EECS 545: Machine Learning

Lecture 21. Reinforcement Learning

Honglak Lee and Michał Dereziński

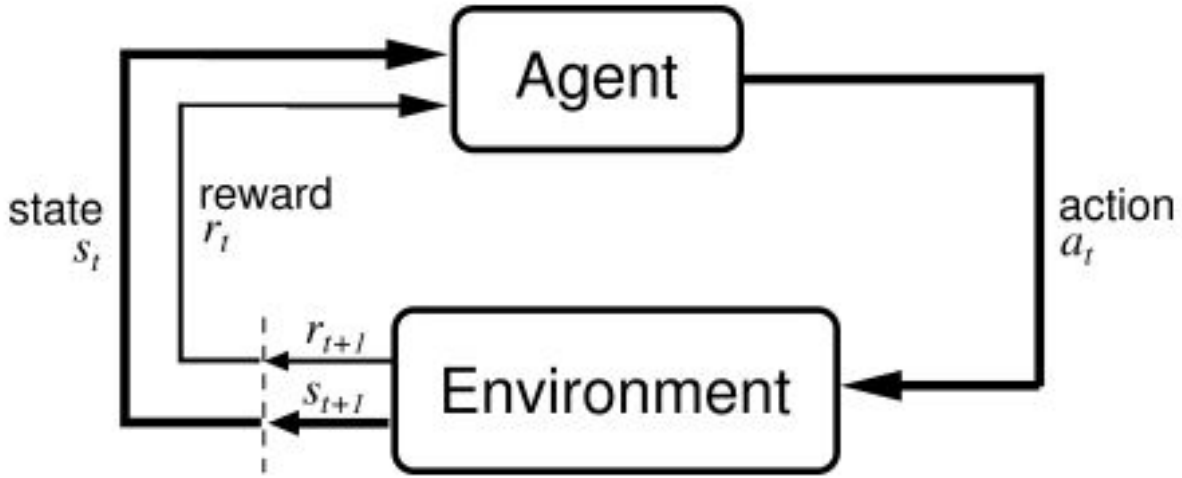
03/28/2022

Outline

- Introduction
- Multi-arm Bandit Problem
- Markov Decision Process
- Value Iteration

Reinforcement Learning (RL)

- The *reinforcement learning problem* is how an agent in an environment can select its actions to **maximize its long-term rewards**.

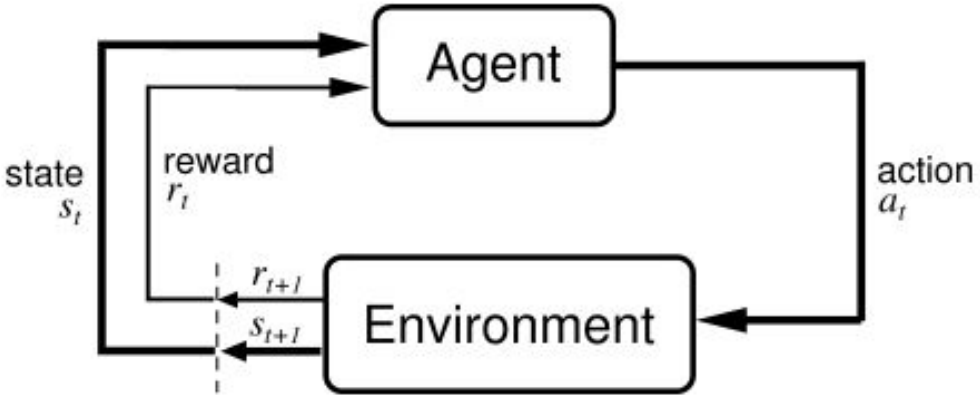


Strengths of the RL Framework

- RL deals with a *complete* (simple) ***agent*** behaving in an environment.
  - Supervised and unsupervised learning are parts of some larger, unspecified, structure.
- RL makes explicit the *trade-off* between
  - **Exploration:** acting to learn the environment,
  - **Exploitation:** acting to maximize reward.

