An Introduction to Reinforcement Learning

Shivaram Kalyanakrishnan
shivaram@csa.iisc.ernet.in

Department of Computer Science and Automation
Indian Institute of Science

August 2014
What is Reinforcement Learning?
What is Reinforcement Learning?

[Video\(^1\) of little girl learning to ride bicycle]

1. https://www.youtube.com/watch?v=Qv43pK1VZXk
What is Reinforcement Learning?

[Video\(^1\) of little girl learning to ride bicycle]

1. https://www.youtube.com/watch?v=Qv43pK1VZXk
What is Reinforcement Learning?

[Video\(^1\) of little girl learning to ride bicycle]

---

Learning to Drive a Bicycle using Reinforcement Learning and Shaping

\(^1\) https://www.youtube.com/watch?v=Qv43pKlVZXk
What is Reinforcement Learning?

[Video\(^1\) of little girl learning to ride bicycle]

Learning to Drive a Bicycle using Reinforcement Learning and Shaping

Learning by trial and error to perform sequential decision making.

\(^1\) https://www.youtube.com/watch?v=Qv43pK1VZXk
Our View of Reinforcement Learning

- Operations Research (Dynamic Programming)
- Control Theory
- Psychology (Animal Behaviour)
- Reinforcement Learning
- Neuroscience
- Artificial Intelligence and Computer Science
Our View of Reinforcement Learning

Operations Research (Dynamic Programming)

Control Theory

Psychology (Animal Behaviour)

Reinforcement Learning

Neuroscience

Artificial Intelligence and Computer Science

B. F. Skinner
Our View of Reinforcement Learning

- Neurosciences
- Reinforcement Learning
- Psychology (Animal Behaviour)
- Artificial Intelligence and Computer Science
- Operations Research (Dynamic Programming)
- Control Theory

B. F. Skinner
R. E. Bellman
Our View of Reinforcement Learning

- Operations Research (Dynamic Programming)
- Control Theory
- Neuroscience
- Artificial Intelligence and Computer Science
- Psychology (Animal Behaviour)
- Reinforcement Learning

- B. F. Skinner
- D. P. Bertsekas
- W. Schultz
Our View of Reinforcement Learning

- R. E. Bellman
- B. F. Skinner
- D. P. Bertsekas
- W. Schultz
- R. S. Sutton

- Operations Research (Dynamic Programming)
- Control Theory
- Psychology (Animal Behaviour)
- Reinforcement Learning
- Neuroscience
- Artificial Intelligence and Computer Science
Reinforcement Learning: A Survey.
Resources

**Reinforcement Learning: A Survey.**

**Reinforcement Learning: An Introduction**

**Algorithms for Reinforcement Learning**
Resources

Reinforcement Learning: A Survey.

Reinforcement Learning: An Introduction

Algorithms for Reinforcement Learning

E-mail List: rl-list@googlegroups.com.
Resources

Reinforcement Learning: A Survey.

Reinforcement Learning: An Introduction

Algorithms for Reinforcement Learning

E-mail List: rl-list@googlegroups.com.

RL Competition: http://www.rl-competition.org/.
Today’s Class

1. Markov decision problems
2. Bellman’s (Optimality) Equations, planning and learning
3. Challenges
4. Summary
Today’s Class

1. Markov decision problems
2. Bellman’s (Optimality) Equations, planning and learning
3. Challenges
4. Summary
Markov Decision Problem

$S$: set of states.
$A$: set of actions.
$T$: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over $S$.
$R$: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
$\gamma$: discount factor. $0 \leq \gamma < 1$. 

Shivaram Kalyanakrishnan
Markov Decision Problem

**S**: set of states.
**A**: set of actions.
**T**: transition function. \( \forall s \in S, \forall a \in A, T(s, a) \) is a distribution over \( S \).
**R**: reward function. \( \forall s, s' \in S, \forall a \in A, R(s, a, s') \) is a finite real number.
**\( \gamma \)**: discount factor. \( 0 \leq \gamma < 1 \).

Trajectory over time: \( s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_t, a_t, r_{t+1}, s_{t+1}, \ldots \).
Markov Decision Problem

$S$: set of states.
$A$: set of actions.
$T$: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over $S$.
$R$: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
$\gamma$: discount factor. $0 \leq \gamma < 1$.

