An Introduction to Reinforcement Learning

Shivaram Kalyanakrishnan
shivaram@cse.iitb.ac.in

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

April 2018

What is Reinforcement Learning?

What is Reinforcement Learning?

[Video[1] of toddler learning to walk]

1. https://www.youtube.com/watch?v=jIzuy9fcf1k

What is Reinforcement Learning?

[Video[1] of toddler learning to walk]

1. `https://www.youtube.com/watch?v=jIzuy9fcf1k`

What is Reinforcement Learning?

[Video[1] of toddler learning to walk]

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

1. https://www.youtube.com/watch?v=jIzuy9fcf1k

What is Reinforcement Learning?

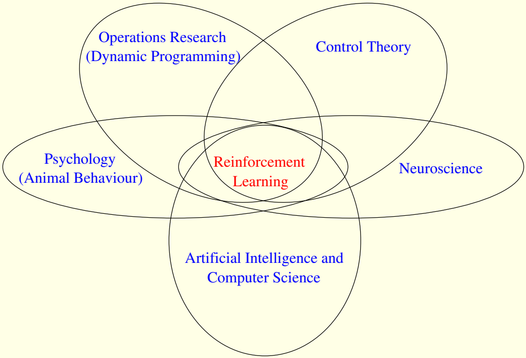
[Video[1] of toddler learning to walk]

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

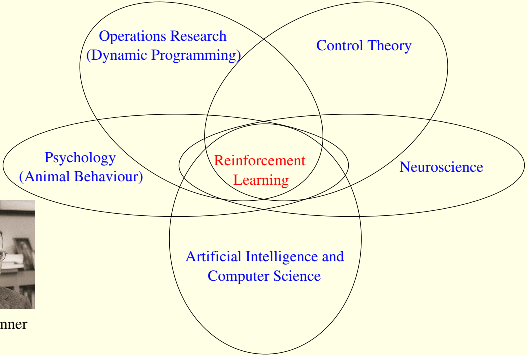
Learning by trial and error to perform *sequential* decision making.

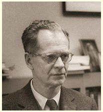
1. https://www.youtube.com/watch?v=jIzuy9fcf1k

