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?

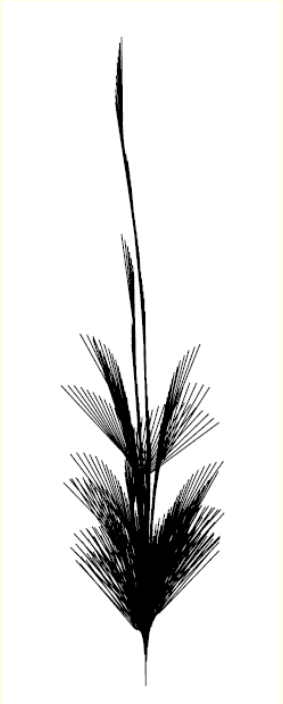
What is Reinforcement Learning?

[Video[1] of little girl learning to ride bicycle]

1. https://www.youtube.com/watch?v=Qv43pKlVZXk

What is Reinforcement Learning?

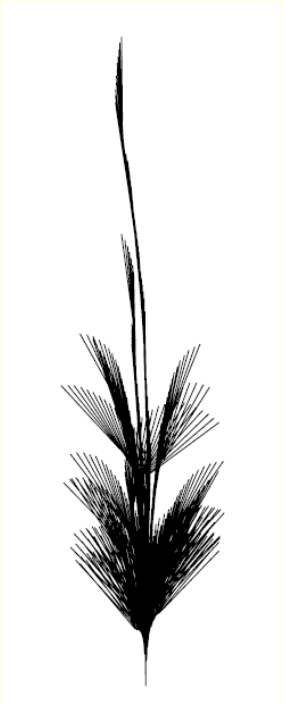
[Video[1] of little girl learning to ride bicycle]



1. https://www.youtube.com/watch?v=Qv43pKlVZXk

What is Reinforcement Learning?

[Video[1] of little girl learning to ride bicycle]

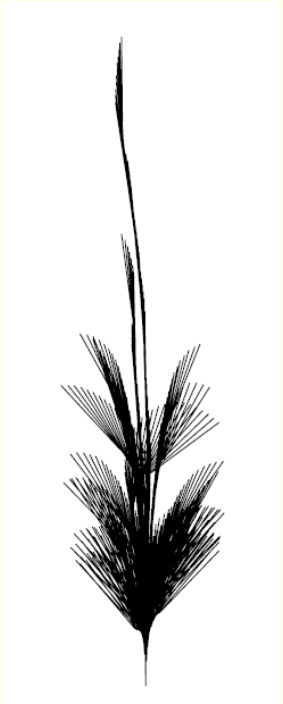


Learning to Drive a Bicycle using Reinforcement Learning and Shaping
Jette Randløv and Preben Alstrøm. ICML 1998.

1. https://www.youtube.com/watch?v=Qv43pKlVZXk

What is Reinforcement Learning?

[Video[1] of little girl learning to ride bicycle]

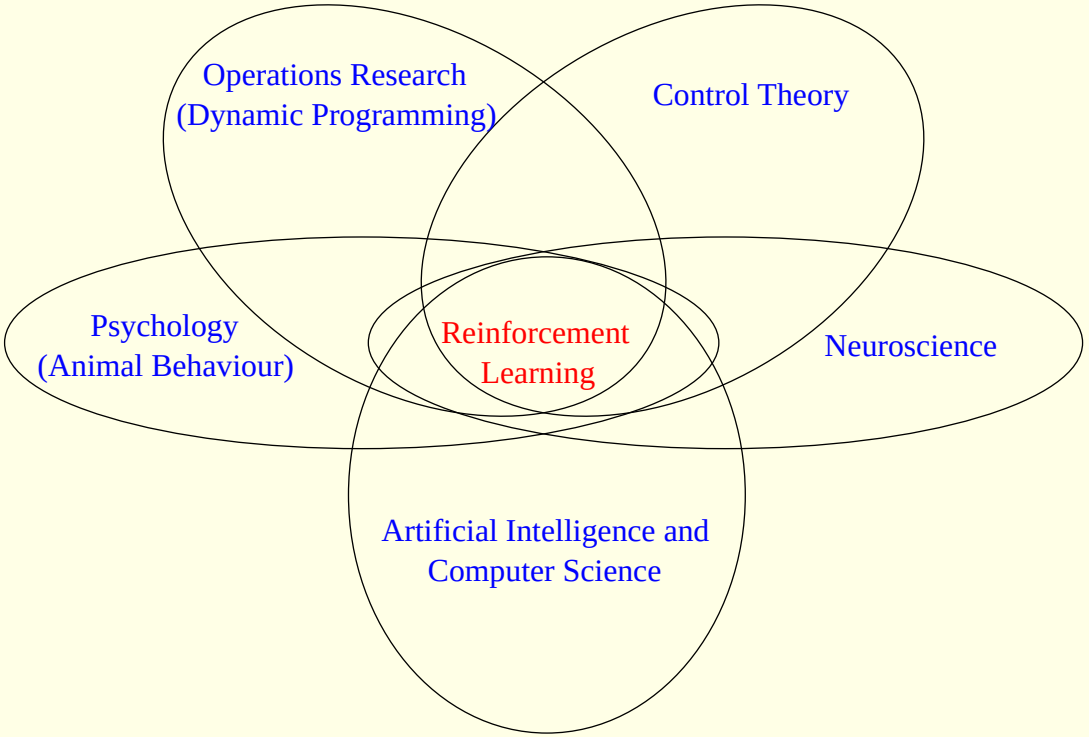


Learning to Drive a Bicycle using Reinforcement Learning and Shaping
Jette Randløv and Preben Alstrøm. ICML 1998.

