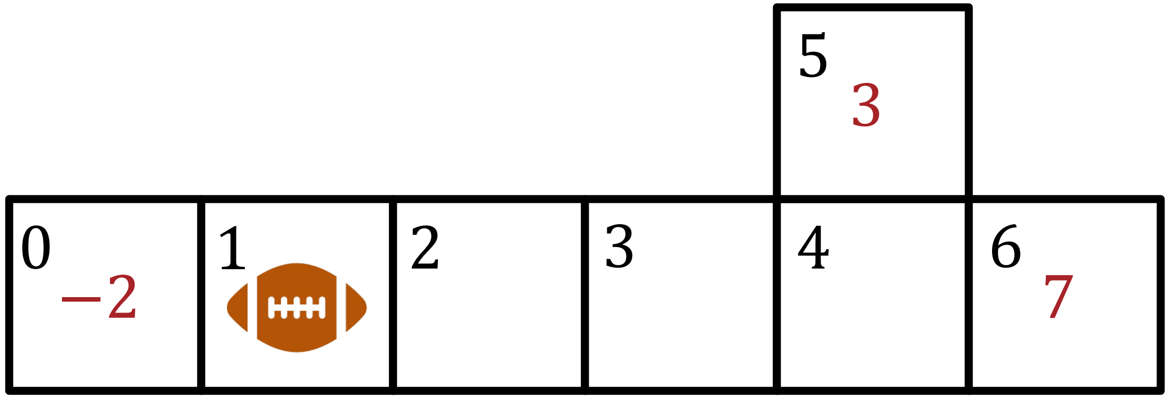
$$= E_{p(s'|s,a)}\left[R(s_0 = s, \pi(s_0)) + \gamma R(s_1, \pi(s_1)) + \gamma^2 R(s_2, \pi(s_2)) + \dots\right]$$