Formalizing the RL Framework

- At each time t = 0, 1, 2, 3, . . .
- The agent perceives a *state* $s_t \in \mathcal{S}$
- It selects an *action* $a_t \in \mathcal{A}(s_t)$
- Then it receives a *reward* $r_{t+1} \in \Re$
- and finds itself in a new state s_{t+1}



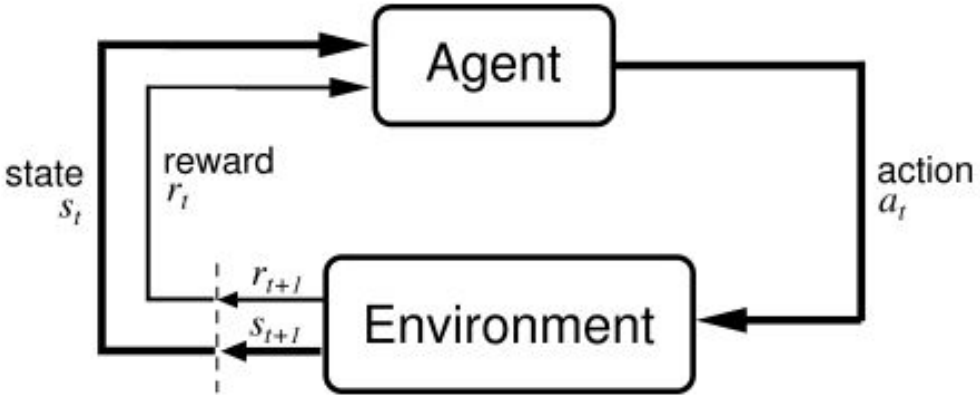
The Environment is Uncertain

- Uncertain result and reward from action.

$$Pr\{s_{t+1} = s', r_{t+1} = r' | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\}$$

- We usually assume the *Markov property*.

$$Pr\{s_{t+1} = s', r_{t+1} = r' | s_t, a_t\}$$



Transitions & Expected Rewards

- State transition probabilities:

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

- Expected rewards:

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} = r' | s_t = s, a_t = a, s_{t+1} = s'\}$$

Objective Function in RL: Expected Future Rewards

- We could just add up future rewards:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots r_T$$

- Typically we *discount* future rewards:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- This permits an infinite time-horizon.
  - Near-term rewards are more important than the long-term future.

Outline

- Introduction
- **Multi-arm Bandit Problem**
- Markov Decision Process
- Value Iteration

Consider a Simplified Problem: Bandit

- One state. No state transitions.
  - In the full RL problem, a more complex version of this problem occurs at each state.
- Choice of actions: 1, …, K
- Unknown *distributions* of rewards per action.
  - Exploration versus Exploitation.

The K-Armed Bandit Problem

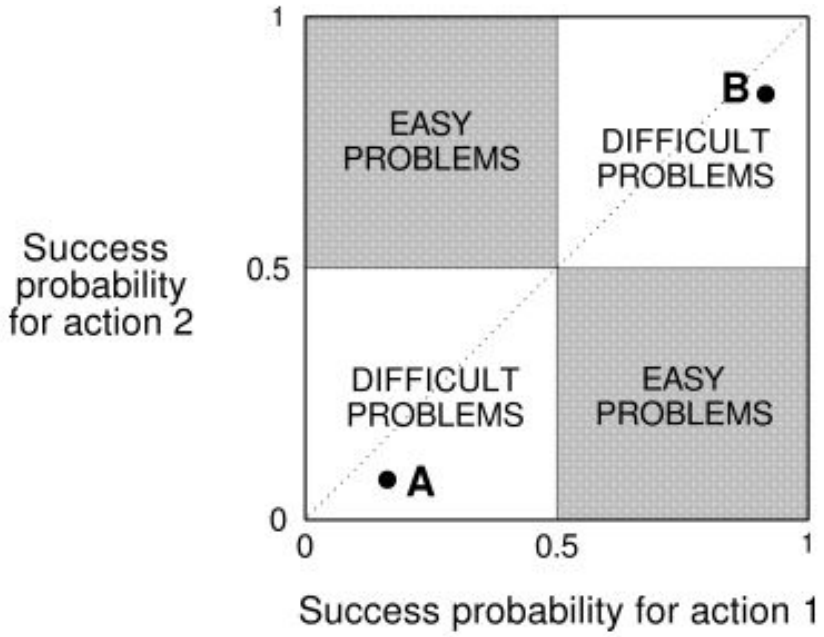
- Each arm has a different, *unknown*, distribution.
- How do you learn the distribution to maximize payoff?

