Reinforcement Learning

Shivaram Kalyanakrishnan
shivaram@cse.iitb.ac.in

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

February 2017

RoboCup Soccer

Objective of the RoboCup Federation:

"By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup."

RoboCup Soccer

Objective of the RoboCup Federation:

"By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup."

[RoboCup 2010: Nao video[1]]

1. `https://www.youtube.com/watch?v=b6Zu5fLUa3c`

Half Field Offense (KLS2007)

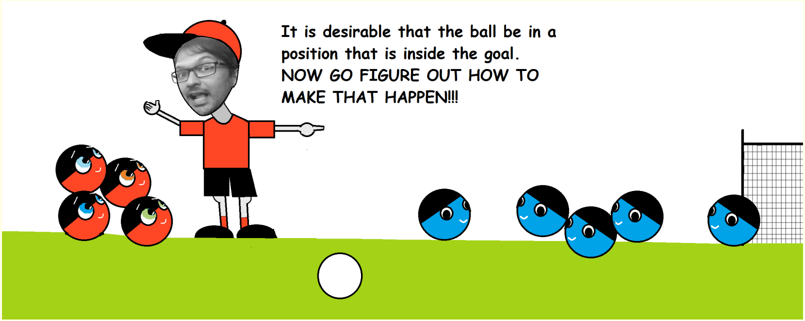
[Video of task[1]]

1. `http://www.cs.utexas.edu/~AustinVilla/sim/halffieldoffense/swfs/Random.swf`

Half Field Offense (KLS2007)

[Video of task[1]]

Training

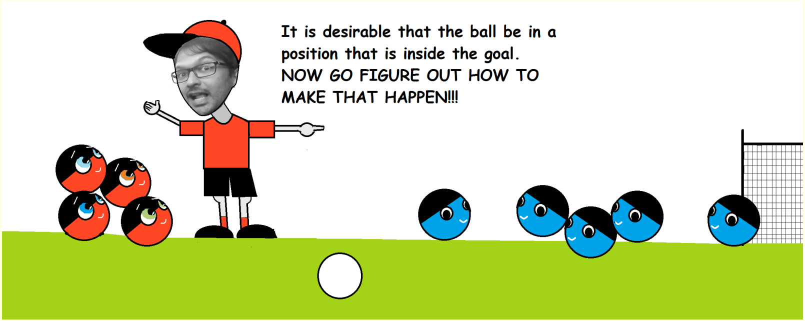


1. `http://www.cs.utexas.edu/~AustinVilla/sim/halffieldoffense/swfs/Random.swf`

Half Field Offense (KLS2007)

[Video of task[1]]

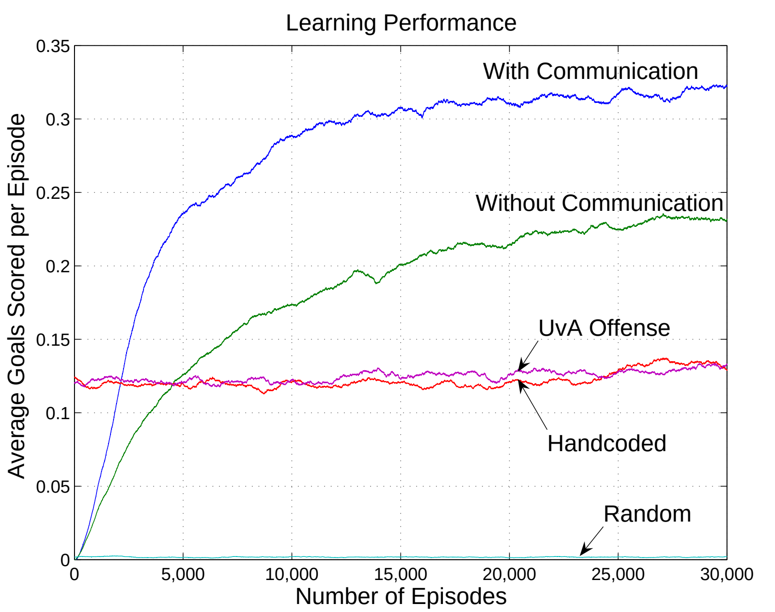
Training



[Video of task after training[2]]

1. `http://www.cs.utexas.edu/~AustinVilla/sim/halffieldoffense/swfs/Random.swf`
2. `http://www.cs.utexas.edu/~AustinVilla/sim/halffieldoffense/swfs/Communication.swf`

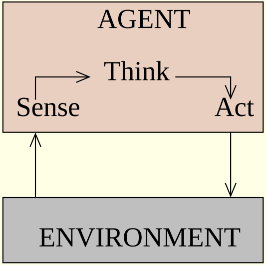
Half Field Offense (KLS2007)



Learning to Act Purposefully

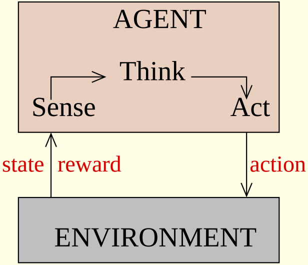
Answer: Reinforcement Learning (RL).