Learning by trial and error to perform *sequential* decision making.

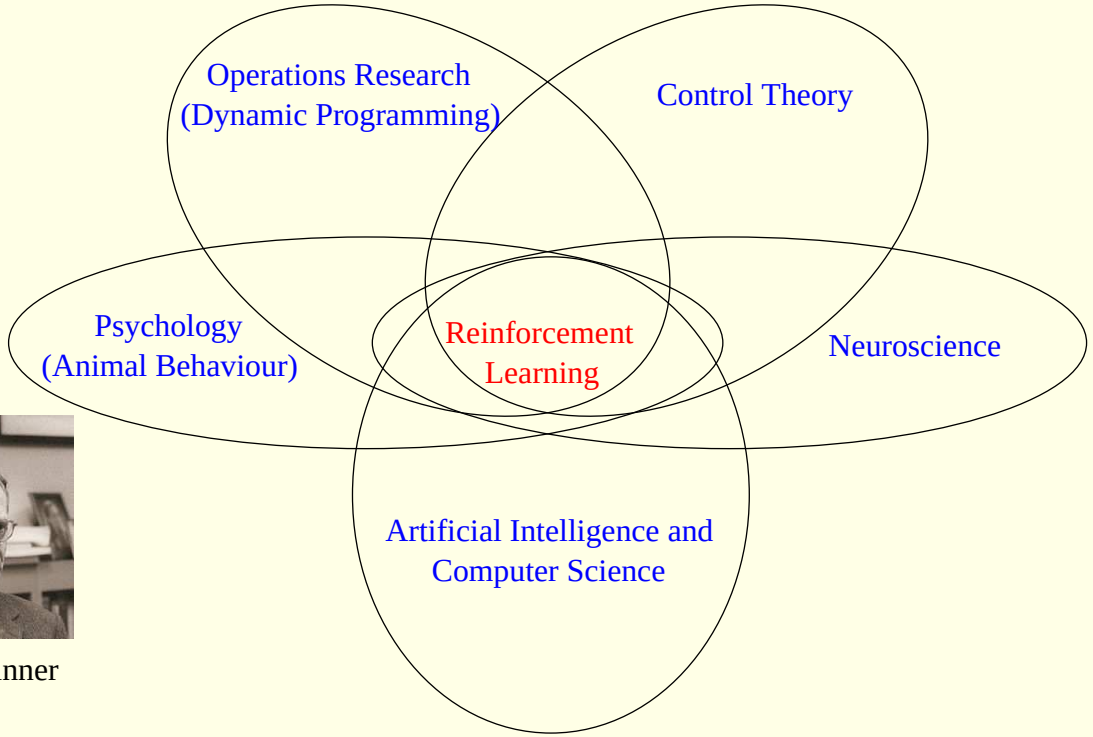
1. https://www.youtube.com/watch?v=Qv43pKlVZXk

Our View of Reinforcement Learning



Our View of Reinforcement Learning

B. F. Skinner



Our View of Reinforcement Learning

R. E. Bellman

B. F. Skinner

Our View of Reinforcement Learning

R. E. Bellman

D. P. Bertsekas

B. F. Skinner

Our View of Reinforcement Learning

R. E. Bellman

D. P. Bertsekas

B. F. Skinner

W. Schultz

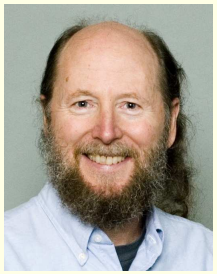
Our View of Reinforcement Learning


R. E. Bellman


D. P. Bertsekas


B. F. Skinner

W. Schultz

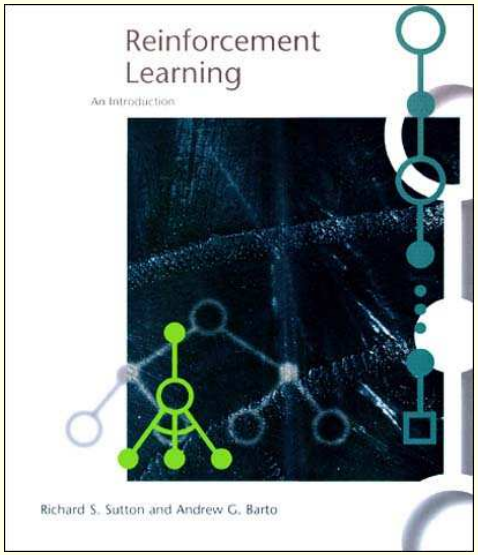

R. S. Sutton

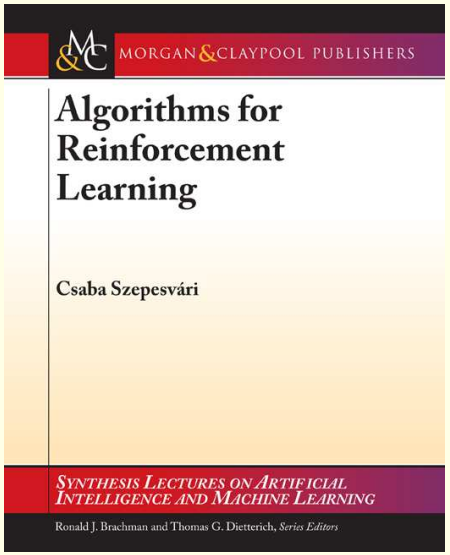
Resources

Reinforcement Learning: A Survey.
Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. JAIR 1996.