K-Armed Bandit Problems

- Let the true value of action a be $Q^*(a)$.
  - One way to estimate: $$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$
- Exploitation: the *greedy* strategy always selects action *a* with the highest $Q_t(a)$.
  - With incomplete knowledge, this may ignore a much better selection.
- Exploration: select action *a* to improve the estimate $Q_t(a)$. (Randomly?)
  - How much exploration still pays off?

K-Armed Bandit Problems

- Some versions are easier than others.



Simulation: The 10-armed bandit testbed

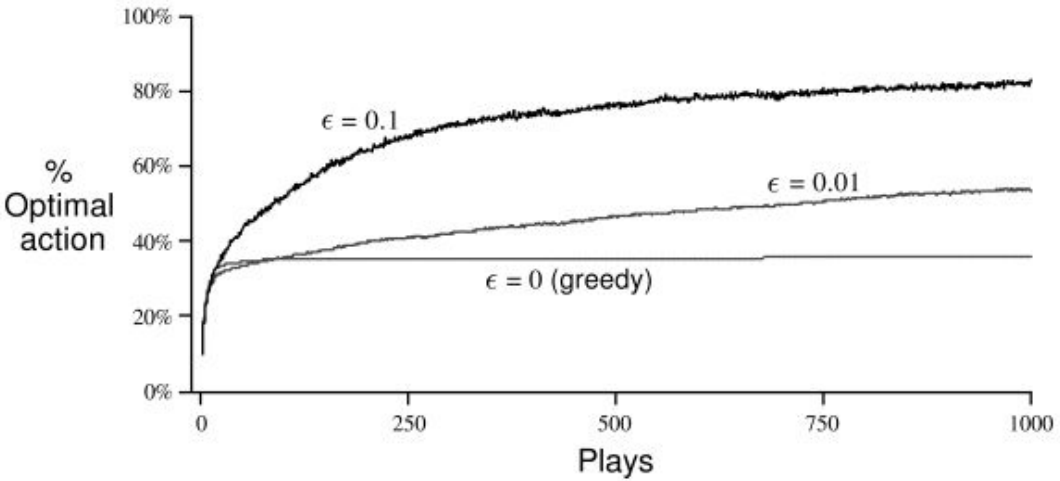
- Create 2000 different 10-armed bandit tasks.
- For each task, select the optimal reward distributions $Q^*(a)$ from $N(0,1)$.
  - For each task, do 1000 plays (actions).
  - For each action a, select the reward from $N(Q^*(a), 1)$.
- Plot averages over the 2000 tasks.

Key Issues in K-armed Bandit Problems

- How to estimate the distribution of reward for each action: 1, …, K?
- How to choose an action (given an estimate of rewards for each action)?
  - Exploration
  - Exploitation

Action-Value method: ϵ-Greedy

- With $p = 1 - \epsilon$
  - Select the greedy action (exploit).
- With $p = \epsilon$
  - Select uniformly across all actions (explore).



What should the estimate be?

- Compute estimate $Q_t(a)$ as the mean reward when action a was performed.

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

- This can be computed incrementally.

$$Q_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = Q_k + \frac{1}{k+1}[r_{k+1} - Q_k]$$

- More generally, the predictor-corrector form:

$$Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k] \text{ where } 0 < \alpha \leq 1$$

Recency-weighted averaging

- Suppose we want new rewards to have the most impact, not the oldest rewards.
  - E.g. when the reward distribution changes over time.

cf: uniform avg. (previous slide)