Learning to Act Purposefully



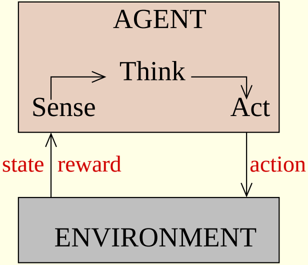
Answer: Reinforcement Learning (RL).

Learning to Act Purposefully



Answer: Reinforcement Learning (RL).

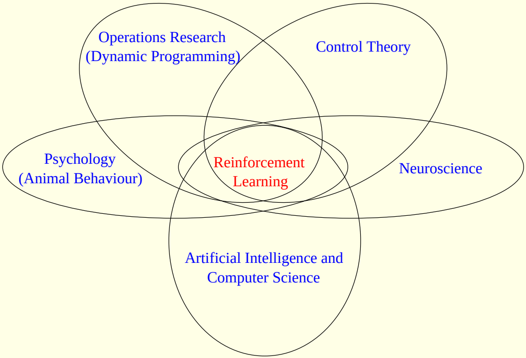
Learning to Act Purposefully



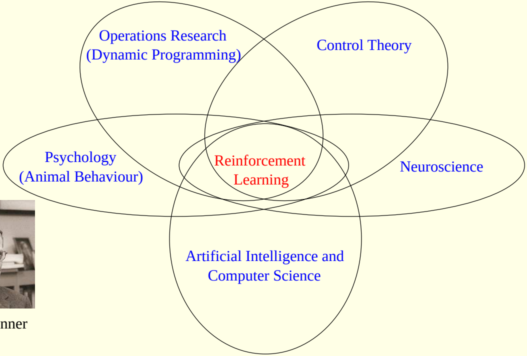
Question: How must an agent in an *unknown* environment act so as to maximise its long-term reward?

Answer: Reinforcement Learning (RL).

Reinforcement Learning: Historical Foundations



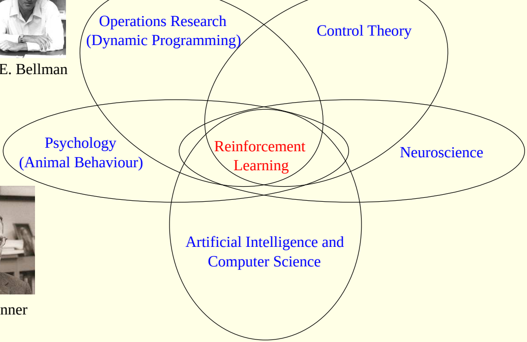
Reinforcement Learning: Historical Foundations




