Introduction to Hierarchical Reinforcement Learning

Lecturer: Bernhard Hengst
email: bernhard.hengst@nicta.com.au

19th October 2005
The goal of reinforcement learning is to decide what action to take at each time-step to maximise the total reward over time.
The key feature of reinforcement learning is reasoning about future rewards.
History of Reinforcement Learning

• Psychology/biology
  (eg BF Skinner - animal training)

• Operations Research
  DP (planning over time)
The Agent View of RL

Environment

Sensors

agent

effectors
One Room Problem

Room States

Exit Reward
$100

cost $1 per time-step
(except $100)

State Transitions

Actions $\in \{N,S,E,W\}$
Sense – Act Cycle

Policy \( \pi : S \rightarrow A \)  
\( \text{e.g. } \pi(1) = \text{E}, \pi(2) = \text{S}, \pi(5) = \text{S}, \ldots \)
Value Function

is the utility of the current state in terms of future rewards given a policy.

$$V^\pi(s_t) \equiv r_t + r_{t+1} + \ldots + r_T$$

$$V^\pi(s_t) \equiv r_t + V^\pi(s_{t+1})$$
Optimal Value Function

\( V^*(s_t) \equiv \max_a [ r_t + V^*(s_{t+1}) ] \)
An Optimal Policy

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$97</td>
<td>$98</td>
<td>$99</td>
</tr>
<tr>
<td>3</td>
<td>$98</td>
<td>$99</td>
<td>$100</td>
</tr>
<tr>
<td>6</td>
<td>$97</td>
<td>$98</td>
<td>$99</td>
</tr>
</tbody>
</table>

\[
\pi^*(s_t) \equiv \arg \max_a \left[ r_t + V^*(s_{t+1}) \right]
\]
Markov Decision Problem (MDP)
tuple \( \langle S, A, \delta, r, s_0 \rangle \)

States \( S = \{0, 1, 2, 3, 4, 5, 6, 7, 8\} \)
\( s_0 = \) starting state (eg random)

Actions \( A = \{N, S, E, W\} \)

Transition function \( \delta(s_t, a_t) = s_{t+1} \)

Reward function \( r(s_t, a_t) = r_t \)

Optimisation criterion eg \( V^\pi(s_t) \equiv r_t + r_{t+1} + r_{t+2} \ldots r_T \)

Optimal policy \( \pi^* \equiv \arg \max_{\pi} V^\pi(s), \forall s \)
The Markov Property

\[ \Pr(s_{t+1} | s_t) = \Pr(s_{t+1} | s_t, s_{t-1}, s_{t-2}, s_{t-3}, \ldots) \]
Solution to MDP - Value Iteration

initialise $V(s) = 0$ for all $s$
repeat for each $s \in S$

$$V(s) \leftarrow \max_a [ r(s, a) + V(\delta(s, a)) ]$$

until $V(s)$ does not change for any $s$

Output $\pi^*(s) \equiv \arg \max_a [ r(s, a) + V(\delta(s, a)) ]$
Towers of Hanoi

States = \{(1,1,1,1,1,1,1), (2,1,1,1,1,1,1), \ldots, (3,3,3,3,3,3,3)\} \quad |S|=2187

$S_0 = (1,1,1,1,1,1,1), \quad s=(3,3,3,3,3,3,3)$ is goal or terminal state

Actions = \{moveDisc12, moveDisc13, moveDisc21, moveDisc23, moveDisc31, moveDisc32\}

$\delta: (1,1,1,1,1,1,1), \text{moveDisc12} \rightarrow (2,1,1,1,1,1,1),$  
\quad (1,1,1,1,1,1,1), \text{moveDisc13} \rightarrow (3,1,1,1,1,1,1), \text{ etc}

$R: \text{ all moves cost } -1$

$V^*(1,1,1,1,1,1,1)=-127$

$\pi^*(1,1,1,1,1,1,1)=\text{moveDisc13, etc} \quad (\text{NB there are } 10^{1701} \text{ polices!!!})$
State Transitions and Rewards can be Stochastic

Deterministic

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Stochastic

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
State Transitions and Rewards must be Markov

- Deterministic
  - $s \xleftarrow{a} s, r$
  - $s \xrightarrow{a} s', r$

- Stochastic
  - $s \xrightarrow{a} s'$

$\times$ = not Markov
Optimality Criteria

Sum of rewards to termination (Stochastic Shortest Path)

\[ V^\pi(s_t) \equiv E\{r_t + r_{t+1} + r_{t+2}...r_T \} \]