$$Q_k + \frac{1}{k+1}[r_{k+1} - Q_k]$$

- With *constant* weight, the recurrence

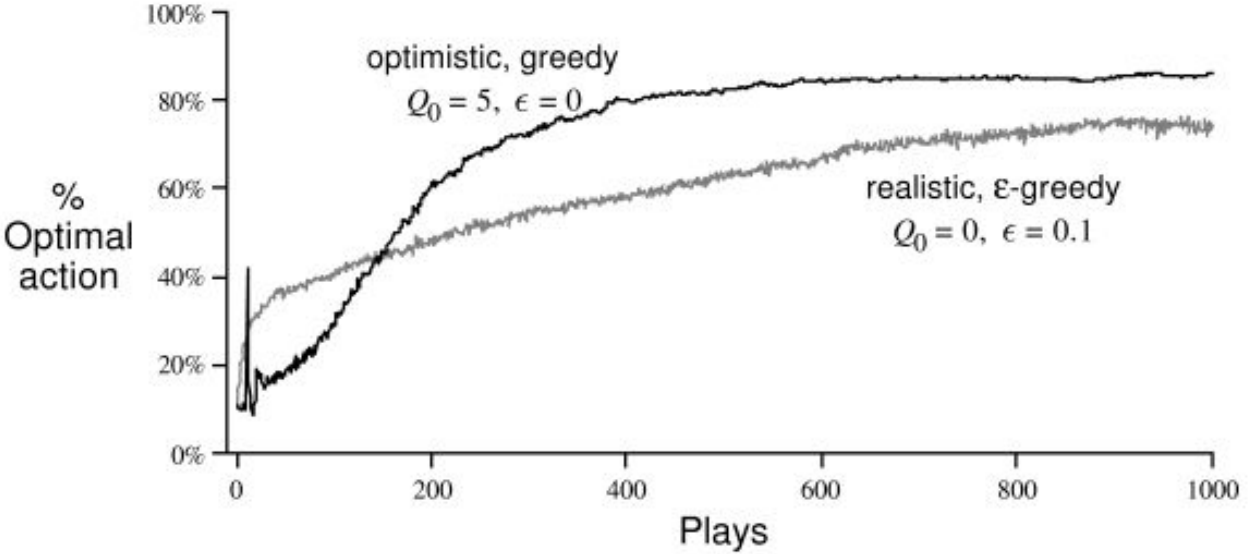
$$Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k] \text{ where } 0 < \alpha \leq 1$$

  - means that past rewards have exponentially decreasing impact on the estimate $Q_t(a)$.

- In this case, the discount rate is $1 - \alpha$

Encouraging Exploration 1

- Optimistic initialization: give every action a an initial high default value, e.g., $Q_0(a) = +5$.
- Now, *greedy* action selection will explore!



Bandit Problems and RL

- In bandit problem, there is only “one state.”
  - Action does not change state. We only need to optimize according to the performance (cumulative rewards) of its actions.
  - However, in other RL problems, the states change too.

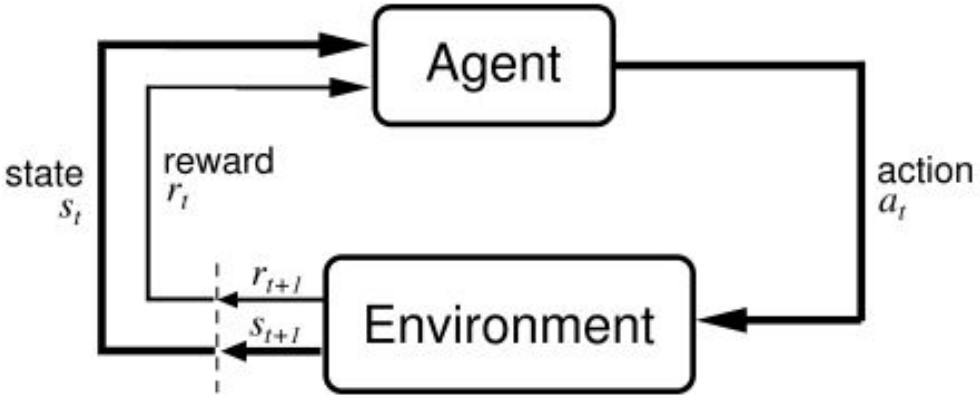
- Now we will see more general RL framework, called Markov Decision Process (MDP).

Outline

- Introduction
- Multi-arm Bandit Problem
- **Markov Decision Process**
- **Value Iteration**

Reviewing the RL Framework

- At each time t = 0, 1, 2, 3, . . .
- The agent perceives a *state* $s_t \in \mathcal{S}$
- It selects an *action* $a_t \in \mathcal{A}(s_t)$
- Then it receives a *reward* $r_{t+1} \in \Re$
- and finds itself in a new state s_{t+1}



Markov Decision Process

- States S
  - E.g., Location of robot
- Actions A
  - E.g., robot's motor commands, etc.
- State transition probabilities:

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

- Expected rewards:

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} = r' | s_t = s, a_t = a, s_{t+1} = s'\}$$

- The goal is to find a "policy" (stochastic mapping from a state to an action) that maximizes the expected cumulative future rewards

State-Value Functions

- A policy $\pi(s, a)$ specifies the probability of selecting action *a* when in state *s*.
- The *value* of a state is the expected future return, starting in *s* and following the policy.

