Lecture 18: Reinforcement Learning
Sanjeev Arora           Elad Hazan

Some slides borrowed from Peter Bodik and David Silver
Course progress

• Learning from examples
  • Definition + fundamental theorem of statistical learning, motivated efficient algorithms/optimization
  • Convexity, greedy optimization – gradient descent
  • Neural networks

• Knowledge Representation
  • NLP
  • Logic
  • Bayes nets
  • Optimization: MCMC
  • HMM

• Today: reinforcement learning part 1
Admin

• (programming) exercise MCMC – announced today
• Due in 1 week in class, as usual
Decisions and planning

• Thus far:
  • Learning from examples
  • Knowledge representation / language
  • inference/prediction
• Missing: actions/decisions
  • Learn from interaction
• RL:
  • no supervisor, only a reward signal
  • Feedback is delayed
  • Time really matters (sequential, non i.i.d data)
  • Agent’s actions affect the subsequent data it receives
RL - examples

• Fly stunt maneuvers in a helicopter
• Defeat the world champion at Backgammon (& Go)
• Control a power station
• Make a humanoid robot walk
• Play Atari games better than humans
Reward hypothesis

• Agent goal: maximize *cumulative* reward
• Hypothesis: *All* goals can be described by the maximization of expected cumulative reward (?)
• Examples:
  • Fly stunt maneuvers in a helicopter:
    +ve reward for following desired trajectory −ve reward for crashing
  • Backgammon:
    +/−ve reward for winning/losing a game
  • Make a humanoid robot walk:
    +ve reward for forward motion −ve reward for falling over
  • Play many different Atari games:
    +/−ve reward for increasing/decreasing score
Sequential decision making

• Agent takes action
• Nature responds with reward
• Agent sees observation
• Agent has internal state (from all previous observations)
  \[ s_t = f(H_t), \quad H_t = \{o_1, r_1, a_1, \ldots, o_{t-1}, r_{t-1}, a_{t-1}, o_t, r_t\} \]
• Markovian assumption: state, observation, reward are independent on past given current state
  \[ \Pr[s_t|s_{t-1}] = \Pr[s_t|s_1, \ldots, s_{t-1}] \]
Markovian?

• State? Actions? Rewards?
Robot in a room

actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT

reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

• states
• actions
• rewards

• what is the solution?
Is this a solution?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>→</td>
<td>→</td>
<td></td>
</tr>
<tr>
<td>↑</td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>↑</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- only if actions deterministic
  - not in this case (actions are stochastic)

- solution/policy
  - mapping from each state to an action
Optimal policy

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>→</td>
<td>→</td>
<td></td>
</tr>
<tr>
<td>↑</td>
<td>↑</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>↑</td>
<td>←</td>
<td>←</td>
<td>←</td>
</tr>
</tbody>
</table>
Reward for each step -2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reward for each step: -0.1

<table>
<thead>
<tr>
<th>→</th>
<th>→</th>
<th>→</th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td></td>
<td>↑</td>
<td>-1</td>
</tr>
<tr>
<td>↑</td>
<td>→</td>
<td>↑</td>
<td>←</td>
</tr>
</tbody>
</table>
Reward for each step: -0.04

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reward for each step: -0.01

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>⇒</td>
<td>⇒</td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>↑</td>
<td></td>
<td>←</td>
<td>-1</td>
</tr>
<tr>
<td>↑</td>
<td>←</td>
<td>←</td>
<td>↓</td>
</tr>
</tbody>
</table>
Reward for each step: +0.01

<table>
<thead>
<tr>
<th></th>
<th>←</th>
<th>←</th>
<th>←</th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td>-1</td>
</tr>
<tr>
<td>↓</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
</tr>
<tr>
<td>←</td>
<td>←</td>
<td>←</td>
<td>←</td>
<td></td>
</tr>
</tbody>
</table>
Formal model: Markov Decision Process

• States: Markov Process (chain)
• Rewards: Markov Reward Process
• Decisions: Markov Decision Process
Markov Process: the student chain

- Facebook (FB)
- Class (C)
- Party (P)
- Sleep (S)

Transition Probabilities:

<table>
<thead>
<tr>
<th></th>
<th>FB</th>
<th>C</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB</td>
<td>0.9</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>0.5</td>
<td></td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Example: the student chain

- Example of episodes (random walks):
  - C C Fb Fb C P S
  - C Fb Fb Fb Fb C P S
Markov Reward Process: the student REWARD chain

Facebook R=-1
Class R=2
Sleep R=0
Party R=1

Transition Probabilities:

<table>
<thead>
<tr>
<th></th>
<th>FB</th>
<th>C</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB</td>
<td>0.9</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>0.5</td>
<td></td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Example: the student REWARD chain

Markov Reward Process, definition:
• Tuple \((S, P, R, \gamma)\) where
  • \(S\) = states, including start state
  • \(P\) = transition matrix \(P_{ss'} = \Pr[S_{t+1} = s'|S_t = s]\)
  • \(R\) = reward function, \(R_S = E[R_{t+1}|S_t = s]\)
  • \(\gamma \in [0,1]\) = discount factor

• Return
  \[ G_t = \sum_{i=1}^{\infty} R_{t+i} \gamma^{i-1} \]

• Exponentially diminishing returns
  why?
• \(\gamma = 0? \gamma = 1?\)
• With discount factor < 1 \(\rightarrow G_t\) always well defined, regardless of stationarity
Example: the student **REWARD** chain