Resources

Reinforcement Learning: A Survey.
Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. JAIR 1996.



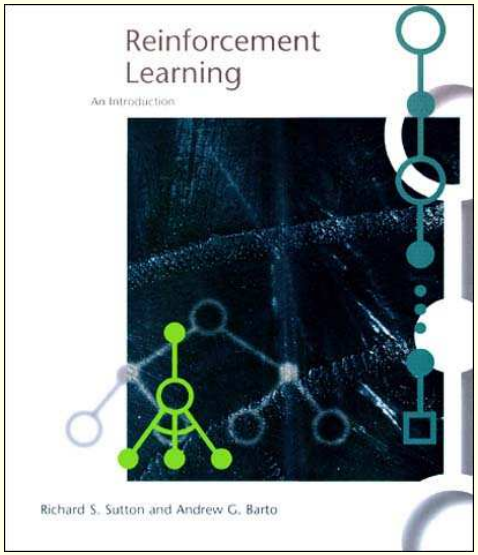


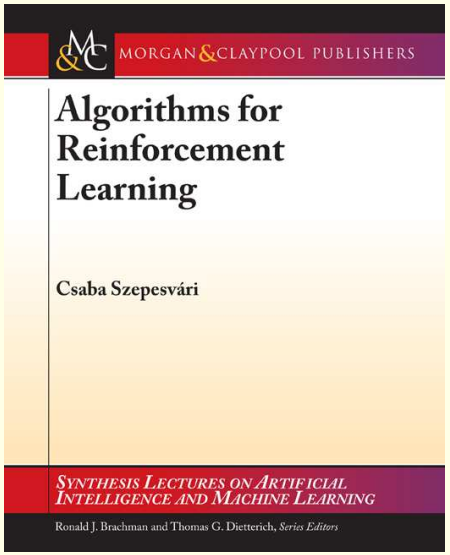
Reinforcement Learning: An Introduction
Richard S. Sutton and Andrew G. Barto. MIT Press, 1998.

Algorithms for Reinforcement Learning
Csaba Szepesvári. Morgan & Claypool, 2010.

Resources

Reinforcement Learning: A Survey.
Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. JAIR 1996.





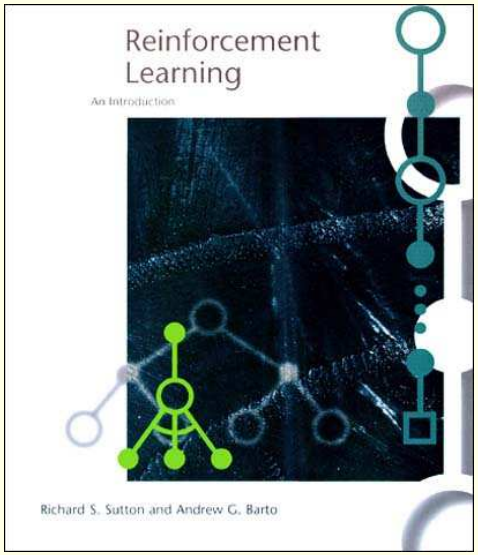
Reinforcement Learning: An Introduction
Richard S. Sutton and Andrew G. Barto. MIT Press, 1998.

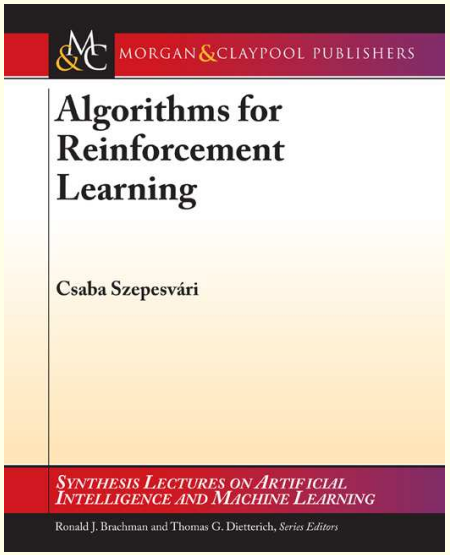
Algorithms for Reinforcement Learning
Csaba Szepesvári. Morgan & Claypool, 2010.

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

Resources

Reinforcement Learning: A Survey.
Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. JAIR 1996.





Reinforcement Learning: An Introduction
Richard S. Sutton and Andrew G. Barto. MIT Press, 1998.

Algorithms for Reinforcement Learning
Csaba Szepesvári. Morgan & Claypool, 2010.

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

RL Competition: `http://www.rl-competition.org/`.