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\}$$

$$= E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

Action-Value Functions

- We describe the value of taking an action a, starting in state s, and following the policy thereafter.

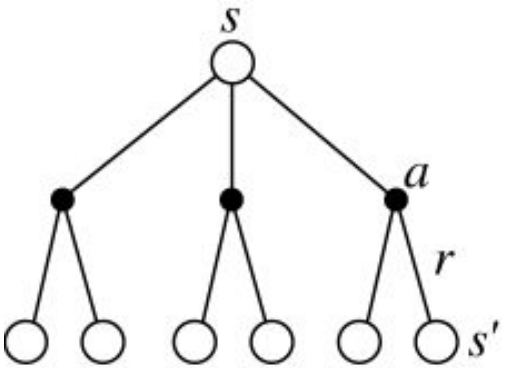
$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\}$$

$$= E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

The Bellman Equation for *V*

- Expresses the value function at a state as a relationship with its immediate successors.

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')]$$



Optimal Value Functions

- There are optimal value functions.
  - Optimal state-value functions:

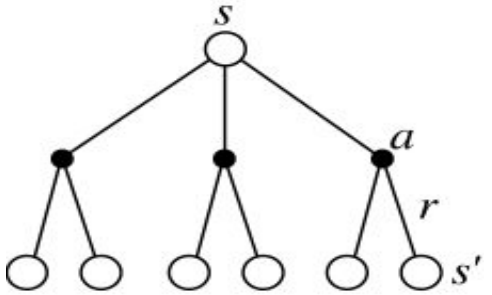
$$V^*(s) = \max_\pi V^\pi(s)$$

  - Optimal action-value functions:

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

- We will use these to find optimal policies.

Learning the Value Function



- We know that $V^\pi(s)$ satisfies:

$$V^\pi(s) = E_\pi\{R_t | s_t = s\}$$

$$= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

$$= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right\}$$

$$= E_\pi\left\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s\right\}$$

Dynamic Programming

- Idea: we can use the Bellman equation:

$$V^{\pi}(s) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right\}$$

- as an update rule:

$$V_{k+1}(s) = E_{\pi}\left\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s\right\}$$

$$= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')]$$

$$V_0 \rightarrow V_1 \rightarrow \ldots \rightarrow V_k \rightarrow V_{k+1} \rightarrow \ldots \rightarrow V^{\pi}$$

- The fixed point is $V_k = V^{\pi}$

Value Iteration

- From Bellman equation:

$$V_{k+1}(s) = E_\pi \{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s\}$$

- The optimal policy should satisfy the following condition: $V^*(s) = \max_{a=\pi(s)} E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s\}$

$$= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$

- Therefore, we use the following updates

$$V_{k+1}(s) = \max_a E\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a\}$$

$$= \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')]$$

- This is called value iteration

Value Iteration

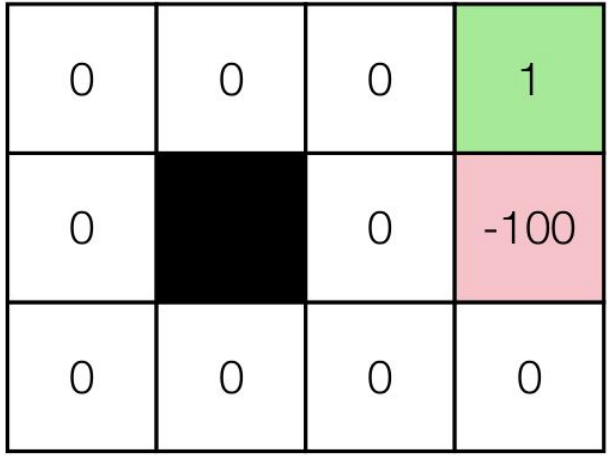
Initialize V arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
$\Delta \leftarrow 0$
For each $s \in \mathcal{S}$:
$v \leftarrow V(s)$
$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Illustration of value iteration

