The Foundations of Deep RL in 6 Lectures

Lecture 1: MDPs and Exact Solution Methods

Pieter Abbeel

Lecture Series

- ***Lecture 1: MDPs Foundations and Exact Solution Methods***
- Lecture 2: Deep Q-Learning
- Lecture 3: Policy Gradients, Advantage Estimation
- Lecture 4: TRPO, PPO
- Lecture 5: DDPG, SAC
- Lecture 6: Model-based RL

Outline for This Lecture

- ***Motivation***
- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - Value Iteration
  - Policy Iteration
- Maximum Entropy Formulation

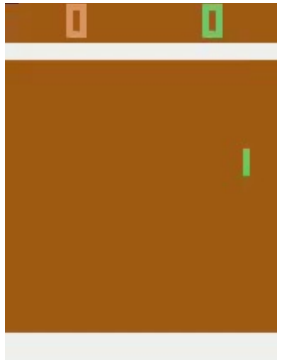
For now: small, discrete state-action spaces as they are simpler to get the main concepts across.

We will consider large state spaces in the next lecture!

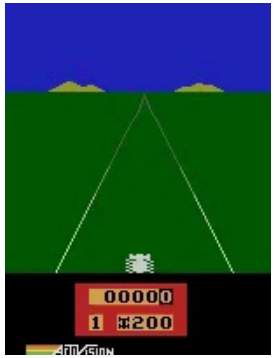
A Few Deep RL Highlights

2013

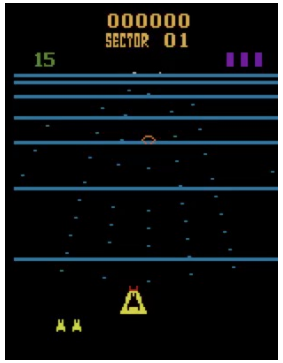
Atari (DQN)
[Deepmind]



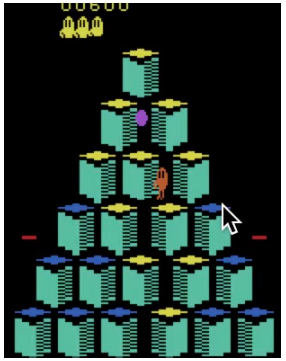
Pong



Enduro



Beamrider

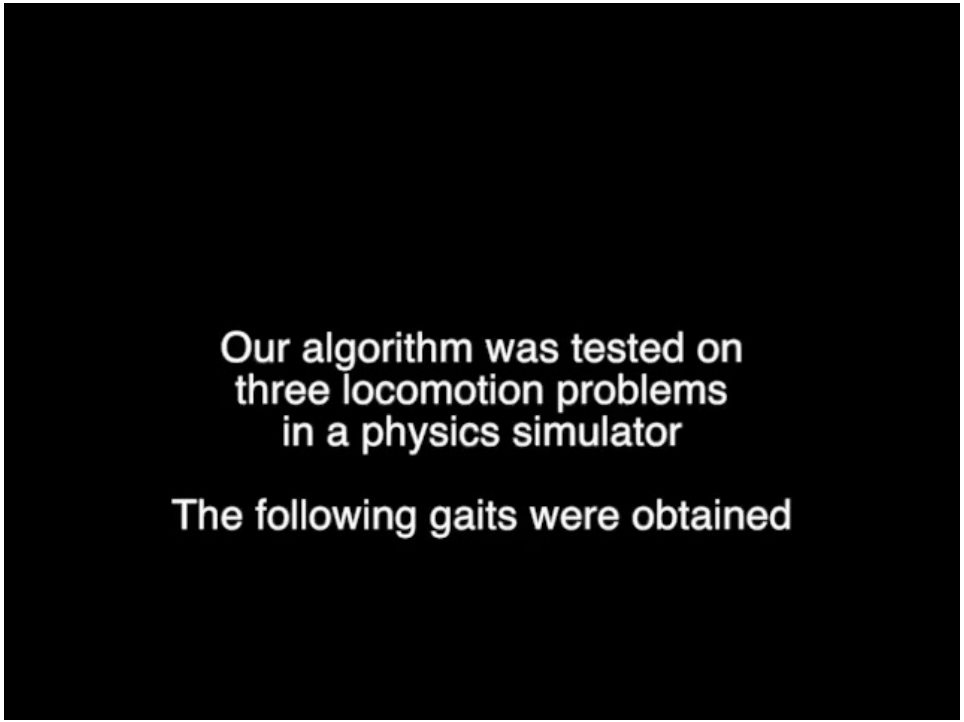


Q*bert

A Few Deep RL Highlights

2013 Atari (DQN) [Deepmind]

2014 2D locomotion (TRPO) [Berkeley]