Our View of Reinforcement Learning



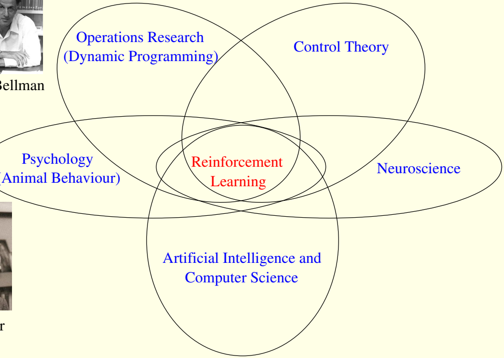
Our View of Reinforcement Learning

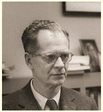



B. F. Skinner

Our View of Reinforcement Learning

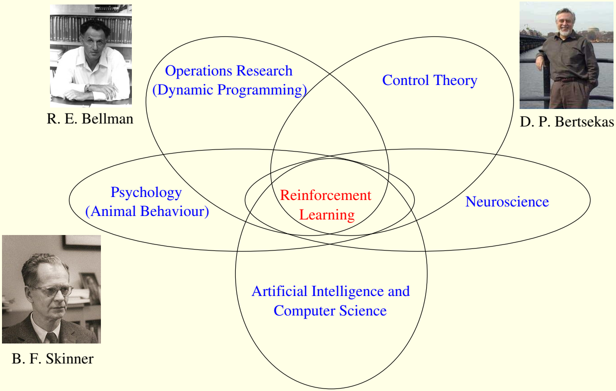
R. E. Bellman



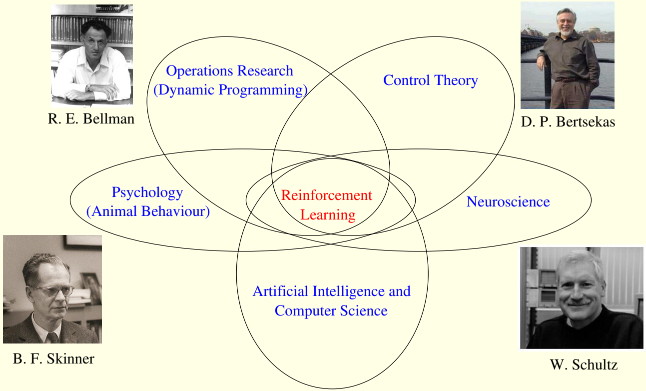


B. F. Skinner

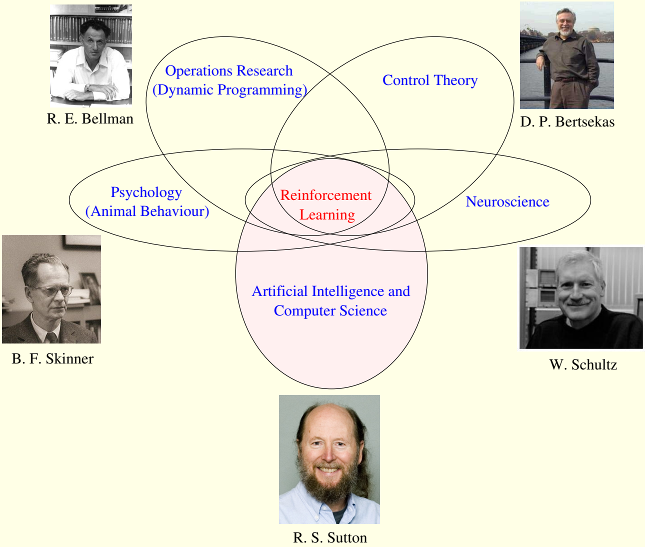
Our View of Reinforcement Learning



Our View of Reinforcement Learning



Our View of Reinforcement Learning

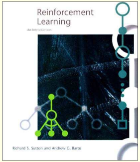


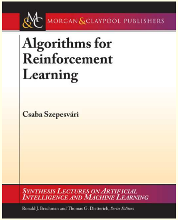
Resources

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. (2017-8 draft also now on-line).

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

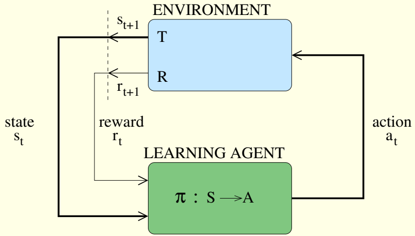
Today's Class

1. Markov Decision Problems
2. Planning and learning
3. Deep Reinforcement Learning
4. Summary

Today's Class

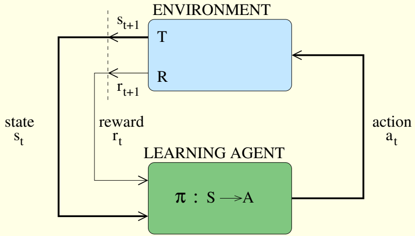
1. Markov Decision Problems
2. Planning and learning
3. Deep Reinforcement Learning
4. Summary

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. $0 \leq \gamma < 1$.

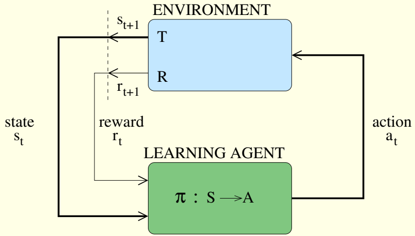
Markov Decision Problem



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: $s^0, a^0, r^0, s^1, a^1, r^1, \ldots, s^t, a^t, r^t, s^{t+1}, \ldots$.

Markov Decision Problem

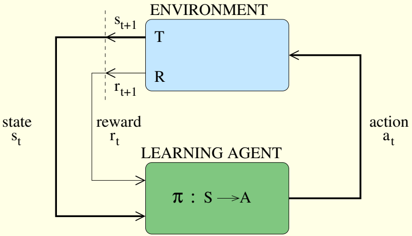


S: set of states.
A: set of actions.
T: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over S.
R: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
γ: discount factor. $0 \leq \gamma < 1$.

Trajectory over time: $s^0, a^0, r^0, s^1, a^1, r^1, \ldots, s^t, a^t, r^t, s^{t+1}, \ldots$.
Value, or expected long-term reward, of state s under policy π:

$$V^\pi(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots \text{ to } \infty | s^0 = s, a^i = \pi(s^i)].$$

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: $s^0, a^0, r^0, s^1, a^1, r^1, \ldots, s^t, a^t, r^t, s^{t+1}, \ldots$.
Value, or expected long-term reward, of state s under policy π:

$$V^\pi(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots \text{ to } \infty | s^0 = s, a^i = \pi(s^i)].$$

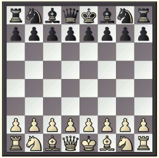
Objective: "Find π such that $V^\pi(s)$ is maximal $\forall s \in S$."

