Imperial College London

TECHNISCHE UNIVERSITÄT DARMSTADT

Tutorial on Reinforcement Learning

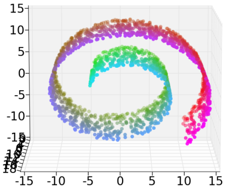
Marc Deisenroth
Department of Computing
Imperial College London

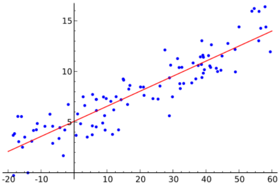
Department of Computer Science
TU Darmstadt

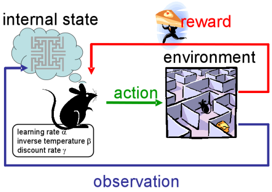
m.deisenroth@imperial.ac.uk

Machine Learning Summer School on Big Data
Hammamet, September 17, 2013

Machine Learning

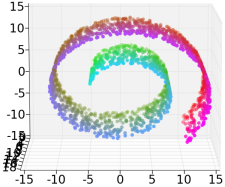


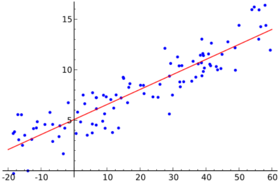


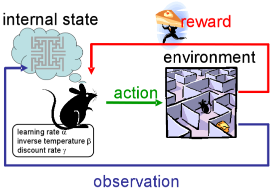


- Unsupervised learning
- Supervised learning
- Reinforcement learning

Machine Learning



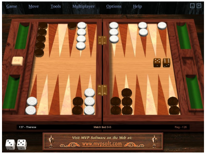


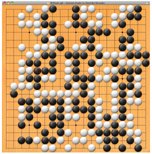


- Unsupervised learning
- Supervised learning
- Reinforcement learning

RL makes decisions!

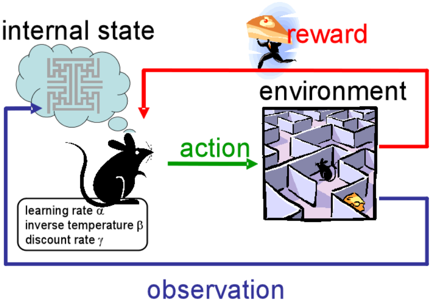
RL Success Stories





- Games (e.g., G. Tesauro, D. Silver)
- Operations research and scheduling (e.g., W. Powell, P. Tadepalli)
- Recently: robotics (e.g., P. Abbeel, J. Peters, P. Stone, M. Riedmiller)

Motivation



- Learning system in an unknown environment
- Knowledge only through interacting with environment
- Explores the environment and receives rewards
- Find strategy/policy, which maximizes overall reward

  ⏩ Optimal behavior

Bayesian Decision Theory

- Make optimal decisions $\boldsymbol{a}^*$ by maximizing an expected utility

$$\boldsymbol{a}^* \in \arg\max_{\boldsymbol{a}} \mathbb{E}[r(\boldsymbol{a})] = \arg\max_{\boldsymbol{a}} \sum_{j=1}^{m} r(\boldsymbol{s}_j, \boldsymbol{a}) p(\boldsymbol{s}_j)$$

$\boldsymbol{a}$: decision

$\boldsymbol{s}$: information about environment/state

Bayesian Decision Theory

- Make optimal decisions $\boldsymbol{a}^*$ by maximizing an expected utility

$$\boldsymbol{a}^* \in \arg\max_{\boldsymbol{a}} \mathbb{E}[r(\boldsymbol{a})] = \arg\max_{\boldsymbol{a}} \sum_{j=1}^{m} r(\boldsymbol{s}_j, \boldsymbol{a}) p(\boldsymbol{s}_j)$$

$\boldsymbol{a}$: decision

$\boldsymbol{s}$: information about environment/state

- Bayesian sequential decision theory (statistics)
- Optimal control theory (engineering)
- Reinforcement learning (computer science, psychology)

Example: Winning the Lottery

Actions	Outcomes
a_1: play	s_1: Win the lottery
a_2: don't play	s_2: Don't win the lottery

Example: Winning the Lottery

Actions	Outcomes
a_1: play	s_1: Win the lottery
a_2: don't play	s_2: Don't win the lottery

Optimal action

$$a^* = \arg\max_{a_i} \sum_{j=1}^{2} r_{ij} p(s_j | a_i)$$

Example: Winning the Lottery

Actions	Outcomes
a_1: play	s_1: Win the lottery
a_2: don't play	s_2: Don't win the lottery

Optimal action

$$a^* = \arg\max_{a_i} \sum_{j=1}^{2} r_{ij} p(s_j | a_i)$$

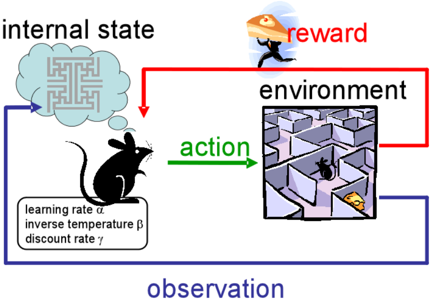
$$\begin{aligned} p(s_1|a_1) &= 10^{-7} & r_{11} &= 500{,}000 \text{ USD} \\ p(s_2|a_1) &= 1 - 10^{-7} & r_{12} &= -1 \text{ USD} \\ p(s_1|a_2) &= 0 & r_{21} &= 0 \text{ USD} \\ p(s_2|a_2) &= 1 & r_{22} &= 0 \text{ USD} \end{aligned}$$

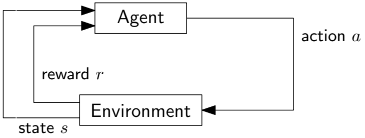
▶▶ What is the optimal action for this decision problem?

From Bayesian Decision Theory to RL

- So far: single decisions. How do we make a sequence of decisions in order to achieve some long-term rewards?
- What about state-dependent actions $\boldsymbol{a}_i|\boldsymbol{s}_j$?

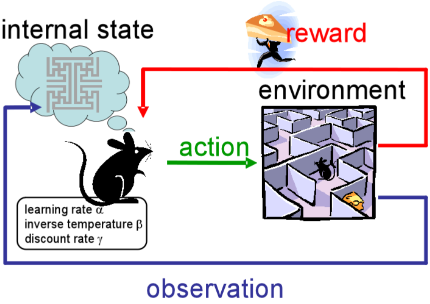
RL Set-up





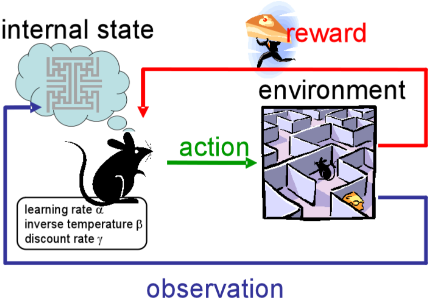
- Agent interacts with environment to gain knowledge
- Explores and receives rewards
- Actions change the state of the environment
- Choose actions to maximize long-term reward

RL Set-up



- Agent interacts with environment to gain knowledge
- Explores and receives rewards
- Actions change the state of the environment
- Choose actions to maximize long-term reward

RL Set-up

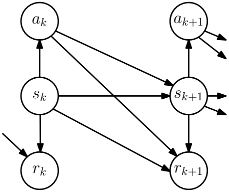


- Agent interacts with environment to gain knowledge
- Explores and receives rewards
- Actions change the state of the environment
- Choose actions to maximize long-term reward

▶▶ Markov Decision Process

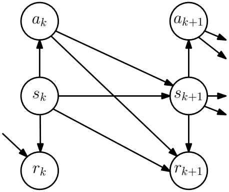
Markov Decision Process: Definition

- $\mathcal{S}$: State space (finite)
- $\mathcal{A}$: Action space (finite)
- $\mathcal{P}$: Transition probability $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$
- r: Reward function
- $\gamma \in [0, 1)$: Discount factor
- π: Policy
  - Deterministic: $\boldsymbol{a} = \pi(\boldsymbol{s})$
  - Stochastic: $\boldsymbol{a} \sim p_\pi(\boldsymbol{a}|\boldsymbol{s})$ alternative notation: $p_\pi(\boldsymbol{a}|\boldsymbol{s}) = \pi(\boldsymbol{a}|\boldsymbol{s})$



Markov Decision Process: Definition

- $\mathcal{S}$: State space (finite)
- $\mathcal{A}$: Action space (finite)
- $\mathcal{P}$: Transition probability $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$
- r: Reward function
- $\gamma \in [0, 1)$: Discount factor
- π: Policy
  - Deterministic: $\boldsymbol{a} = \pi(\boldsymbol{s})$
  - Stochastic: $\boldsymbol{a} \sim p_\pi(\boldsymbol{a}|\boldsymbol{s})$ alternative notation: $p_\pi(\boldsymbol{a}|\boldsymbol{s}) = \pi(\boldsymbol{a}|\boldsymbol{s})$



Objective

Find a policy π^* that maximizes the expected long-term reward

$$V^\pi(\boldsymbol{s}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} \middle| \boldsymbol{s}_0 = \boldsymbol{s}, \pi\right], \qquad r_{k+1} = r_{k+1}(\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{s}_{k+1})$$

Categorization of RL Algorithms

- Value function methods
  - ▶▶ Use structure of a value function to discover optimal policies
- Value function-free methods (e.g., policy search)
  - ▶▶ Search in policy space directly

Value Functions

- State Value Function: How good is it to be in a particular state $\boldsymbol{s}$? Well, this depends on the current policy:

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}\big] = \mathbb{E}\big[\textstyle\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$

$$= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \qquad \text{Self-consistency}$$

Value Functions

- State Value Function: How good is it to be in a particular state $\boldsymbol{s}$? Well, this depends on the current policy:

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}\big] = \mathbb{E}\big[\sum\nolimits_{k=0}^{\infty}\gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$
$$= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \qquad \text{Self-consistency}$$

- State-Action Value Function: How good is it to be in a particular state $\boldsymbol{s}$ and apply action $\boldsymbol{a}$, and afterwards follow policy π?

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big]$$
$$= \mathbb{E}\big[\sum\nolimits_{k=0}^{\infty}\gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big]$$

Value Functions

- State Value Function: How good is it to be in a particular state $\boldsymbol{s}$? Well, this depends on the current policy:

$$V^\pi(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}\big] = \mathbb{E}\big[\sum_{k=0}^{\infty}\gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$
$$= \mathbb{E}[r_1 + \gamma V^\pi(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \qquad \text{Self-consistency}$$

- State-Action Value Function: How good is it to be in a particular state $\boldsymbol{s}$ and apply action $\boldsymbol{a}$, and afterwards follow policy π?

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big]$$
$$= \mathbb{E}\big[\sum_{k=0}^{\infty}\gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big]$$
$$= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma\sum_{k=0}^{\infty}\gamma^k r_{k+2}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big]$$
$$=$$

Value Functions

- State Value Function: How good is it to be in a particular state $\boldsymbol{s}$? Well, this depends on the current policy:

$$V^\pi(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}\big] = \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$

$$= \mathbb{E}[r_1 + \gamma V^\pi(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \qquad \text{Self-consistency}$$

- State-Action Value Function: How good is it to be in a particular state $\boldsymbol{s}$ and apply action $\boldsymbol{a}$, and afterwards follow policy π?

$$\begin{aligned} Q^\pi(\boldsymbol{s}, \boldsymbol{a}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \\ &= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \\ &= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \\ &= \end{aligned}$$

Value Functions

- State Value Function: How good is it to be in a particular state $\boldsymbol{s}$? Well, this depends on the current policy:

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}\big] = \mathbb{E}\big[\textstyle\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$
$$= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \qquad \text{Self-consistency}$$

- State-Action Value Function: How good is it to be in a particular state $\boldsymbol{s}$ and apply action $\boldsymbol{a}$, and afterwards follow policy π?

$$\begin{aligned} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \\ &= \mathbb{E}\big[\textstyle\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \\ &= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma \textstyle\sum_{k=0}^{\infty} \gamma^k r_{k+2}|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \\ &= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi\big] \end{aligned}$$

Bellman Operator

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \\
&= T^{\pi}[V^{\pi}](\boldsymbol{s}) \\
T^{\pi}[V^{\pi}](\boldsymbol{s}) &:= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]
\end{aligned}
$$

Bellman Operator

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \\
&= T^{\pi}[V^{\pi}](\boldsymbol{s}) \\
T^{\pi}[V^{\pi}](\boldsymbol{s}) &:= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]
\end{aligned}
$$

- $V^{\pi} = T^{\pi}[V^{\pi}]$, $\qquad T^{\pi}$: Bellman operator (linear affine)

Bellman Operator

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \\
&= T^{\pi}[V^{\pi}](\boldsymbol{s}) \\
T^{\pi}[V^{\pi}](\boldsymbol{s}) &:= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]
\end{aligned}
$$

- $V^{\pi} = T^{\pi}[V^{\pi}]$, $\quad T^{\pi}$: Bellman operator (linear affine)
  - Fixed point equation with a unique solution for V^{π} (Banach's FP theorem)

Bellman Operator

$$
\begin{aligned}
V^\pi(\boldsymbol{s}) &= \mathbb{E}[r_1 + \gamma V^\pi(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi] \\
&= T^\pi[V^\pi](\boldsymbol{s}) \\
T^\pi[V^\pi](\boldsymbol{s}) &:= \mathbb{E}\big[r_1 + \gamma V^\pi(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]
\end{aligned}
$$

- $V^\pi = T^\pi[V^\pi]$, $\quad T^\pi$: Bellman operator (linear affine)
  - Fixed point equation with a unique solution for V^π (Banach's FP theorem)

$$
\begin{aligned}
T^\pi[V^\pi] &= r^\pi + \gamma \mathcal{P}^\pi V^\pi \\
\implies V^\pi &= r^\pi + \gamma \mathcal{P}^\pi V^\pi
\end{aligned}
$$

for suitable representations $r^\pi, \mathcal{P}^\pi, V^\pi$

Optimal Policies and Value Functions

- Optimal policy π^* ensures that $V^{\pi^*}(\boldsymbol{s}) \geqslant V^{\pi}(\boldsymbol{s}) \quad \forall \boldsymbol{s} \in \mathcal{S}, \pi$
- Existence of π^*? Uniqueness?