Play 0:06 – 0:25

A Few Deep RL Highlights

2013 Atari (DQN) [Deepmind]

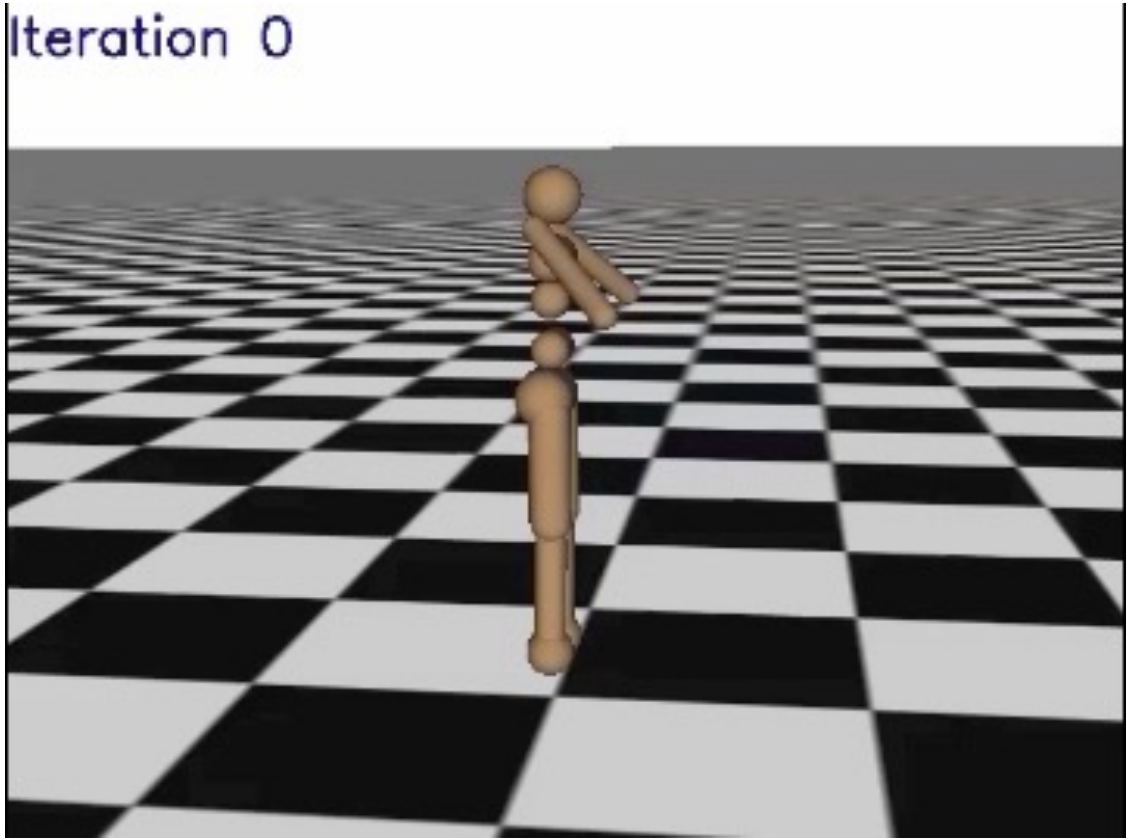
2014 2D locomotion (TRPO) [Berkeley]

2015 AlphaGo [Deepmind]

Tian et al, 2016; Maddison et al, 2014; Clark et al, 2015

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	**3D locomotion (TRPO+GAE) [Berkeley]**



[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016]

A Few Deep RL Highlights

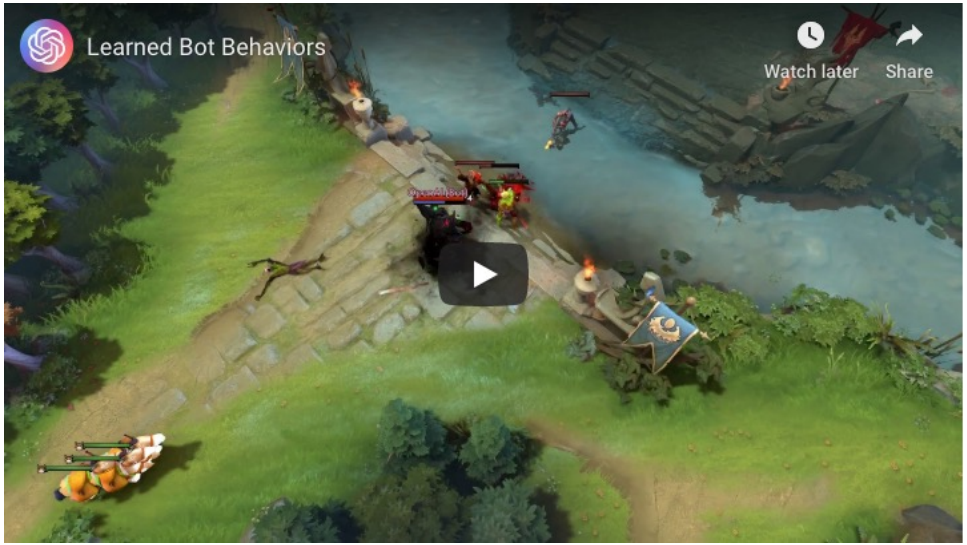
2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	**Real Robot Manipulation (GPS) [Berkeley]**

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016]


r3d11
r3d24
r3d32

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	**Dota2 (PPO) [OpenAI]**



OpenAI Dota Bot beat best humans 1:1 (Aug 2018)

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	Dota2 (PPO) [OpenAI]
2018	**DeepMimic [Berkeley]**

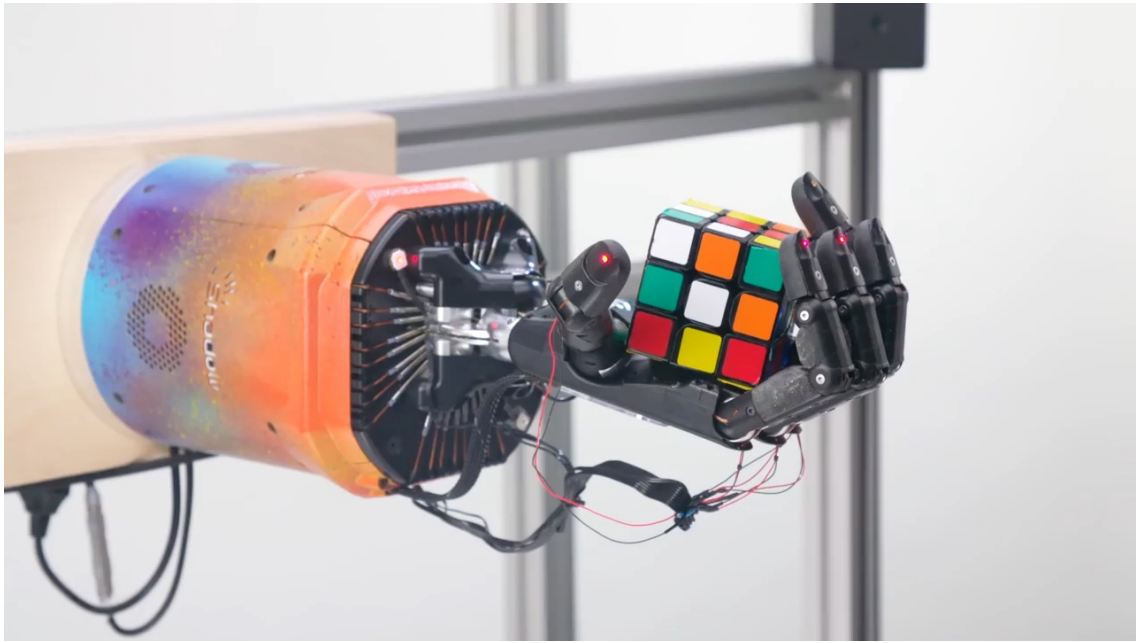
[Peng, Abbeel, Levine, van de Panne, 2018]