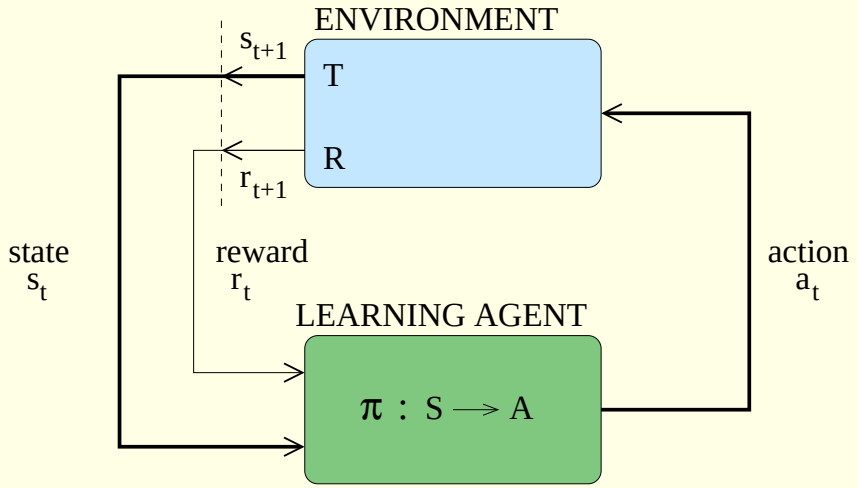
Today's Class

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

Today's Class

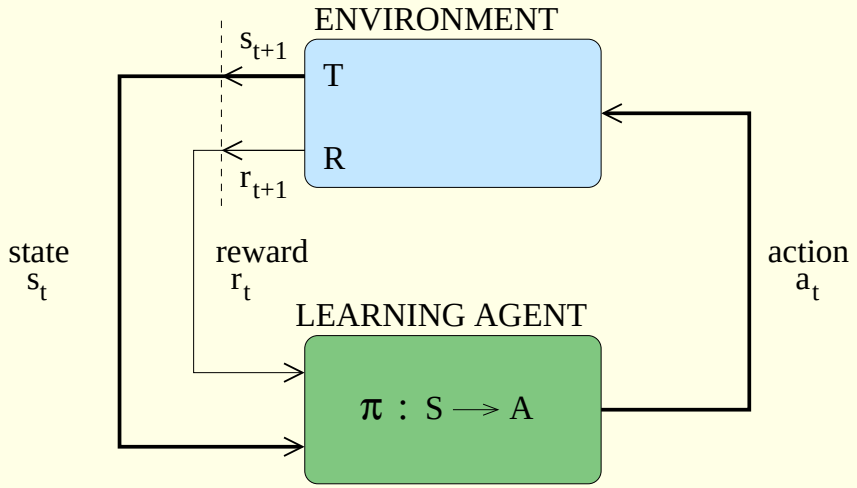
1. Markov decision problems
2. Bellman's (Optimality) Equations, planning and learning
3. Challenges
4. Summary

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.
γ: discount factor. $0 \leq \gamma < 1$.

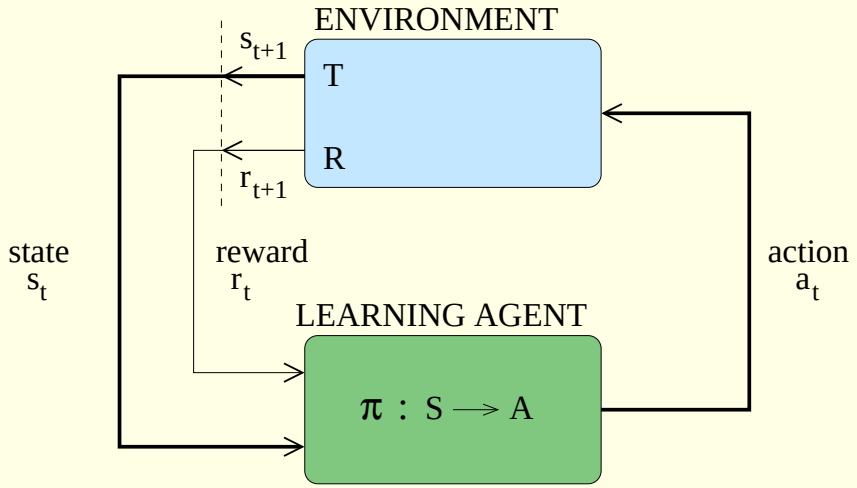
Markov Decision Problem



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.
γ: 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

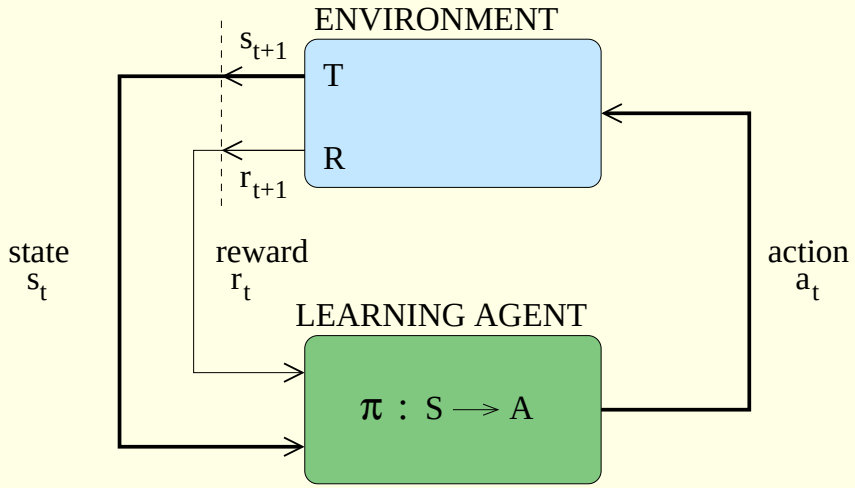


S: set of states.
A: set of actions.
T: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over S.
R: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
γ: discount factor. $0 \leq \gamma < 1$.