Optimal Policies and Value Functions

- Optimal policy π^* ensures that $V^{\pi^*}(\boldsymbol{s}) \geqslant V^{\pi}(\boldsymbol{s}) \quad \forall \boldsymbol{s} \in \mathcal{S}, \pi$
- Existence of π^*? Uniqueness?
- Optimal state value function:
  $\forall s \in \mathcal{S} : V^*(\boldsymbol{s}) = V^{\pi^*}(\boldsymbol{s}) = \max_{\pi} V^{\pi}(\boldsymbol{s})$

Optimal Policies and Value Functions

- Optimal policy π^* ensures that $V^{\pi^*}(\boldsymbol{s}) \geqslant V^{\pi}(\boldsymbol{s}) \quad \forall \boldsymbol{s} \in \mathcal{S}, \pi$
- Existence of π^*? Uniqueness?
- Optimal state value function:
  $\forall s \in \mathcal{S} : V^*(\boldsymbol{s}) = V^{\pi^*}(\boldsymbol{s}) = \max_\pi V^\pi(\boldsymbol{s})$
- Optimal state-action value function:
  $\forall s \in \mathcal{S} : Q^*(\boldsymbol{s}, \boldsymbol{a}) = \max_\pi Q^\pi(\boldsymbol{s}, \boldsymbol{a})$

  ▶▶ Expected return of choosing action $\boldsymbol{a}$ in state $\boldsymbol{s}$ and afterwards following the optimal policy π^*. Note that

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[r_{t+1} + \gamma V^*(\boldsymbol{s}_{t+1}) | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}]$$

Bellman Optimality Equations

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a})$$

Bellman Optimality Equations

$$
\begin{aligned}
V^*(\boldsymbol{s}) &= \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big]
\end{aligned}
$$

Bellman Optimality Equations

$$
\begin{aligned}
V^*(\boldsymbol{s}) &= \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big]
\end{aligned}
$$

Bellman Optimality Equations

$$
\begin{aligned}
V^*(\boldsymbol{s}) &= \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^* \Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^* \Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\big[r_1 + \gamma V^*(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^* \big]
\end{aligned}
$$

Bellman Optimality Equations

$$
\begin{aligned}
V^*(\boldsymbol{s}) &= \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^* \Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^* \Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\big[r_1 + \gamma V^*(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^* \big]
\end{aligned}
$$

- $V^* = T^*[V^*]$ $\quad$ T^* : Bellman optimality operator (nonlinear)

Bellman Optimality Equations

$$
\begin{aligned}
V^*(\boldsymbol{s}) &= \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\big[r_1 + \gamma V^*(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\big]
\end{aligned}
$$

▶▶ $V^* = T^*[V^*]$ T^* : Bellman optimality operator (nonlinear)

$$
\begin{aligned}
Q^*(\boldsymbol{s}, \boldsymbol{a}) &= \mathbb{E}[r_1 + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}_1, \boldsymbol{a}') | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}] \\
&=
\end{aligned}
$$

Bellman Optimality Equations

$$
\begin{aligned}
V^*(\boldsymbol{s}) &= \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}) \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\Big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\Big] \\
&= \max_{\boldsymbol{a}} \mathbb{E}\big[r_1 + \gamma V^*(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \pi^*\big]
\end{aligned}
$$

▸▸ $V^* = T^*[V^*]$ T^* : Bellman optimality operator (nonlinear)

$$
\begin{aligned}
Q^*(\boldsymbol{s}, \boldsymbol{a}) &= \mathbb{E}[r_1 + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}_1, \boldsymbol{a}') | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}] \\
&= \mathbb{E}[r_1 + \gamma V^*(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}]
\end{aligned}
$$

Solving MDPs

- Assume you know V^*
- **One-step search**: be "greedy" with respect to V^*

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$$

Solving MDPs

- Assume you know V^*
- **One-step search**: be "greedy" with respect to V^*

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$$

- Assume you know Q^*
- **Zero-step search**:

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

Solving MDPs

- Assume you know V^*
- **One-step search**: be "greedy" with respect to V^*

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$$

- Assume you know Q^*
- **Zero-step search**:

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

Assumptions for solving the Bellman equations exactly:

- Know the transition probabilities $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$
- Sufficient computational resources available
- Markov property holds

Solving MDPs

- Assume you know V^*
- **One-step search**: be "greedy" with respect to V^*

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$$

- Assume you know Q^*
- **Zero-step search**:

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

Assumptions for solving the Bellman equations exactly:

- Know the transition probabilities $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$
- Sufficient computational resources available
- Markov property holds

▶▶ Approximate solutions in practice

Solving MDPs (2)

- Exact: Dynamic programming
- Approximate: Monte Carlo, Temporal Difference Learning

Dynamic Programming

Assumptions:

- Perfect model $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ is known
- Typically finite state spaces $\mathcal{S}$ and action spaces $\mathcal{A}$
- Expected immediate rewards $\mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')]$ are known

▶▶ Use value functions to structure the search for good policies

Policy Evaluation

Objective

For a given policy π, find the corresponding value function V^π

- Exploit the fixed-point property of the value function $V^\pi = T^\pi[V^\pi]$:
  - Initialize V_0^π arbitrarily
  - Find V^π as the limit of the sequence $V_0^\pi, V_1^\pi, \ldots$

Policy Evaluation

Objective

For a given policy π, find the corresponding value function V^π

- Exploit the fixed-point property of the value function $V^\pi = T^\pi[V^\pi]$:
  - Initialize V_0^π arbitrarily
  - Find V^π as the limit of the sequence $V_0^\pi, V_1^\pi, \dots$

Update Rule

$$\forall \boldsymbol{s} \in \mathcal{S}: \; V_{k+1}^\pi(\boldsymbol{s}) \leftarrow \mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V_k^\pi(\boldsymbol{s}')|\boldsymbol{s}, \pi]$$

$$\boldsymbol{a} \sim p_\pi(\boldsymbol{a}|\boldsymbol{s}), \quad \boldsymbol{s}' \sim p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$$

▸▸ Bootstrapping

Policy Improvement

- So far: We know V^{π}, but we want V^*
- Find a better policy π'

Objective

Find a policy $\pi' \geqslant \pi$, i.e., $V^{\pi'} \geqslant V^{\pi}$

Policy Improvement Theorem

Policy Improvement Theorem

If π, π' are two (deterministic) policies with

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) = V^{\pi}(\boldsymbol{s})$$

then $\pi' \geqslant \pi$ and $V^{\pi'} \geqslant V^{\pi}$, i.e., π' improves π.

Policy Improvement Theorem

Policy Improvement Theorem

If π, π' are two (deterministic) policies with

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) = V^{\pi}(\boldsymbol{s})$$

then $\pi' \geqslant \pi$ and $V^{\pi'} \geqslant V^{\pi}$, i.e., π' improves π.

- For stochastic policies:

$$\begin{aligned} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) &= \mathbb{E}_{\boldsymbol{a}}[Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})] \qquad \qquad \boldsymbol{a} \sim p_{\pi'}(\boldsymbol{a}|\boldsymbol{s}) \\ &= \sum_{\boldsymbol{a}} p_{\pi'}(\boldsymbol{a}|\boldsymbol{s}) Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) \end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$V^{\pi}(\boldsymbol{s}) \overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s}))$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\ &= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S} : \quad Q^\pi(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^\pi(\boldsymbol{s}) = Q^\pi(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned} V^\pi(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^\pi(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\ &= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^\pi(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \end{aligned}$$

Proof

$$\forall s \in \mathcal{S}: \quad Q^\pi(s, \pi'(s)) \geqslant V^\pi(s) = Q^\pi(s, \pi(s)) \qquad (*)$$

$$
\begin{aligned}
V^\pi(s) &\overset{(*)}{\leqslant} Q^\pi(s, \pi'(s)) \\
&= \mathbb{E}\big[r_1(s_0, a_0, s_1) + \gamma V^\pi(s_1) | s_0 = s, a_0 = \pi'(s_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^\pi(s_1, \pi'(s_1)) | s_0 = s, a_0 = \pi'(s_0), a_1 = \pi'(s_1)\big]
\end{aligned}
$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\ &= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\ &\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\ &= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^{\pi}(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^{\pi}(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^{\pi}(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 Q^{\pi}(\boldsymbol{s}_2, \pi'(\boldsymbol{s}_2)) | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^{\pi}(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 Q^{\pi}(\boldsymbol{s}_2, \pi'(\boldsymbol{s}_2)) | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^\pi(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^\pi(\boldsymbol{s}) = Q^\pi(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^\pi(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^\pi(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^\pi(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^\pi(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^\pi(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^\pi(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 Q^\pi(\boldsymbol{s}_2, \pi'(\boldsymbol{s}_2)) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\pi(\boldsymbol{s}_3) | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^{\pi}(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 Q^{\pi}(\boldsymbol{s}_2, \pi'(\boldsymbol{s}_2)) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^{\pi}(\boldsymbol{s}_3) | \boldsymbol{s}_0 = \boldsymbol{s}\big]
\end{aligned}$$

Proof

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \geqslant V^{\pi}(\boldsymbol{s}) = Q^{\pi}(\boldsymbol{s}, \pi(\boldsymbol{s})) \qquad (*)$$

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &\overset{(*)}{\leqslant} Q^{\pi}(\boldsymbol{s}, \pi'(\boldsymbol{s})) \\
&= \mathbb{E}\big[r_1(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1) + \gamma V^{\pi}(\boldsymbol{s}_1) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0)\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma Q^{\pi}(\boldsymbol{s}_1, \pi'(\boldsymbol{s}_1)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \pi'(\boldsymbol{s}_0), \boldsymbol{a}_1 = \pi'(\boldsymbol{s}_1)\big] \\
&= \mathbb{E}\big[r_1 + \gamma \mathbb{E}[r_2 + \gamma V^{\pi}(\boldsymbol{s}_2)] | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 V^{\pi}(\boldsymbol{s}_2) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&\overset{(*)}{\leqslant} \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 Q^{\pi}(\boldsymbol{s}_2, \pi'(\boldsymbol{s}_2)) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&= \mathbb{E}\big[r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^{\pi}(\boldsymbol{s}_3) | \boldsymbol{s}_0 = \boldsymbol{s}\big] \\
&\ldots \\
&\leqslant \mathbb{E}\big[\textstyle\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}\big] = V^{\pi'}(\boldsymbol{s})
\end{aligned}$$

Policy Improvement

- Easy to evaluate a change in the policy at a single state s

Policy Improvement

- Easy to evaluate a change in the policy at a single state $\boldsymbol{s}$
- Extend this idea to all states:

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad \pi'(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^{\pi}(\boldsymbol{s}')\big]$$
$$= \arg\max_{\boldsymbol{a}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$$

Policy Improvement

- Easy to evaluate a change in the policy at a single state $\boldsymbol{s}$
- Extend this idea to all states:

$$\forall \boldsymbol{s} \in \mathcal{S} : \quad \pi'(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^{\pi}(\boldsymbol{s}')\big]$$
$$= \arg\max_{\boldsymbol{a}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$$

- When $\pi' = \pi$ then $V^{\pi'} = V^{\pi}$ ⏩ Convergence (Why?)

Policy Improvement

- Easy to evaluate a change in the policy at a single state $\boldsymbol{s}$
- Extend this idea to all states:

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad \pi'(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^{\pi}(\boldsymbol{s}')\big]$$
$$= \arg\max_{\boldsymbol{a}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$$

- When $\pi' = \pi$ then $V^{\pi'} = V^{\pi}$ ⏩ Convergence (Why?)
- If V^{π} is known, we need a model $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ for the one-step look-ahead

Policy Improvement

- Easy to evaluate a change in the policy at a single state $\boldsymbol{s}$
- Extend this idea to all states:

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad \pi'(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^{\pi}(\boldsymbol{s}')\big]$$
$$= \arg\max_{\boldsymbol{a}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$$

- When $\pi' = \pi$ then $V^{\pi'} = V^{\pi}$ ▸▸ Convergence (Why?)
- If V^{π} is known, we need a model $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ for the one-step look-ahead
- If Q^{π} is known, we don't need a model (no prediction required)

Policy Improvement

- Easy to evaluate a change in the policy at a single state $\boldsymbol{s}$
- Extend this idea to all states:

$$\forall \boldsymbol{s} \in \mathcal{S}: \quad \pi'(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V^{\pi}(\boldsymbol{s}')\big]$$
$$= \arg\max_{\boldsymbol{a}} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$$

- When $\pi' = \pi$ then $V^{\pi'} = V^{\pi}$ ⏩ Convergence (Why?)
- If V^{π} is known, we need a model $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ for the one-step look-ahead
- If Q^{π} is known, we don't need a model (no prediction required)
- Greedy policy update with respect to the value function (but look implicitly at long-term rewards)

Policy Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} V^* \xrightarrow{I} \pi^*$$

- E: policy evaluation
- I: policy improvement

Policy Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} V^* \xrightarrow{I} \pi^*$$

- E: policy evaluation
- I: policy improvement
- Strict policy improvement at each step (unless policy is already optimal)
- Converges often after a few iterations

Policy Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} V^* \xrightarrow{I} \pi^*$$

- E: policy evaluation
- I: policy improvement
- Strict policy improvement at each step (unless policy is already optimal)
- Converges often after a few iterations
- Each policy evaluation is itself an iterative process
  ▸▸ Can be really slow!

Value Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{E} V^* \xrightarrow{I} \pi^*$$

- Stop policy evaluation after a single update

  ▶▶ No longer an iterative process

Value Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} V^* \xrightarrow{I} \pi^*$$

- Stop policy evaluation after a single update
  - No longer an iterative process

Update Rule

$$\begin{aligned} V_{k+1}(\boldsymbol{s}) &= \max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma V_k(\boldsymbol{s}')|\boldsymbol{s}, \boldsymbol{a}\big] \\ &= \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})\big(\mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')] + \gamma V_k(\boldsymbol{s}')\big) \end{aligned}$$

Value Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} V^* \xrightarrow{I} \pi^*$$

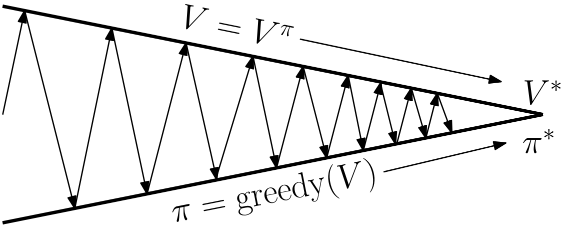
- Stop policy evaluation after a single update
  - No longer an iterative process

Update Rule

$$\begin{aligned} V_{k+1}(\boldsymbol{s}) &= \max_{\boldsymbol{a}} \mathbb{E}\big[r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V_k(\boldsymbol{s}')|\boldsymbol{s},\boldsymbol{a}\big] \\ &= \max_{\boldsymbol{a}} \sum\nolimits_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\big(\mathbb{E}[r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')] + \gamma V_k(\boldsymbol{s}')\big) \end{aligned}$$

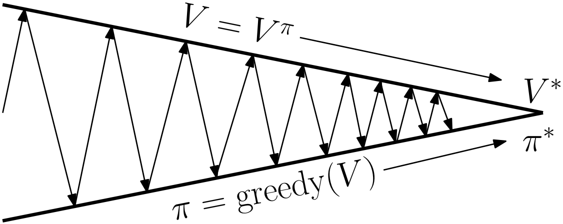
- Bootstrapping
- Bellman optimality equation as an update rule: $V_{k+1} \leftarrow T^*[V_k]$
- Identical to policy evaluation backup if you add the max operator

Generalized Policy Iteration (GPI)



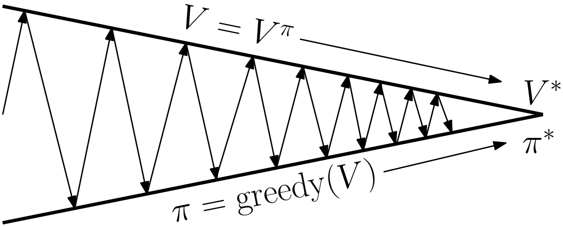
- Abstraction/generalization of policy iteration
- Two interacting processes: policy evaluation/improvement

Generalized Policy Iteration (GPI)



- Abstraction/generalization of policy iteration
- Two interacting processes: policy evaluation/improvement
- Update details abstracted away (policy needs to be greedy)
- Don't need to go full way to π or V^π, just "in the direction"

Generalized Policy Iteration (GPI)



- Abstraction/generalization of policy iteration
- Two interacting processes: policy evaluation/improvement
- Update details abstracted away (policy needs to be greedy)
- Don't need to go full way to π or V^π, just "in the direction"
- Both processes converge to a single joint solution (V^*, π^*)
- Value iteration (incomplete value function updates) is one example of GPI

Summary: Dynamic Programming

- Find optimal policies via value functions and bootstrapping
- Exact method (standard method in optimal control)

Summary: Dynamic Programming

- Find optimal policies via value functions and bootstrapping
- Exact method (standard method in optimal control)
- Computationally expensive (sweeps through state-action spaces)
  ▶▶ Curse of dimensionality

Summary: Dynamic Programming

- Find optimal policies via value functions and bootstrapping
- Exact method (standard method in optimal control)
- Computationally expensive (sweeps through state-action spaces)
  ⏩ Curse of dimensionality
- Exponentially faster than any direct policy search method (if the policy space is not restricted)
- Requires a model $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ and the knowledge of $\mathbb{E}[r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')]$ before DP can be applied.
- 2 algorithms: Policy iteration, value iteration

Approximate Value Function Methods

Look trajectory samples

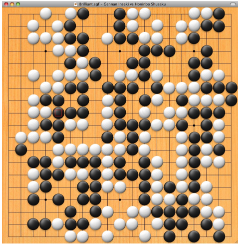
- Monte Carlo methods
- Temporal difference learning

Monte Carlo Methods

Key Idea

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \pi, \boldsymbol{s}_0 = \boldsymbol{s}\Big]$$

Estimate value function by averaging returns of sampled trajectories

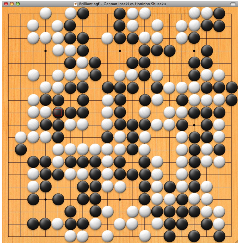


Monte Carlo Methods

Key Idea

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \pi, \boldsymbol{s}_0 = \boldsymbol{s}\Big]$$

Estimate value function by averaging returns of sampled trajectories

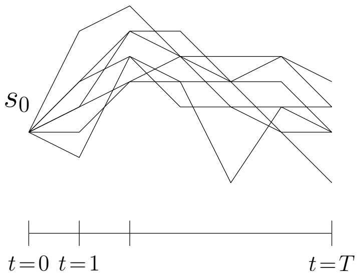


Properties:

- Model free (no knowledge of $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ required) ⏩ very general
- Learn from online experience (sampled trajectories of states, actions, rewards)
- Compute the same value function as DP (in the limit)

Episodic Set-up

- Consider a finite time horizon of length T
- Usually a fixed set of initial states $\boldsymbol{s}_0$
- Observe rewards $r_1, r_2, \ldots, r_T | \boldsymbol{s}_0, \pi$
- Value estimates $V^\pi(\boldsymbol{s}_0)$ updated at the end of an episode (not after each time step)



First-Visit Monte Carlo Policy Evaluation

- Generate trajectories with policy π
- Record reward after visiting state $\boldsymbol{s}$, average at the end

```
for i = 1 to ∞ do
    Generate trajectory τ_i^π with policy π
    for all states s ∈ τ_i do
        r ← sum of rewards following the first occurrence of s
        R(s) ← [R(s) , r]                                   ▷ Append r to array
    end for
    V^π(s) ≈ E[R(s)|π] ≈ 1/|R(s)| Σ_{j=1}^{|R(s)|} R(s)[j]
end for
```

$$V^{\pi}(\boldsymbol{s}) \approx \mathbb{E}[R(\boldsymbol{s})|\pi] \approx \frac{1}{|R(\boldsymbol{s})|}\sum_{j=1}^{|R(\boldsymbol{s})|} R(\boldsymbol{s})[j]$$

- Convergence to the correct value V^{π} in the limit

Properties

- Computational complexity independent of the size of the state space
- Very valuable if we are only interested in values starting from a small set of states s_0 ⏩ episodic set-up
- No bootstrapping (unlike DP)
- Learn from actual experience (DP only uses the model)

Properties

- Computational complexity independent of the size of the state space
- Very valuable if we are only interested in values starting from a small set of states s_0 ⏩ episodic set-up
- No bootstrapping (unlike DP)
- Learn from actual experience (DP only uses the model)

Next steps?

