11. Reinforcement Learning

Varun Kanade

University of Oxford
March 9, 2016
Textbooks

  - Available online (html format)
  - Draft second edition

  - Available online
  - Terse and mathematical

- Course by David Silver (lectures on youtube)
Outline

Overview of reinforcement learning.

- Formulation - difference from other learning paradigms
- Markov Decision Processes
- Reward, Return, Value function, Policy
- Algorithms for policy evaluation and optimisation
Reinforcement Learning

How is RL different from other paradigms in ML?

- No supervisor, only a reward signal
- Unlike unsupervised learning not really looking for hidden structure; goal is to maximise reward
- Feedback may be delayed, long-term effects of actions
- Data is sequential and not i.i.d.; time plays an important role
- Tradeoff between exploration and exploitation
Examples of Reinforcement Learning

- Beat world champion at Go
- Fly helicopter and perform stunts [video]
- Make(?) money on the stock market
- Make robots walk
- Play video games
Reward

- A reward $R_t$ at time $t$ is a scalar signal
- Indicates performance of agent at time $t$
- Agent’s goal is to maximise cumulative reward

**Reward Hypothesis**

All goals can be described by the maximisation of expected cumulative reward
Examples of Reward

Playing Go
- $1 million for winning
- -$1 million for losing

Flying a helicopter
- Positive reward for doing tricks
- Negative reward for crashing

Investing on the stock market
- $$$
Reinforcement Learning: Sequential Decision Making

**Goal:** Select actions to maximise total future reward

Actions have long term consequences

Reward may be delayed

At times, it may be imperative to sacrifice immediate reward to get long-term reward

**Examples**
- Blocking an opponent’s move, sacrificing a rook
- Financial investment
- Refuelling a helicopter
- Getting oxygen in seaquest [video]
Agent and Environment

- $t$ denotes discrete time

- At time step $t$ the agent does the following:
  - Receive reward $R_t$ (from the previous step)
  - Observe state $S_t$
  - Execute action $A_t$

- At time step $t$ the environment “does” the following:
  - Update state to $S_{t+1}$ based on action $A_t$
  - Emit reward $R_{t+1}$
A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$

- $S$ is a finite set of states
- $A$ is a finite set of actions
- $P$ is a state transition probability matrix
  \[ P_{s,s'}^a = \Pr[S_{t+1} = s' \mid S_t = s, A_t = a] \]
- $R$ is a reward function
  \[ R_s^a = \Pr[R_{t+1} = r \mid S_t = s, A_t = a] \]
- $\gamma \in [0, 1]$ is a discount factor

Are real-world problems really Markovian?
Example: Student MDP

Source: David Silver
Components of an RL agent

**Goal:** Maximise expected cumulative discounted future reward

\[
G_t = \mathbb{E}\left[\sum_{j=1}^{\infty} \gamma^{j-1} R_{t+j}\right]
\]

**Model:** Agent’s representation of the environment (transitions)

**Policy:** How the agent chooses actions given a state

**Value Function:** How much long term reward can be achieved from a given state
Why discount?

Why should we consider discounted reward rather than just add up?

- Mathematically more convenient formulation to deal with
- Don’t have infinite returns because of positive reward cycles in MDP
- Captures the idea that future may be “uncertain”
- Bird in hand vs two in bush (especially true for monetary reward)
- Can use $\gamma = 1$ if MDP is episodic
Model

- A model helps predict future states and rewards given current state and action

- $\mathcal{P}$ (stochastically) determines the next state

  $$\mathcal{P}_{s,s'}^a = \Pr[S_{t+1} = s' \mid S_t = s, A_t = a]$$

- $\mathcal{R}$ (stochastically) determines the immediate reward

  $$\mathcal{R}_s^a = \Pr[R_{t+1} = r \mid S_t = s, A_t = a]$$

- Model-based reinforcement learning (planning)

- Model-free reinforcement learning (trial and error)
Policy

- A **policy** describes the agent’s behaviour or strategy
- Map each state to an action
- **Deterministic Policy**: $a = \pi(s)$
- **Stochastic Policy**: $\pi(a \mid s) = \Pr[A_t = a \mid S_t = s]$
Reinforcement Learning: Prediction and Control