Trajectory over time: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_t, a_t, r_{t+1}, s_{t+1}, \ldots$

Value, or expected long-term reward, of state $s$ under policy $\pi$:

$$V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \text{ to } \infty | s_0 = s, a_i = \pi(s_i)]$$
Markov Decision Problem

$S$: set of states.
$A$: set of actions.
$T$: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over $S$.
$R$: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
$\gamma$: discount factor. $0 \leq \gamma < 1$.

Trajectory over time: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_t, a_t, r_{t+1}, s_{t+1}, \ldots$

Value, or expected long-term reward, of state $s$ under policy $\pi$:

$$V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \text{ to } \infty | s_0 = s, a_i = \pi(s_i)].$$

Objective: “Find $\pi$ such that $V^\pi(s)$ is maximal $\forall s \in S$.”
Examples

What are the agent and environment? What are $S$, $A$, $T$, and $R$?
Examples

What are the agent and environment? What are $S, A, T, \text{ and } R$?

Examples

What are the agent and environment? What are $S$, $A$, $T$, and $R$?

An Application of Reinforcement Learning to Aerobatic Helicopter Flight
Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. NIPS 2006.

Examples

What are the **agent** and **environment**? What are $S$, $A$, $T$, and $R$?

[Video$^3$ of Tetris]

An Application of Reinforcement Learning to Aerobatic Helicopter Flight
Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. NIPS 2006.

3. https://www.youtube.com/watch?v=khHzyghXseE
Illustration: MDPs as State Transition Diagrams

Notation: "transition probability, reward" marked on each arrow

States: \( s_1, s_2, s_3, \) and \( s_4 \).

Actions: Red (solid lines) and blue (dotted lines).

Transitions: Red action leads to same state with 20\% chance, to next-clockwise state with 80\% chance. Blue action leads to next-clockwise state or 2-removed-clockwise state with equal (50\%) probability.

Rewards: \( R(\ast, \ast, s_1) = 0, R(\ast, \ast, s_2) = 1, R(\ast, \ast, s_3) = -1, R(\ast, \ast, s_4) = 2 \).

Discount factor: \( \gamma = 0.9 \).
Today’s Class

1. Markov decision problems
2. Bellman’s (Optimality) Equations, planning and learning
3. Challenges
4. Summary
Bellman’s Equations

Recall that

$$V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s, a_i = \pi(s_i)].$$

Bellman’s Equations (\(\forall s \in S\)):

$$V^\pi(s) = \sum_{s' \in S} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V^\pi(s')].$$

\(V^\pi\) is called the value function of \(\pi\).
Bellman’s Equations

Recall that
\[ V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s, a_i = \pi(s_i)]. \]

Bellman’s Equations (\( \forall s \in S \)):

\[
V^\pi(s) = \sum_{s' \in S} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V^\pi(s')].
\]

\( V^\pi \) is called the value function of \( \pi \).

Define (\( \forall s \in S, \forall a \in A \)):

\[
Q^\pi(s, a) = \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma V^\pi(s')].
\]

\( Q^\pi \) is called the action value function of \( \pi \).

\[
V^\pi(s) = Q^\pi(s, \pi(s)).
\]
Bellman’s Equations

Recall that

\[ V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s, a_i = \pi(s_i)]. \]

Bellman’s Equations (\( \forall s \in S \)):

\[ V^\pi(s) = \sum_{s' \in S} T(s, \pi(s), s') \left[ R(s, \pi(s), s') + \gamma V^\pi(s') \right]. \]

\( V^\pi \) is called the value function of \( \pi \).

Define (\( \forall s \in S, \forall a \in A \)):

\[ Q^\pi(s, a) = \sum_{s' \in S} T(s, a, s') \left[ R(s, a, s') + \gamma V^\pi(s') \right]. \]

\( Q^\pi \) is called the action value function of \( \pi \).

\[ V^\pi(s) = Q^\pi(s, \pi(s)). \]

The variables in Bellman’s equation are the \( V^\pi(s) \). \( |S| \) linear equations in \( |S| \) unknowns.
Bellman’s Equations

Recall that
\[ V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s, a_i = \pi(s_i)]. \]

Bellman’s Equations (\(\forall s \in S\)):
\[ V^\pi(s) = \sum_{s' \in S} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V^\pi(s')]. \]

\(V^\pi\) is called the value function of \(\pi\).