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	Dota2 (PPO) [OpenAI]
2018	DeepMimic [Berkeley]
2019	**AlphaStar [Deepmind]**

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	Dota2 (PPO) [OpenAI]
2018	DeepMimic [Berkeley]
2019	AlphaStar [Deepmind]
2019	**Rubik's Cube (PPO+DR) [OpenAI]**



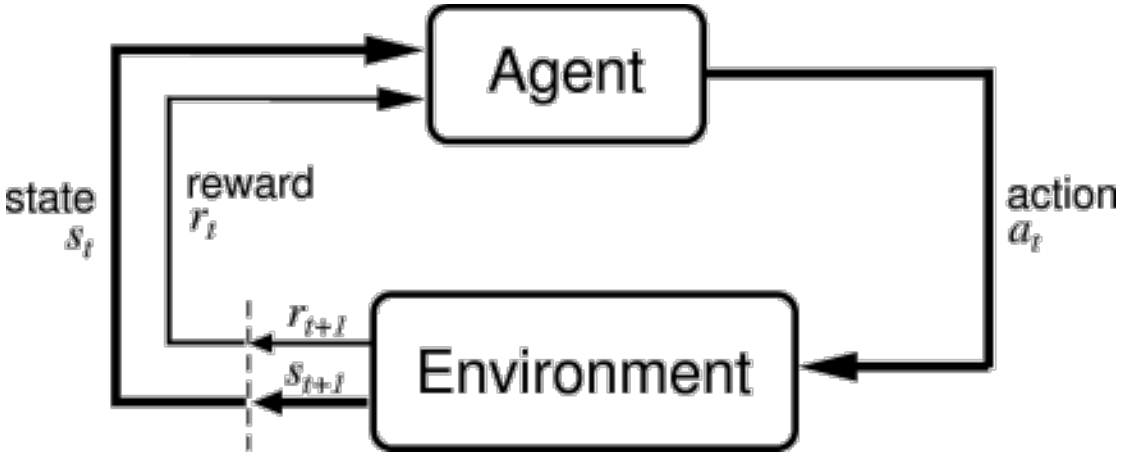
Outline for This Lecture

- Motivation
- ***Markov Decision Processes (MDPs)***
- Exact Solution Methods
  - Value Iteration
  - Policy Iteration
- Maximum Entropy Formulation

For now: small, discrete state-action spaces as they are simpler to get the main concepts across.

We will consider large state spaces in the next lecture!

Markov Decision Process



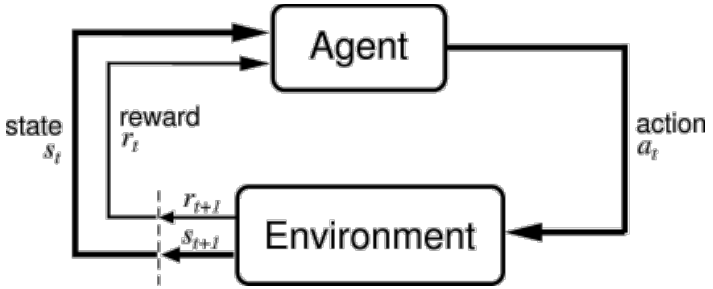
Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Markov Decision Process (MDP)

An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



Examples

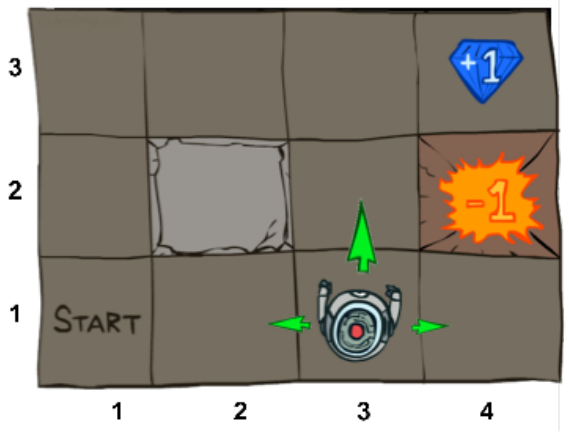
MDP (S, A, T, R, γ, H), goal: $max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1})|\pi]$

- Cleaning robot
- Walking robot
- Pole balancing
- Games: tetris, backgammon
- Server management
- Shortest path problems
- Model for animals, people

Example MDP: Gridworld

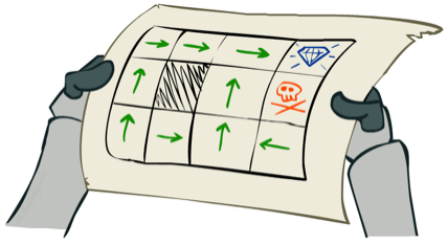
An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



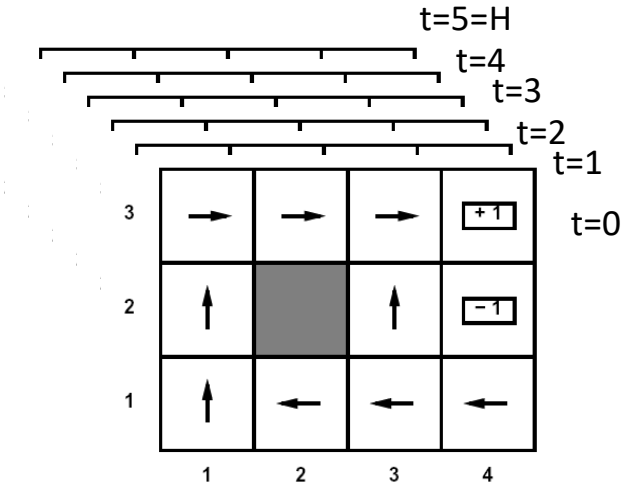
Goal: $$max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$$