Examples

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

Examples

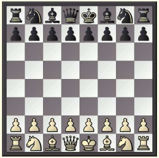
What are the agent and environment? What are S, A, T, and R?



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?

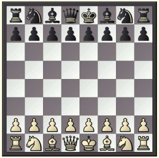


An Application of Reinforcement Learning to Aerobatic Helicopter Flight
Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. NIPS 2006.

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

Today's Class

1. Markov decision problems
2. Planning and learning
3. Deep Reinforcement Learning
4. Summary

Bellman's Equations

Recall that

$$V^{\pi}(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^i = \pi(s^i)].$$

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

$$V^\pi(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^i = \pi(s^i)].$$

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

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

V^π is called the value function of π.

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

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

Q^π is called the action value function of π.

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

Bellman's Equations

Recall that

$$V^{\pi}(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^i = \pi(s^i)].$$

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 Equations are the $V^{\pi}(s)$. $|S|$ linear equations in $|S|$ unknowns.

Bellman's Equations

Recall that

$$V^{\pi}(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^i = \pi(s^i)].$$

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 Equations 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

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

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 ($\forall s \in S$):

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

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 π^* $\in \Pi$ such that

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

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 ($\forall s \in S$):

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

V^* can be computed up to arbitrary precision using **Value Iteration**.

Planning and Learning

Planning problem:

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

Planning and Learning

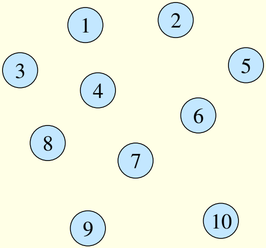
Planning problem:

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

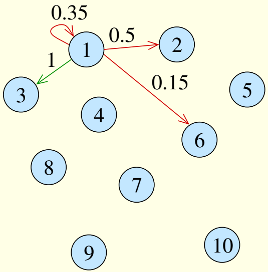
Learning problem:

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.