- Low-cost (computation) solution to computer V^{π}
- Is this useful for policy improvement? Why? (not?)

Properties

- Computational complexity independent of the size of the state space
- Very valuable if we are only interested in values starting from a small set of states $\boldsymbol{s}_0$ ⏩ episodic set-up
- No bootstrapping (unlike DP)
- Learn from actual experience (DP only uses the model)

Next steps?

- Low-cost (computation) solution to computer V^π
- Is this useful for policy improvement? Why? (not?)
- If we only know V^π, we need to perform a one-step search for policy improvement...

  ⏩ Need an estimate of Q^π if we don't have a model $p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})$

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
    Generate trajectory τ_i^π with policy π
    for all states $\boldsymbol{s} \in \tau_i$ **do**
        $r \leftarrow$ sum of rewards following the first occurrence of $\boldsymbol{s}$
        $R(\boldsymbol{s}) \leftarrow [R(\boldsymbol{s})\,,\ r]$ ▷ Append r to array
    end for
    $V^\pi(\boldsymbol{s}) \approx \mathbb{E}[R(\boldsymbol{s})|\pi] \approx \frac{1}{|R(\boldsymbol{s})|}\sum_{j=1}^{|R(\boldsymbol{s})|} R(\boldsymbol{s})[j]$
end for

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**

Generate trajectory τ_i^π with policy π

for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**

$r \leftarrow$ sum of rewards following the <u>first</u> occurrence of $(\boldsymbol{s}, \boldsymbol{a})$

$R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ ▷ Append r to array

end for

$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|} \sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$

end for

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
  Generate trajectory τ_i^π with policy π
  for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**
    $r \leftarrow$ sum of rewards following the first occurrence of $(\boldsymbol{s}, \boldsymbol{a})$
    $R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ ▷ Append r to array
  end for
  $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|} \sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$
end for

- Any potential problems?

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
  Generate trajectory τ_i^π with policy π
  for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**
    $r \leftarrow$ sum of rewards following the <u>first</u> occurrence of $(\boldsymbol{s}, \boldsymbol{a})$
    $R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ ▷ Append r to array
  end for
  $$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|}\sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$$
end for

- Any potential problems?

▶▶ ∞ many trajectories (use GPI idea)

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
    Generate trajectory τ_i^π with policy π
    for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**
        $r \leftarrow$ sum of rewards following the <u>first</u> occurrence of $(\boldsymbol{s}, \boldsymbol{a})$
        $R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ ▷ Append r to array
    end for
    $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|} \sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$
end for

- Any potential problems?
- Deterministic policy?

▶▶ ∞ many trajectories (use GPI idea)

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
    Generate trajectory τ_i^π with policy π
    for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**
        $r \leftarrow$ sum of rewards following the <u>first</u> occurrence of $(\boldsymbol{s}, \boldsymbol{a})$
        $R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ ▷ Append r to array
    end for
    $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|} \sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$
end for

- Any potential problems?
- Deterministic policy? Not many state-action pairs are visited!

▶▶ ∞ many trajectories (use GPI idea)

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
    Generate trajectory τ_i^π with policy π
    for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**
        $r \leftarrow$ sum of rewards following the first occurrence of $(\boldsymbol{s}, \boldsymbol{a})$
        $R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ ▷ Append r to array
    end for
    $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|} \sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$
end for

- Any potential problems?
- Deterministic policy? Not many state-action pairs are visited!
- Ideally, estimate $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ for all actions $\boldsymbol{a} \in \mathcal{A}$

▶▶ ∞ many trajectories (use GPI idea)

Monte Carlo for Q Function

for $i = 1$ to ∞ **do**
    Generate trajectory τ_i^π with policy π
    for all state-action pairs $(\boldsymbol{s}, \boldsymbol{a}) \in \tau_i$ **do**
        $r \leftarrow$ sum of rewards following the <u>first</u> occurrence of $(\boldsymbol{s}, \boldsymbol{a})$
        $R(\boldsymbol{s}, \boldsymbol{a}) \leftarrow [R(\boldsymbol{s}, \boldsymbol{a}),\ r]$ $\quad \triangleright$ Append r to array
    end for
    $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \approx \mathbb{E}[R(\boldsymbol{s}, \boldsymbol{a})|\pi] \approx \frac{1}{|R(\boldsymbol{s},\boldsymbol{a})|} \sum_{j=1}^{|R(\boldsymbol{s},\boldsymbol{a})|} R(\boldsymbol{s}, \boldsymbol{a})[j]$
end for

- Any potential problems?
- Deterministic policy? Not many state-action pairs are visited!
- Ideally, estimate $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ for all actions $\boldsymbol{a} \in \mathcal{A}$

➡ ∞ many trajectories (use GPI idea)

➡ Maintain exploration

Exploration

- Exploring starts: non-zero probability that each state-action pair is chosen as the start (difficult in practice, not commonly used)

Exploration

- Exploring starts: non-zero probability that each state-action pair is chosen as the start (difficult in practice, not commonly used)
- Stochastic policies with non-zero probability on each action
  - $\forall \boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}: \quad p_\pi(\boldsymbol{a}|\boldsymbol{s}) > 0$
  - Example: ϵ-greedy policies, softmax policies
  - ϵ-greedy:

$$p_\pi(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} \pi(\boldsymbol{s}) & \text{with probability } 1-\epsilon \\ \mathcal{U}(\mathcal{A}) & \text{with probability } \epsilon \end{cases}$$

Exploration

- Exploring starts: non-zero probability that each state-action pair is chosen as the start (difficult in practice, not commonly used)
- Stochastic policies with non-zero probability on each action
  - $\forall \boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}: \quad p_\pi(\boldsymbol{a}|\boldsymbol{s}) > 0$
  - Example: ϵ-greedy policies, softmax policies
  - ϵ-greedy:

$$p_\pi(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} \pi(\boldsymbol{s}) & \text{with probability } 1-\epsilon \\ \mathcal{U}(\mathcal{A}) & \text{with probability } \epsilon \end{cases}$$

▸▸ Trade off exploring the world and exploiting current knowledge

On-Policy Monte Carlo Control

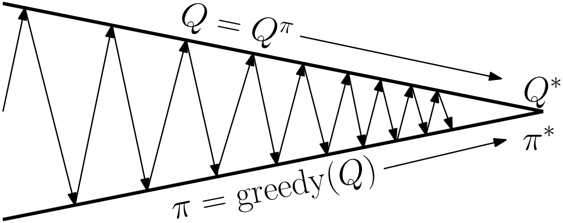
$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} Q^* \xrightarrow{I} \pi^*$$

- Approximate optimal policies
- Follow GPI idea: ϵ-greedy policy

On-Policy Monte Carlo Control

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \cdots \xrightarrow{E} Q^* \xrightarrow{I} \pi^*$$

- Approximate optimal policies
- Follow GPI idea: ϵ-greedy policy
- MC version of policy iteration:
  - Policy evaluation via Monte Carlo estimates (stochastic policy)
    ▶▶ get Q^π
  - Policy improvement: select greedy policy with respect to Q^π



Off-Policy Monte Carlo Control

- So far on-policy: Evaluate Q^{π} and, subsequently, apply π when interacting with the environment

Off-Policy Monte Carlo Control

- So far on-policy: Evaluate Q^π and, subsequently, apply π when interacting with the environment
- Off-policy: Evaluate Q^π but when interacting with the environment, follow $\pi' \neq \pi$

  ▶▶ Experience only generated from π'. Can we still estimate Q^π?

Off-Policy Monte Carlo Control

- So far on-policy: Evaluate Q^π and, subsequently, apply π when interacting with the environment
- Off-policy: Evaluate Q^π but when interacting with the environment, follow $\pi' \neq \pi$

  ▶▶ Experience only generated from π'. Can we still estimate Q^π?
- Yes, if we ensure that every action under π is also (occasionally) taken under π', i.e.,

$$\text{if } p_\pi(\boldsymbol{a}|\boldsymbol{s}) > 0 \text{ then } p_{\pi'}(\boldsymbol{a}|\boldsymbol{s}) > 0$$

Off-Policy Monte Carlo Control

- So far on-policy: Evaluate Q^π and, subsequently, apply π when interacting with the environment
- Off-policy: Evaluate Q^π but when interacting with the environment, follow $\pi' \neq \pi$

  ⏩ Experience only generated from π'. Can we still estimate Q^π?
- Yes, if we ensure that every action under π is also (occasionally) taken under π', i.e.,

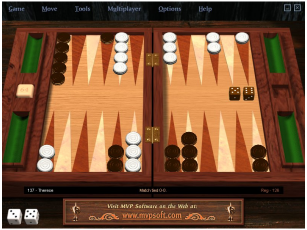
$$\text{if } p_\pi(\boldsymbol{a}|\boldsymbol{s}) > 0 \text{ then } p_{\pi'}(\boldsymbol{a}|\boldsymbol{s}) > 0$$

- Learning can be slow if the policies are very explorative, i.e., $\epsilon \gg 0$

Summary: Monte Carlo Methods

- Learn optimal behavior from interaction
- Easy to focus them on a small set of start states
- Incremental implementation (updates after each episode) possible
- Exploration required!
- No bootstrapping (unlike DP)
- Model free (unlike DP)

Temporal-Difference Learning



- Between MC and DP
- Bootstrapping
- Model-free
- MC waits until the end of the episode to update V^π, Q^π
- TD only waits until the next time step. Update value functions based on observed reward and the current estimate of V^π, Q^π

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[R(\boldsymbol{s})|\boldsymbol{s}_0 = \boldsymbol{s}] \\ &= \mathbb{E}\big[\textstyle\sum_k \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}\big] \\ &= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[R(\boldsymbol{s})|\boldsymbol{s}_0 = \boldsymbol{s}] \\ &= \mathbb{E}\big[\sum\nolimits_k \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}\big] \\ &= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

- MC uses an estimate of $R(\boldsymbol{s})$ as target κ (sample average)

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[R(\boldsymbol{s})|\boldsymbol{s}_0 = \boldsymbol{s}] \\ &= \mathbb{E}\big[\textstyle\sum_k \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}\big] \\ &= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

- MC uses an estimate of $R(\boldsymbol{s})$ as target κ (sample average)
- DP uses an estimate of $r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)$ as target κ (V_k instead of V^{π})

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[R(\boldsymbol{s})|\boldsymbol{s}_0 = \boldsymbol{s}] \\ &= \mathbb{E}\big[\sum_k \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}\big] \\ &= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

- MC uses an estimate of $R(\boldsymbol{s})$ as target κ (sample average)
- DP uses an estimate of $r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)$ as target κ (V_k instead of V^{π})
- TD target: $\kappa = r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)$

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

$$\begin{aligned} V^\pi(\boldsymbol{s}) &= \mathbb{E}[R(\boldsymbol{s})|\boldsymbol{s}_0 = \boldsymbol{s}] \\ &= \mathbb{E}\big[\sum\nolimits_k \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}\big] \\ &= \mathbb{E}\big[r_1 + \gamma V^\pi(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

- MC uses an estimate of $R(\boldsymbol{s})$ as target κ (sample average)
- DP uses an estimate of $r_1 + \gamma V^\pi(\boldsymbol{s}_1)$ as target κ (V_k instead of V^π)
- TD target: $\kappa = r_1 + \gamma V^\pi(\boldsymbol{s}_1)$
  - Approximate V^π by V_k (same as DP)

TD Policy Evaluation

Generic Update Rule

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(\kappa - V(\boldsymbol{s})\big) = (1-\alpha)V(\boldsymbol{s}) + \alpha\kappa$$

κ: "target"

$$\begin{aligned} V^{\pi}(\boldsymbol{s}) &= \mathbb{E}[R(\boldsymbol{s})|\boldsymbol{s}_0 = \boldsymbol{s}] \\ &= \mathbb{E}\big[\textstyle\sum_k \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}\big] \\ &= \mathbb{E}\big[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}\big] \end{aligned}$$

- MC uses an estimate of $R(\boldsymbol{s})$ as target κ (sample average)
- DP uses an estimate of $r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)$ as target κ (V_k instead of V^{π})
- TD target: $\kappa = r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)$
  - Approximate V^{π} by V_k (same as DP)
  - Sample trajectories (same as MC)
  - ⏩ Combine MC sampling with DP bootstrapping

TD(0)

Init.: Set $V(\boldsymbol{s})$ arbitrarily
repeat for each episode
  $\boldsymbol{a} \leftarrow p_\pi(\boldsymbol{a}|\boldsymbol{s})$ ▷ Sample action in current state
  Apply $\boldsymbol{a}$, observe $r, \boldsymbol{s}'$ ▷ Transition to next state
  $V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(r + \gamma V(\boldsymbol{s}') - V(\boldsymbol{s})\big)$ ▷ Update V
  $\boldsymbol{s} \leftarrow \boldsymbol{s}'$ ▷ Re-set current state
until $\boldsymbol{s}$ is terminal

TD(0)

Init.: Set $V(\boldsymbol{s})$ arbitrarily
repeat for each episode
  $\boldsymbol{a} \leftarrow p_\pi(\boldsymbol{a}|\boldsymbol{s})$ ▻ Sample action in current state
  Apply $\boldsymbol{a}$, observe $r, \boldsymbol{s}'$ ▻ Transition to next state
  $V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha\big(r + \gamma V(\boldsymbol{s}') - V(\boldsymbol{s})\big)$ ▻ Update V
  $\boldsymbol{s} \leftarrow \boldsymbol{s}'$ ▻ Re-set current state
until $\boldsymbol{s}$ is terminal

- Temporal difference error: $r + \gamma V(\boldsymbol{s}') - V(\boldsymbol{s})$ $\quad (= V^{\text{new}} - V^{\text{old}})$
- Difference to MC and DP:
  - MC waits until the end of the episode to update V
  - DP needs complete distribution $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ of successor states to update V

TD(λ)

- TD(0) uses 1-step returns to update V^π
- MC uses full trajectories to update V^π
- TD(λ), $\lambda \in [0,1]$, blends between them
  - $\lambda = 0$: TD(0), $\quad \lambda = 1$: MC
  - TD(λ) update rule given as a mixture of multi-step returns
  - Mixing coefficients $(1-\lambda)\lambda^k, k \geqslant 0$

SARSA: On-Policy TD Control

- Thus far, we only learned V^π using TD
- Not very useful without model when we want to do control

  ⏩ Learn Q^π

SARSA: On-Policy TD Control

- Thus far, we only learned V^π using TD
- Not very useful without model when we want to do control

  ⏩ Learn Q^π

TD Update for Q^π

$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow \underbrace{Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\big(r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a})\big)}_{=(1-\alpha)Q^{\text{old}}+\alpha Q^{\text{new}}}$$

SARSA: On-Policy TD Control

- Thus far, we only learned V^π using TD
- Not very useful without model when we want to do control