Define (\(\forall s \in S, \forall a \in A\)):
\[ Q^\pi(s, a) = \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma V^\pi(s')]. \]

\(Q^\pi\) is called the action value function of \(\pi\).
\[ V^\pi(s) = Q^\pi(s, \pi(s)). \]

The variables in Bellman’s equation are the \(V^\pi(s)\). \(|S|\) linear equations in \(|S|\) unknowns.

Thus, given \(S, A, T, R, \gamma\), and a fixed policy \(\pi\), we can solve Bellman’s equations efficiently to obtain, \(\forall s \in S, \forall a \in A\), \(V^\pi(s)\) and \(Q^\pi(s, a)\).
Bellman’s Optimality Equations

Let $\Pi$ be the set of all policies. What is its cardinality?
Bellman’s Optimality Equations

Let \( \Pi \) be the set of all policies. What is its cardinality?

It can be shown that there exists a policy \( \pi^* \in \Pi \) such that

\[
\forall \pi \in \Pi \ \forall s \in S: V_{\pi^*}(s) \geq V_\pi(s).
\]

\( V_{\pi^*} \) is denoted \( V^* \), and \( Q_{\pi^*} \) is denoted \( Q^* \).

There could be multiple optimal policies \( \pi^* \), but \( V^* \) and \( Q^* \) are unique.
Bellman’s Optimality Equations

Let $\Pi$ be the set of all policies. What is its cardinality?

It can be shown that there exists a policy $\pi^* \in \Pi$ such that

$$\forall \pi \in \Pi \ \forall s \in S: V_{\pi^*}(s) \geq V_\pi(s).$$

$V_{\pi^*}$ is denoted $V^*$, and $Q_{\pi^*}$ is denoted $Q^*$.
There could be multiple optimal policies $\pi^*$, but $V^*$ and $Q^*$ are unique.

Bellman’s Optimality Equations ($\forall s \in S$):

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma V^*(s')].$$
Bellman’s Optimality Equations

Let $\Pi$ be the set of all policies. What is its cardinality?

It can be shown that there exists a policy $\pi^* \in \Pi$ such that

$\forall \pi \in \Pi \forall s \in S: V^{\pi^*}(s) \geq V^\pi(s).$

$V^{\pi^*}$ is denoted $V^*$, and $Q^{\pi^*}$ is denoted $Q^*$.
There could be multiple optimal policies $\pi^*$, but $V^*$ and $Q^*$ are unique.

Bellman’s Optimality Equations ($\forall s \in S$):

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

Planning problem:

Given $S, A, T, R, \gamma$, how can we find an optimal policy $\pi^*$? We need to be computationally efficient.
Bellman’s Optimality Equations

Let \( \Pi \) be the set of all policies. What is its cardinality?

It can be shown that there exists a policy \( \pi^* \in \Pi \) such that
\[
\forall \pi \in \Pi \ \forall s \in S: V_{\pi^*}(s) \geq V_{\pi}(s).
\]

\( V_{\pi^*} \) is denoted \( V^* \), and \( Q_{\pi^*} \) is denoted \( Q^* \).

There could be multiple optimal policies \( \pi^* \), but \( V^* \) and \( Q^* \) are unique.

Bellman’s Optimality Equations (\( \forall s \in S \)):
\[
V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma V^*(s')].
\]

Planning problem:

Given \( S, A, T, R, \gamma \), how can we find an optimal policy \( \pi^* \)? We need to be computationally efficient.

Learning problem:

Given \( S, A, \gamma \), and the facility to follow a trajectory by sampling from \( T \) and \( R \), how can we find an optimal policy \( \pi^* \)? We need to be sample-efficient.
Given $S, A, T, R, \gamma$, how can we find an optimal policy $\pi^*$?
Given $S$, $A$, $T$, $R$, $\gamma$, how can we find an optimal policy $\pi^*$?