B. F. Skinner

Reinforcement Learning: Historical Foundations

R. E. Bellman

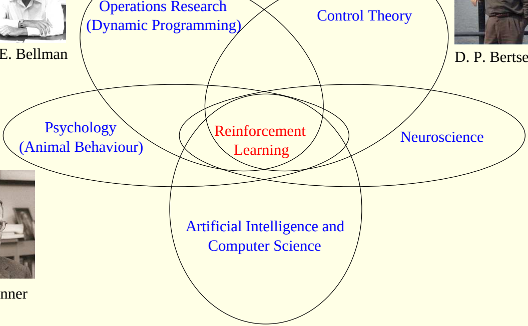


B. F. Skinner

Reinforcement Learning: Historical Foundations

R. E. Bellman

D. P. Bertsekas

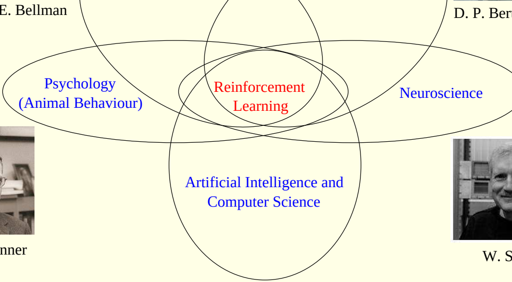


B. F. Skinner

Reinforcement Learning: Historical Foundations

R. E. Bellman

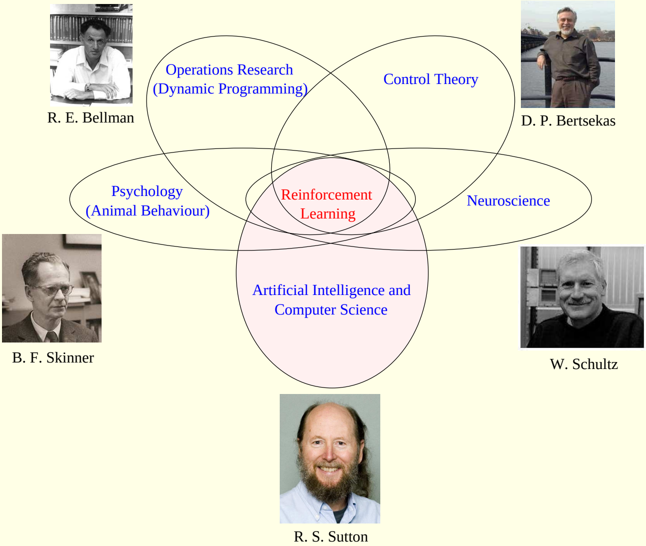
D. P. Bertsekas



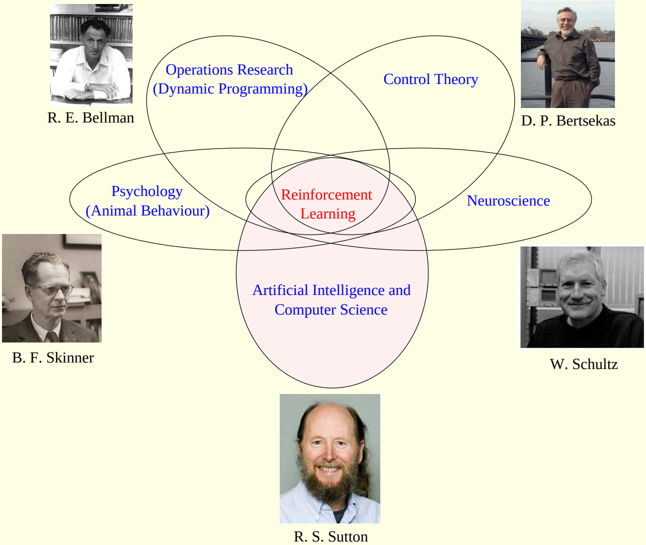
B. F. Skinner

W. Schultz

Reinforcement Learning: Historical Foundations



Reinforcement Learning: Historical Foundations



References: KLM1996, SB1998.

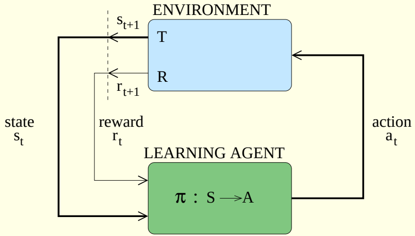
Outline

1. Markov Decision Problems
2. Bellman's (Optimality) Equations, planning and learning
3. Challenges
4. RL in practice
5. Summary

Outline

1. Markov Decision Problems
2. Bellman's (Optimality) Equations, planning and learning
3. Challenges
4. RL in practice
5. Summary

Markov Decision Problem



S: set of states.

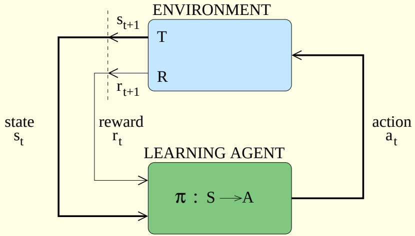
A: set of actions.

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. $0 \leq \gamma < 1$.

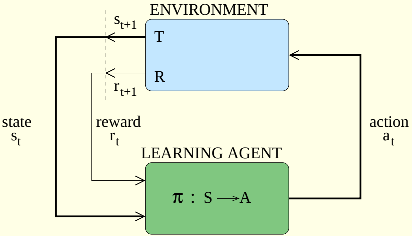
Markov Decision Problem



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

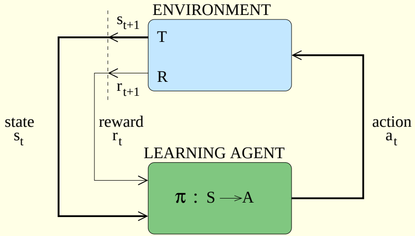


S: set of states.
A: set of actions.
T: transition function. $\forall s \in S, \forall a \in A$, $T(s, a)$ is a distribution over S.
R: reward function. $\forall s, s' \in S, \forall a \in A$, $R(s, a, s')$ is a finite real number.
γ: discount factor. $0 \leq \gamma < 1$.