  ▶▶ Learn Q^π

TD Update for Q^π

$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow \underbrace{Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\big(r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a})\big)}_{=(1-\alpha)Q^{\text{old}}+\alpha Q^{\text{new}}}$$

- Update rule needs $(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}',\boldsymbol{a}')$ ▶▶ SARSA
- On-policy algorithm
- $\boldsymbol{a}'$ chosen from $\boldsymbol{s}'$ using policy derived from Q (e.g., ϵ-greedy)
- Convergence proofs for ϵ-greedy policies

SARSA: On-Policy TD Control

- Thus far, we only learned V^π using TD
- Not very useful without model when we want to do control
  ⏩ Learn Q^π

TD Update for Q^π

$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow \underbrace{Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\Big(r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a})\Big)}_{=(1-\alpha)Q^{\text{old}}+\alpha Q^{\text{new}}}$$

- Update rule needs $(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}',\boldsymbol{a}')$ ⏩ SARSA
- On-policy algorithm
- $\boldsymbol{a}'$ chosen from $\boldsymbol{s}'$ using policy derived from Q (e.g., ϵ-greedy)
- Convergence proofs for ϵ-greedy policies

DEMO gridworld

Q-Learning: Off-Policy TD Control

TD Update for Q^*

$$Q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow Q(\boldsymbol{s}, \boldsymbol{a}) + \alpha\Big(r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a})\Big)$$

Q-Learning: Off-Policy TD Control

TD Update for Q^*

$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\big(r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a})\big)$$

- Off-policy TD control
- SARSA: learn Q^π, Q-learning: learn Q^*
- Update the value function Q independent of the policy the agent actually follows to generate the samples (max ...)
- For convergence: keep updating all state-action pairs

Q-Learning: Off-Policy TD Control

TD Update for Q^*

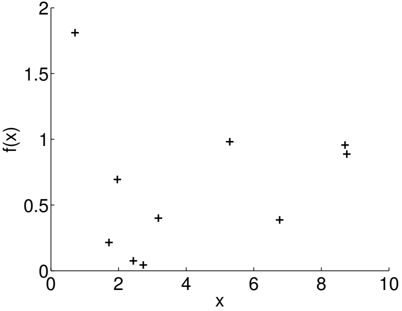
$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\big(r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a})\big)$$

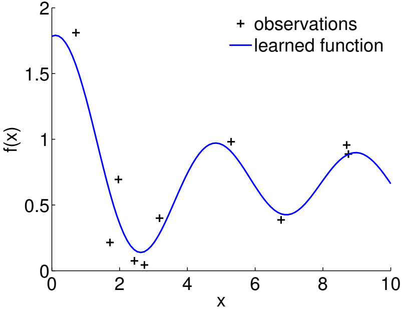
- Off-policy TD control
- SARSA: learn Q^π, Q-learning: learn Q^*
- Update the value function Q independent of the policy the agent actually follows to generate the samples (max ...)
- For convergence: keep updating all state-action pairs

DEMO gridworld 2

RL in Continuous Spaces: Function Approximation

- So far: discrete states and actions
  - ⏩ Table representation sufficient
- In continuous spaces: Function approximation for better generalization:





Function Approximation

- Typically: (linear) basis function representation

$$V(\boldsymbol{s}) = \sum_i \theta_i \phi_i(\boldsymbol{s})$$

- Basis functions ϕ_i are fixed, only parameters θ_i need to be learned

Function Approximation

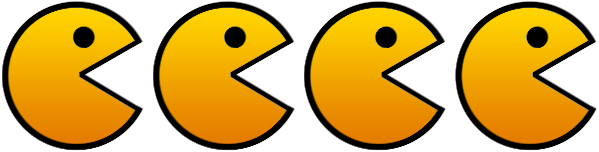
- Typically: (linear) basis function representation

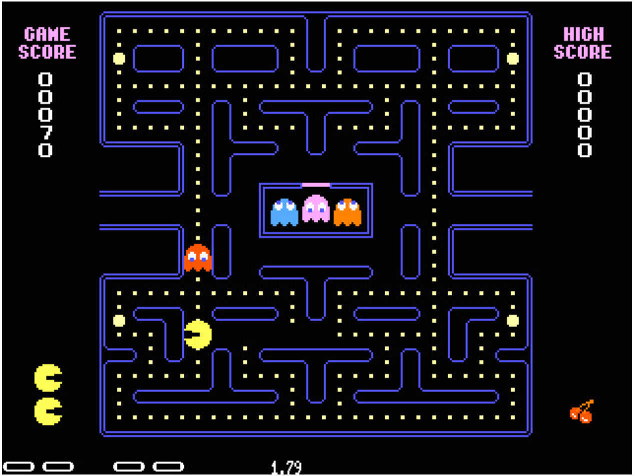
$$V(\boldsymbol{s}) = \sum_i \theta_i \phi_i(\boldsymbol{s})$$

- Basis functions ϕ_i are fixed, only parameters θ_i need to be learned
- Gradient descent to update the parameters. Example TD(0):

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha\big(v_k - V_k(\boldsymbol{s}_k)\big)\frac{\partial V_k(\boldsymbol{s}_k)}{\partial \boldsymbol{\theta}_k}$$

  v_k: approximation/estimate of $V^\pi(\boldsymbol{s}_k)$, e.g., MC estimate
  α: learning rate
- Convergence if v_k is unbiased, i.e., $\mathbb{E}[v_k] = V^\pi(\boldsymbol{s}_k)$

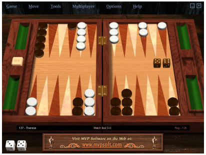


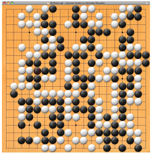
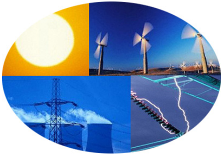

GAME
SCORE
00070
HIGH
SCORE
00000
1.79

Summary: RL with Value Functions

- Learn policies exploiting properties of the value functions V, Q
- Bellman equations/optimality principle
- Policy evaluation/improvement
- Exact solution: Dynamic programming
- Approximation solution: MC, TD
- Exploration
- Function approximation

Applications




- Board games (e.g., Tesauro, Silver, Riedmiller)
- Power systems (e.g., D. Ernst)
- Robocup (e.g., Stone, Riedmiller)
- Scheduling tasks (e.g., W. Powell)

Motivation

Reinforcement Learning Set-Up

Value Function Methods

Policy Search

RL without Value Functions: Policy Search

- Value function approximation is hard, especially in continuous domains
- Search directly in policy space?

RL without Value Functions: Policy Search

- Value function approximation is hard, especially in continuous domains
- Search directly in policy space?
- Only feasible in a restricted class Π of policies

$$\pi = \pi(\boldsymbol{\theta})$$
$$V^{\pi}(\boldsymbol{\theta}) = \mathbb{E}\big[\sum_{k=0}^{T} r_{k+1}|\boldsymbol{\theta}\big]$$

- Consider an episodic set-up, i.e., $p(\boldsymbol{s}_0)$ is given

RL without Value Functions: Policy Search

- Value function approximation is hard, especially in continuous domains
- Search directly in policy space?
- Only feasible in a restricted class Π of policies

$$\pi = \pi(\boldsymbol{\theta})$$

$$V^{\pi}(\boldsymbol{\theta}) = \mathbb{E}\big[\textstyle\sum_{k=0}^{T} r_{k+1}|\boldsymbol{\theta}\big]$$

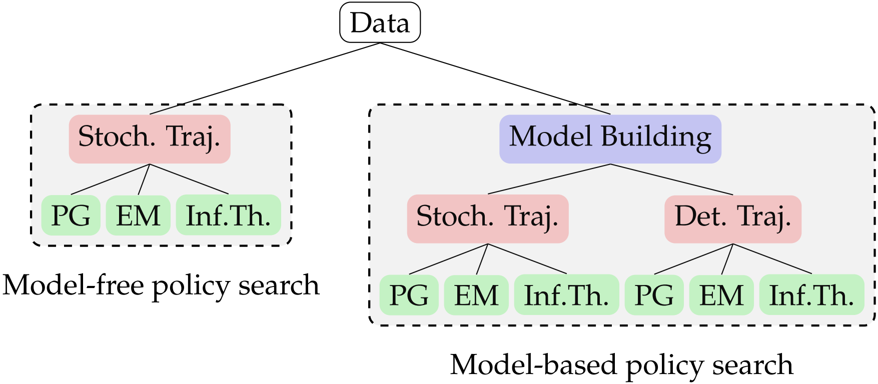
- Consider an episodic set-up, i.e., $p(\boldsymbol{s}_0)$ is given

Objective (Policy Search)

For a given class $\Pi(\boldsymbol{\theta})$ of policies, find an optimal policy

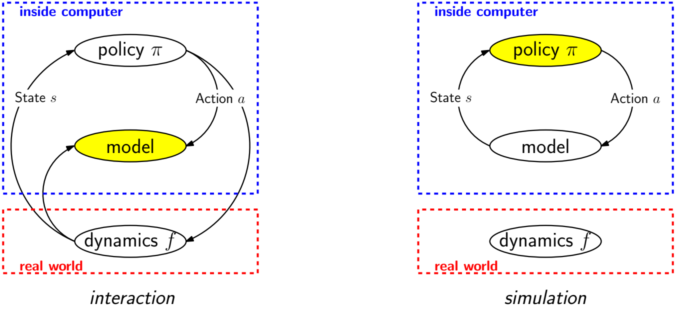
$$\pi^* \in \arg\max_{\pi \in \Pi(\boldsymbol{\theta})} V^{\pi}(\boldsymbol{\theta})$$

Categorization of Policy Search Methods



- Data: $(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{r}_i), i = 1, \ldots, n$
- Model-based vs. model-free policy search
- Policy evaluation (red), policy improvement (green)

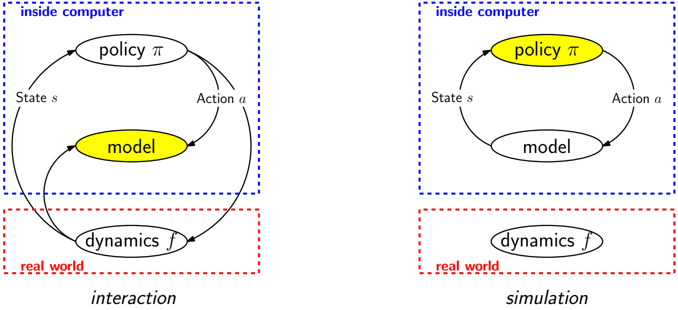
Model-based Set-up: Interaction and Simulation



Two alternating phases:

- **Interaction:** internal model is refined using experience from interacting with the real system

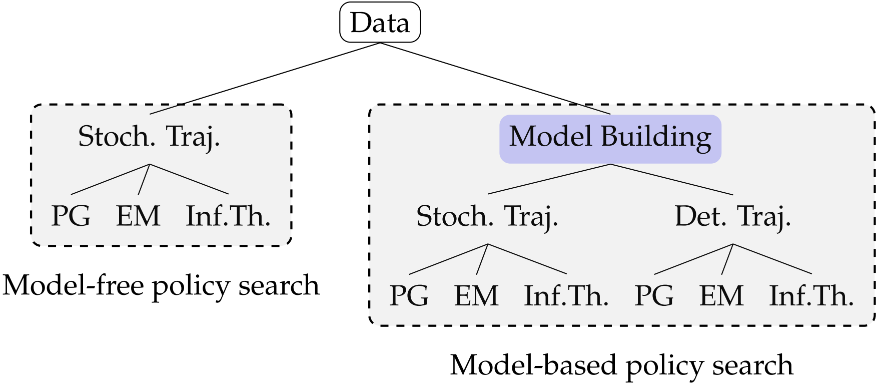
Model-based Set-up: Interaction and Simulation



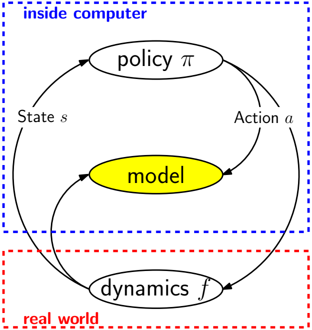
Two alternating phases:

- **Interaction:** internal model is refined using experience from interacting with the real system
- **Simulation:** internal model simulates consequences of actions in the real system, policy is refined

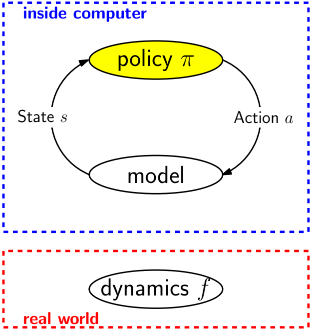
Model Building



Model Learning



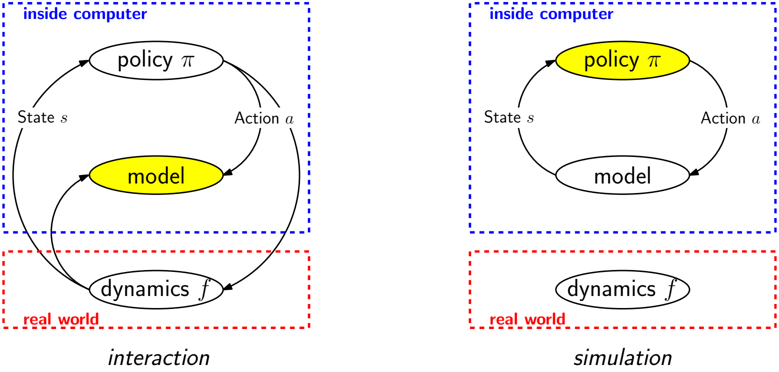
interaction



simulation

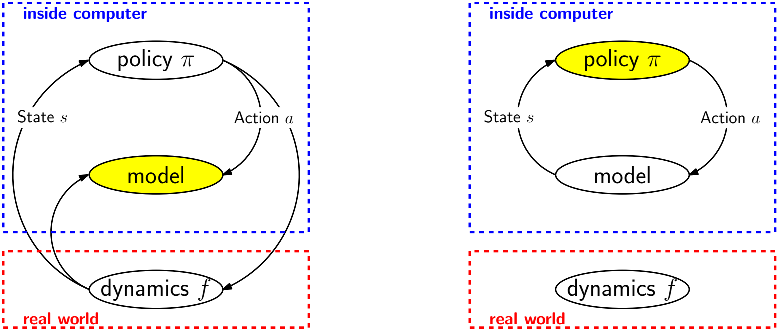
- Pro:

Model Learning



- Pro: No "real" experiments with robot for policy evaluation and improvement (just simulate!) ⏩ Protect hardware
- Con:

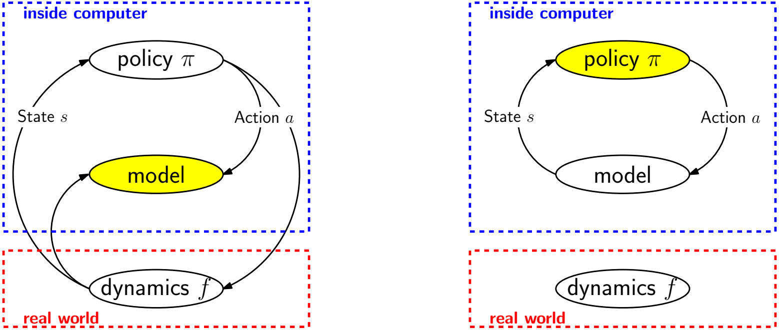
Model Learning



interaction *simulation*

- Pro: No "real" experiments with robot for policy evaluation and improvement (just simulate!) ⏩ Protect hardware
- Con: Model errors ⏩ Effects of "wrong" models?

Model Learning

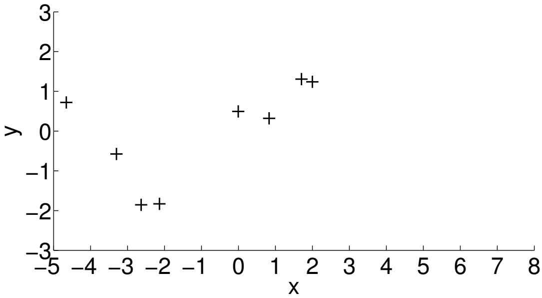


interaction *simulation*

- Pro: No "real" experiments with robot for policy evaluation and improvement (just simulate!) ⏩ Protect hardware
- Con: Model errors ⏩ Effects of "wrong" models?
- What are good models?