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 π:

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

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.
γ: 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 π:

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

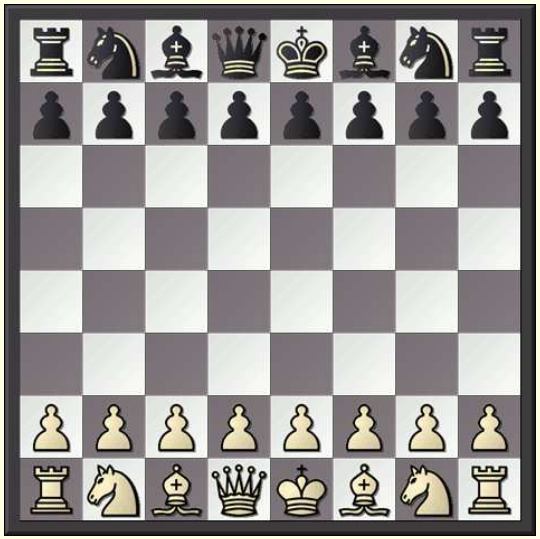
Objective: "Find π such that $V^\pi(s)$ is maximal $\forall s \in S$."

Examples

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

Examples

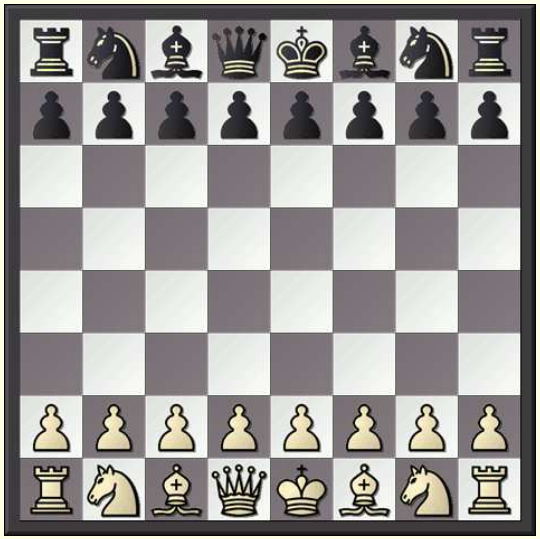
What are the agent and environment? What are S, A, T, and R?



1. http://www.chess-game-strategies.com/images/kqa_chessboard_large-picture_2d.gif

Examples

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

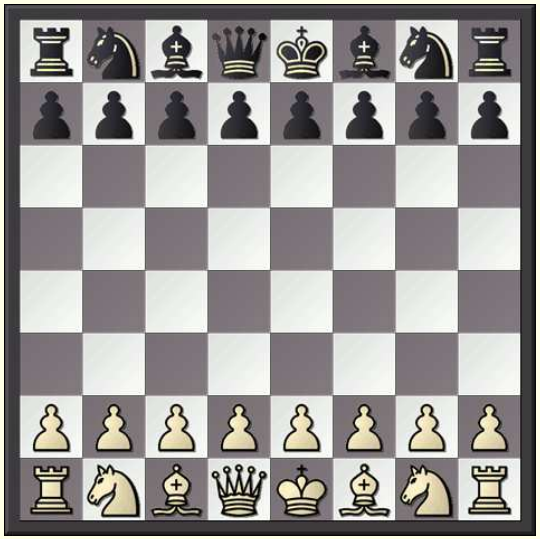


An Application of Reinforcement Learning to Aerobatic Helicopter Flight
Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. NIPS 2006.

1. http://www.chess-game-strategies.com/images/kqa_chessboard_large-picture_2d.gif
2. http://www.aviationspectator.com/files/images/SH-3-Sea-King-helicopter-191.preview.jpg

Examples

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

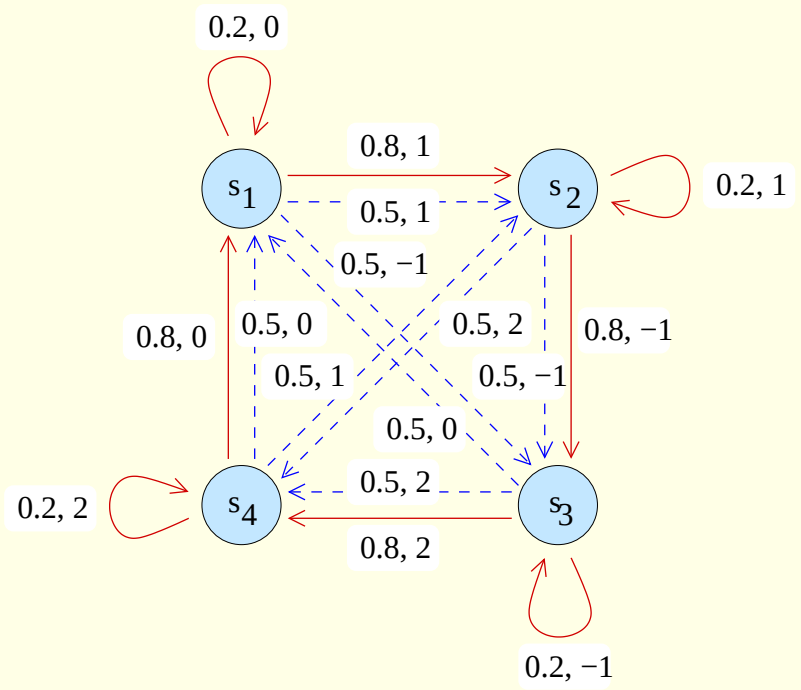


[Video[3] of Tetris]

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

1. http://www.chess-game-strategies.com/images/kqa_chessboard_large-picture_2d.gif
2. http://www.aviationspectator.com/files/images/SH-3-Sea-King-helicopter-191.preview.jpg
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: $R(*, *, s_1) = 0$, $R(*, *, s_2) = 1$, $R(*, *, s_3) = -1$, $R(*, *, s_4) = 2$.

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 ($\forall s \in S$):

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

V^{π} is called the value function of π.

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 ($\forall s \in S$):

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

V^{π} is called the value function of π.