**One way.** We can pose Bellman’s optimality equations as a linear program, solve for $V^*$, derive $Q^*$, and induce $\pi^*(s) = \text{argmax}_a Q^*(s, a)$. 
Given $S, A, T, R, \gamma$, how can we find an optimal policy $\pi^*$?

**One way.** We can pose Bellman’s optimality equations as a linear program, solve for $V^*$, derive $Q^*$, and induce $\pi^*(s) = \text{argmax}_a Q^*(s, a)$.

**Another way.** We can apply the policy iteration algorithm, which is typically more efficient in practice.

- Pick an initial policy $\pi$ arbitrarily.
- Compute $Q^\pi$ using Bellman’s equations.
- converged $\leftarrow$ false.
- Repeat
  - Set $\pi'$ as: $\forall s \in S, \pi'(s) = \text{argmax}_a Q^\pi(s, a)$ (break ties arbitrarily). **[Improvement]**
  - Compute $Q^{\pi'}$ using Bellman’s equations. **[Evaluation]**
  - If ($Q^{\pi'} = Q^\pi$), converged $\leftarrow$ true.
  - $\pi \leftarrow \pi'$, $Q^\pi \leftarrow Q^{\pi'}$.
- Until converged.
- Return $\pi$ (which is provably optimal).
Planning

Given $S, A, T, R, \gamma$, how can we find an optimal policy $\pi^*$?

**One way.** We can pose Bellman’s optimality equations as a linear program, solve for $V^*$, derive $Q^*$, and induce $\pi^*(s) = \arg\max_a Q^*(s, a)$.

**Another way.** We can apply the policy iteration algorithm, which is typically more efficient in practice.

- Pick an initial policy $\pi$ arbitrarily.
- Compute $Q^\pi$ using Bellman’s equations.
- converged ← false.
- Repeat
  - Set $\pi'$ as: $\forall s \in S$, $\pi'(s) = \arg\max_a Q^\pi(s, a)$ (break ties arbitrarily).[Improvement]
  - Compute $Q^{\pi'}$ using Bellman’s equations. [Evaluation]
  - If $(Q^{\pi'} = Q^\pi)$, converged ← true.
  - $\pi ← \pi'$, $Q^\pi ← Q^{\pi'}$.
- Until converged.
- Return $\pi$ (which is provably optimal).

**Other ways.** Value iteration and its various “mixtures” with policy iteration.
Learning

Given $S$, $A$, $\gamma$, and the facility to follow a trajectory by sampling from $T$ and $R$, how can we find an optimal policy $\pi^*$?
Given $S$, $A$, $\gamma$, and the facility to follow a trajectory by sampling from $T$ and $R$, how can we find an optimal policy $\pi^*$?

Various classes of learning methods exist. We will consider a simple one called Q-learning, which is a temporal difference learning algorithm.

- Let $Q$ be our “guess” of $Q^*$: for every state $s$ and action $a$, initialise $Q(s, a)$ arbitrarily. We will start in some state $s_0$.
- For $t = 0, 1, 2, \ldots$
  - Take an action $a_t$, chosen uniformly at random with probability $\epsilon$, and to be $\text{argmax}_a Q(s_t, a)$ with probability $1 - \epsilon$.
  - The environment will generate next state $s_{t+1}$ and reward $r_{t+1}$.
  - Update: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t (r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t))$.

[$\epsilon$: parameter for “$\epsilon$-greedy” exploration] [$\alpha_t$: learning rate] [$r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)$: temporal difference prediction error]
Given $S$, $A$, $\gamma$, and the facility to follow a trajectory by sampling from $T$ and $R$, how can we find an optimal policy $\pi^*$?

Various classes of learning methods exist. We will consider a simple one called **Q-learning**, which is a **temporal difference learning** algorithm.

- Let $Q$ be our “guess” of $Q^*$: for every state $s$ and action $a$, initialise $Q(s, a)$ arbitrarily. We will start in some state $s_0$.
- For $t = 0, 1, 2, \ldots$,
  - Take an action $a_t$, chosen uniformly at random with probability $\epsilon$, and to be $\text{argmax}_a Q(s_t, a)$ with probability $1 - \epsilon$.
  - The environment will generate next state $s_{t+1}$ and reward $r_{t+1}$.
  - Update: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t (r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t))$.