Model Learning

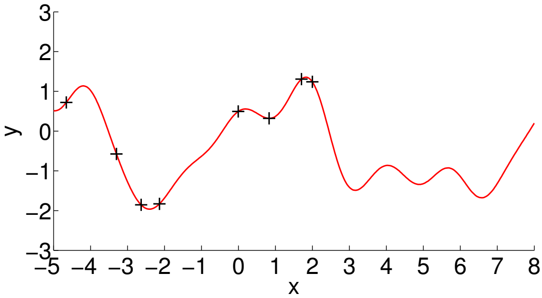
Model learning problem: Find a function $f : x \mapsto f(x) = y$



Observed function values

Model Learning

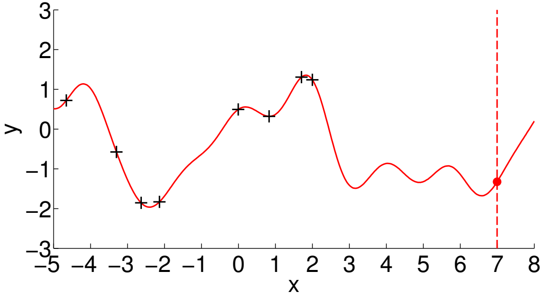
Model learning problem: Find a function $f : x \mapsto f(x) = y$



Plausible function approximators

Model Learning

Model learning problem: Find a function $f : x \mapsto f(x) = y$

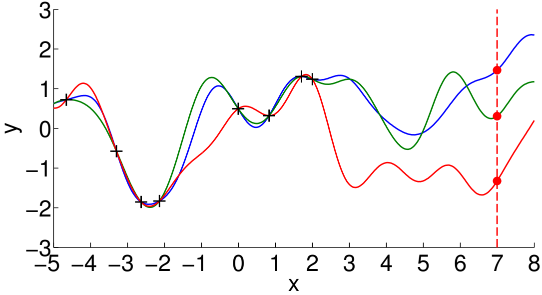


Plausible function approximators

Predictions? Decision Making?

Model Learning

Model learning problem: Find a function $f : x \mapsto f(x) = y$

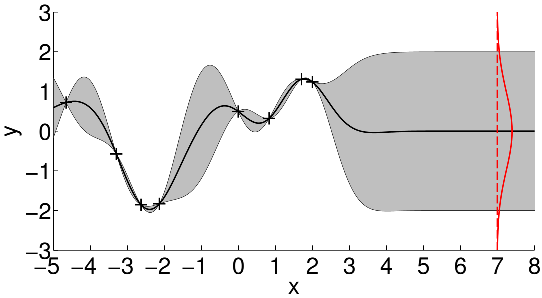


Plausible function approximators

Predictions? Decision Making? Model Errors!

Model Learning

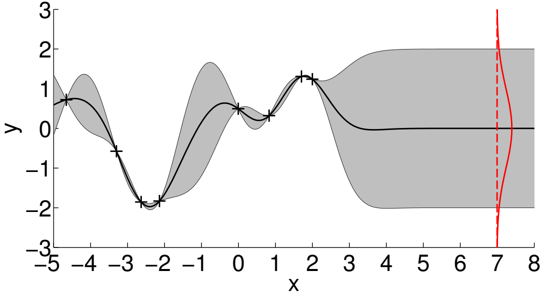
Model learning problem: Find a function $f : x \mapsto f(x) = y$



Distribution over plausible functions

Model Learning

Model learning problem: Find a function $f : x \mapsto f(x) = y$



Distribution over plausible functions

- Express uncertainty about the underlying function
- Bayesian models (Bayesian linear regression, Gaussian process, ...)

Useful Models

- Probabilistic models!
- Reduce model errors and simulation/optimization bias
- Examples of probabilistic models
    - Bayesian linear regression
    - Gaussian process

Bayesian Linear Regression: Model

- Model: $\boldsymbol{y} = \boldsymbol{\theta}^\top \phi(\boldsymbol{x}) = \sum_i \theta_i \phi_i(\boldsymbol{x})$
- $\phi(\boldsymbol{x})$: "Features", e.g., $\phi(x) = [x,\, x^2]^\top$

Bayesian Linear Regression: Model

- Model: $\boldsymbol{y} = \boldsymbol{\theta}^\top \phi(\boldsymbol{x}) = \sum_i \theta_i \phi_i(\boldsymbol{x})$
- $\phi(\boldsymbol{x})$: "Features", e.g., $\phi(x) = [x,\, x^2]^\top$
- Distribution over model parameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{\theta}\,|\,\boldsymbol{m},\, \boldsymbol{S}\big)$$

Bayesian Linear Regression: Model

- Model: $\boldsymbol{y} = \boldsymbol{\theta}^\top \phi(\boldsymbol{x}) = \sum_i \theta_i \phi_i(\boldsymbol{x})$
- $\phi(\boldsymbol{x})$: "Features", e.g., $\phi(x) = [x,\, x^2]^\top$
- Distribution over model parameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{\theta} \,|\, \boldsymbol{m},\, \boldsymbol{S}\big)$$

▶▶ For any $\boldsymbol{\theta}$, one particular function is defined

▶▶ Distribution over $\boldsymbol{\theta}$ induces distribution over functions

Bayesian Linear Regression: Model

- Model: $\boldsymbol{y} = \boldsymbol{\theta}^\top \phi(\boldsymbol{x}) = \sum_i \theta_i \phi_i(\boldsymbol{x})$
- $\phi(\boldsymbol{x})$: "Features", e.g., $\phi(x) = [x,\, x^2]^\top$
- Distribution over model parameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{\theta} \,|\, \boldsymbol{m},\, \boldsymbol{S}\big)$$

▶▶ For any $\boldsymbol{\theta}$, one particular function is defined

▶▶ Distribution over $\boldsymbol{\theta}$ induces distribution over functions

DEMO Bayesian regression

Bayesian Linear Regression: Predictions

- Model: $y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta}, \quad \boldsymbol{\theta} \sim \mathcal{N}\big(\boldsymbol{m}, \boldsymbol{S}\big)$

Bayesian Linear Regression: Predictions

- Model: $y = \phi^\top(\boldsymbol{x})\boldsymbol{\theta}, \quad \boldsymbol{\theta} \sim \mathcal{N}\big(\boldsymbol{m}, \boldsymbol{S}\big)$
- Predict function values $\boldsymbol{y} = [y_1, \ldots, y_n]^\top$ at inputs $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$

Define $\boldsymbol{\Phi} = \phi(\boldsymbol{X}) \Rightarrow \boldsymbol{y} = \boldsymbol{\Phi}^\top\boldsymbol{\theta}$

$$\mathbb{E}[\boldsymbol{y}] =$$

$$\mathbb{V}[\boldsymbol{y}] =$$

Bayesian Linear Regression: Predictions

- Model: $y = \phi^\top(\boldsymbol{x})\boldsymbol{\theta}, \quad \boldsymbol{\theta} \sim \mathcal{N}\big(\boldsymbol{m}, \boldsymbol{S}\big)$
- Predict function values $\boldsymbol{y} = [y_1, \dots, y_n]^\top$ at inputs $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]$

Define $\boldsymbol{\Phi} = \phi(\boldsymbol{X}) \Rightarrow \boldsymbol{y} = \boldsymbol{\Phi}^\top\boldsymbol{\theta}$

$$\mathbb{E}[\boldsymbol{y}] = \mathbb{E}[\boldsymbol{\Phi}^\top\boldsymbol{\theta}] = \boldsymbol{\Phi}^\top\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{\Phi}^\top\boldsymbol{m}$$

$$\mathbb{V}[\boldsymbol{y}] =$$

Bayesian Linear Regression: Predictions

- Model: $y = \phi^\top(\boldsymbol{x})\boldsymbol{\theta}, \quad \boldsymbol{\theta} \sim \mathcal{N}\big(\boldsymbol{m}, \boldsymbol{S}\big)$
- Predict function values $\boldsymbol{y} = [y_1, \dots, y_n]^\top$ at inputs $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]$

Define $\boldsymbol{\Phi} = \phi(\boldsymbol{X}) \Rightarrow \boldsymbol{y} = \boldsymbol{\Phi}^\top\boldsymbol{\theta}$

$$\mathbb{E}[\boldsymbol{y}] = \mathbb{E}[\boldsymbol{\Phi}^\top\boldsymbol{\theta}] = \boldsymbol{\Phi}^\top\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{\Phi}^\top\boldsymbol{m}$$

$$\mathbb{V}[\boldsymbol{y}] = \mathbb{V}[\boldsymbol{\Phi}^\top\boldsymbol{\theta}] = \boldsymbol{\Phi}^\top\mathbb{V}[\boldsymbol{\theta}]\boldsymbol{\Phi} = \boldsymbol{\Phi}^\top\boldsymbol{S}\boldsymbol{\Phi}$$

Introduction to Gaussian Processes

- Generalization of Bayesian linear regression
- Nonparametric Bayesian regression method
- Probability distribution over functions
- Fully specified by
  - Mean function m (average function)
  - Covariance function/kernel k (assumptions on structure)

$$\mathrm{Cov}[f(\boldsymbol{x}_p), f(\boldsymbol{x}_q)] = k(\boldsymbol{x}_p, \boldsymbol{x}_q)$$

Introduction to Gaussian Processes

- Generalization of Bayesian linear regression
- Nonparametric Bayesian regression method
- Probability distribution over functions
- Fully specified by
  - Mean function m (average function)
  - Covariance function/kernel k (assumptions on structure)

$$\mathrm{Cov}[f(\boldsymbol{x}_p), f(\boldsymbol{x}_q)] = k(\boldsymbol{x}_p, \boldsymbol{x}_q)$$

- Posterior predictive distribution at $\boldsymbol{x}_*$ is Gaussian:

$$p(f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(f(\boldsymbol{x}_*)\,|\,m(\boldsymbol{x}_*),\, \sigma^2(\boldsymbol{x}_*)\big)$$

Test input ($\boldsymbol{x}_*$) — Training data ($\boldsymbol{X}, \boldsymbol{y}$)

Gaussian Process: Definition

Definition

A Gaussian process is a collection of random variables, any finite number of which has a joint Gaussian distribution.

Gaussian Process: Definition

Definition

A Gaussian process is a collection of random variables, any finite number of which has a joint Gaussian distribution.

- Look at Gaussian distributions of function values $f_1, f_2, \ldots$
- All of them are jointly Gaussian distributed

$$\mathbb{E}[f(\boldsymbol{x})] = m(\boldsymbol{x})$$
$$\mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Gaussian Process: Predictions

- Given a training set $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))_{i=1}^n$, we can predict function values f_{*_j} at test inputs $\boldsymbol{x}_{*_j}$
- First, compute the joint distribution:

$$p(\boldsymbol{f}, \boldsymbol{f}_*|\boldsymbol{X}, \boldsymbol{X}_*) = \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & k(\boldsymbol{X}, \boldsymbol{X}_*) \\ k(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}_* \end{bmatrix}\right)$$

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)]$$

Gaussian Process: Predictions

- Given a training set $\big(\boldsymbol{x}_i, f(\boldsymbol{x}_i)\big)_{i=1}^n$, we can predict function values f_{*_j} at test inputs $\boldsymbol{x}_{*_j}$
- First, compute the joint distribution:

$$p(\boldsymbol{f}, \boldsymbol{f}_*|\boldsymbol{X}, \boldsymbol{X}_*) = \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & k(\boldsymbol{X}, \boldsymbol{X}_*) \\ k(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}_* \end{bmatrix}\right)$$

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)]$$

- Second, compute the conditional $p(\boldsymbol{f}_*|\boldsymbol{X}, \boldsymbol{X}_*, \boldsymbol{f})$ by plain Gaussian conditioning:

Gaussian Process: Predictions

- Given a training set $\big(\boldsymbol{x}_i, f(\boldsymbol{x}_i)\big)_{i=1}^n$, we can predict function values f_{*_j} at test inputs $\boldsymbol{x}_{*_j}$
- First, compute the joint distribution:

$$p(\boldsymbol{f}, \boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*) = \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & k(\boldsymbol{X}, \boldsymbol{X}_*) \\ k(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}_* \end{bmatrix}\right)$$

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)]$$

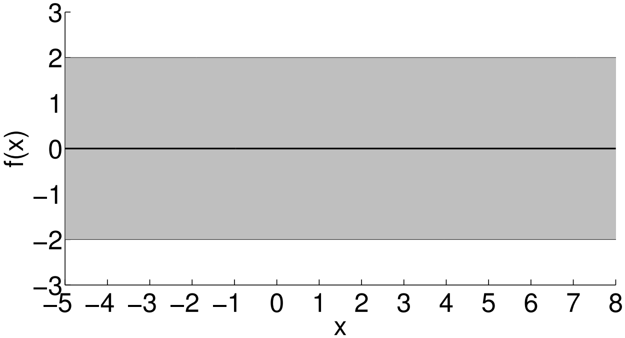
- Second, compute the conditional $p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*, \boldsymbol{f})$ by plain Gaussian conditioning:

$$p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*, \boldsymbol{f}) = \mathcal{N}\big(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*\big)$$

$$\boldsymbol{\mu}_* = m(\boldsymbol{X}_*) + k(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{f} - m(\boldsymbol{X}))$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{K}_* - k(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X}, \boldsymbol{X}_*)$$

Intuitive Introduction to Gaussian Processes

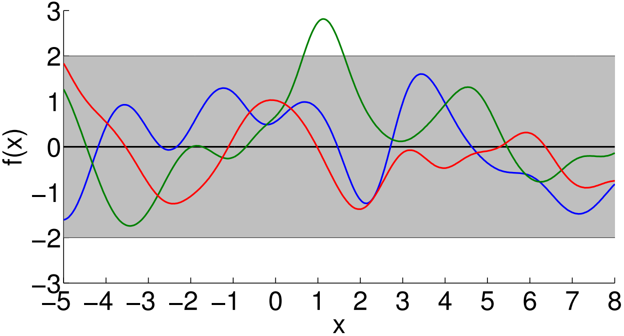


Prior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\varnothing] = m(\boldsymbol{x}_*) = 0$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\varnothing] = \sigma^2(\boldsymbol{x}_*) = \mathrm{Cov}[f(\boldsymbol{x}_*), f(\boldsymbol{x}_*)] = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

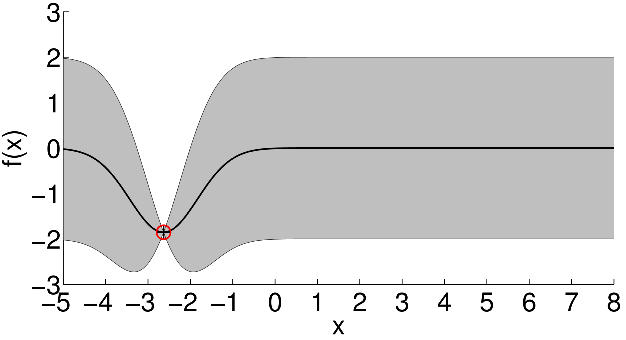


Prior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\varnothing] = m(\boldsymbol{x}_*) = 0$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\varnothing] = \sigma^2(\boldsymbol{x}_*) = \mathrm{Cov}[f(\boldsymbol{x}_*), f(\boldsymbol{x}_*)] = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes



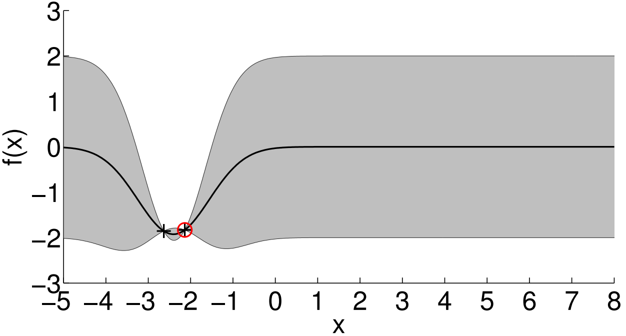
Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$