Sum of rewards for next \(N\) time steps

\[ V^\pi(s_t) \equiv E\{r_t + r_{t+1} + r_{t+2}...r_N \} \]

Discounted sum of rewards

\[ V^\pi(s_t) \equiv E\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2}...\}, \quad 0 \leq \gamma < 1 \]

Average reward per step

\[ V^\pi(s_t) \equiv \lim_{n \to \infty} \frac{1}{n} E\{r_t + r_{t+1} + r_{t+2}...r_n \} \]
Be Careful Defining the MDP!

\[
V^\pi(s_t) \equiv r_t + r_{t+1} + r_{t+2} \ldots r_T
\]
Discounted Value Function

\[ V^\pi(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots \gamma^T r_T, \quad \gamma = 0.9 \]
Infinite Horizon (Continuing) Tasks

Checkers

Episodic

Continuing

MDP terminates

$S_{T-2} \xrightarrow{r_{T-2}} S_{T-1} \xrightarrow{r_{T-1}} S_T$

Absorbing state

$r = 0$

unify episodic and continuing
Model Free Learning

Model \( s_{t+1} = \delta(s_t, a_t) \) and \( r_t = r(s_t, a_t) \)

The transition and reward function may not be known by the agent.

But, in the Value Iteration algorithm

\[
V(s) \leftarrow \max_a [ r(s, a) + V(\delta(s, a)) ]
\]

Major breakthrough is to learn the action value function \( Q(s, a) \) instead of the value function \( V(s) \)

Define \( Q^*(s, a) \equiv r(s, a) + V^*(\delta(s, a)) \)

\[
\begin{array}{ccc}
0 & 1 & 2 \\
\$97 & \$98 & \$99 \\
3 & 4 & 5 \\
\$98 & \$99 & \$100 \\
6 & 7 & 8 \\
\$97 & \$98 & \$99 \\
\end{array}
\]

\[
Q(3, \text{N}) = -1 + 97 = 96 \\
Q(3, \text{E}) = -1 + 99 = 98 \\
Q(3, \text{S}) = -1 + 97 = 96 \\
Q(3, \text{W}) = -1 + 98 = 97 \\
\]

\[\pi^*(s) = \arg\max_a Q^*(s, a)\]

\[\pi^*(3) = \text{E}\]
Q-Learning (Temporal Difference)

Training Rule: \( Q(s, a) \leftarrow r + \max_{a'} Q(s', a') \)
Training Rule:  \( Q(s, a) \leftarrow r + \max_{a'} Q(s', a') \)
Training Rule to Learn $Q(s,a)$

$s, a \rightarrow r, s'$

$$Q(s, a) \leftarrow r + \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow (1 - \alpha_n) Q(s, a) + \alpha_n [r + \gamma \max_{a'} Q(s', a')]$$
Q Learning

initialise $Q(s, a) = 0$ for all $s$ and $a$

observe current state $s$

repeat

select an action $a$ and execute it

observe immediate reward $r$ and next state $s'$

$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$

$s \leftarrow s'$

$\pi^*(s) = \arg \max_a Q^*(s, a)$
Exploration vs Exploitation
Some Exploration Policies

- **Greedy**: Choose the action with the highest estimated value
- $\varepsilon$-greedy: Behave greedily most of the time, but occasionally, with probability $\varepsilon$, choose an action at random.
- **Softmax**: Vary the probability of action as a graded function of estimated value:
  \[
  \Pr(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}
  \]
  \[
  \tau = \text{temperature parameter}
  \]
- **Random**
Eligibility Traces

\[
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\]

\[
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\]
Function Approximation
(Generalising from Examples)

What if we have continuous or large state and/or action values?

