Reinforcement Learning

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

February 2023

What is Reinforcement Learning?

- Watch this YouTube video[1] of a toddler.

.

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

What is Reinforcement Learning?

- Watch this YouTube video[1] of a toddler.
  The toddler was ____________.

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

What is Reinforcement Learning?

- Watch this YouTube video[1] of a toddler.
  The toddler was learning to walk.

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

What is Reinforcement Learning?

- Watch this YouTube video[1] of a toddler.
  The toddler was learning to walk.

[2]

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

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

What is Reinforcement Learning?

- Watch this YouTube video[1] of a toddler.
  The toddler was learning to walk.

[2]

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

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

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

References

- **Reinforcement Learning: An Introduction**, Richard S. Sutton and Andrew G. Barto, 2nd edition, MIT Press, 2018. Available on-line at `http://www.incompleteideas.net/book/the-book-2nd.html`.

References

- **Reinforcement Learning: An Introduction**, Richard S. Sutton and Andrew G. Barto, 2nd edition, MIT Press, 2018. Available on-line at `http://www.incompleteideas.net/book/the-book-2nd.html`.

- **Reinforcement Learning: A Survey**. Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore, Journal of Artificial Intelligence Research, 4(1): 237–285, 1996. Available on-line at `https://arxiv.org/pdf/cs/9605103.pdf`.

References

- **Reinforcement Learning: An Introduction**, Richard S. Sutton and Andrew G. Barto, 2nd edition, MIT Press, 2018. Available on-line at `http://www.incompleteideas.net/book/the-book-2nd.html`.

- **Reinforcement Learning: A Survey**. Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore, Journal of Artificial Intelligence Research, 4(1): 237–285, 1996. Available on-line at `https://arxiv.org/pdf/cs/9605103.pdf`.

- My course at IIT Bombay.

  Foundations of Intelligent and Learning Agents

  Topics: Multi-armed Bandits, Markov Decision Problems, Reinforcement Learning.

  Course page: `https://www.cse.iitb.ac.in/~shivaram/teaching/old/cs747-a2020/index.html`.

  1.5–2 hours of video lectures per week.

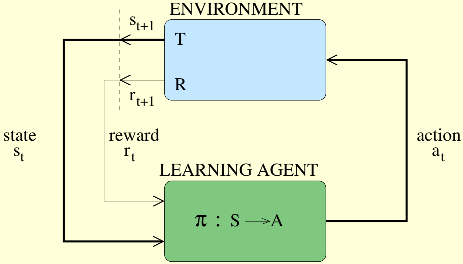
This Lecture

- Markov Decision Problems
- Reinforcement Learning problem
- Q-learning algorithm
- Deep Reinforcement Learning

This Lecture

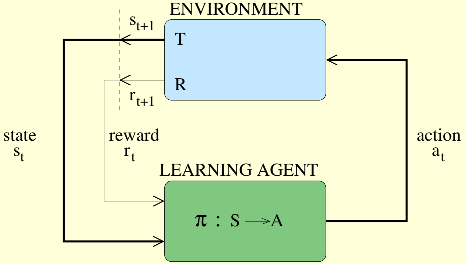
- Markov Decision Problems
- Reinforcement Learning problem
- Q-learning algorithm
- Deep Reinforcement Learning

Markov Decision Problem (MDP)



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. $0 \leq \gamma < 1$.

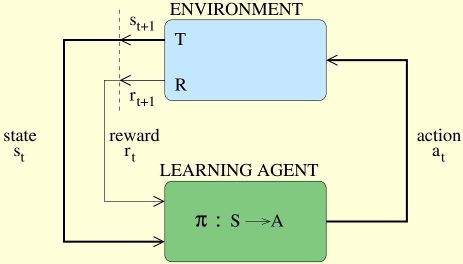
Markov Decision Problem (MDP)



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 (MDP)

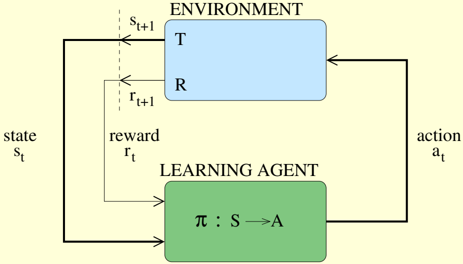


S: set of states.
A: set of actions.
T: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over S.
R: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
γ: discount factor. $0 \leq \gamma < 1$.