π:



Solving MDPs

- In an MDP, we want to find an optimal policy $\pi^*: S \times 0{:}H \rightarrow A$
  - A policy π gives an action for each state for each time



  - An optimal policy maximizes expected sum of rewards

- Contrast: If environment were deterministic, then would just need an optimal plan, or sequence of actions, from start to a goal

Outline for This Lecture

- Motivation
- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - ***Value Iteration***
  - Policy Iteration
- Maximum Entropy Formulation

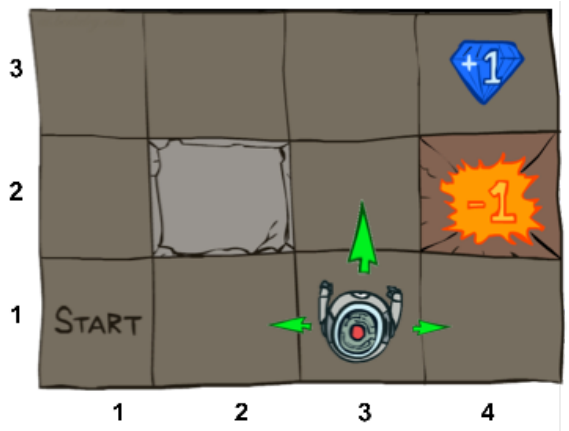
For now: small, discrete state-action spaces as they are simpler to get the main concepts across.

We will consider large state spaces in the next lecture!

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

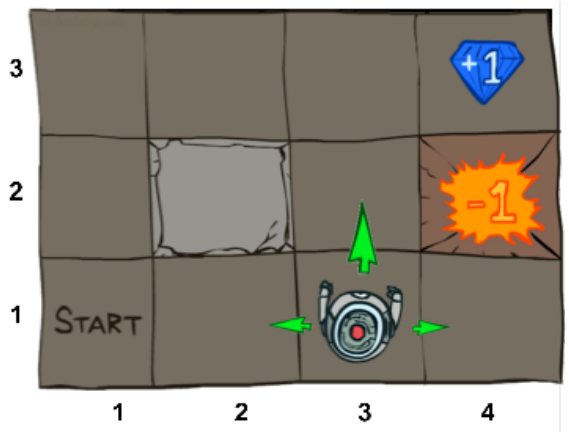
= sum of discounted rewards when starting from state s and acting optimally



Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume:

actions deterministically successful, gamma = 1, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

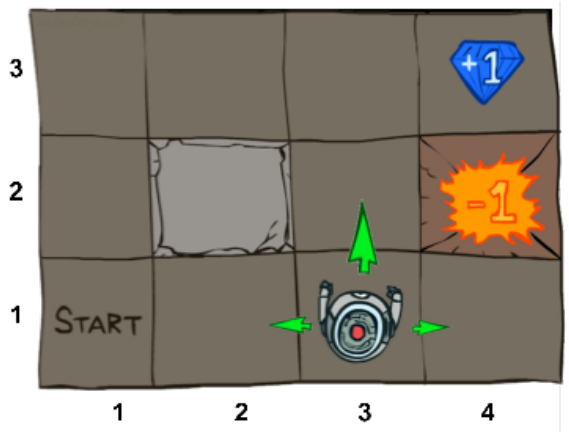
V*(1,1) =

V*(4,2) =

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume:

actions deterministically successful, gamma = 0.9, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

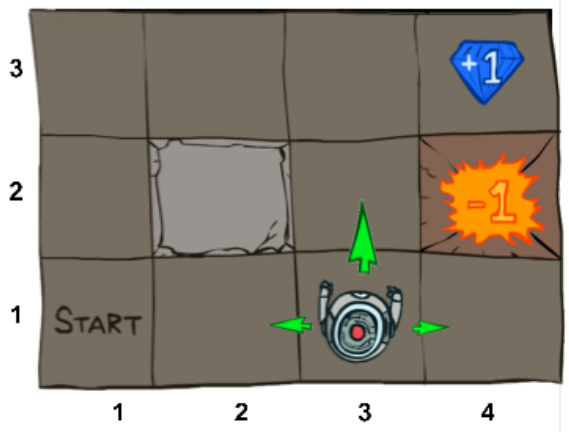
V*(1,1) =

V*(4,2) =

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume:
actions successful w/probability 0.8, gamma = 0.9, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

V*(1,1) =

V*(4,2) =

Value Iteration

- $V_0^*(s)$ = optimal value for state s when H=0
  - $V_0^*(s) = 0 \quad \forall s$
- $V_1^*(s)$ = optimal value for state s when H=1
  - $V_1^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0^*(s'))$
- $V_2^*(s)$ = optimal value for state s when H=2
  - $V_2^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when H = k
  - $V_k^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For k = 1, … , H:

For all states s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

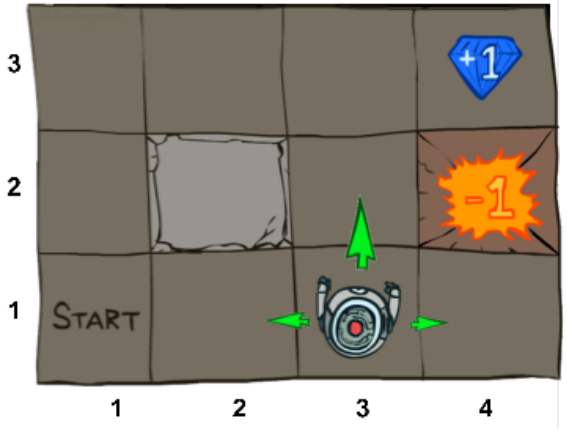
$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