Running value iteration with $\gamma = 0.9$



Original reward function

Noisy State Transition Dynamics: e.g.,

$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

Rewards:

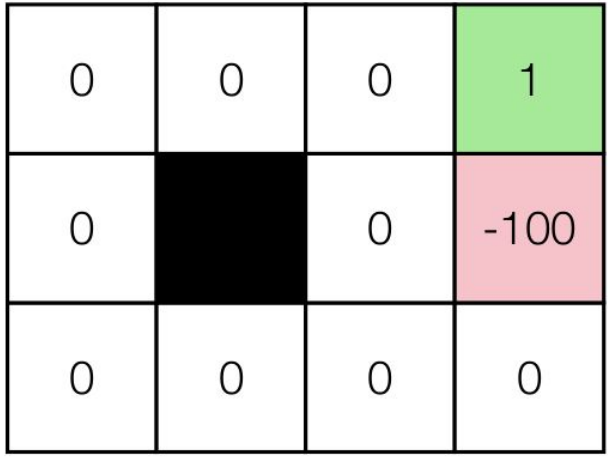
- just a function of s: i.e. $R^a_{ss'} = R(s)$

$$\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')] = R(s) + \gamma \sum_{s'} P^a_{ss'} V(s')$$

Slide credit: J. Zico Kolter

Illustration of value iteration

Running value iteration with $\gamma = 0.9$



At iteration 1

Noisy State Transition Dynamics: e.g.,

$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{V}(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a)\, \hat{V}(s'), \quad \forall s \in \mathcal{S}$$

Slide credit: J. Zico Kolter

Illustration of value iteration

Running value iteration with $\gamma = 0.9$

0	0	0.72	1.81
0		0	-99.91
0	0	0	0

At iteration 2

Noisy State Transition Dynamics: e.g.,

$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{V}(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a)\, \hat{V}(s'), \quad \forall s \in \mathcal{S}$$

Slide credit: J. Zico Kolter

Illustration of value iteration

Running value iteration with $\gamma = 0.9$

0.809	1.598	2.475	3.745
0.268		0.302	-99.59
0	0.034	0.122	0.004

At iteration 6

Noisy State Transition Dynamics: e.g.,

$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{V}(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a)\, \hat{V}(s'), \quad \forall s \in \mathcal{S}$$

Slide credit: J. Zico Kolter

Illustration of value iteration

Running value iteration with $\gamma = 0.9$

2.686	3.527	4.402	5.812
2.021		1.095	-98.82
1.390	0.903	0.738	0.123

At iteration 11

Noisy State Transition Dynamics: e.g.,

$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{V}(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a)\, \hat{V}(s'), \quad \forall s \in \mathcal{S}$$

Slide credit: J. Zico Kolter

Illustration of value iteration

Running value iteration with $\gamma = 0.9$

5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

At iteration 1000

Noisy State Transition Dynamics: e.g.,

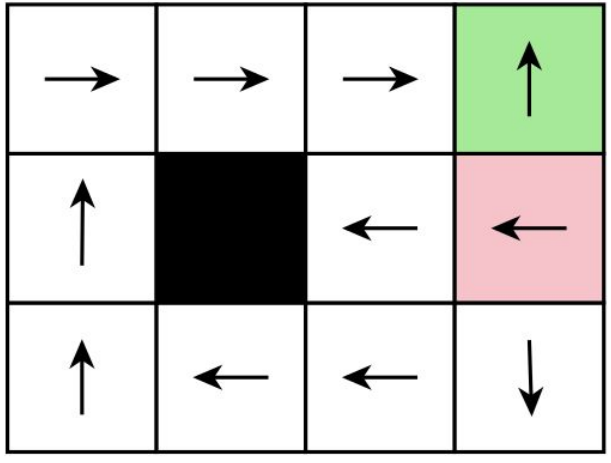
$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{V}(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a)\, \hat{V}(s'), \quad \forall s \in \mathcal{S}$$