• We may have an infinite number of states and/or actions!!
• Use a kind of generalisation called function approximation.
  – State Aggregation: States are grouped together, with one table entry (value estimate) used for the group.
  – Linear Methods, eg as introduced in Ch1 ML (Mitchell), page 9 for checkers
  – Instance based methods: value approximated by nearest neighbour
  – Artificial neural networks
  – Others, eg tile coding or CMAC (cerebella module articulator controller)

solution may not converge!
Example: Pole Balancing

Markov State
- pole angle (A)
- pole angular velocity (\( \dot{A} \))
- cart position (X)
- cart velocity (\( \dot{X} \))

Actions
- push left
- push right
Pole Balancing: Function Approximation

reward = -1.0

A = 1

+/- 0.017 radians
+/- 0.1 radians
+/- 0.2 radians

X = 0 1 3

1.4 2.0 1.4
Pole Balancing

- State: continuous, discretised to $|A|=6$, $|\dot{A}|=4$, $|X|=3$, $|\dot{X}|=4$. Total $|S|=288$
- Actions: push left, push right
- Reward: -1.0 if leaning too much or run off track, 0.0 otherwise
- Infinite horizon
- Model free
- Q learning (one step backups)
- Optimality criteria: maximise sum of discounted future rewards
- Parameters:
  - learning rate (alpha) fixed at .2
  - discount rate (gamma) = 0.99
Hidden state
New York City Driving

Ref: Andrew McCallum

New action = 2
Reward = 0.1
Hear horn = 1
Gaze object = 1
Gaze side = 0
Gaze direction = 0
Gaze speed = 0
Gaze distance = 0
Gaze ref. dist = 0
Gaze colour = 0
Last action = 0

time = 90443
nn (#nodes used) = 10808
ln (#leaf nodes or states) = 133

Before learning - Mean Scrapes 75 per 1000
Before learning - Mean Horn 240 per 1000
Co-Evolution
The Taxi Domain with Fuel

6500 states
Raymond Sheh’s’s Flipper 2005

Untrained
Trained
MENACE
RL Examples

- Trial and Error [Michie, 1961]
- TD-Gammon [Tesauro, 1995]
- Elevator Dispatching [Crites & Barto, 1996]
- Dynamic Channel Allocation [Singh and Bertsekas, 1997]
- KeepAway Soccer [Stone and Sutton, 2001]
References


*Reinforcement Learning, An Introduction*
Richard R Sutton and Andrew G. Barto,
MIT Press 1998
Issues with ‘Flat’ RL

- Long time to converge.
- Huge memory requirements
- Poor generalisation over states and actions
- Hidden State

Remainder of this lecture will look at how hierarchical RL can address some of these issues.
Repetitive Models
An Illustrative Example
Objective: Minimise distance to goal
Value Function

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th></th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Basis of HRL

• Abstract Actions (or temporally extended actions)
  – Macros
  – Options (Sutton, R.S., Precup, D.)
  – Machines (R. Parr - HAMQ)
  – Sub-tasks (T. Dietterich - MAXQ)

• Semi-MDPs: MDPs where the actions are generalised to be temporally extended

• State abstraction
Options \( \langle I, \pi, \beta \rangle \)
HAMQ

Machine (e.g., Stochastic Finite State Automata)

Reduced Abstract Machine
MAXQ

a typical room
Showing possible terminations
MAXQ
task hierarchy
The Abstract Problem

Room 0

Room 1

Room 2
Create Top Level Abstract MDP

Level 2
(Colour)

- 3 rooms
- Goal

A typical room

Level 1
(Position)

- Rooms become abstract states
- Room leaving policies become abstract actions

1. Subtask reuse
2. Value Function decomposition
3. State abstraction
Hand Simulation

Initial state =
(red room, position 15)

Room level action =

argmin \{Q^2(\text{top, red, exit}) \text{plus min } Q^1(\text{exit,15, a})\}

Choose abstract action:
leave-room-at-bottom

Invoke leave-room-at-bottom sub-MDP

Robot executes sub-MDP policy until it exits
Hand Simulation

Control is passed back up to top level MDP.

Decide next room leaving action to minimise total cost to goal

Choose leave-room-to-right action and invoke this sub-MDP.

Execute sub-MDP exiting room to right to complete the problem.
HRL and Optimality

goal
Hierarchically Optimal
(not globally optimal)
Recursively Optimal
(not globally optimal or hierarchically optimal)
Optimal?
Hierarchical Pole Balancing & Translation

Markov State
- pole angle (\(A\))
- pole angular velocity (\(\dot{A}\))
- cart position (\(X\))
- cart velocity (\(\dot{X}\))

Actions
- push left
- push right
Pole Translation

40m
Pole Translation

+ve velocity

-ve velocity

-20 meters +20 meters
Towers of Hanoi

Start

Goal

(a)

(b)

(c)

discs

pegs
ToH Hierarchical Solution

Diagram showing the comparison between Flat and Hierarchical solutions over different trials.
Simulated Soccer Agent