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

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

Markov Decision Problem (MDP)



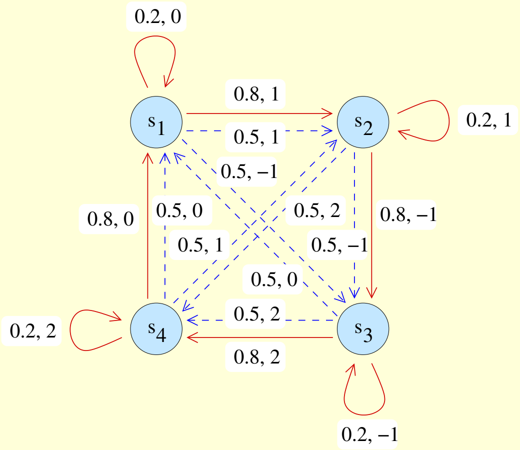
S: set of states.
A: set of actions.
T: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over S.
R: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
γ: discount factor. $0 \leq \gamma < 1$.

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:

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

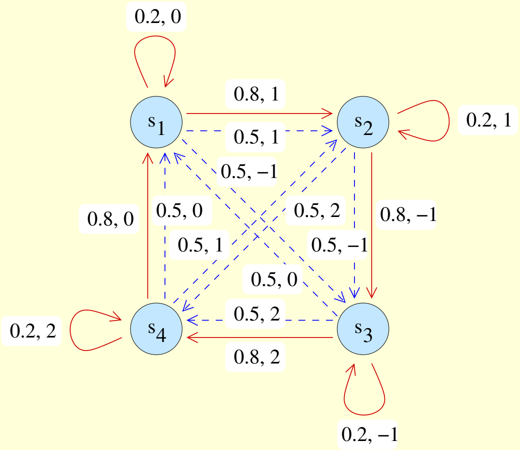
Objective: "Find π such that $V^\pi(s)$ is maximal $\forall s \in S$."

MDPs as State-transition Diagrams



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

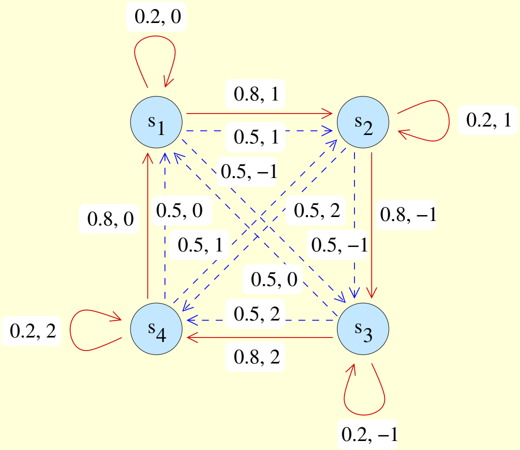
MDPs as State-transition Diagrams



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

- How many policies does this MDP have?

MDPs as State-transition Diagrams



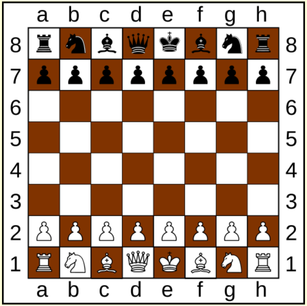
Notation: "transition probability, reward" marked on each arrow

- How many policies does this MDP have?

  $2 \times 2 \times 2 \times 2 = 16$.

Examples: MDP Formulation

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



[1]

[2]

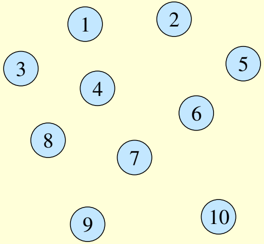
[1] https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg. CC image courtesy of ILA-boy on WikiMedia Commons licensed under CC-BY-SA-3.0-migrated.