[$\epsilon$: parameter for “$\epsilon$-greedy” exploration] [$\alpha_t$: learning rate] [$r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)$: temporal difference prediction error]

For $\epsilon \in (0, 1]$ and $\alpha_t = \frac{1}{t}$, it can be proven that as $t \to \infty$, $Q \to Q^*$.

**Q-Learning**
Today’s Class

1. Markov decision problems
2. Bellman’s (Optimality) Equations, planning and learning
3. Challenges
4. Summary
Practical Implementation and Evaluation of Learning Algorithms
Practical Implementation and Evaluation of Learning Algorithms

Generalized Model Learning for Reinforcement Learning on a Humanoid Robot
Todd Hester, Michael Quinlan, and Peter Stone. ICRA 2010.

[Video\textsuperscript{1} of RL on a humanoid robot]

\textsuperscript{1} http://www.youtube.com/watch?v=mRpX9DFCdwI
Practical Implementation and Evaluation of Learning Algorithms

Generalized Model Learning for Reinforcement Learning on a Humanoid Robot
Todd Hester, Michael Quinlan, and Peter Stone. ICRA 2010.

[Video¹ of RL on a humanoid robot]

¹ http://www.youtube.com/watch?v=mRpX9DFCdwI
Research Challenges

- Exploration
- Generalisation (over states and actions)
- State aliasing (partial observability)
- Multiple agents, nonstationary rewards and transitions
- Abstraction (over states and over time)
- Proofs of convergence, sample-complexity bounds
Research Challenges

- Exploration
- Generalisation (over states and actions)
- State aliasing (partial observability)
- Multiple agents, nonstationary rewards and transitions
- Abstraction (over states and over time)
- Proofs of convergence, sample-complexity bounds

My thesis question:

“How well do different learning methods for sequential decision making perform in the presence of state aliasing and generalization; can we develop methods that are both sample-efficient and capable of achieving high asymptotic performance in their presence?”

Learning Methods for Sequential Decision Making with Imperfect Representations
# Practice \(\rightarrow\) Imperfect Representations

<table>
<thead>
<tr>
<th>Task</th>
<th>State Aliasing</th>
<th>State Space</th>
<th>Policy Representation (Number of features)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backgammon (T1992)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (198)</td>
</tr>
<tr>
<td>Job-shop scheduling (ZD1995)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (20)</td>
</tr>
<tr>
<td>Tetris (BT1906)</td>
<td>Absent</td>
<td>Continuous</td>
<td>Linear (22)</td>
</tr>
<tr>
<td>Elevator dispatching (CB1996)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (46)</td>
</tr>
<tr>
<td>Acrobot control (S1996)</td>
<td>Absent</td>
<td>Continuous</td>
<td>Tile coding (4)</td>
</tr>
<tr>
<td>Dynamic channel allocation (SB1997)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (100’s)</td>
</tr>
<tr>
<td>Active guidance of finless rocket (GM2003)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (14)</td>
</tr>
<tr>
<td>Robot sensing strategy (KF2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (36)</td>
</tr>
<tr>
<td>Helicopter control (NKJS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (10)</td>
</tr>
<tr>
<td>Dynamic bipedal locomotion (TZS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Feedback control policy (2)</td>
</tr>
<tr>
<td>Robot soccer keepaway (SSK2005)</td>
<td>Present</td>
<td>Continuous</td>
<td>Tile coding (13)</td>
</tr>
<tr>
<td>Robot obstacle negotiation (LSYSN2006)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (10)</td>
</tr>
<tr>
<td>Optimized trade execution (NFK2007)</td>
<td>Present</td>
<td>Discrete</td>
<td>Tabular (2-5)</td>
</tr>
<tr>
<td>Blimp control (RPHB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Gaussian Process (2)</td>
</tr>
<tr>
<td>9 × 9 Go (SSM2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear ((\approx 1.5) million)</td>
</tr>
<tr>
<td>Ms. Pac-Man (SL2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Rule list (10)</td>
</tr>
<tr>
<td>Autonomic resource allocation (TJDB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (2)</td>
</tr>
<tr>
<td>General game playing (FB2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tabular (part of state space)</td>
</tr>
<tr>
<td>Soccer opponent “hasling” (GRT2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (9)</td>
</tr>
<tr>
<td>Adaptive epilepsy treatment (GVAP2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Extremely rand. trees (114)</td>
</tr>
<tr>
<td>Computer memory scheduling (IMMC2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tile coding (6)</td>
</tr>
<tr>
<td>Motor skills (PS2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Motor primitive coeff. (100’s)</td>
</tr>
<tr>
<td>Combustion Control (HNGK2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Parameterized policy (2-3)</td>
</tr>
</tbody>
</table>
### Practice → Imperfect Representations