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^{π} is called the action value function of π.

$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$.

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 ($\forall s \in S$):

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

V^{π} is called the value function of π.

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^{π} is called the action value function of π.

$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$.

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 ($\forall s \in S$):

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

V^{π} is called the value function of π.

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^{π} is called the action value function of π.

$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$.

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

Thus, given S, A, T, R, γ, and a fixed policy π, 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 Π be the set of all policies. What is its cardinality?

Bellman's Optimality Equations

Let Π 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^{π^*} is denoted V^*, and Q^{π^*} is denoted Q^*.

There could be multiple optimal policies π^*, but V^* and Q^* are unique.

Bellman's Optimality Equations

Let Π 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^{π^*} is denoted V^*, and Q^{π^*} is denoted Q^*.

There could be multiple optimal policies π^*, but V^* and Q^* are unique.

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

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

Bellman's Optimality Equations

Let Π 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^{π^*} is denoted V^*, and Q^{π^*} is denoted Q^*.

There could be multiple optimal policies π^*, but V^* and Q^* are unique.

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

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

Planning problem:

Given S, A, T, R, γ, how can we find an optimal policy π^*? We need to be computationally efficient.

Bellman's Optimality Equations

Let Π 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^{π^*} is denoted V^*, and Q^{π^*} is denoted Q^*.

There could be multiple optimal policies π^*, but V^* and Q^* are unique.

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

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

Planning problem:

Given S, A, T, R, γ, how can we find an optimal policy π^*? We need to be computationally efficient.

Learning problem:

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

Planning

Given S, A, T, R, γ, how can we find an optimal policy π^*?

Planning

Given S, A, T, R, γ, how can we find an optimal policy π^*?

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

Planning

Given S, A, T, R, γ, how can we find an optimal policy π^*?

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 π arbitrarily.
- Compute Q^π using Bellman's equations.
- converged $\leftarrow$ `false`.
- Repeat
  - Set π' 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 π (which is provably optimal).

Planning

Given S, A, T, R, γ, how can we find an optimal policy π^*?

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 π arbitrarily.
- Compute Q^π using Bellman's equations.
- converged $\leftarrow$ `false`.
- Repeat
  - Set π' 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 π (which is provably optimal).

Other ways. Value iteration and its various "mixtures" with policy iteration.

Learning

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

Learning

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

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 ϵ, 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: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t))$.

  [ϵ: parameter for "ϵ-greedy" exploration] [α_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]

Learning

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

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 ϵ, 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: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t))$.

  [ϵ: parameter for "ϵ-greedy" exploration] [α_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

Christopher J. C. H. Watkins and Peter Dayan. Machine Learning, 1992.

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[1] of RL on a humanoid robot]

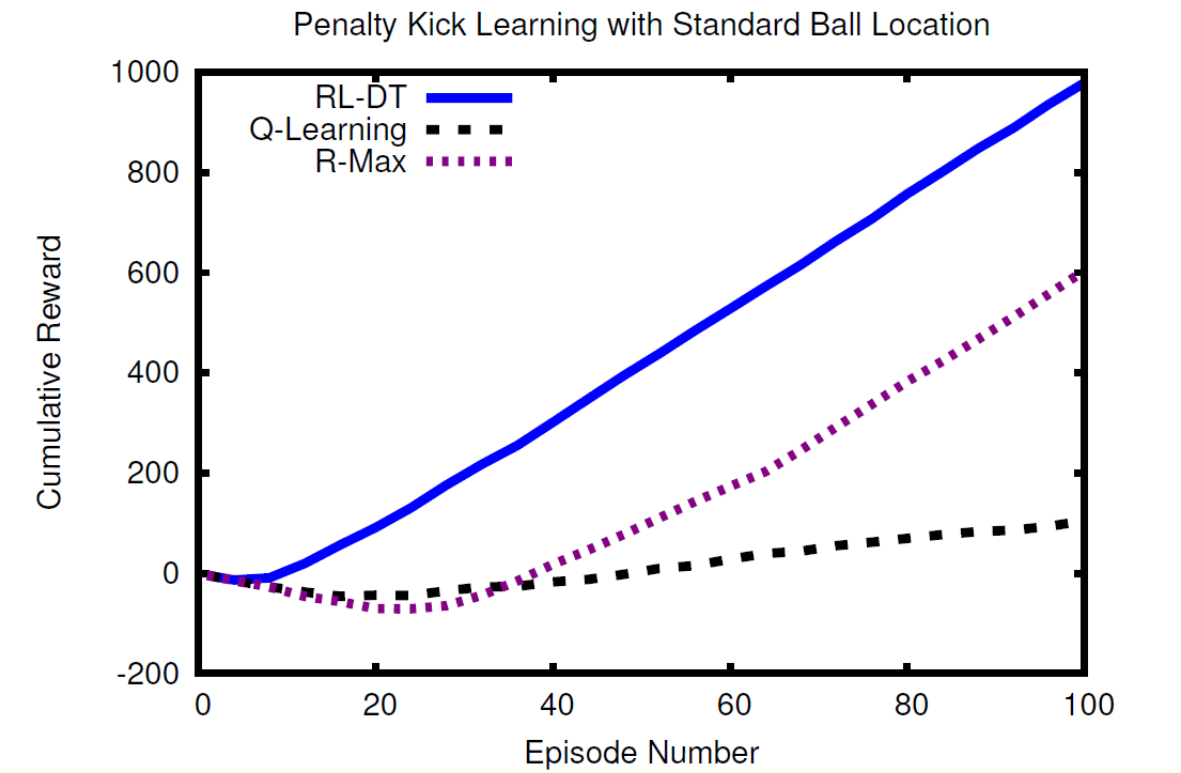
1. http://www.youtube.com/watch?v=mRpX9DFCdwI

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.