$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

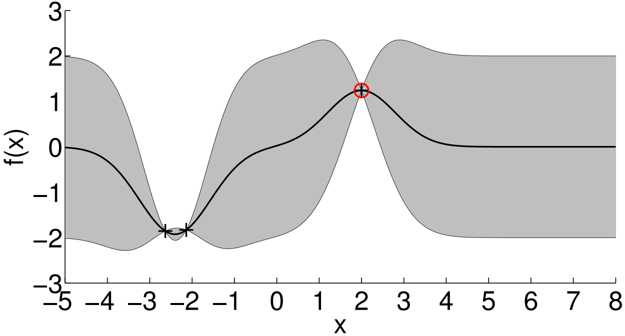


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

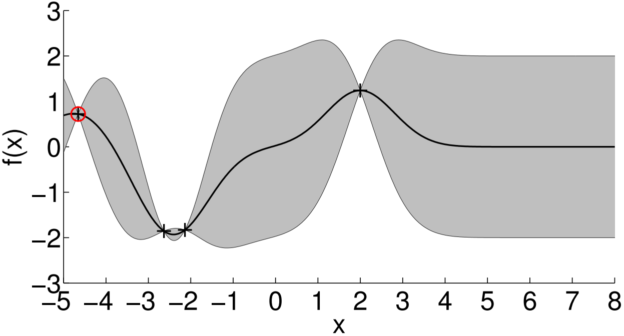


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

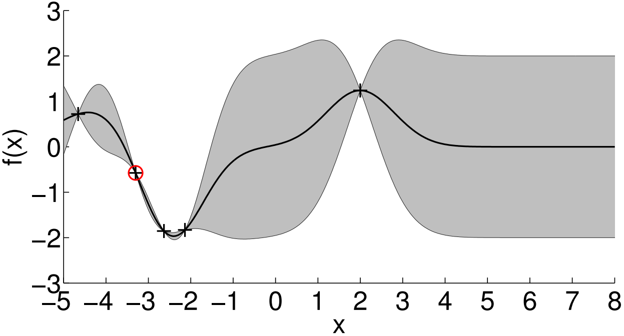


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

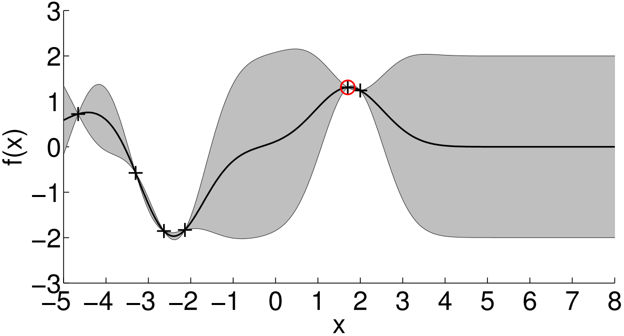


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes



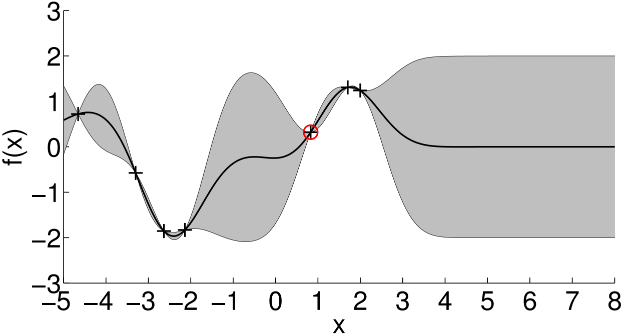
Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$

$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes



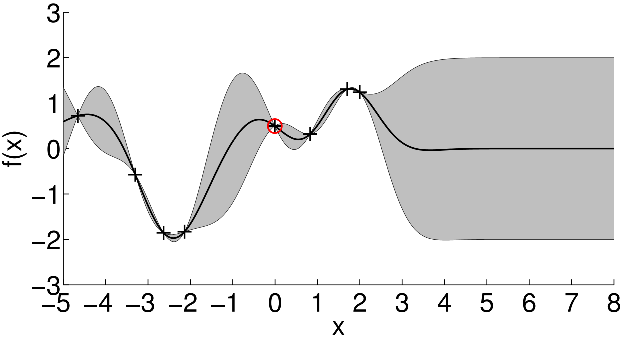
Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$

$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

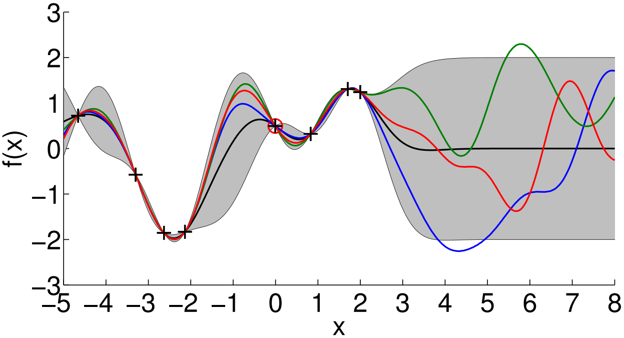


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes



Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{X},\boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{X},\boldsymbol{x}_*)^\top k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)$$

Properties

- Universal function approximator ⏩ extremely expressive
- Model gives "free" variance estimates

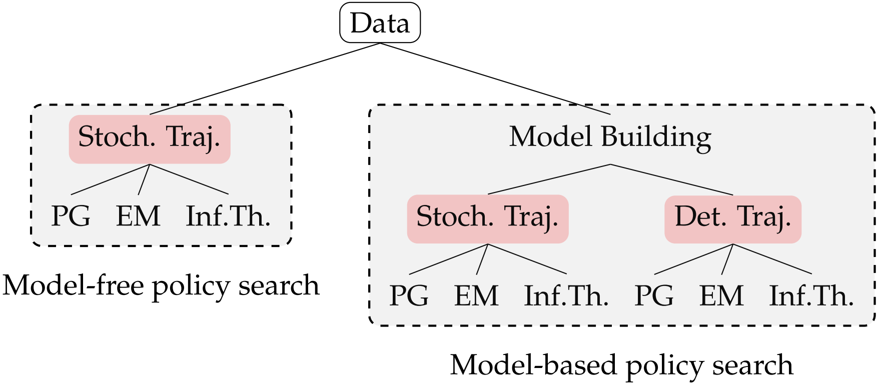
Properties

- Universal function approximator ⏩ extremely expressive
- Model gives "free" variance estimates
- Computationally involved:
  - Training: $\mathcal{O}(N^3)$ ⏩ Repeated inversion of $N \times N$ matrix
  - Mean prediction: $\mathcal{O}(N)$ ⏩ Scalar product
  - Variance prediction: $\mathcal{O}(N^2)$ ⏩ Matrix-vector multiplication

Properties

- Universal function approximator ⏩ extremely expressive
- Model gives "free" variance estimates
- Computationally involved:
  - Training: $\mathcal{O}(N^3)$ ⏩ Repeated inversion of $N \times N$ matrix
  - Mean prediction: $\mathcal{O}(N)$ ⏩ Scalar product
  - Variance prediction: $\mathcal{O}(N^2)$ ⏩ Matrix-vector multiplication
- Sparse approximations exist
- Code/book online: `http://www.gaussianprocess.org`

Policy Evaluation



Policy evaluation: Compute expected long-term reward

- Stochastic trajectory evaluation
- Deterministic trajectory evaluation

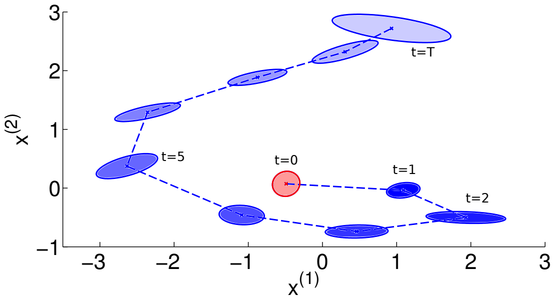
Policy Evaluation

- Stochastic inference (sampling) using either the learned model (simulator) or the real system
- Deterministic inference—only with a learned model

Stochastic Inference

- Sample trajectories $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i)$ ⏩ Monte Carlo
- Conceptually very simple
- Requires a lot of "interactions" (if you don't have a model or a good simulator)
  ⏩ Potentially impractical (e.g., in robotics)

Deterministic Inference



- Analytically propagate uncertainty through the model
- Computationally/mathematically more involved
- Can't do this for arbitrary systems, but for some.

Deterministic Inference: Example

Linear system

$$p(\boldsymbol{s}_t) = \mathcal{N}\big(\boldsymbol{s}_t \,|\, \boldsymbol{m}_t, \boldsymbol{S}_t\big)$$
$$\boldsymbol{s}_{t+1} = \boldsymbol{A}\boldsymbol{s}_t$$

Successor state distribution $p(\boldsymbol{s}_{t+1})$?

Deterministic Inference: Example

Linear system

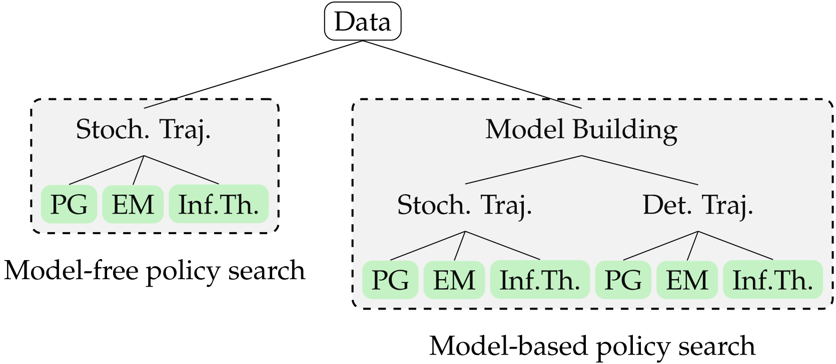
$$p(\boldsymbol{s}_t) = \mathcal{N}\big(\boldsymbol{s}_t \,|\, \boldsymbol{m}_t,\, \boldsymbol{S}_t\big)$$
$$\boldsymbol{s}_{t+1} = \boldsymbol{A}\boldsymbol{s}_t$$

Successor state distribution $p(\boldsymbol{s}_{t+1})$?

$$p(\boldsymbol{s}_{t+1}) = \mathcal{N}\big(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{m}_{t+1},\, \boldsymbol{S}_{t+1}\big)$$
$$\boldsymbol{m}_{t+1} = \boldsymbol{A}\boldsymbol{m}_t\,, \quad \boldsymbol{S}_{t+1} = \boldsymbol{A}\boldsymbol{S}_t\boldsymbol{A}^\top$$

- In nonlinear/non-Gaussian systems, we need approximations (e.g., linearization, moment matching)

Policy Improvement



Policy improvement (green)

- Policy gradients
- Expectation Maximization
- Information theory

Policy Search: Policy Improvement

Objective

Find policy parameters $\boldsymbol{\theta}^*$, which maximize the expected long-term reward

$$V^{\pi}(\boldsymbol{\theta}) = \mathbb{E}\Big[\sum_{k=0}^{T} \gamma^k r_{k+1} | \boldsymbol{\theta}\Big], \qquad \boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$$

- No global value function model V^{π} or Q^{π}
- Search directly in (policy) parameter space

Policy Search: Policy Improvement

Objective

Find policy parameters $\boldsymbol{\theta}^*$, which maximize the expected long-term reward

$$V^\pi(\boldsymbol{\theta}) = \mathbb{E}\Big[\sum_{k=0}^{T} \gamma^k r_{k+1} | \boldsymbol{\theta}\Big], \qquad \boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$$

- No global value function model V^π or Q^π
- Search directly in (policy) parameter space

▶▶ One way: gradient-based optimization

- Compute V^π with corresponding gradients $\mathrm{d}V^\pi/\mathrm{d}\boldsymbol{\theta}$
- Gradient-based optimizer for maximization (e.g., CG, BFGS)

Gradient Estimation for Stochastic Inference

Gradient Estimation for Stochastic Inference

- Finite (central) differences

$$\frac{\mathrm{d}V^{\pi}(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} \approx \frac{V^{\pi}(\boldsymbol{\theta}+\epsilon) - V^{\pi}(\boldsymbol{\theta}-\epsilon)}{2\epsilon}$$

  - Model $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$ not required but useful
  - Large variance of estimator ⏩ many samples needed

Gradient Estimation for Stochastic Inference

- Finite (central) differences

$$\frac{\mathrm{d}V^{\pi}(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} \approx \frac{V^{\pi}(\boldsymbol{\theta}+\epsilon) - V^{\pi}(\boldsymbol{\theta}-\epsilon)}{2\epsilon}$$

  - Model $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$ not required but useful
  - Large variance of estimator ⏩ many samples needed
- PEGASUS trick:
  - Fix the random seed and re-set
  - Smaller variance of the estimate of V^{π} and its gradient
  - No model $p(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$ required
  - Only with simulator (where we can run exactly the same experiment)

Gradient Estimation for Deterministic Inference

- Finite differences and PEGASUS still work
- Analytic (=exact) gradients. Example (assume $r = r(\boldsymbol{s})$):

$$\frac{\mathrm{d}V(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \frac{\mathrm{d}r(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t} \frac{\mathrm{d}\boldsymbol{s}_t}{\mathrm{d}\boldsymbol{\theta}}$$

$$= \sum_t \gamma^t \frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t} \left(\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_{t-1}} \frac{\mathrm{d}\boldsymbol{s}_{t-1}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{a}_{t-1}} \frac{\partial \boldsymbol{a}_{t-1}}{\partial \boldsymbol{\theta}} \right)$$

Gradient Estimation for Deterministic Inference

- Finite differences and PEGASUS still work
- Analytic (=exact) gradients. Example (assume $r = r(\boldsymbol{s})$):

$$\frac{\mathrm{d}V(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \frac{\mathrm{d}r(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t}\frac{\mathrm{d}\boldsymbol{s}_t}{\mathrm{d}\boldsymbol{\theta}}$$

$$= \sum_t \gamma^t \frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t}\left(\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_{t-1}}\frac{\mathrm{d}\boldsymbol{s}_{t-1}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{a}_{t-1}}\frac{\partial \boldsymbol{a}_{t-1}}{\partial \boldsymbol{\theta}}\right)$$

- Requires
  - Forward model $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$
  - Differentiable policy $\boldsymbol{a} = \pi(\boldsymbol{s}, \boldsymbol{\theta})$

Gradient Estimation for Deterministic Inference

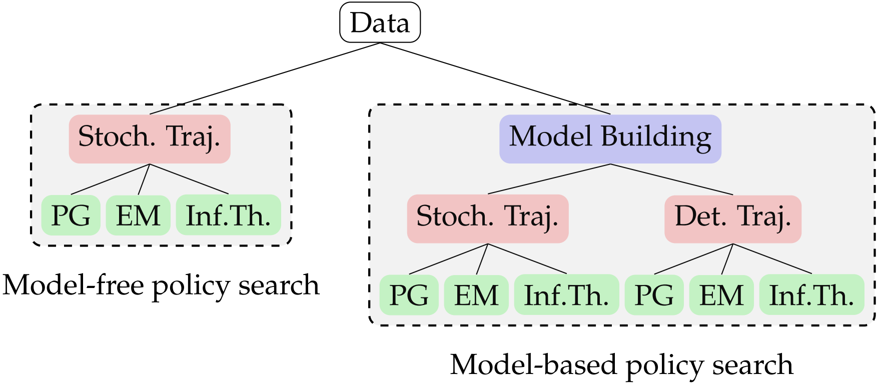
- Finite differences and PEGASUS still work
- Analytic (=exact) gradients. Example (assume $r = r(\boldsymbol{s})$):

$$\frac{\mathrm{d}V(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \frac{\mathrm{d}r(\boldsymbol{s}_t)}{\mathrm{d}\boldsymbol{\theta}} = \sum_t \gamma^t \frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t} \frac{\mathrm{d}\boldsymbol{s}_t}{\mathrm{d}\boldsymbol{\theta}}$$

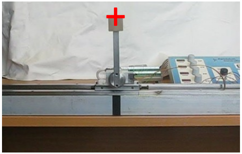
$$= \sum_t \gamma^t \frac{\partial r(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t} \left(\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{s}_{t-1}} \frac{\mathrm{d}\boldsymbol{s}_{t-1}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{a}_{t-1}} \frac{\partial \boldsymbol{a}_{t-1}}{\partial \boldsymbol{\theta}} \right)$$

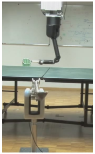
- Requires
  - Forward model $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$
  - Differentiable policy $\boldsymbol{a} = \pi(\boldsymbol{s}, \boldsymbol{\theta})$
- Mathematically more involved
- Gradients are exact (no variance): single trajectory evaluation

Policy Search



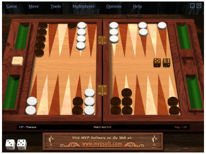
Applications in Robotics and Control

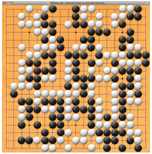



- Cart-pole: e.g., Riedmiller (2005), Deisenroth & Rasmussen (2011)
- Throttle valve control: Bischoff et al. (2013)
- Autonomous helicopter: e.g., Abbeel, Ng et al. (2003–2010), Bagnell & Schneider (2001)
- Pancake flipping (Kormushev et al., 2010)
- Throwing and catching balls (Kober et al., 2012)

Summary



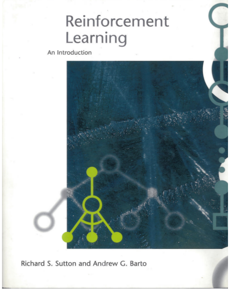


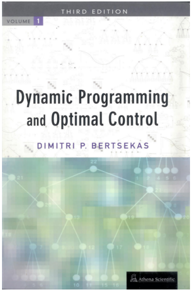
- RL is a principled framework for sequential decision making under uncertainty
- Value functions V, Q
- Exact RL: Dynamic programming
- Approximate RL: Monte Carlo, TD
- Policy Search with applications in robotics

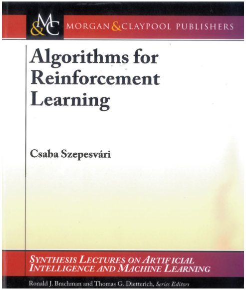
m.deisenroth@imperial.ac.uk

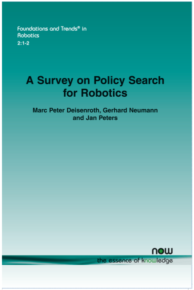
Thank you for your attention

Key References









- Sutton, Barto: *Reinforcement Learning: An Introduction* (online)
- Bertsekas: *Dynamic Programming and Optimal Control, Vol. 1–2*
- Szepesvári: *Algorithms for Reinforcement Learning* (online)
- Deisenroth et al.: *A Survey on Policy Search for Robotics* (online)

RL Software Packages

- RLGlue: `http://glue.rl-community.org/`
- RLPy: `http://acl.mit.edu/RLPy/`
- CLSquare: `http://www.ni.uos.de/index.php?id=70`
- PIQLE: `http://piqle.sourceforge.net/`
- RL Toolbox: `http://www.igi.tugraz.at/ril-toolbox/`
- LibPG: `http://code.google.com/p/libpgrl/`
- PILCO (policy search): `http://mlg.eng.cam.ac.uk/pilco`

References

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, volume 19, page 2007. The MIT Press, Cambridge, MA, USA, 2007.