Prediction

- Policy is fixed
- Evaluate the future reward (return) from each state

Control

- Find the policy that maximises future reward
Value Function

- Value function defines expected future reward
- Useful for defining quality (goodness/badness) of a given state
- Useful to select a suitable action (policy improvement)

\[ v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] \]
Learning vs Planning

Two fundamental problems in sequential decision making

Reinforcement Learning
- Environment is unknown (observed through trial and error)
- Agent interacts with the environment
- Agent improves policy/strategy/behaviour through this interaction

Planning
- Model of the environment is known
- Agent does not play directly with the environment
- Compute good (optimal) policy/strategy through simulation, reasoning, search
Value Function and Action-Value Function

State-Value Function

- The *state-value function* \( v_\pi(s) \) for an MDP is the expected return starting from state \( s \), and then following policy \( \pi \)

\[
v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]
\]

Action-Value Function

- The *action-value function* \( q_\pi(s, a) \) for an MDP is the expected return starting from state \( s \), taking action \( a \), and then following policy \( \pi \)

\[
q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]
\]
State-Value Function

- The *state-value function* satisfies the fixed-point equation. It can be decomposed into reward at current time, plus the discounted value at the successor state.

\[ v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \]

Action-Value Function

- The *action-value function* also satisfies a similar relationship.

\[ q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \]
Evaluating a Random Policy in the Small Gridworld

- No discounting, $\gamma = 1$
- States 1 to 14 are not terminal, the grey state is terminal
- All transitions have reward $-1$, no transitions out of terminal states
- If transitions lead out of grid, stay where you are
- **Policy**: Move north, south, east, west with equal probability
### Policy Evaluation

<table>
<thead>
<tr>
<th>$k = 0$</th>
<th>0.0</th>
<th>0.0</th>
<th>0.0</th>
<th>0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k = 1$</th>
<th>0.0</th>
<th>-1.0</th>
<th>-1.0</th>
<th>-1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-1.7</td>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k = 2$</th>
<th>0.0</th>
<th>-1.7</th>
<th>-2.0</th>
<th>-2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.7</td>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>-1.7</td>
<td>0.0</td>
</tr>
<tr>
<td>-2.0</td>
<td>-2.0</td>
<td>-2.0</td>
<td>-1.7</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k = 3$</th>
<th>0.0</th>
<th>-2.4</th>
<th>-2.9</th>
<th>-3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.4</td>
<td>-2.9</td>
<td>-3.0</td>
<td>-2.9</td>
<td>0.0</td>
</tr>
<tr>
<td>-2.9</td>
<td>-3.0</td>
<td>-2.9</td>
<td>-2.4</td>
<td>0.0</td>
</tr>
<tr>
<td>-3.0</td>
<td>-2.9</td>
<td>-2.4</td>
<td>-0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k = 10$</th>
<th>0.0</th>
<th>-6.1</th>
<th>-8.4</th>
<th>-9.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6.1</td>
<td>-7.7</td>
<td>-8.4</td>
<td>-8.4</td>
<td>0.0</td>
</tr>
<tr>
<td>-8.4</td>
<td>-8.4</td>
<td>-7.7</td>
<td>-6.1</td>
<td>0.0</td>
</tr>
<tr>
<td>-9.0</td>
<td>-8.4</td>
<td>-6.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k = \infty$</th>
<th>0.0</th>
<th>-14.0</th>
<th>-20.0</th>
<th>-22.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-14.0</td>
<td>-18.0</td>
<td>-20.0</td>
<td>-20.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-20.0</td>
<td>-20.0</td>
<td>-18.0</td>
<td>-14.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-22.0</td>
<td>-20.0</td>
<td>-14.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Policy Improvement

$k = 0$

\[
\begin{array}{cccc}
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
\end{array}
\]

Random Policy

$k = 1$

\[
\begin{array}{cccc}
0.0 & -1.0 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.0 & -1.0 \\
-1.0 & -1.0 & -1.0 & 0.0 \\
\end{array}
\]