[2] . CC image courtesy of Wilfredor on WikiMedia Commons licensed under CC-BY-SA-4.0.

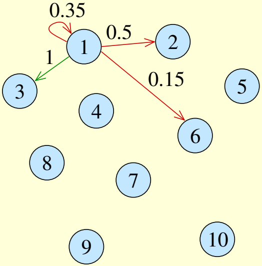
This Lecture

- Markov Decision Problems
- Reinforcement Learning problem
- Q-learning algorithm
- Deep Reinforcement Learning

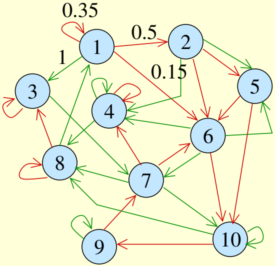
The Learning Problem



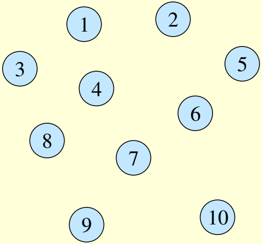
The Learning Problem



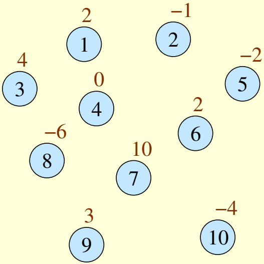
The Learning Problem



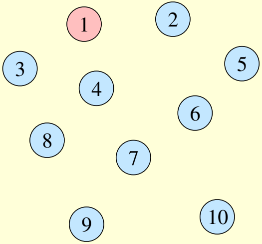
The Learning Problem



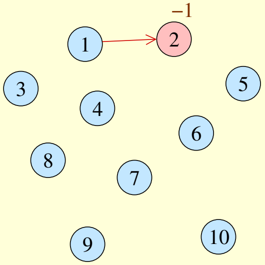
The Learning Problem



The Learning Problem

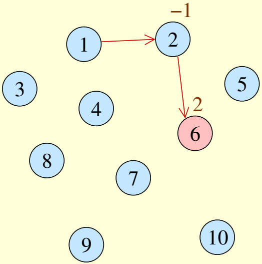


The Learning Problem



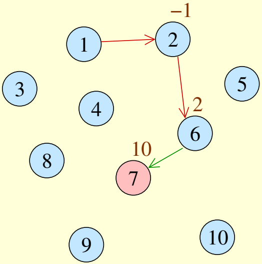
$r^0 = -1.$

The Learning Problem



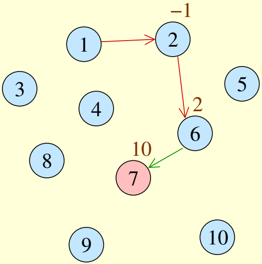
$r^0 = -1.$
$r^1 = 2.$

The Learning Problem



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

The Learning Problem

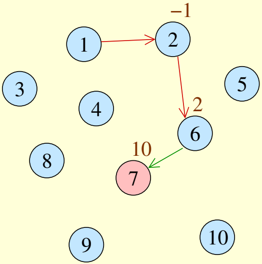


$r^0 = -1.$
$r^1 = 2.$
$r^2 = 10.$

- 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



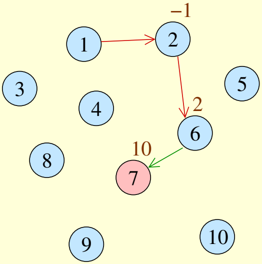
$r^0 = -1.$
$r^1 = 2.$
$r^2 = 10.$

- 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 $\pi^\star$.

The Learning Problem



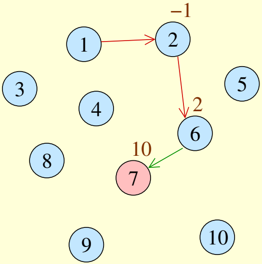
$r^0 = -1.$
$r^1 = 2.$
$r^2 = 10.$

- 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 $\pi^\star$.
- Can we learn to perform as well as $\pi^\star$?

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 $\pi^\star$.
- Can we learn to perform as well as $\pi^\star$? Eventually?