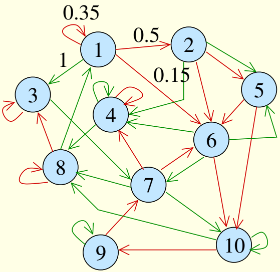
The Learning Problem



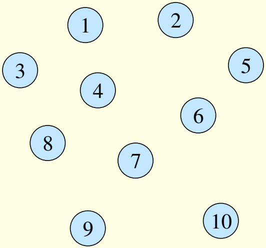
The Learning Problem



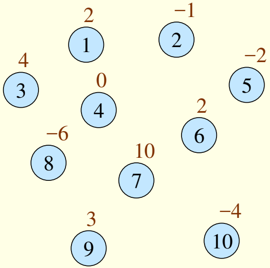
The Learning Problem



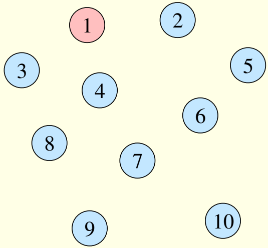
The Learning Problem



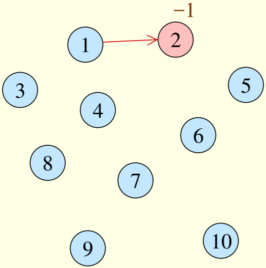
The Learning Problem



The Learning Problem

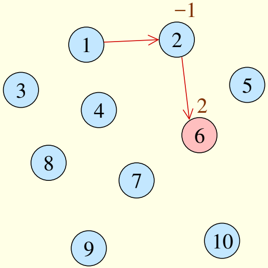


The Learning Problem



$r^0 = -1.$

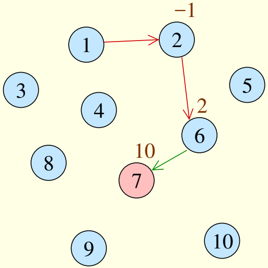
The Learning Problem



$r^0 = -1.$

$r^1 = 2.$

The Learning Problem

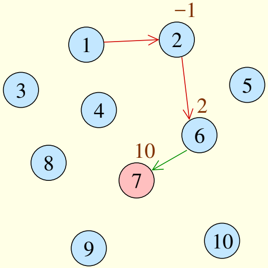


$r^0 = -1.$

$r^1 = 2.$

$r^2 = 10.$

The Learning Problem



$r^0 = -1.$

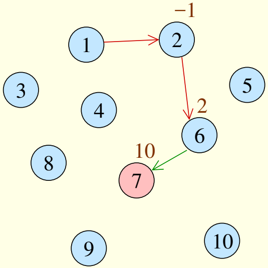
$r^1 = 2.$

$r^2 = 10.$

How to take actions so as to maximise expected long-term reward

$$\mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots]?$$

The Learning Problem



$r^0 = -1.$
$r^1 = 2.$
$r^2 = 10.$

How to take actions so as to maximise expected long-term reward

$$\mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots]?$$

[Note that there exists an (unknown) *optimal policy*.]

Q-Learning

- Keep a running estimate of the expected long-term reward obtained by taking each action from each state s, and acting *optimally* thereafter.

Q	red	green
1	-0.2	10
2	4.5	13
3	6	-8
4	0	0.2
5	-4.2	-4.2
6	1.2	1.6
7	10	6
8	4.8	9.9
9	5.0	-3.4
10	-1.9	2.3

Q-Learning

- Keep a running estimate of the expected long-term reward obtained by taking each action from each state s, and acting *optimally* thereafter.

Q	red	green
1	-0.2	10
2	4.5	13
3	6	-8
4	0	0.2
5	-4.2	-4.2
6	1.2	1.6
7	10	6
8	4.8	9.9
9	5.0	-3.4
10	-1.9	2.3

- Update these estimates based on experience (s^t, a^t, r^t, s^{t+1}):

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t\{r^t + \max_a Q(s^{t+1}, a) - Q(s^t, a^t)\}.$$

Q-Learning

- Keep a running estimate of the expected long-term reward obtained by taking each action from each state s, and acting *optimally* thereafter.

Q	red	green
1	-0.2	10
2	4.5	13
3	6	-8
4	0	0.2
5	-4.2	-4.2
6	1.2	1.6
7	10	6
8	4.8	9.9
9	5.0	-3.4
10	-1.9	2.3

- Update these estimates based on experience (s^t, a^t, r^t, s^{t+1}):

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t \{ r^t + \max_a Q(s^{t+1}, a) - Q(s^t, a^t) \}.$$

Q-Learning

- Keep a running estimate of the expected long-term reward obtained by taking each action from each state s, and acting *optimally* thereafter.

Q	red	green
1	-0.2	10
2	4.5	13
3	6	-8
4	0	0.2
5	-4.2	-4.2
6	1.2	1.6
7	10	6
8	4.8	9.9
9	5.0	-3.4
10	-1.9	2.3

- Update these estimates based on experience (s^t, a^t, r^t, s^{t+1}):

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t\{r^t + \max_a Q(s^{t+1}, a) - Q(s^t, a^t)\}.$$

Q-Learning

- Keep a running estimate of the expected long-term reward obtained by taking each action from each state s, and acting *optimally* thereafter.

Q	red	green
1	-0.2	10
2	4.5	13
3	6	-8
4	0	0.2
5	-4.2	-4.2
6	1.2	1.6
7	10	6
8	4.8	9.9
9	5.0	-3.4
10	-1.9	2.3

- Update these estimates based on experience (s^t, a^t, r^t, s^{t+1}):

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t\{r^t + \max_a Q(s^{t+1}, a) - Q(s^t, a^t)\}.$$

- Act greedily based on the estimates (exploit) most of the time, but still
- Make sure to explore each action enough times.

Q-Learning

- Keep a running estimate of the expected long-term reward obtained by taking each action from each state s, and acting *optimally* thereafter.

Q	red	green
1	-0.2	10
2	4.5	13
3	6	-8
4	0	0.2
5	-4.2	-4.2
6	1.2	1.6
7	10	6
8	4.8	9.9
9	5.0	-3.4
10	-1.9	2.3

- Update these estimates based on experience (s^t, a^t, r^t, s^{t+1}):

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t\{r^t + \max_a Q(s^{t+1}, a) - Q(s^t, a^t)\}.$$

- Act greedily based on the estimates (exploit) most of the time, but still
- Make sure to explore each action enough times.

Q-learning will converge and induce an optimal policy!

Q-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]

Q-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.

Practice In Spite of the Theory!

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 In Spite of the Theory!

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 In Spite of the Theory!

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 In Spite of the Theory!