[Video[1] 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
Shivaram Kalyanakrishnan. Ph.D. dissertation, University of Texas at Austin, 2011.

Practice $\Longrightarrow$ Imperfect Representations

Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network (46)
Acrobot control (S1996)	Absent	Continuous	Tile coding (4)
Dynamic channel allocation (SB1997)	Absent	Discrete	Linear (100's)
Active guidance of finless rocket (GM2003)	Present	Continuous	Neural network (14)
Fast quadrupedal locomotion (KS2004)	Present	Continuous	Parameterized policy (12)
Robot sensing strategy (KF2004)	Present	Continuous	Linear (36)
Helicopter control (NKJS2004)	Present	Continuous	Neural network (10)
Dynamic bipedal locomotion (TZS2004)	Present	Continuous	Feedback control policy (2)
Adaptive job routing/scheduling (WS2004)	Present	Discrete	Tabular (4)
Robot soccer keepaway (SSK2005)	Present	Continuous	Tile coding (13)
Robot obstacle negotiation (LSYSN2006)	Present	Continuous	Linear (10)
Optimized trade execution (NFK2007)	Present	Discrete	Tabular (2-5)
Blimp control (RPHB2007)	Present	Continuous	Gaussian Process (2)
9 $\times$ 9 Go (SSM2007)	Absent	Discrete	Linear ($\approx$1.5 million)
Ms. Pac-Man (SL2007)	Absent	Discrete	Rule list (10)
Autonomic resource allocation (TJDB2007)	Present	Continuous	Neural network (2)
General game playing (FB2008)	Absent	Discrete	Tabular (part of state space)
Soccer opponent "hassling" (GRT2009)	Present	Continuous	Neural network (9)
Adaptive epilepsy treatment (GVAP2008)	Present	Continuous	Extremely rand. trees (114)
Computer memory scheduling (IMMC2008)	Absent	Discrete	Tile coding (6)
Motor skills (PS2008)	Present	Continuous	Motor primitive coeff. (100's)
Combustion Control (HNGK2009)	Present	Continuous	Parameterized policy (2-3)

Practice $\Longrightarrow$ Imperfect Representations

Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network (46)
Acrobot control (S1996)	Absent	Continuous	Tile coding (4)
Dynamic channel allocation (SB1997)	Absent	Discrete	Linear (100's)
Active guidance of finless rocket (GM2003)	Present	Continuous	Neural network (14)
Fast quadrupedal locomotion (KS2004)	Present	Continuous	Parameterized policy (12)
Robot sensing strategy (KF2004)	Present	Continuous	Linear (36)
Helicopter control (NKJS2004)	Present	Continuous	Neural network (10)
Dynamic bipedal locomotion (TZS2004)	Present	Continuous	Feedback control policy (2)
Adaptive job routing/scheduling (WS2004)	Present	Discrete	Tabular (4)
Robot soccer keepaway (SSK2005)	Present	Continuous	Tile coding (13)
Robot obstacle negotiation (LSYSN2006)	Present	Continuous	Linear (10)
Optimized trade execution (NFK2007)	Present	Discrete	Tabular (2-5)
Blimp control (RPHB2007)	Present	Continuous	Gaussian Process (2)
9 $\times$ 9 Go (SSM2007)	Absent	Discrete	Linear ($\approx$1.5 million)
Ms. Pac-Man (SL2007)	Absent	Discrete	Rule list (10)
Autonomic resource allocation (TJDB2007)	Present	Continuous	Neural network (2)
General game playing (FB2008)	Absent	Discrete	Tabular (part of state space)
Soccer opponent "hassling" (GRT2009)	Present	Continuous	Neural network (9)
Adaptive epilepsy treatment (GVAP2008)	Present	Continuous	Extremely rand. trees (114)
Computer memory scheduling (IMMC2008)	Absent	Discrete	Tile coding (6)
Motor skills (PS2008)	Present	Continuous	Motor primitive coeff. (100's)
Combustion Control (HNGK2009)	Present	Continuous	Parameterized policy (2-3)

Practice $\Longrightarrow$ Imperfect Representations

Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network (46)
Acrobot control (S1996)	Absent	Continuous	Tile coding (4)
Dynamic channel allocation (SB1997)	Absent	Discrete	Linear (100's)
Active guidance of finless rocket (GM2003)	Present	Continuous	Neural network (14)
Fast quadrupedal locomotion (KS2004)	Present	Continuous	Parameterized policy (12)
Robot sensing strategy (KF2004)	Present	Continuous	Linear (36)
Helicopter control (NKJS2004)	Present	Continuous	Neural network (10)
Dynamic bipedal locomotion (TZS2004)	Present	Continuous	Feedback control policy (2)
Adaptive job routing/scheduling (WS2004)	Present	Discrete	Tabular (4)
Robot soccer keepaway (SSK2005)	Present	Continuous	Tile coding (13)
Robot obstacle negotiation (LSYSN2006)	Present	Continuous	Linear (10)
Optimized trade execution (NFK2007)	Present	Discrete	Tabular (2-5)
Blimp control (RPHB2007)	Present	Continuous	Gaussian Process (2)
9 $\times$ 9 Go (SSM2007)	Absent	Discrete	Linear ($\approx$1.5 million)
Ms. Pac-Man (SL2007)	Absent	Discrete	Rule list (10)
Autonomic resource allocation (TJDB2007)	Present	Continuous	Neural network (2)
General game playing (FB2008)	Absent	Discrete	Tabular (part of state space)
Soccer opponent "hassling" (GRT2009)	Present	Continuous	Neural network (9)
Adaptive epilepsy treatment (GVAP2008)	Present	Continuous	Extremely rand. trees (114)
Computer memory scheduling (IMMC2008)	Absent	Discrete	Tile coding (6)
Motor skills (PS2008)	Present	Continuous	Motor primitive coeff. (100's)
Combustion Control (HNGK2009)	Present	Continuous	Parameterized policy (2-3)