Slide credit: J. Zico Kolter

Illustration of value iteration

Running value iteration with $\gamma = 0.9$



Resulting policy after 1000 iterations

Noisy State Transition Dynamics: e.g.,

$$P(s'|s, a = \text{“up”}) = \begin{cases} 0.8 & s' \text{ is above } s \\ 0.1 & s' \text{ is left of } s \\ 0.1 & s' \text{ is right of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{V}(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a)\, \hat{V}(s'), \quad \forall s \in \mathcal{S}$$

Slide credit: J. Zico Kolter

Convergence of Value Iteration

Theorem: Value iteration converges to optimal value: $\hat{V} \rightarrow V^{\star}$

Proof: For any estimate of the value function $\hat{V}$, we define the Bellman backup operator $B : \mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^{|\mathcal{S}|}$

$$B\,\hat{V}(s) = R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a)\,\hat{V}(s')$$

We will show that Bellman operator is a *contraction*, that for any value function estimates $V_1,\ V_2$

$$\max_{s \in \mathcal{S}} |BV_1(s) - BV_2(s)| \leq \gamma \max_{s \in \mathcal{S}} |V_1(s) - V_2(s)|$$

Since $BV^{\star} = V^{\star}$ (definition of Bellman equation; the contraction property also implies existence and uniqueness of this fixed point), we have:

$$\max_{s \in \mathcal{S}} \left| B\,\hat{V}(s) - V^{\star}(s) \right| \leq \gamma \max_{s \in \mathcal{S}} \left| \hat{V}(s) - V^{\star}(s) \right| \implies \hat{V} \rightarrow V^{\star}$$

Slide credit: J. Zico Kolter

Proof of the contraction property (details)

$$
\begin{aligned}
&|BV_1(s) - BV_2(s)| \\
&\quad = \gamma \left| \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a) V_1(s') - \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a) V_2(s') \right| \\
&\quad \leq \gamma \max_{a \in \mathcal{A}} \left| \sum_{s' \in \mathcal{S}} P(s'|s,a) V_1(s') - \sum_{s' \in \mathcal{S}} P(s'|s,a) V_2(s') \right| \\
&\quad \leq \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a) \left| V_1(s') - V_2(s') \right| \\
&\quad \leq \gamma \max_{s \in \mathcal{S}} |V_1(s) - V_2(s)|
\end{aligned}
$$

where third line follows from property that

$$
\left| \max_x f(x) - \max_x g(x) \right| \leq \max_x |f(x) - g(x)|
$$

and final line because $P(s'|s,a)$ are non-negative and sum to one

Slide credit: J. Zico Kolter

Value Iteration Convergence

How many iterations will it take to find the optimal policy?

Assume rewards are in $[0, R_{\max}]$, then

$$V^\star(s) \leq \sum_{t=1}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma}$$

Then letting V^k be the value after the k-th iteration

$$\max_{s \in \mathcal{S}} |\, V^k(s) - \, V^\star(s)| \leq \frac{\gamma^k R_{\max}}{1-\gamma}$$

i.e., we have a linear convergence towards the optimal value function

But, the time to find the optimal policy depends on the separation between the value of the optimal and second suboptimal policy, and is difficult to bound