Sensor = (ball position, ball distance, robot stance)

Goal = Only Reward

Actions = \{move legs, turn\}
Robot-Stance Variable

State = (ball-position-on-field, robot-relative-to-ball, robot-stance)

- 2 legs
- 4 settings per leg
- 6 directions on field
- 4 positions per direction

384 state values
Sensor =
(robot relative to ball, ball position on field, robot stance)
Sensor =
(robot relative to ball, ball position on field, robot stance)
Task Hierarchy using HEXQ
Hierarchical RL (HEXQ)

Learning to walk and turn

Learning to kick

Learning to shoot goals

MDP solved
Robot Soccer MDP Q Values

**Flat MDP**
|States| = 600 x 2501 x 384 = 5.76 x 10e8
|Actions| = 21
|Q values| = 1.21 x 10e10

**HEXQ**
Level 1: 384 states x 21 actions x 6 sub-MDPs
Level 2: 2501 states x 6 actions x 6 sub-MDPs
Level 3: 600 states x 6 actions
|Q values| = 1.42 x 10e5

Saving: 5 orders of magnitude
Abstraction Hierarchy

Concept
- Dribble ball on field

Subgoal
- Kick goal

Ball approach
- kick in direction

Gait and turn
- Step in direction
Infinite Horizon or Continuing HRL
HRL Approximations

- Hierarchical depth of value function
- Coarseness of subtask termination conditions
Racing Cars

Adam Schuck
Low-level Learner

- Racetracks are composed from simple pieces
- Lower level treats track segment as MDP
- Determines optimal policy for different exit strategies (position, angle, speed)
High-level Learner

- Determine optimal policy for abstracted SMDP
High-level Learner

- Learns how to use low level skills appropriately
- Context has large effect on which policy to use in a particular segment:
Two Taxi Task
Two Taxi Task Example

Environment

Percept = <t1,t2,p1,d1,p2,d2>, Primitive actions A=<a1,a2>, a1,a2 in {N,S,E,W,U1,U2,D,0}
Two Taxi Task
LEAR: Machine learn strategies

The distinguishing research features of this project are

(1) the study of operational art and

(2) machine learning and representation.
A LEAR Task Hierarchy

Effects (root)

Attacking force

Defending force

Target(t)

Release weapon

Max intercept

Target dist

Out mission

In mission

Avoid threats

Take off

Straight Nav(p)

Land

Tarmac wait

Shoot

Strike mission

Mission(t)

Defend(d)

Diversionary(t)

CAP(d)

Intercept(j)

CAP intercept
AIBO Visual Sensor

176 pixels

1 pixel

{256 y values x 256 u values x 256 v values}

$25^{(144 \times 176)^{256^3}}$
Effector

Controlled in micro-radians
125 times a second

Head: pan, tilt, yaw

Legs: shoulder, hip, knee
Fukuoka, Japan 2002
Selective Perception & Serialise Subgoals

Weakly coupled environment

60 orders of magnitude saving.
RoboCup

Diagram showing the following:
- Strategies
- Roles
- Skills
- Vision
- Localisation
- Locomotion
- Frame
- Wireless
- Motors
- Field
- Robots
Hierarchical Appearance Based Object Recognition

Ref: Randal Nelson

Tom Vogelgesang 2004
Wen Tao Lu 2005
Agent sensors are diectic by default and ground the agent primitive sensor array of variables.

- **Spatial Sensor Arrangement**
  - colour, shape, texture categories
  - Visual components
  - syllable
  - spatial relation

- **Visual components**
  - movement, size position spaces

- **Spatial Sensor Arrangement**
  - goal
  - higher level task
  - distance relation

- **Maps (Memory)**
  - higher level task

- **Deliberative**
  - goal
  - distance relation

- **Relational representations**
  - higher level task

- **Abstraction level**
  - syllable
  - House

**Learning Grounded Dynamic Conceptual Hierarchies**

- **primitive sensor array of variables**
- Agent sensors are diectic by default and ground the agent.
Hierarchical Reinforcement Learning

References:


Andrew Barto and Sridhar Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning", volume 13, Special Issue on Reinforcement Learning, Discrete Event Systems journal, pp. 41-77, 2003


Reinforcement Learning with Hierarchies of Machines, Ronald Parr and Stuart Russell. NIPS 97.
The Next Step