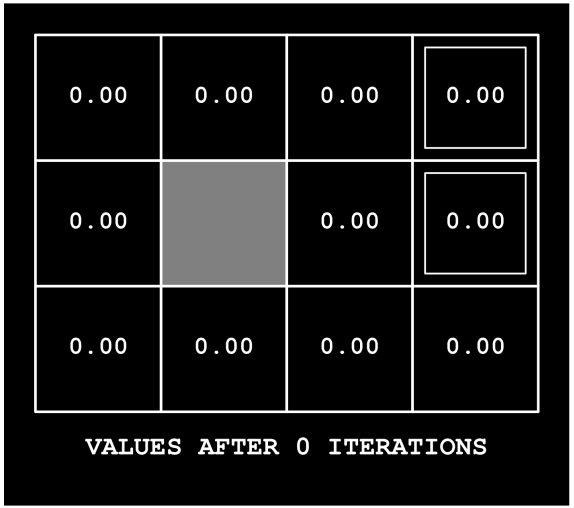
This is called a value update or Bellman update/back-up

Value Iteration

$$V_0(s) \leftarrow 0$$

k = 0



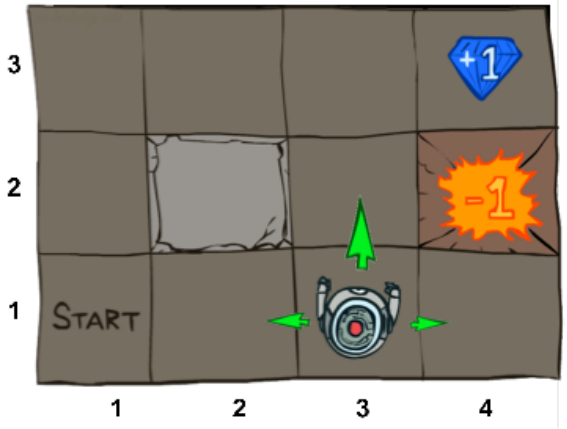


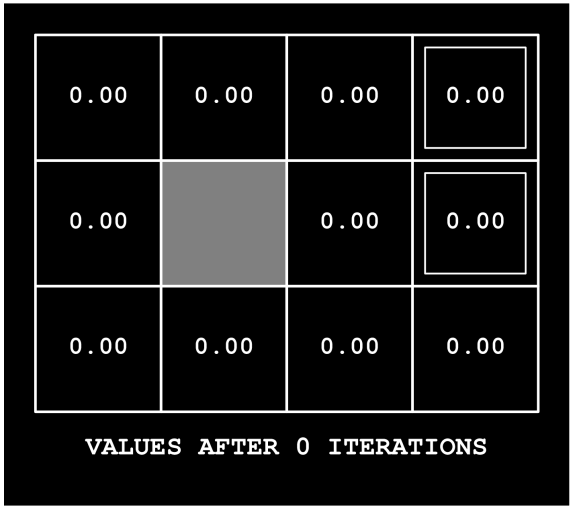
Noise = 0.2
Discount = 0.9

Value Iteration

$$V_1(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 0





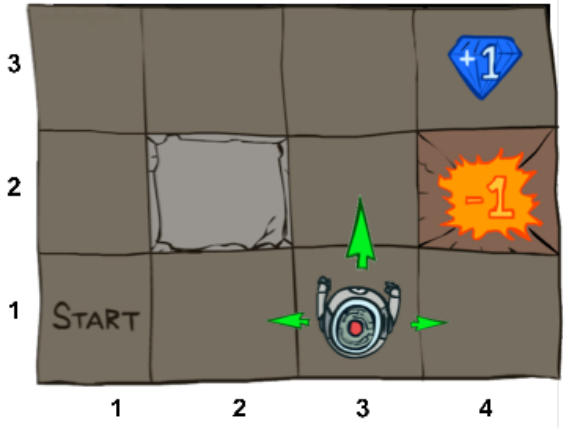
Noise = 0.2

Discount = 0.9

Value Iteration

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 1



Noise = 0.2

Discount = 0.9

Value Iteration

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 2

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 3

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 4

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 5

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 6

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 7

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 8

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 9

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 10

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 11

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 12

Noise = 0.2

Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 100

Noise = 0.2

Discount = 0.9

Value Iteration Convergence

Theorem. Value iteration converges. At convergence, we have found the optimal value function V* for the discounted infinite horizon problem, which satisfies the Bellman equations

$$\forall S \in S: \quad V^*(s) = \max_A \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

- Now we know how to act for infinite horizon with discounted rewards!
  - Run value iteration till convergence.
  - This produces V*, which in turn tells us how to act, namely following:

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- Note: the infinite horizon optimal policy is stationary, i.e., the optimal action at a state s is the same action at all times. (Efficient to store!)

Convergence: Intuition

- $V^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for ∞ steps
- $V_H^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for H steps

- Additional reward collected over time steps H+1, H+2, …

$$\gamma^{H+1}R(s_{H+1}) + \gamma^{H+2}R(s_{H+2}) + \ldots \leq \gamma^{H+1}R_{max} + \gamma^{H+2}R_{max} + \ldots = \frac{\gamma^{H+1}}{1-\gamma}R_{max}$$

goes to zero as H goes to infinity

Hence $V_H^* \xrightarrow{H\to\infty} V*$

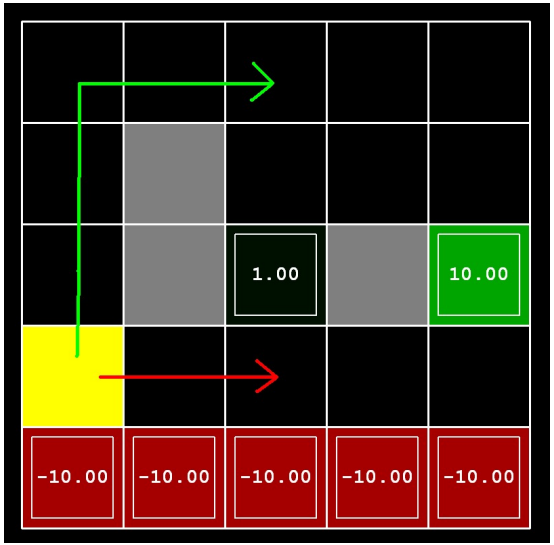
For simplicity of notation in the above it was assumed that rewards are always greater than or equal to zero. If rewards can be negative, a similar argument holds, using max |R| and bounding from both sides.