Practice $\Longrightarrow$ Imperfect Representations

Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network (46)
Acrobot control (S1996)	Absent	Continuous	Tile coding (4)
Dynamic channel allocation (SB1997)	Absent	Discrete	Linear (100's)
Active guidance of finless rocket (GM2003)	Present	Continuous	Neural network (14)
Fast quadrupedal locomotion (KS2004)	Present	Continuous	Parameterized policy (12)
Robot sensing strategy (KF2004)	Present	Continuous	Linear (36)
Helicopter control (NKJS2004)	Present	Continuous	Neural network (10)
Dynamic bipedal locomotion (TZS2004)	Present	Continuous	Feedback control policy (2)
Adaptive job routing/scheduling (WS2004)	Present	Discrete	Tabular (4)
Robot soccer keepaway (SSK2005)	Present	Continuous	Tile coding (13)
Robot obstacle negotiation (LSYSN2006)	Present	Continuous	Linear (10)
Optimized trade execution (NFK2007)	Present	Discrete	Tabular (2-5)
Blimp control (RPHB2007)	Present	Continuous	Gaussian Process (2)
9 $\times$ 9 Go (SSM2007)	Absent	Discrete	Linear ($\approx$1.5 million)
Ms. Pac-Man (SL2007)	Absent	Discrete	Rule list (10)
Autonomic resource allocation (TJDB2007)	Present	Continuous	Neural network (2)
General game playing (FB2008)	Absent	Discrete	Tabular (part of state space)
Soccer opponent "hassling" (GRT2009)	Present	Continuous	Neural network (9)
Adaptive epilepsy treatment (GVAP2008)	Present	Continuous	Extremely rand. trees (114)
Computer memory scheduling (IMMC2008)	Absent	Discrete	Tile coding (6)
Motor skills (PS2008)	Present	Continuous	Motor primitive coeff. (100's)
Combustion Control (HNGK2009)	Present	Continuous	Parameterized policy (2-3)

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

Summary

- Learning by trial and error to perform sequential decision making.

Summary

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

Summary

- Learning by trial and error to perform sequential decision making.
- Given an MDP (S, A, T, R, γ), 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 π^* (several optimal policies might exist).

Summary

- Learning by trial and error to perform sequential decision making.
- Given an MDP (S, A, T, R, γ), 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 π^* (several optimal policies might exist).
- Under planning, we are given S, A, T, R, and γ. We may compute V^* and π^* using a dynamic programming algorithm such as policy iteration.

Summary

- Learning by trial and error to perform sequential decision making.
- Given an MDP (S, A, T, R, γ), 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 π^* (several optimal policies might exist).
- Under planning, we are given S, A, T, R, and γ. We may compute V^* and π^* using a dynamic programming algorithm such as policy iteration.
- In the learning context, we are given S, A, and γ: we may sample T and R in a sequential manner. We can still converge to V^* and π^* by applying a temporal difference learning method such as Q-learning.

Summary

- Learning by trial and error to perform sequential decision making.
- Given an MDP (S, A, T, R, γ), 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 π^* (several optimal policies might exist).
- Under planning, we are given S, A, T, R, and γ. We may compute V^* and π^* using a dynamic programming algorithm such as policy iteration.
- In the learning context, we are given S, A, and γ: we may sample T and R in a sequential manner. We can still converge to V^* and π^* 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, γ), we have to find a policy $\pi : S \to A$ that yields high expected long-term reward from states.
- An optimal value function V^* exists, and it induces an optimal policy π^* (several optimal policies might exist).
- Under planning, we are given S, A, T, R, and γ. We may compute V^* and π^* using a dynamic programming algorithm such as policy iteration.
- In the learning context, we are given S, A, and γ: we may sample T and R in a sequential manner. We can still converge to V^* and π^* 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

Christopher J. C. H. Watkins and Peter Dayan, 1992. Q-Learning. *Machine Learning*, 8(3–4):279–292, 1992.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Jette Randløv and Preben Alstrøm, 1998. Learning to Drive a Bicycle using Reinforcement Learning and Shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pp. 463–471, Morgan Kaufmann, 1998.

Richard S. Sutton and Andrew G. Barto, 1998. Reinforcement Learning: An Introduction. MIT Press, 1998.

Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng, 2006. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In *Advances in Neural Information Processing Systems 19*, pp. 1–8, MIT Press, 2006.

Todd Hester, Michael Quinlan, and Peter Stone, 2010. Generalized Model Learning for Reinforcement Learning on a Humanoid Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2010)*, pp. 2369–2374, IEEE, 2010.

Csaba Szepesvári, 2010. Algorithms for Reinforcement Learning. Morgan & Claypool, 2010.

Shivaram Kalyanakrishnan, 2011. Learning Methods for Sequential Decision Making with Imperfect Representations. Ph.D. dissertation, published as UT Austin Computer Science Technical Report TR-11-41, 2011.