[2] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 of *Optimization and Computation Series*. Athena Scientific, Belmont, MA, USA, 3rd edition, 2005.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2 of *Optimization and Computation Series*. Athena Scientific, Belmont, MA, USA, 3rd edition, 2007.

[4] B. Bischoff, D. Nguyen-Tuong, T. Koller, H. Markert, and A. Knoll. Learning Throttle Valve Control Using Policy Search. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

[5] M. P. Deisenroth, G. Neumann, and J. Peters. *A Survey on Policy Search for Robotics*, volume 2 of *Foundations and Trends in Robotics*. NOW Publishers, 2013.

[6] M. P. Deisenroth and C. E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the International Conference on Machine Learning*, pages 465–472, New York, NY, USA, June 2011. ACM.

[7] S. Gelly, M. Schoenauer, M. Sebag, O. Teytaud, L. Kocsis, D. Silver, and C. Szepesvári. The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions. *Communications of the ACM*, 55(3), 2012.

[8] Q. Gemine, E. Karangelos, D. Ernst, and B. Cornélusse. Active Network Management: Planning under Uncertainty for Exploiting Load Modulation. In *Proceedings of the 2013 IREP Symposium-Bulk Power System Dynamics and Control*, 2013.

[9] J. Kober, K. Mülling, and J. Peters. Learning Throwing and Catching Skills. In *Proceedings of the IEEE/RSJ International Conference on Robot Systems*, 2012.

[10] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot Motor Skill Coordination with EM-based Reinforcement Learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[11] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, volume 703. John Wiley & Sons, 2007.

[12] M. Riedmiller. Neural Fitted Q Iteration—First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Proceedings of the 16th European Conference on Machine Learning*, Porto, Portugal, 2005.

[13] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 1998.

[14] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[15] G. J. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

Appendix

Self-Consistency of Value Functions

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$

Self-Consistency of Value Functions

$$V^\pi(\boldsymbol{s}) = \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]$$

Self-Consistency of Value Functions

$$
\begin{aligned}
V^\pi(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]
\end{aligned}
$$

Self-Consistency of Value Functions

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} |\boldsymbol{s}_0 = \boldsymbol{s}, \pi\Big]
\end{aligned}
$$

Self-Consistency of Value Functions

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\Big] \\
&= \mathbb{E}\Big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\Big]
\end{aligned}
$$

Self-Consistency of Value Functions

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= {\color{red}\mathbb{E}}\big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big]
\end{aligned}
$$

Self-Consistency of Value Functions

$$
\begin{aligned}
V^\pi(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \sum_{a} p_\pi(\boldsymbol{a}|\boldsymbol{s}) \sum_{s'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \left(\mathbb{E}[r_1] + \gamma \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_1 = \boldsymbol{s}', \pi\big]\right)
\end{aligned}
$$

Self-Consistency of Value Functions

$$\begin{aligned}
V^\pi(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \sum_{a} p_\pi(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \left(\mathbb{E}[r_1] + \gamma \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+2}|\boldsymbol{s}_1 = \boldsymbol{s}', \pi\big]\right)
\end{aligned}$$

Self-Consistency of Value Functions

$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \sum_a p_\pi(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \left(\mathbb{E}[r_1] + \gamma \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_1 = \boldsymbol{s}', \pi\big] \right) \\
&= \sum_{\boldsymbol{a}} p_\pi(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \big(\mathbb{E}[r_1(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')] + \gamma V^{\pi}(\boldsymbol{s}') \big)
\end{aligned}
$$

Self-Consistency of Value Functions

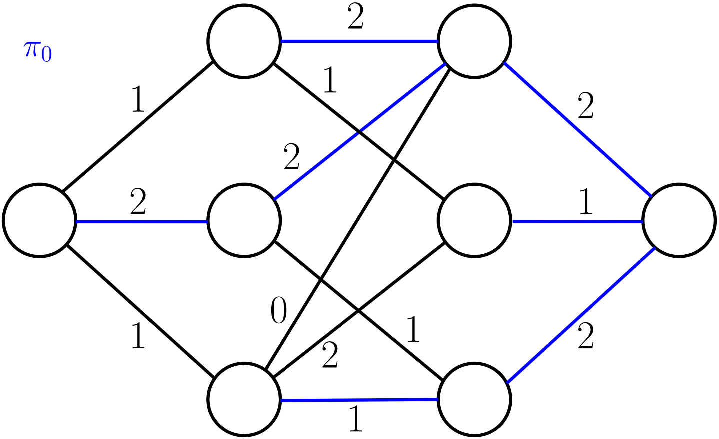
$$
\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \sum_{a} p_{\pi}(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \left(\mathbb{E}[r_1] + \gamma \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+2} | \boldsymbol{s}_1 = \boldsymbol{s}', \pi\big] \right) \\
&= \sum_{\boldsymbol{a}} p_{\pi}(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \big(\mathbb{E}[r_1(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')] + \gamma V^{\pi}(\boldsymbol{s}') \big)
\end{aligned}
$$

Self-Consistency of Value Functions

$$\begin{aligned}
V^{\pi}(\boldsymbol{s}) &= \mathbb{E}\big[R|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+1}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \mathbb{E}\big[r_1 + \gamma \sum_{k=0}^{\infty} \gamma^k r_{k+2}|\boldsymbol{s}_0 = \boldsymbol{s}, \pi\big] \\
&= \sum_{a} p_{\pi}(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \left(\mathbb{E}[r_1] + \gamma \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{k+2}|\boldsymbol{s}_1 = \boldsymbol{s}', \pi\big]\right) \\
&= \sum_{\boldsymbol{a}} p_{\pi}(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \big(\mathbb{E}[r_1(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')] + \gamma V^{\pi}(\boldsymbol{s}')\big) \\
&= \mathbb{E}[r_1 + \gamma V^{\pi}(\boldsymbol{s}_1)|\boldsymbol{s}_0 = \boldsymbol{s}, \pi]
\end{aligned}$$