This Lecture

- Markov Decision Problems
- Reinforcement Learning problem
- Q-learning algorithm
- Deep Reinforcement Learning

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}):

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

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}):

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

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}):

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

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}):

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

- 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}):

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

- 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!

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)
9×9 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!

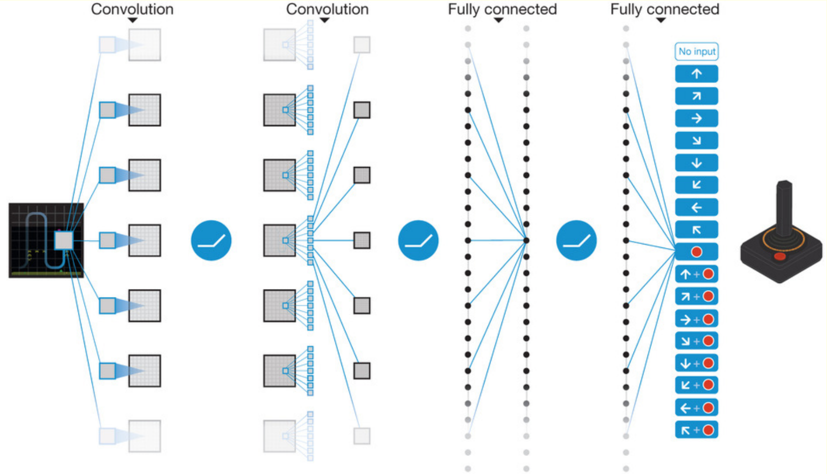
Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network (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)
9×9 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.

Typical Neural Network-based Representation of Q



[1]

[1] **Human-level control through deep reinforcement learning**. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Nature, 518: 529–533, 2015.

This Lecture

- Markov Decision Problems
- Reinforcement Learning problem
- Q-learning algorithm
- Deep Reinforcement Learning

ATARI 2600 Games

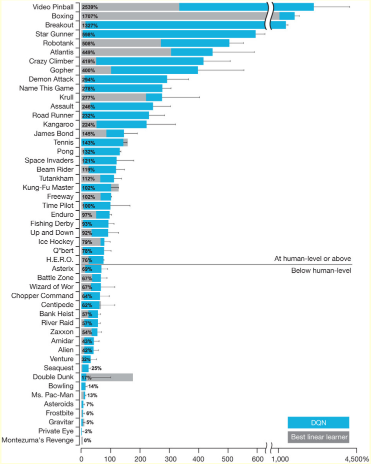
Watch this YouTube video[1] of "Breakout".

[1] `https://www.youtube.com/watch?v=TmPfTpjtdgg`.

[2] **Human-level control through deep reinforcement learning**. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Nature, 518: 529–533, 2015.

ATARI 2600 Games

Watch this YouTube video[1] of “Breakout”.



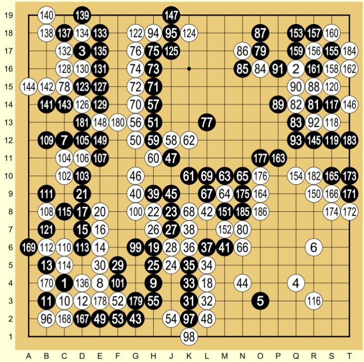
[2]

[1] `https://www.youtube.com/watch?v=TmPfTpjtdgg`.

[2] **Human-level control through deep reinforcement learning**. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Nature, 518: 529–533, 2015.

AlphaGo

- In March 2016, Google DeepMind's AlphaGo[1] program beat Lee Sedol (winner of 18 international titles), 4–1.



Lee Sedol (B) vs AlphaGo (W) - Game 1

[2]

[1] **Mastering the game of Go with deep neural networks and tree search**. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Nature, 529:484–489, 2016.

[2] `https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg/734px-Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg.png`. CC image courtesy of Wesalius on WikiMedia Commons licensed under CC-BY-SA-4.0.

Learning Algorithm in ATARI, AlphaGo: 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 in ATARI, AlphaGo: 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.

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.

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.
- 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 optimal behaviour by applying a temporal difference learning method such as Q-learning.

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.
- 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 optimal behaviour 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.