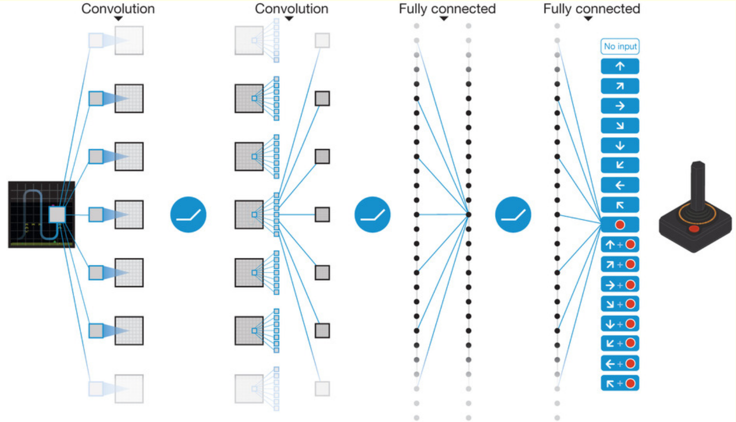
Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network (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. Planning and learning
3. Deep Reinforcement Learning
4. Summary

Typical Neural Network-based Representation of Q



1. `http://www.nature.com/nature/journal/v518/n7540/carousel/nature14236-f1.jpg`

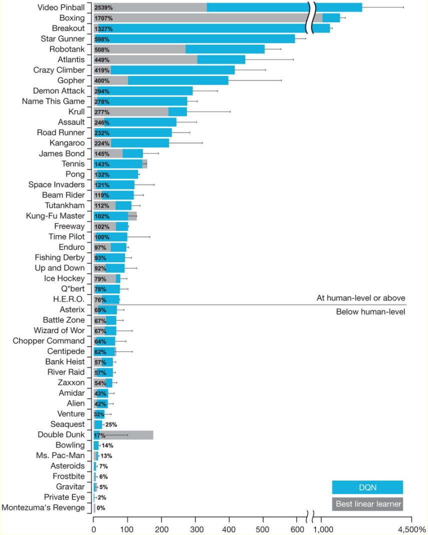
ATARI 2600 Games (MKSRVBGRFOPBSAKKWLH2015)

[Breakout video[1]]

1. http://www.nature.com/nature/journal/v518/n7540/extref/nature14236-sv2.mov

ATARI 2600 Games (MKSRVBGRFOPBSAKKWLH2015)

[Breakout video[1]]



1. http://www.nature.com/nature/journal/v518/n7540/extref/nature14236-sv2.mov

AlphaGo (SHMGSDSAPLDGNKSLLKGH2016)

March 2016: DeepMind's program beats Go champion Lee Sedol 4-1.

1. http://www.kurzweilai.net/images/AlphaGo-vs.-Sedol.jpg

AlphaGo (SHMGSDSAPLDGNKSLLKGH2016)

1. http://static1.uk.businessinsider.com/image/56e0373052bcd05b008b5217-810-602/screen%20shot%202016-03-09%20at%2014.png

Learning Algorithm: Batch Q-learning

1. Represent action value function Q as a neural network.

2. Gather data (on the simulator) by taking ϵ-greedy actions w.r.t. Q:
   $(s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \ldots s_D, a_D, r_D, s_{D+1})$.

3. Train the network such that $Q(s_t, a_t) \approx r_t + \max_a Q(s_{t+1}, a)$.
   Go to 2.

Learning Algorithm: Batch Q-learning

1. Represent action value function Q as a neural network.
   AlphaGo: Use both a policy network and an action value network.

2. Gather data (on the simulator) by taking ϵ-greedy actions w.r.t. Q:
   $(s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \ldots s_D, a_D, r_D, s_{D+1})$.
   AlphaGo: Use Monte Carlo Tree Search for action selection

3. Train the network such that $Q(s_t, a_t) \approx r_t + \max_a Q(s_{t+1}, a)$.
   Go to 2.

AlphaGo: Trained using self-play.

Today's Class

1. Markov Decision Problems
2. Planning and learning
3. Deep Reinforcement Learning
4. Summary

Summary

- Learning by trial and error to perform sequential decision making.
- Do not program behaviour! Rather, specify goals.
- Rich history, at confluence of several fields of study, firm foundation.

- 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.
- Limited in practice by quality of the representation used.
- Deep neural networks address the representation problem in some domains, and have yielded impressive results.