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:

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

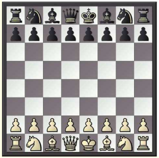
Objective: "Find π such that $V^\pi(s)$ is maximal $\forall s \in S$."

Examples

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

Examples

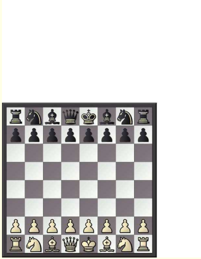
What are the agent and environment? What are S, A, T, and R?



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?

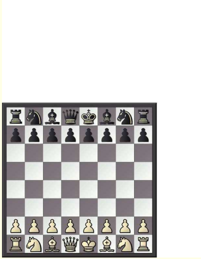


(ACQN2006)

1. http://www.chess-game-strategies.com/images/kqa_chessboard_large-picture_2d.gif
2. http://scd.france24.com/en/files/imagecache/france24_ct_api_bigger_169/article/image/101016-airbus-pologne-characal-m.jpg

Examples

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

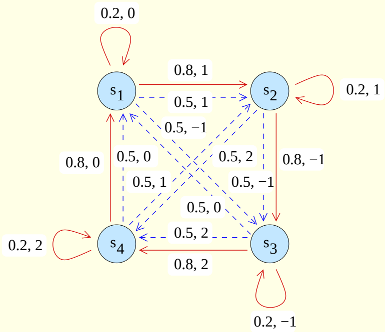


(ACQN2006)

[Video[3] of Tetris]

1. http://www.chess-game-strategies.com/images/kqa_chessboard_large-picture_2d.gif
2. http://scd.france24.com/en/files/imagecache/france24_ct_api_bigger_169/article/image/101016-airbus-pologne-characal-m.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: $\gamma = 0.9$.

Outline

1. Markov Decision Problems
2. Bellman's (Optimality) Equations, planning and learning
3. Challenges
4. RL in practice
5. 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 ($\forall s \in S$):

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

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 ($\forall s \in S$):

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

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.

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

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

Bellman's Equations

Recall that

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

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 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

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

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

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 ($\forall s \in S$):

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

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 $\pi^* \in \Pi$ 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 ($\forall s \in S$):

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

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

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

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 ($\forall s \in S$):

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

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 method. 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 method. 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 method to find $V^\star$. Value Iteration.

- Initialise $V^0 : S \to \mathbb{R}$ arbitrarily.
- $t \leftarrow 0$.
- Repeat
  - For all $s \in S$,
    - $V^{t+1}(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma V^t(s)\right]$.
  - $t \leftarrow t + 1$.
- Until $\|V^t - V^{t-1}\|$ is small enough.

Planning

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

One method. 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 method to find $V^\star$. Value Iteration.

■Initialise $V^0 : S \to \mathbb{R}$ arbitrarily.
■ $t \leftarrow 0$.
■Repeat
  ■For all $s \in S$,
    ■ $V^{t+1}(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma V^t(s)\right]$.
  ■ $t \leftarrow t + 1$.
■Until $\|V^t - V^{t-1}\|$ is small enough.

Other methods Policy iteration, and mixtures with Value 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 $t \to \infty$, $Q \to Q^*$.

(WD1992)

Outline

Challenges

- Exploration
- Generalisation (over states and actions)
- State aliasing (partial observability)
- Multiple agents, nonstationary rewards and transitions
- Abstraction (over states and over time)

Challenges

- Exploration
- Generalisation (over states and actions)
- State aliasing (partial observability)
- Multiple agents, nonstationary rewards and transitions
- Abstraction (over states and over time)

My thesis question (K2011):

"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?"