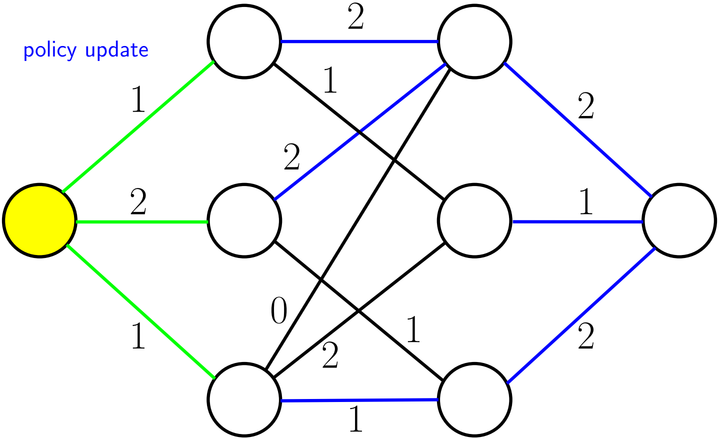
Demo: Policy Iteration

Example: Shortest-path problem



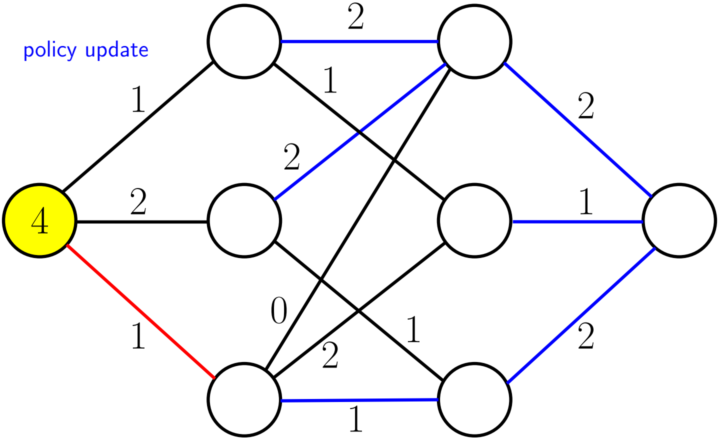
Demo: Policy Iteration

Example: Shortest-path problem



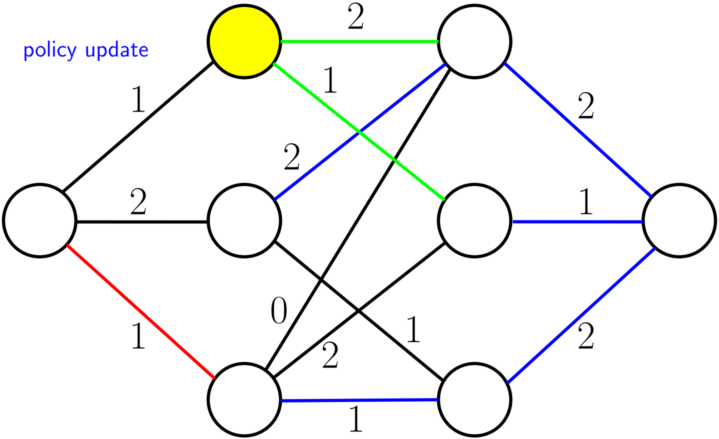
Demo: Policy Iteration

Example: Shortest-path problem



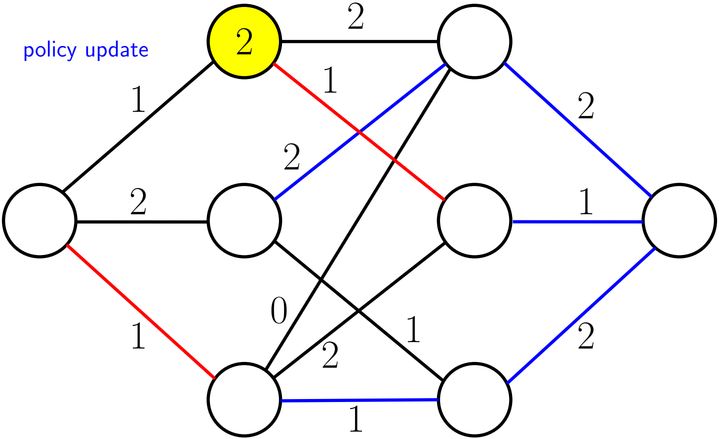
Demo: Policy Iteration

Example: Shortest-path problem



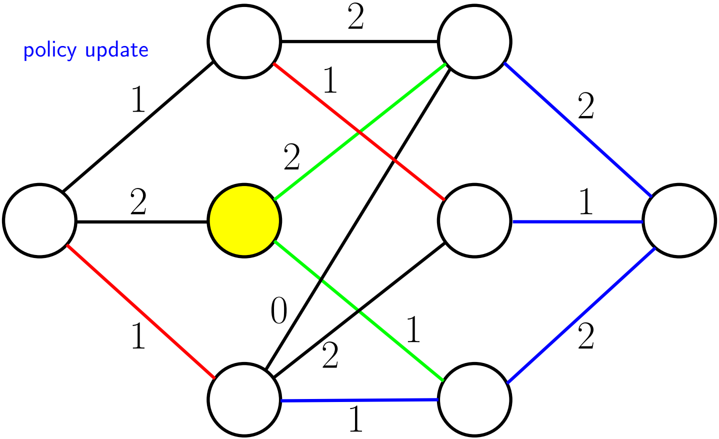
Demo: Policy Iteration

Example: Shortest-path problem



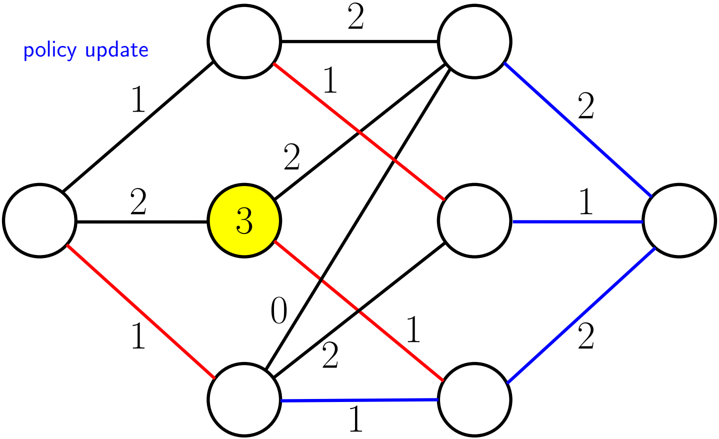
Demo: Policy Iteration

Example: Shortest-path problem



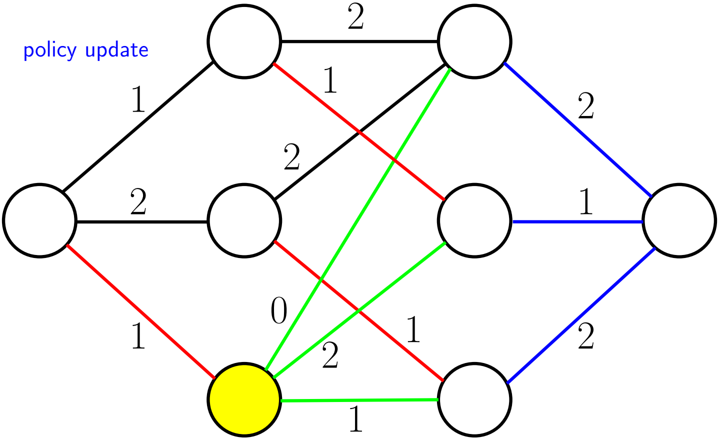
Demo: Policy Iteration

Example: Shortest-path problem



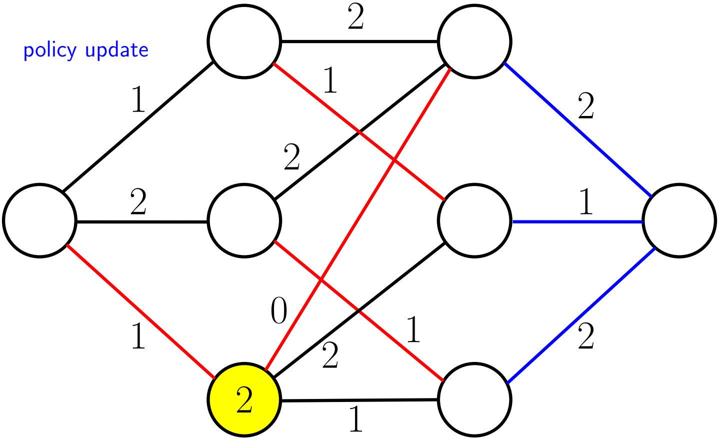
Demo: Policy Iteration

Example: Shortest-path problem



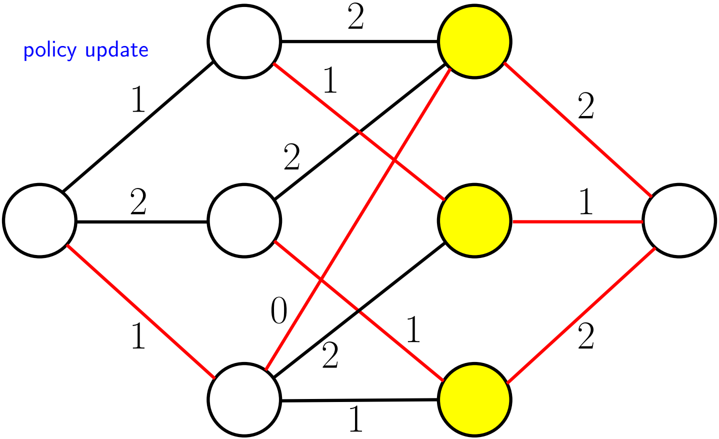
Demo: Policy Iteration

Example: Shortest-path problem



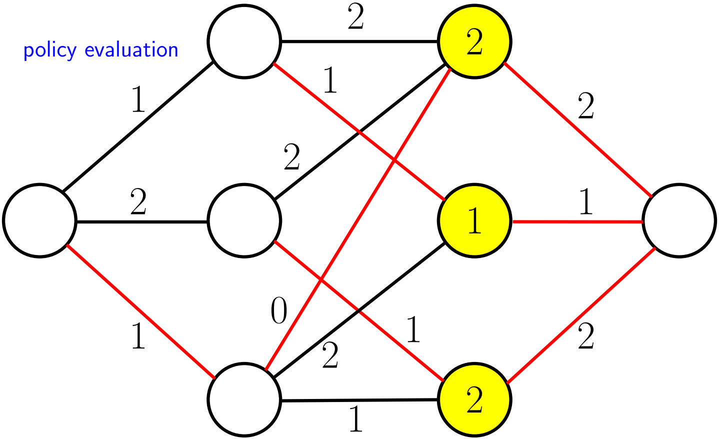
Demo: Policy Iteration

Example: Shortest-path problem



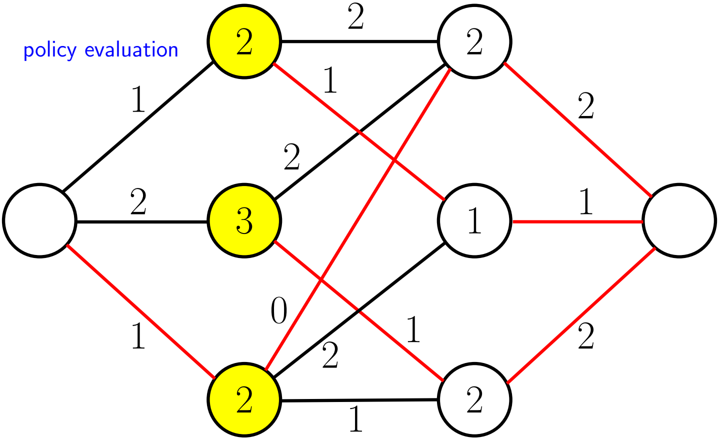
Demo: Policy Iteration

Example: Shortest-path problem



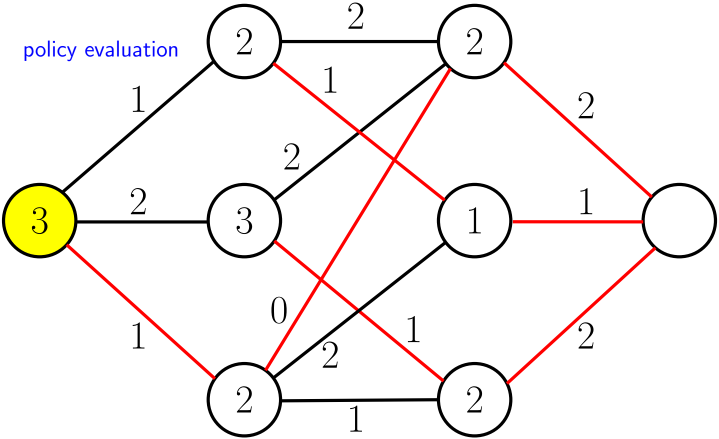
Demo: Policy Iteration

Example: Shortest-path problem



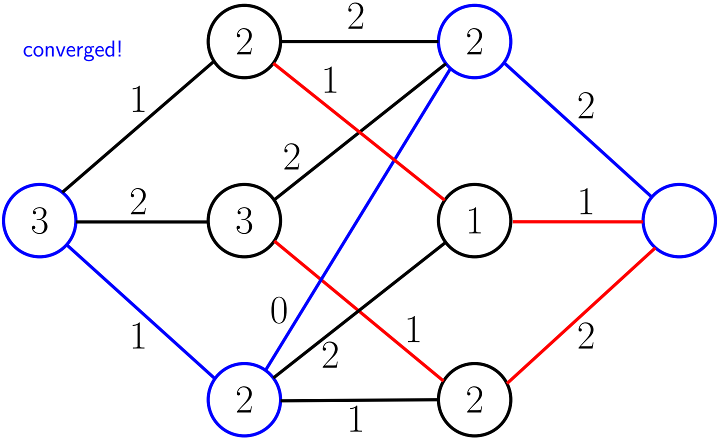
Demo: Policy Iteration

Example: Shortest-path problem



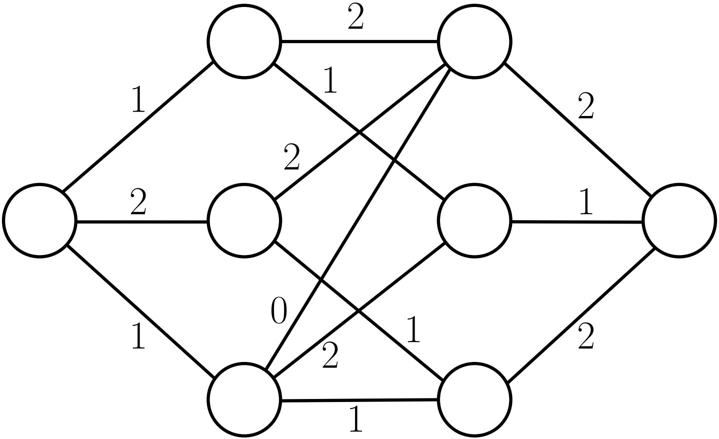
Demo: Policy Iteration

Example: Shortest-path problem



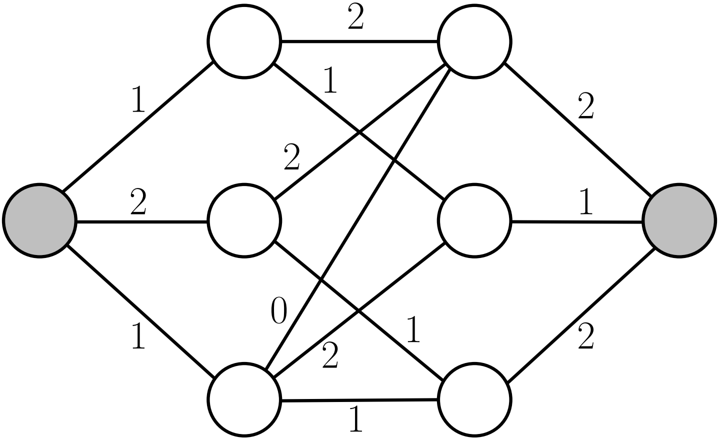
Demo: Value Iteration

Example: Shortest-path problem



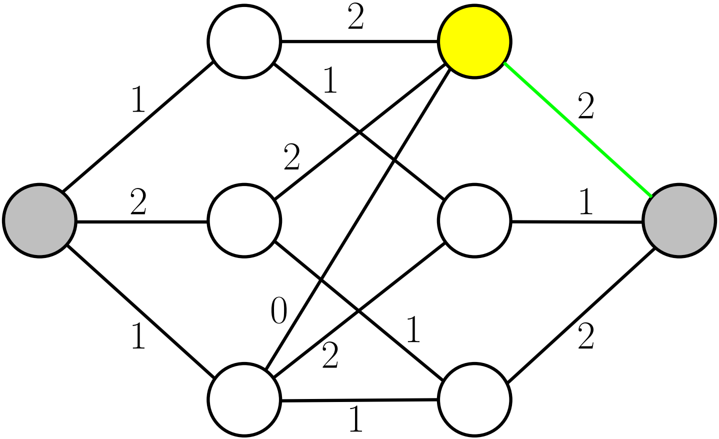
Demo: Value Iteration

Example: Shortest-path problem



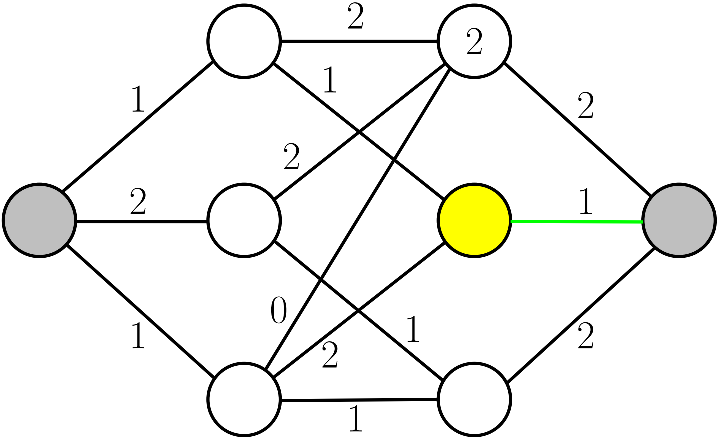
Demo: Value Iteration

Example: Shortest-path problem



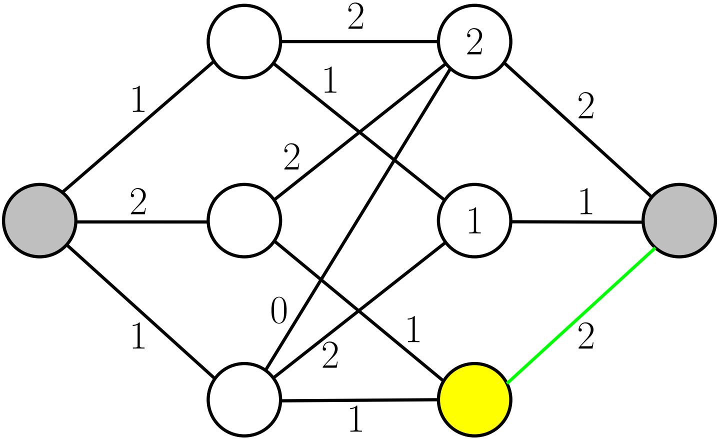
Demo: Value Iteration

Example: Shortest-path problem



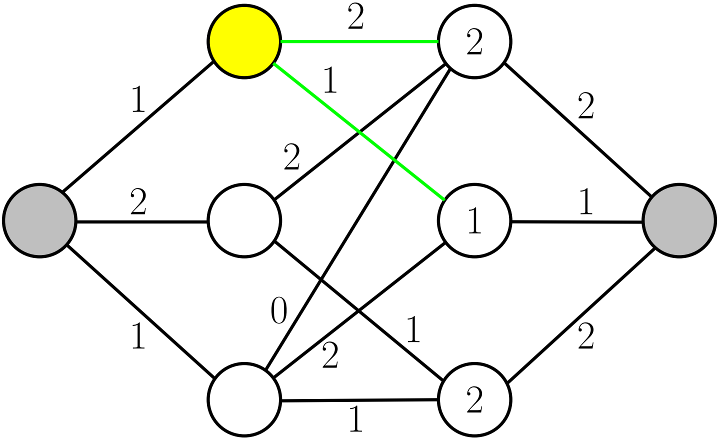
Demo: Value Iteration

Example: Shortest-path problem



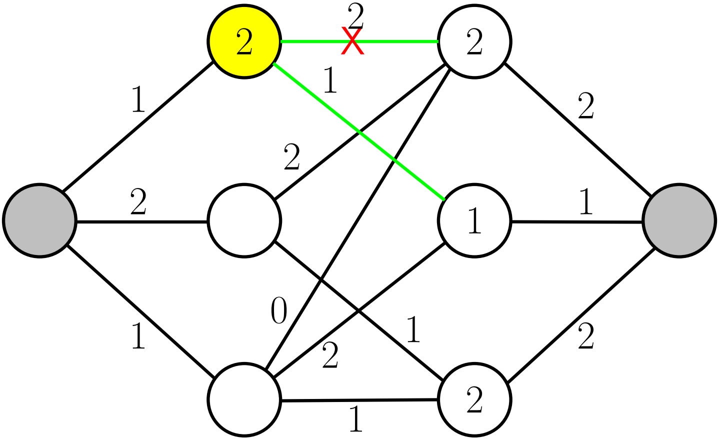
Demo: Value Iteration

Example: Shortest-path problem



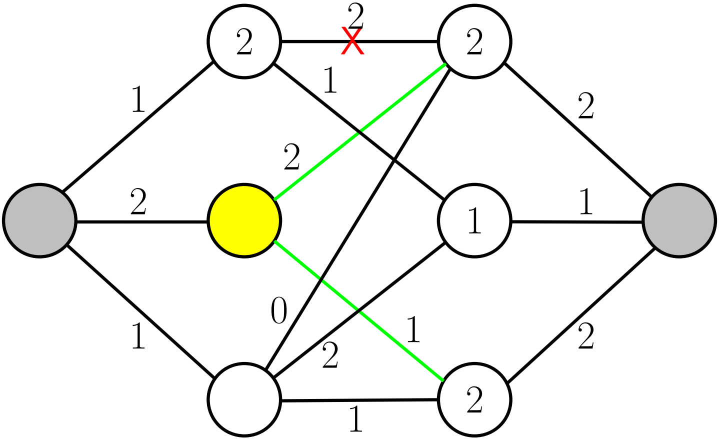
Demo: Value Iteration

Example: Shortest-path problem



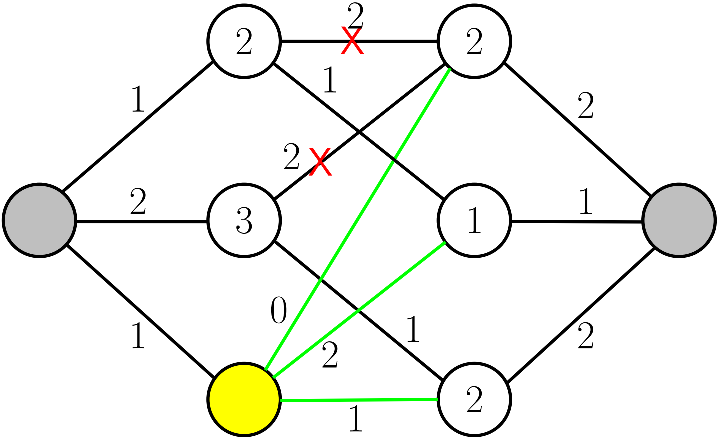
Demo: Value Iteration

Example: Shortest-path problem



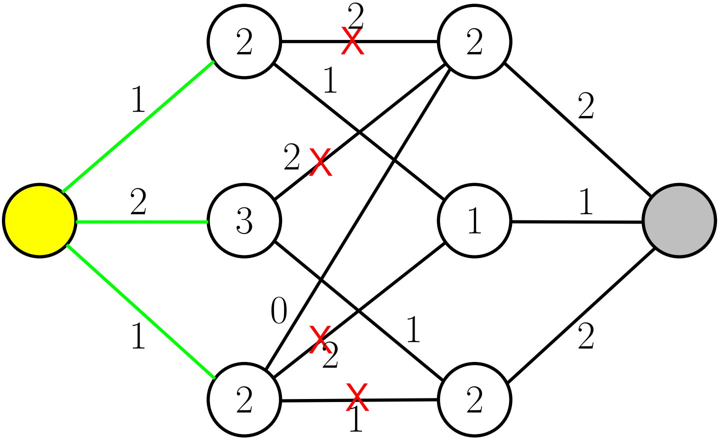
Demo: Value Iteration

Example: Shortest-path problem



Demo: Value Iteration

Example: Shortest-path problem



Demo: Value Iteration

Example: Shortest-path problem