<table>
<thead>
<tr>
<th>Task</th>
<th>State Aliasing</th>
<th>State Space</th>
<th>Policy Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><strong>Aliasing</strong></td>
<td><strong>Space</strong></td>
<td><strong>(Number of features)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backgammon (T1992)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (198)</td>
</tr>
<tr>
<td>Job-shop scheduling (ZD1995)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (20)</td>
</tr>
<tr>
<td>Tetris (BT1996)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (22)</td>
</tr>
<tr>
<td>Elevator dispatching (CB1996)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (46)</td>
</tr>
<tr>
<td>Acrobot control (S1996)</td>
<td>Absent</td>
<td>Continuous</td>
<td>Tile coding (4)</td>
</tr>
<tr>
<td>Dynamic channel allocation (SB1997)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (100’s)</td>
</tr>
<tr>
<td>Active guidance of finless rocket (GM2003)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (14)</td>
</tr>
<tr>
<td>Robot sensing strategy (KF2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (36)</td>
</tr>
<tr>
<td>Helicopter control (NKJS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (10)</td>
</tr>
<tr>
<td>Dynamic bipedal locomotion (TZS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Feedback control policy (2)</td>
</tr>
<tr>
<td>Robot soccer keepaway (SSK2005)</td>
<td>Present</td>
<td>Continuous</td>
<td>Tile coding (13)</td>
</tr>
<tr>
<td>Robot obstacle negotiation (LSYSN2006)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (10)</td>
</tr>
<tr>
<td>Optimized trade execution (NFK2007)</td>
<td>Present</td>
<td>Discrete</td>
<td>Tabular (2-5)</td>
</tr>
<tr>
<td>Blimp control (RPHB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Gaussian Process (2)</td>
</tr>
<tr>
<td>9 × 9 Go (SSM2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (≈1.5 million)</td>
</tr>
<tr>
<td>Ms. Pac-Man (SL2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Rule list (10)</td>
</tr>
<tr>
<td>Autonomic resource allocation (TJDB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (2)</td>
</tr>
<tr>
<td>General game playing (FB2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tabular (part of state space)</td>
</tr>
<tr>
<td>Soccer opponent “hassling” (GRT2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (9)</td>
</tr>
<tr>
<td>Adaptive epilepsy treatment (GVAP2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Extremely rand. trees (114)</td>
</tr>
<tr>
<td>Computer memory scheduling (IMMC2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tile coding (6)</td>
</tr>
<tr>
<td>Motor skills (PS2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Motor primitive coeff. (100’s)</td>
</tr>
<tr>
<td>Combustion Control (HNGK2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Parameterized policy (2-3)</td>
</tr>
</tbody>
</table>
## Practice → Imperfect Representations