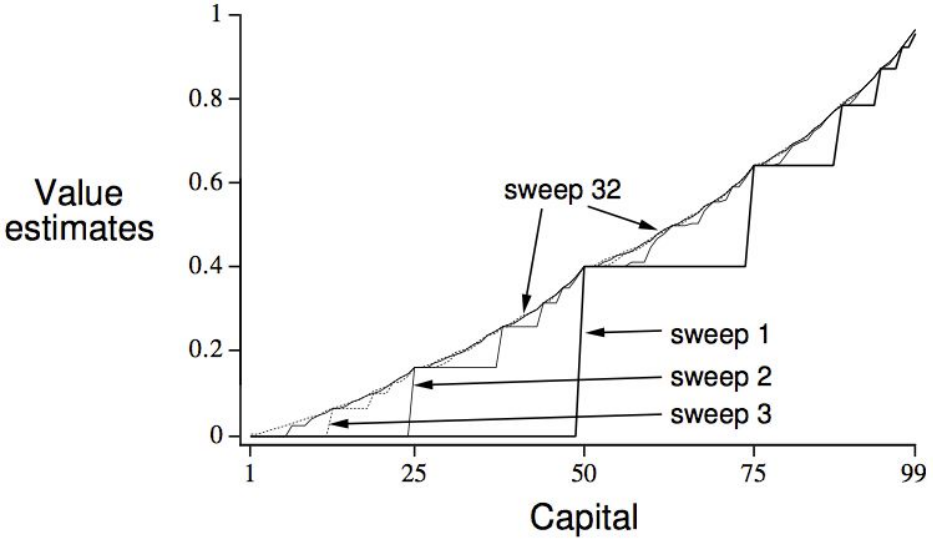
Slide credit: J. Zico Kolter

Example: Gambling

- A gambler makes bets on the outcomes of a sequence of coin flips.
  - If the coin comes up heads, he wins as many dollars as he has staked on that flip;
    - p: the probability of the coin resulting in heads.
  - If it is tails, he loses his stake.
- On each flip, the gambler decides stakes out of his capital.
- The game ends when the gambler wins by reaching his goal of $100, or loses by running out of money.

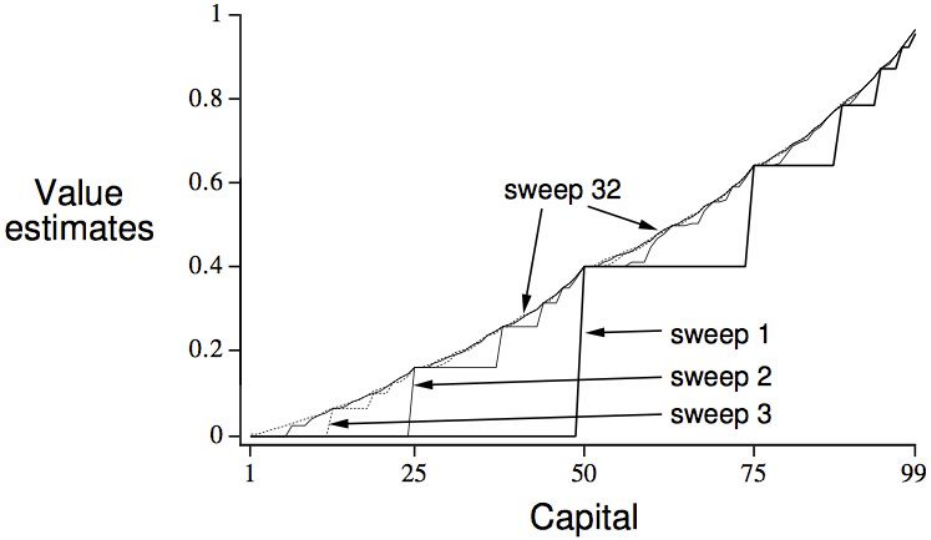
- MDP
  - State: gambler's capital $s \in \{1, ..., 99\}$
  - Action: stakes $a \in \{0, 1, ..., \min(s, 100 - s)\}$
  - Reward is zero on all transitions except reaching the goal of $100 (which gives the reward of +1)
  - A policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal

Gambling

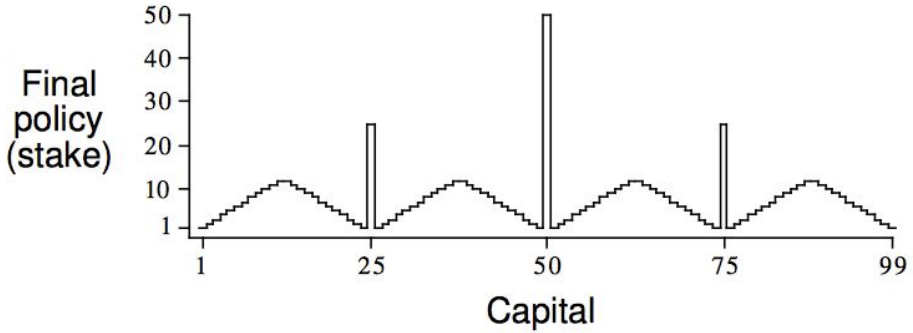


P=0.4

Gambling



P=0.4



RL Demos

- https://waxworksmath.com/Authors/N_Z/Sutton/RLAI_1st_Edition/sutton.html

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

Quiz

https://forms.gle/LFHL2cFGvCKuQmbPA