Convergence and Contractions

- Definition: max-norm: $||U|| = \max_s |U(s)|$
- Definition: An update operation is a γ-contraction in max-norm if and only if

  for all U_i, V_i: $||U_{i+1} - V_{i+1}|| \leq \gamma ||U_i - V_i||$
- Theorem: A contraction converges to a unique fixed point, no matter initialization.
- Fact: the value iteration update is a γ-contraction in max-norm
- Corollary: value iteration converges to a unique fixed point
- Additional fact: $||V_{i+1} - V_i|| < \epsilon, \Rightarrow ||V_{i+1} - V^*|| < 2\epsilon\gamma/(1-\gamma)$
  - I.e. once the update is small, it must also be close to converged

Exercise 1: Effect of Discount and Noise



(a) Prefer the close exit (+1), risking the cliff (-10)

(b) Prefer the close exit (+1), but avoiding the cliff (-10)

(c) Prefer the distant exit (+10), risking the cliff (-10)

(d) Prefer the distant exit (+10), avoiding the cliff (-10)

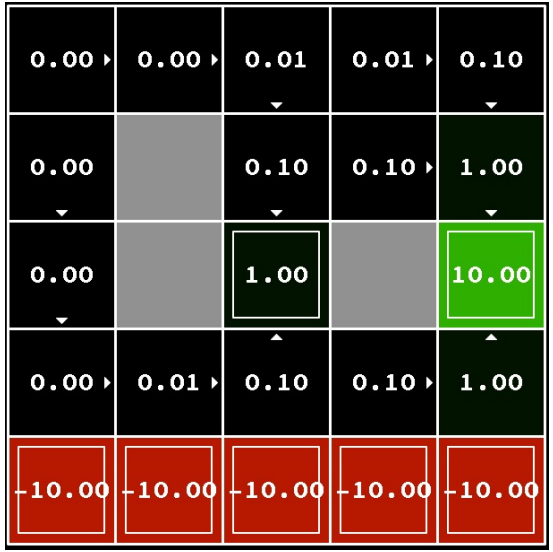
(1) $\gamma = 0.1$, noise = 0.5

(2) $\gamma = 0.99$, noise = 0

(3) $\gamma = 0.99$, noise = 0.5

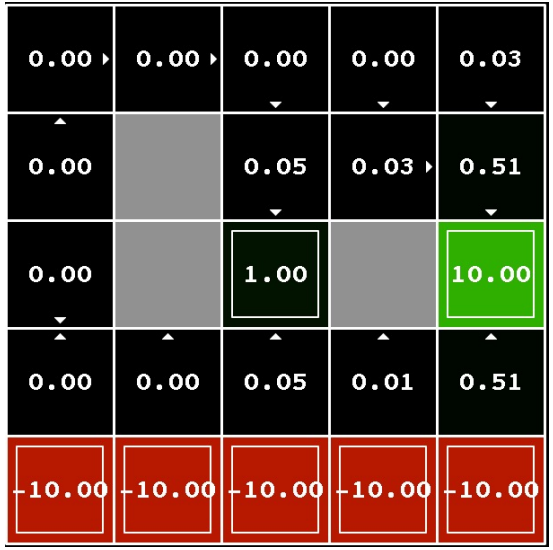
(4) $\gamma = 0.1$, noise = 0

Exercise 1 Solution



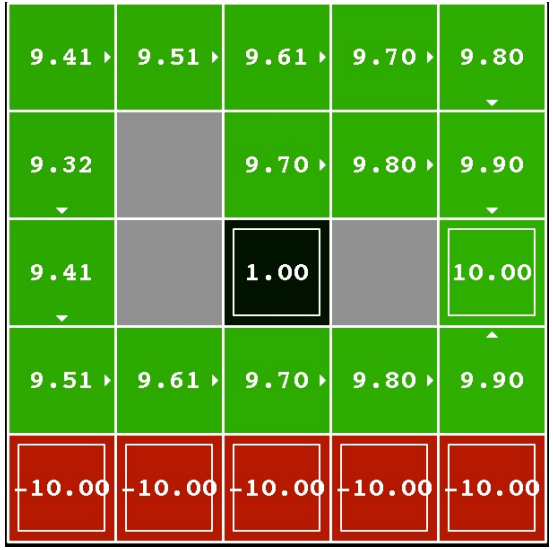
(a) Prefer close exit (+1), risking the cliff (-10) --- (4) γ = 0.1, noise = 0

Exercise 1 Solution



(b) Prefer close exit (+1), avoiding the cliff (-10) --- (1) $\gamma = 0.1$, noise = 0.5

Exercise 1 Solution



(c) Prefer distant exit (+10), risking the cliff (-10) --- (2) γ = 0.99, noise = 0

Exercise 1 Solution

(d) Prefer distant exit (+10), avoid the cliff (-10) --- (3) γ = 0.99, noise = 0.5

Q-Values

$Q^*(s, a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally

Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

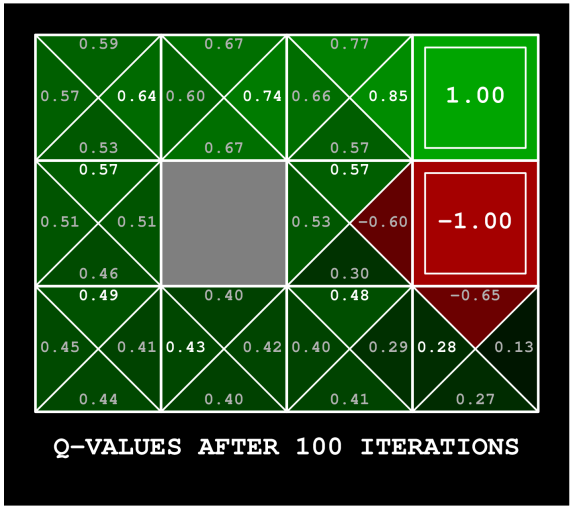
Q-Value Iteration:

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

Q-Value Iteration

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

k = 100



Noise = 0.2

Discount = 0.9

Outline for This Lecture

- Motivation
- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - Value Iteration
  - ***Policy Iteration***
- Maximum Entropy Formulation

For now: small, discrete state-action spaces as they are simpler to get the main concepts across.