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 $\implies$ Imperfect Representations

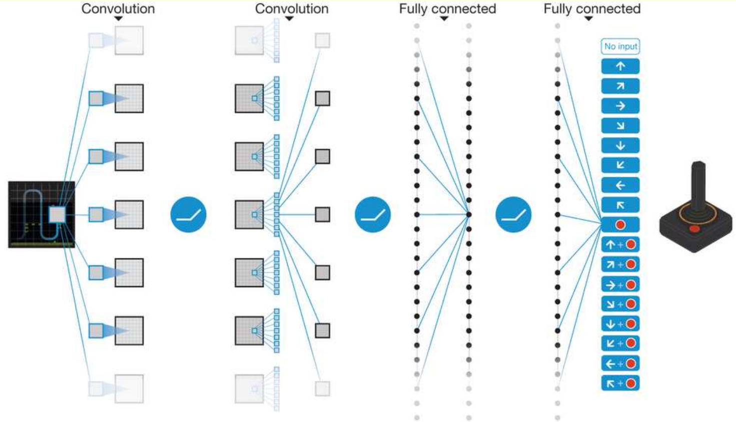
Task	State Aliasing	State Space	Policy Representation (Number of features)
Backgammon (T1992)	Absent	Discrete	Neural network (198)
Job-shop scheduling (ZD1995)	Absent	Discrete	Neural network (20)
Tetris (BT1906)	Absent	Discrete	Linear (22)
Elevator dispatching (CB1996)	Present	Continuous	Neural network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

Outline

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

Typical Neural Network-based Representation of Q



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

Practical Implementation and Evaluation of Learning Algorithms

(HQS2010)

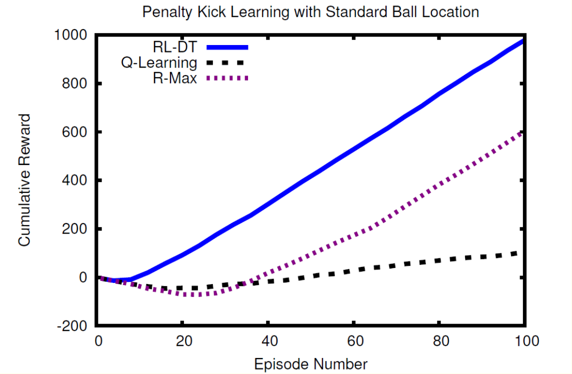
[Video[1] of RL on a humanoid robot]

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

Practical Implementation and Evaluation of Learning Algorithms

(HQS2010)

[Video[1] of RL on a humanoid robot]



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

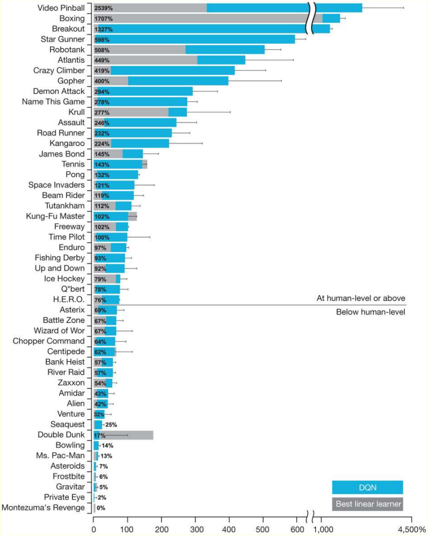
ATARI 2600 Games (MKSRVBGRFOPBSAKKWLH2015)

[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

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

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.

References

(For references on slide 17, see Kalyanakrishnan's thesis (K2011).)

[WD1992] Christopher J. C. H. Watkins and Peter Dayan, 1992. Q-Learning. *Machine Learning*, 8(3–4):279–292, 1992.

[P1994] Martin L. Puterman. Markov Decision Processes. Wiley, 1994.

[KLM1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[SB1998] Richard S. Sutton and Andrew G. Barto, 1998. Reinforcement Learning: An Introduction. MIT Press, 1998.

[HOT2006] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, 2006. A Fast Learning Algorithm for Deep Belief Nets, *Neural Computation*, 18:1527–1554, 2006.

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.

References

[KLS2007] Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone. Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study. In *RoboCup 2006: Robot Soccer World Cup X*, pp. 72–85, Springer, 2007.

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.

[K2011] Shivaram Kalyanakrishnan. Learning Methods for Sequential Decision Making with Imperfect Representations. *Ph.D. Thesis, Department of Computer Science, The University of Texas at Austin*, 2011.

[MKSRVBGRFOPBSAKKWLH2015] 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. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533, 2015.

[SHMGSDSAPLDGNKSLLKGH2016] 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, 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529: 484–489, 2016.

Summary and Conclusion

Reinforcement Learning

Do not program behaviour! Rather, specify goals.

Rich history, at confluence of several fields of study, firm foundation.

Limited in practice by quality of the representation used.

Recent advances in deep learning have reinvigorated the field of RL.

Very promising technology that is changing the face of AI.