<table>
<thead>
<tr>
<th>Task</th>
<th>State Aliasing</th>
<th>State Space</th>
<th>Policy Representation (Number of features)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backgammon (T1992)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (198)</td>
</tr>
<tr>
<td>Job-shop scheduling (ZD1995)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (20)</td>
</tr>
<tr>
<td>Tetris (BT1906)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (22)</td>
</tr>
<tr>
<td>Elevator dispatching (CB1996)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (46)</td>
</tr>
<tr>
<td>Acrobot control (S1996)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (100’s)</td>
</tr>
<tr>
<td>Dynamic channel allocation (SB1997)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (100’s)</td>
</tr>
<tr>
<td>Active guidance of finless rocket (GM2003)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (14)</td>
</tr>
<tr>
<td>Fast quadrupedal locomotion (KS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (36)</td>
</tr>
<tr>
<td>Robot sensing strategy (KF2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (10)</td>
</tr>
<tr>
<td>Helicopter control (NKJS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Feedback control policy (2)</td>
</tr>
<tr>
<td>Dynamic bipedal locomotion (TZS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (10)</td>
</tr>
<tr>
<td>Robot soccer keepaway (SSK2005)</td>
<td>Present</td>
<td>Continuous</td>
<td>Tile coding (13)</td>
</tr>
<tr>
<td>Robot obstacle negotiation (LSYN2006)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (10)</td>
</tr>
<tr>
<td>Optimized trade execution (NFK2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Gaussian Process (2)</td>
</tr>
<tr>
<td>Blimp control (RPHB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (≈1.5 million)</td>
</tr>
<tr>
<td>9 × 9 Go (SSM2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Rule list (10)</td>
</tr>
<tr>
<td>Ms. Pac-Man (SL2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (2)</td>
</tr>
<tr>
<td>Autonomic resource allocation (TJDB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (9)</td>
</tr>
<tr>
<td>General game playing (FB2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tabular (part of state space)</td>
</tr>
<tr>
<td>Soccer opponent “hassling” (GRT2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Extremely rand. trees (114)</td>
</tr>
<tr>
<td>Adaptive epilepsy treatment (GVAP2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Tile coding (6)</td>
</tr>
<tr>
<td>Computer memory scheduling (IMMC2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Motor primitive coeff. (100’s)</td>
</tr>
<tr>
<td>Motor skills (PS2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Parameterized policy (2-3)</td>
</tr>
<tr>
<td>Combustion Control (HNGK2009)</td>
<td>Present</td>
<td>Continuous</td>
<td></td>
</tr>
</tbody>
</table>
Practice ➞ Imperfect Representations

<table>
<thead>
<tr>
<th>Task</th>
<th>State Aliasing</th>
<th>State Space</th>
<th>Policy Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backgammon (T1992)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (198)</td>
</tr>
<tr>
<td>Job-shop scheduling (ZD1995)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Neural network (20)</td>
</tr>
<tr>
<td>Tetris (BT1906)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (22)</td>
</tr>
<tr>
<td>Elevator dispatching (CB1996)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (46)</td>
</tr>
<tr>
<td>Acrobot control (S1996)</td>
<td>Absent</td>
<td>Continuous</td>
<td>Linear (100’s)</td>
</tr>
<tr>
<td>Dynamic channel allocation (SB1997)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (100’s)</td>
</tr>
<tr>
<td>Active guidance of finless rocket (GM2003)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (14)</td>
</tr>
<tr>
<td>Fast quadrupedal locomotion (KS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (36)</td>
</tr>
<tr>
<td>Robot sensing strategy (KF2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (10)</td>
</tr>
<tr>
<td>Helicopter control (NKJS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (10)</td>
</tr>
<tr>
<td>Dynamic bipedal locomotion (TZS2004)</td>
<td>Present</td>
<td>Continuous</td>
<td>Feedback control policy (2)</td>
</tr>
<tr>
<td>Robot soccer keepaway (SSK2005)</td>
<td>Present</td>
<td>Continuous</td>
<td>Tile coding (13)</td>
</tr>
<tr>
<td>Robot obstacle negotiation (LSASN2006)</td>
<td>Present</td>
<td>Continuous</td>
<td>Linear (10)</td>
</tr>
<tr>
<td>Optimized trade execution (NFK2007)</td>
<td>Present</td>
<td>Discrete</td>
<td>Tabular (2-5)</td>
</tr>
<tr>
<td>Blimp control (RPHB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Gaussian Process (2)</td>
</tr>
<tr>
<td>9 × 9 Go (SSM2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Linear (≈1.5 million)</td>
</tr>
<tr>
<td>Ms. Pac-Man (SL2007)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Rule list (10)</td>
</tr>
<tr>
<td>Autonomic resource allocation (TJDB2007)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (2)</td>
</tr>
<tr>
<td>General game playing (FB2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tabular (part of state space)</td>
</tr>
<tr>
<td>Soccer opponent “hassling” (GRT2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Neural network (9)</td>
</tr>
<tr>
<td>Adaptive epilepsy treatment (GVAP2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Extremely rand. trees (114)</td>
</tr>
<tr>
<td>Computer memory scheduling (IMMC2008)</td>
<td>Absent</td>
<td>Discrete</td>
<td>Tile coding (6)</td>
</tr>
<tr>
<td>Motor skills (PS2008)</td>
<td>Present</td>
<td>Continuous</td>
<td>Motor primitive coeff. (100’s)</td>
</tr>
<tr>
<td>Combustion Control (HNGK2009)</td>
<td>Present</td>
<td>Continuous</td>
<td>Parameterized policy (2-3)</td>
</tr>
</tbody>
</table>