$$= \sum_{t=0}^{\infty} \gamma^t E_{p(s'|s,a)}\left[R(s_t, \pi(s_t))\right]$$

π

$E[a+b] = E[a] + E[b]$

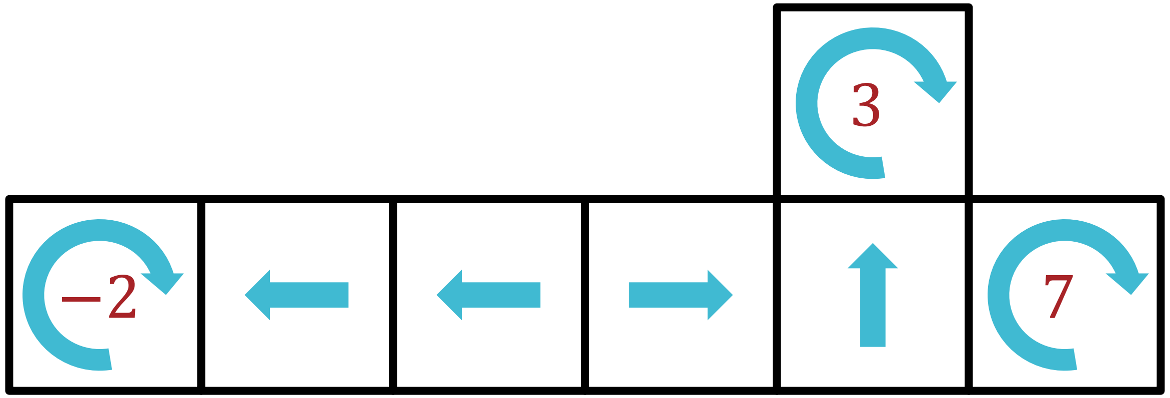
Value Function: Example



$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

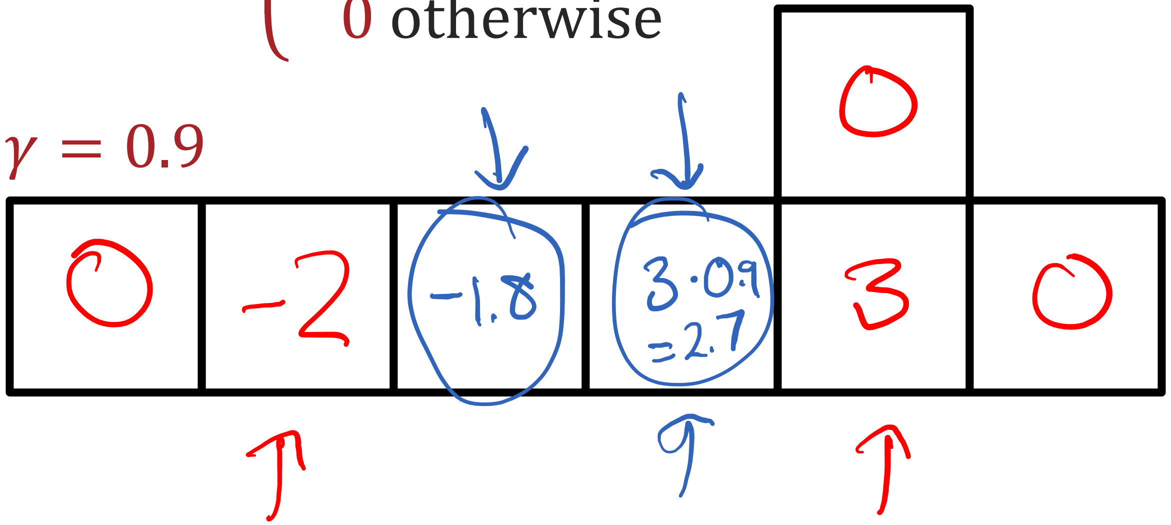
$$\gamma = 0.9$$

Value Function: Example

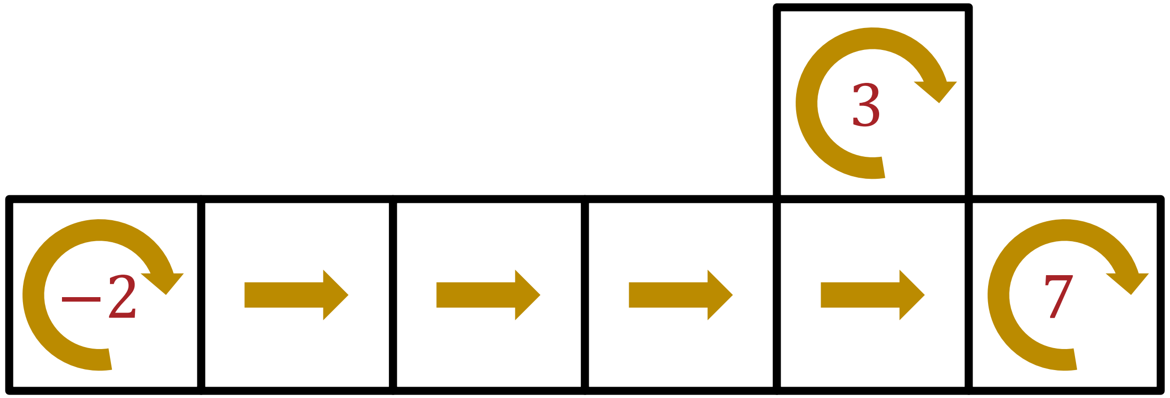


$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

$\gamma = 0.9$

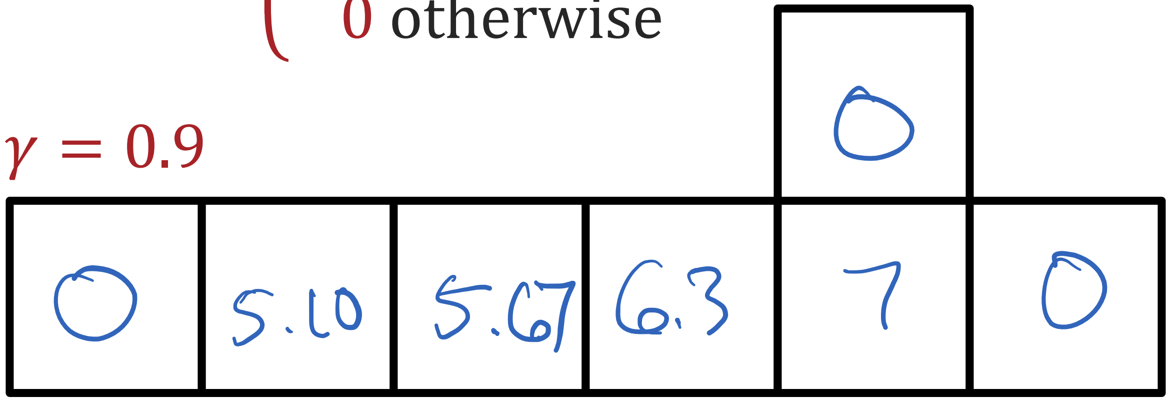


Value Function: Example

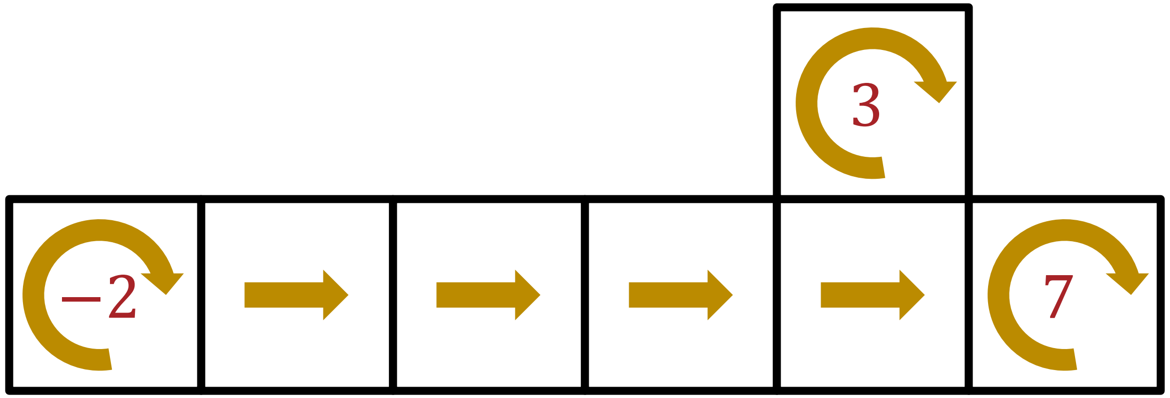


$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

$\gamma = 0.9$



Okay, now how do we go about learning this optimal policy?



$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

$\gamma = 0.9$