We will consider large state spaces in the next lecture!

Policy Evaluation

- Recall value iteration:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

- Policy evaluation for a given $\pi(s)$:

$$V_k^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V_{k-1}^\pi(s))$$

At convergence:

$$\forall s \quad V^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V^\pi(s))$$

Exercise 2

Consider a *stochastic* policy $\pi(a|s)$, where $\pi(a|s)$ is the probability of taking action a when in state s. Which of the following is the correct update to perform policy evaluation for this stochastic policy?

1. $V^{\pi}_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V^{\pi}_{k}(s')\right)$
2. $V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} \sum_{a} \pi(a|s) P(s'|s,a)\left(R(s,a,s') + \gamma V^{\pi}_{k}(s')\right)$
3. $V^{\pi}_{k+1}(s) \leftarrow \sum_{a} \pi(a|s) \max_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V^{\pi}_{k}(s')\right)$

Policy Iteration

One iteration of policy iteration:

- Policy evaluation for current policy π_k :
  - Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) \left[R(s, \pi(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

- Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

- Repeat until policy converges
- At convergence: optimal policy; and converges faster than value iteration under some conditions

Policy Iteration Guarantees

Policy Iteration iterates over:

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates:
  - Iterate until values converge

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) \left[R(s, \pi(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

Theorem. Policy iteration is guaranteed to converge and at convergence, the current policy and its value function are the optimal policy and the optimal value function!

Proof sketch:

(1) *Guarantee to converge*: In every step the policy improves. This means that a given policy can be encountered at most once. This means that after we have iterated as many times as there are different policies, i.e., (number actions)$^{(\text{number states})}$, we must be done and hence have converged.

(2) *Optimal at convergence*: by definition of convergence, at convergence $\pi_{k+1}(s) = \pi_k(s)$ for all states s. This means $\forall s \;\; V^{\pi_k}(s) = \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_i^{\pi_k}(s')\right]$

Hence V^{π_k} satisfies the Bellman equation, which means V^{π_k} is equal to the optimal value function V*.

Outline for This Lecture

- Motivation
- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - Value Iteration
  - Policy Iteration
- ***Maximum Entropy Formulation***

For now: small, discrete state-action spaces as they are simpler to get the main concepts across.

We will consider large state spaces in the next lecture!

What if we could find a distribution over near-optimal solutions?

- More robust policy: if the environment changes, the distribution over near-optimal solutions might still have some good ones for the new situation
- More robust learning: if we can retain a distribution over near-optimal solutions our agent will collect more interesting exploratory data during learning

Entropy

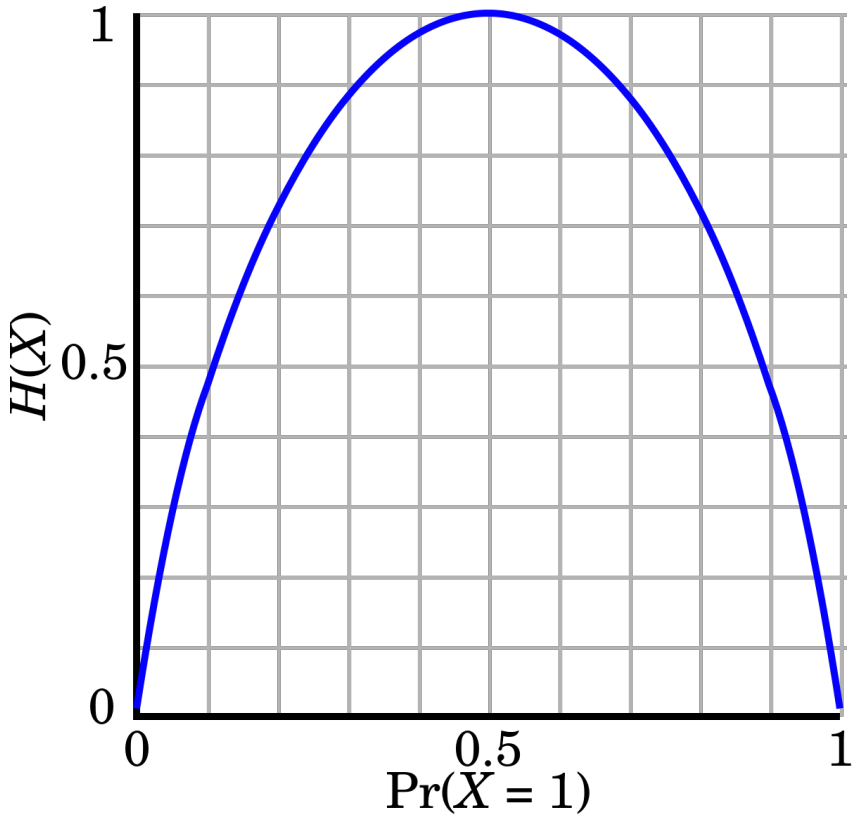
- Entropy = measure of uncertainty over random variable X

  = number of bits required to encode X (on average)

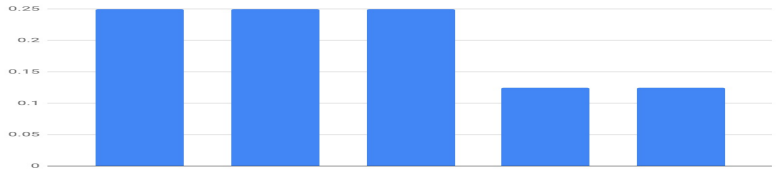
$$\mathcal{H}(X) = \sum_i p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_i p(x_i) \log_2 p(x_i)$$