Perfect representations (fully observable, enumerable states) are impractical.
Today’s Class

1. Markov decision problems
2. Bellman’s (Optimality) Equations, planning and learning
3. Challenges
4. Summary
Learning by trial and error to perform sequential decision making.
Learning by trial and error to perform sequential decision making.

Given an MDP \((S, A, T, R, \gamma)\), we have to find a policy \(\pi : S \rightarrow A\) that yields high expected long-term reward from states.
Learning by trial and error to perform sequential decision making.

Given an MDP \((S, A, T, R, \gamma)\), we have to find a policy \(\pi : S \rightarrow A\) that yields high expected long-term reward from states.

An optimal value function \(V^*\) exists, and it induces an optimal policy \(\pi^*\) (several optimal policies might exist).
Learning by trial and error to perform sequential decision making.

Given an MDP \((S, A, T, R, \gamma)\), we have to find a policy \(\pi : S \rightarrow A\) that yields high expected long-term reward from states.

An optimal value function \(V^*\) exists, and it induces an optimal policy \(\pi^*\) (several optimal policies might exist).

Under planning, we are given \(S, A, T, R, \) and \(\gamma\). We may compute \(V^*\) and \(\pi^*\) using a dynamic programming algorithm such as policy iteration.
Learning by trial and error to perform sequential decision making.

Given an MDP \( (S, A, T, R, \gamma) \), we have to find a policy \( \pi : S \rightarrow A \) that yields high expected long-term reward from states.

An optimal value function \( V^* \) exists, and it induces an optimal policy \( \pi^* \) (several optimal policies might exist).

Under planning, we are given \( S, A, T, R \), and \( \gamma \). We may compute \( V^* \) and \( \pi^* \) using a dynamic programming algorithm such as policy iteration.

In the learning context, we are given \( S, A \), and \( \gamma \): we may sample \( T \) and \( R \) in a sequential manner. We can still converge to \( V^* \) and \( \pi^* \) by applying a temporal difference learning method such as Q-learning.
Learning by trial and error to perform sequential decision making.

Given an MDP \((S, A, T, R, \gamma)\), we have to find a policy \(\pi : S \rightarrow A\) that yields high expected long-term reward from states.

An optimal value function \(V^*\) exists, and it induces an optimal policy \(\pi^*\) (several optimal policies might exist).

Under planning, we are given \(S, A, T, R,\) and \(\gamma\). We may compute \(V^*\) and \(\pi^*\) using a dynamic programming algorithm such as policy iteration.

In the learning context, we are given \(S, A,\) and \(\gamma\): we may sample \(T\) and \(R\) in a sequential manner. We can still converge to \(V^*\) and \(\pi^*\) by applying a temporal difference learning method such as Q-learning.

Theory \(\neq\) Practice! In particular, convergence and optimality are difficult to achieve when state spaces are large, and when state aliasing exists.
Summary

- Learning by trial and error to perform sequential decision making.
- Given an MDP \((S, A, T, R, \gamma)\), we have to find a policy \(\pi : S \rightarrow A\) that yields high expected long-term reward from states.
- An optimal value function \(V^*\) exists, and it induces an optimal policy \(\pi^*\) (several optimal policies might exist).
- Under planning, we are given \(S, A, T, R, \gamma\). We may compute \(V^*\) and \(\pi^*\) using a dynamic programming algorithm such as policy iteration.
- In the learning context, we are given \(S, A, \gamma\): we may sample \(T\) and \(R\) in a sequential manner. We can still converge to \(V^*\) and \(\pi^*\) by applying a temporal difference learning method such as Q-learning.
- Theory \(\neq\) Practice! In particular, convergence and optimality are difficult to achieve when state spaces are large, and when state aliasing exists.

Thank you!
References