$k = 2$

\[
\begin{array}{cccc}
0.0 & -1.7 & -2.0 & -2.0 \\
-1.7 & -2.0 & -2.0 & -2.0 \\
-2.0 & -2.0 & -2.0 & -1.7 \\
-2.0 & -2.0 & -1.7 & 0.0 \\
\end{array}
\]
Policy Improvement

\[ k = 3 \]

\[
\begin{array}{cccc}
0.0 & -2.4 & -2.9 & -3.0 \\
-2.4 & -2.9 & -3.0 & -2.9 \\
-2.9 & -3.0 & -2.9 & -2.4 \\
-3.0 & -2.9 & -2.4 & 0.0 \\
\end{array}
\]

Optimal Policy

\[ k = 10 \]

\[
\begin{array}{cccc}
0.0 & -6.1 & -8.4 & -9.0 \\
-6.1 & -7.7 & -8.4 & -8.4 \\
-8.4 & -8.4 & -7.7 & -6.1 \\
-9.0 & -8.4 & -6.1 & 0.0 \\
\end{array}
\]

Optimal Policy

\[ k = \infty \]

\[
\begin{array}{cccc}
0.0 & -14. & -20. & -22. \\
-22. & -20. & -14. & 0.0 \\
\end{array}
\]

Optimal Policy
Bellman Optimality Equations

State-Value Function

- The *optimal state-value function* satisfies the fixed point equation.

\[ v_*(s) = \max_{a \in A} \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \]

\[ = \max_{a \in A} q_*(s, a) \]

Action-Value Function

- The *optimal action-value function* also satisfies a fixed point equation.

\[ q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \]

\[ = \mathbb{E}[R_{t+1} + \gamma \max_{a' \in A} q_*(S_{t+1}, a') | S_t = s, A_t = a] \]

Optimal Policy

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

\[ = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \]
Model-Free Reinforcement Learning

- If we know the model, based on optimality equations we can, possibly with great computational effort, solve the MDP to find an optimal policy.

- In reality, we don’t know the MDP but can try to learn $v_*$ and $q_*$ approximately through interaction.

- Monte Carlo Methods

  $$ V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t)) $$

- Temporal difference method, *e.g.*, TD(0)

  $$ V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1} - V(S_t)) $$

- Important to explore as well as exploit.
Reinforcement Learning: Exploration vs Exploitation

- Learning is through trial and error
- Discovering good policy requires diverse experiences
- Should not lose too much reward during exploration
- **Exploration**: Discover more information about the environment
- **Exploitation**: Use known information to maximise reward
Examples: Exploration vs Exploitation

Video Game Playing
- In Seaquest, if you never try to get oxygen, only limited potential for reward

Restaurant Selection
- Try the new American burger place, or go to your favourite curry place?

Online Advertisements
- Keep showing money-making ads or try new ones?
Very Large MDPs

State ($S_{t+1}$)

Reward ($R_{t+1}$)

Action ($A_t$)

Source: David Silver
Function approximation

- Approximate $q_*(s, a)$ using a convnet
- Requires new training procedures

Source: Mnih et al. (Nature 2015)
Summary: What we did

In the past 8 weeks we’ve seen machine learning techniques from Gauss to the present day

**Supervised Learning**
- Linear regression, logistic regression, SVMs
- Neural networks, deep learning, convolutional networks
- Loss functions, regularisation, maximum likelihood, basis expansion, kernel trick

**Unsupervised Learning**
- Dimensionality reduction: PCA, Johnson Lindenstrauss
- Clustering: $\ell$-means, hierarchical clustering, spectral clustering

**Reinforcement Learning**
- MDPs, prediction, control, Bellman equations
Summary: What we did not

ML Topics
▶ Boosting, bagging, decision trees, random forests
▶ Bayesian approaches, graphical models, inference
▶ Dealing with very high-dimensional data
▶ More than half of Murphy’s book

Further Exploration
▶ Lots of online videos, videolectures.net, ML summer schools
▶ ML toolkits: torch, theano, tensor flow, sci-kit learn, R
▶ Conferences: NIPS, ICML, COLT, UAI, ICLR, AISTATS, ACL, CVPR, ICCV
▶ Arxiv: cs.LG, cs.AI, ml-news mailing list, podcasts, blogs, reddit