Entropy

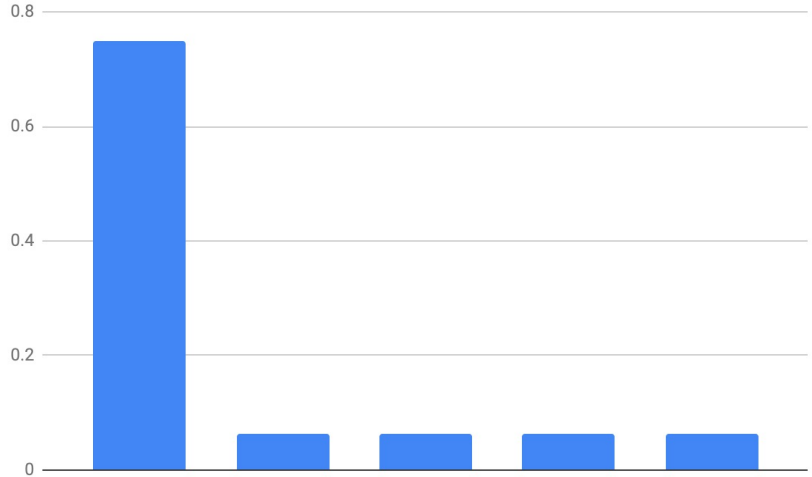
E.g. binary random variable



Entropy



$$
\begin{aligned}
p(S) &= \{0.25, 0.25, 0.25, 0.125, 0.125\} \\
H &= 3 \times 0.25 \times \log_2 4 + 2 \times 0.125 \times \log_2 8 \\
&= 1.5 + 0.75 \\
&= 2.25
\end{aligned}
$$



$$
\begin{aligned}
p(s) &= \{0.75, 0.0625, 0.0625, 0.0625, 0.0625\} \\
H &= 0.75 \times \log_2(\frac{4}{3}) + 4 \times 0.0625 \times \log_2 16 \\
&= 0.3 + 1 \\
&= 1.3
\end{aligned}
$$

Maximum Entropy MDP

- Regular formulation:

$$\max_{\pi} E\left[\sum_{t=0}^{H} r_t\right]$$

- Max-ent formulation:

$$\max_{\pi} E\left[\sum_{t=0}^{H} r_t + \beta\mathcal{H}(\pi(\cdot \mid s_t))\right]$$

Max-ent Value Iteration

- Yes, we'll cover soon, but we first need intermezzo on constrained optimization…

Constrained Optimization

- Original problem:

$$\max_x \quad f(x)$$
$$\text{s.t.} \quad g(x) = 0$$

- Lagrangian:

$$\max_x \min_\lambda \mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$$

- At optimum:

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial x} = 0$$

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial \lambda} = 0$$

Max-ent for 1-step problem

$$\max_{\pi(a)} \; E\left[r(a)\right] + \beta \mathcal{H}(\pi(a))$$

$$\max_{\pi(a)} \; \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a)$$

$$\max_{\pi(a)} \min_{\lambda} \; \mathcal{L}(\pi(a), \lambda) = \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a) + \lambda \left(\sum_a \pi(a) - 1\right)$$

$$\frac{\partial}{\partial \pi(a)} \mathcal{L}(\pi(a), \lambda) = 0$$

$$\frac{\partial}{\partial \pi(a)} \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a) + \lambda \left(\sum_a \pi(a) - 1\right) = 0$$

$$r(a) - \beta \log \pi(a) - \beta + \lambda = 0$$

$$\beta \log \pi(a) = r(a) - \beta + \lambda$$

$$\pi(a) = \exp\left[\frac{1}{\beta}\left(r(a) - \beta + \lambda\right)\right]$$

$$\frac{\partial}{\partial \lambda} \mathcal{L}(\pi(a), \lambda) = 0$$

$$\sum_a \pi(a) - 1 = 0$$

$$\pi(a) = \frac{1}{Z} \exp\left(\frac{1}{\beta} r(a)\right) \qquad Z = \sum_a \exp\left(\frac{1}{\beta} r(a)\right)$$

Max-ent for 1-step problem

$$\max_{\pi(a)} \; E\left[r(a)\right] + \beta \mathcal{H}(\pi(a))$$

$$\max_{\pi(a)} \sum_a \pi(a) r(a) - \beta \sum_a \pi(a) \log \pi(a)$$

$$\pi(a) = \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \qquad Z = \sum_a \exp(\frac{1}{\beta} r(a))$$

$$\begin{aligned}
V = & \quad \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) r(a) - \beta \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \log \left(\frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \right) \\
= & \quad \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \left(r(a) - \beta \log \left(\exp(\frac{1}{\beta} r(a)) \right) \right) - \beta \sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \log \frac{1}{Z} \\
= & \quad 0 - \beta \log \frac{1}{Z} \textstyle\sum_a \frac{1}{Z} \exp(\frac{1}{\beta} r(a)) \\
= & \quad -\beta \log \frac{1}{Z} \\
= & \quad \beta \log \textstyle\sum_a \exp(\frac{1}{\beta} r(a)) \qquad \text{= softmax}
\end{aligned}$$

Max-ent Value Iteration

$$\max_{\pi} E\left[\sum_{t=0}^{H} r_t + \beta\mathcal{H}(\pi(\cdot \mid s_t))\right] \qquad V_k(s) = \max_{\pi} E\left[\sum_{t=H-k}^{H} r(s_t, a_t) + \beta\mathcal{H}(\pi(a_t|s_t)\right]$$

$$\begin{aligned} V_k(s) = &\quad \max_{\pi} E\left[r(s,a) + \beta\mathcal{H}(\pi(a|s) + V_{k-1}(s')\right] \\ = &\quad \max_{\pi} E\left[Q_k(s,a) + \beta\mathcal{H}(\pi(a|s)\right] \end{aligned}$$

$$Q_k(s,a) = E\left[r(s,a) + V_{k-1}(s')\right]$$

= 1-step problem (with Q instead of r), so we can directly transcribe solution:

$$V_k(s) = \beta \log \sum_a \exp(\frac{1}{\beta} Q_k(s,a)) \qquad \pi_k(a|s) = \frac{1}{Z} \exp(\frac{1}{\beta} Q_k(s,a))$$

Outline for This Lecture

- Motivation
- Markov Decision Processes (MDPs)
- Exact Solution Methods
  - Value Iteration
  - Policy Iteration
- Maximum Entropy Formulation

For now: small, discrete state-action spaces as they are simpler to get the main concepts across.

We will consider large state spaces in the next lecture!