- Example of episode (random walks): discount factor = $\frac{1}{2}$

  - **C C Fb Fb S**
  
  total reward =
  
  $$G_1 = 2 + 2 \times \frac{1}{2} + (-1) \times \left(\frac{1}{2}\right)^2 + (-1) \times 0.5^3 + 0 = 2 + 1 - \frac{1}{4} - \frac{1}{8}$$

  $$= 3 - 0.365 = 2.635$$

- **C Fb Fb Fb C P S**

  $$G_1 = ?$$
The Value function

- Mapping from states to real numbers:

\[ \nu(s) = E[G_t | S_t = s_t] \]
Value function, $\gamma = 0$

$$v(s) = E[G_t | S_t = s_t]$$

$$G_t = \sum_{i=1}^{\infty} R_{t+i} \gamma^{i-1}$$
Computing the value function

• How can we compute it?

\[ v(s) = E[G_t | S_t = s_t] \]
The Bellman equation for MRP

$$v(s) = R_s + \gamma \sum_{s'} P_{ss'} v(s')$$

$$R_s = E[R_{t+1} | S_t = s]$$

$$P_{ss'} = \text{transition probability from } s \text{ to } s'$$
The Bellman equation for MRP

\[ v(s) = E[G_t | S_t = s] \]
\[ = E \left[ \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i} | S_t = s \right] \]
\[ = E \left[ R_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+1+i} | S_t = s \right] \]
\[ = E[R_{t+1} + \gamma G_{t+1} | S_t = s] \]
\[ = E[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \]
\[ = R_s + \gamma \sum_{s'} P_{ss'} v(s') \]
Bellman equation in matrix form

• How can we compute it?

\[
v(s) = R_s + \gamma \sum_s P_{ss'} v(s')
\]

\[
v = R + \gamma P v
\]

For \(v\) being the vector of values \(v(s)\), \(R\) being vector in same space of \(R(s)\), \(\forall s \in S\), and \(P\) being the transition matrix. Thus,

\[
v = (I - \gamma P)^{-1} R
\]

System of linear equations (Gaussian elimination, cubic time)
Markov Decision Process

Markov Reward Process, definition:
- Tuple $(S, P, R, A, \gamma)$ where
  - $S =$ states, including start state
  - $A =$ set of possible actions
  - $P =$ transition matrix $P_{ss'} = \Pr[S_{t+1} = s'|S_t = s, A_t = a]$
  - $R =$ reward function, $R_{ss'} = E[R_{t+1}|S_t = s, A_t = a]$
  - $\gamma \in [0,1] =$ discount factor

- Return

$$G_t = \sum_{i=1}^{\infty} R_{t+i} \gamma^{i-1}$$

- Goal: take actions to maximize expected return
Policies

The Markovian structure ➔ best action depends only on current state!

- Policy = mapping from state to distribution over actions
  \[ \pi: S \mapsto \Delta(A), \quad \pi(a|s) = \Pr[A_t = a|S_t = s] \]

- Given a policy, the MDP reduces to a Markov Reward Process
Reminder: MDP1

actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT

reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

• states
• actions
• rewards
• Policies?
Reminder 2

• State? Actions? Rewards? Policy?
Policies: action = study

- Facebook (R=-1)
- Class (R=2)
- Sleep (R=0)
- Party (R=1)
Policies: action = facebook

- Facebook: R=-1
- Class: R=2
- Sleep: R=0
- Party: R=1
Fixed policy $\rightarrow$ Markov Reward process

- Given a policy, the MDP reduces to a Markov Reward Process

  - $P_{ss'}^\pi = \sum_{a \in A} \pi(a | s) P_{ss'}^a$
  - $R_s^\pi = \sum_{a \in A} \pi(a | s) R_s^a$

- Value function for policy: $v_\pi(s) = E_\pi[G_t | S_t = s]$
- Action-Value function for policy: $q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$

- How to compute the best policy?
The Bellman equation

• Policies satisfy the Bellman equation:

\[
\nu_\pi(s) = E_\pi [R_{t+1} + \gamma \nu_\pi(S_{t+1}) | S_t = s] = R^\pi_s + \gamma \sum_s P^\pi_{ss'} \nu_\pi(s')
\]

• And similarly for value-action function:

\[
\nu_\pi(s) = \sum_{a \in A} \pi(a | s) q_\pi(s, a)
\]

• Optimal value function, and value-action function

\[
\nu^*(s) = \max_\pi \{\nu_\pi(s)\} \quad q^*(s, a) = \max_\pi \{q_\pi(s, a)\}
\]

• Important: \( \nu^*(s) = \max_a q^*(s, a) \), why?
Theorem

• There exists an optimal policy $\pi_*$ (it is deterministic!)

• All optimal policy achieve the same optimal value $v_*(s)$ at every state, and the same optimal value-action function $q_*(s, a)$ at every state and for every action.

• How can we find it? Bellman equation: $v_*(s) = \max_a \{q_*(s, a)\}$ implies Bellman optimality equations:

$$q_*(s, a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} \{q_*(s', a')\}$$

$$v_*(s) = \max_a \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s') \right\}$$
Summary

• Markov Reward Process – generalization of Markov Chains

• Markov Decision Processes – formalization of learning with state from environment observations in a Markovian world.

• Bellman equation: fundamental recursive property of MDPs

• Will enable algorithms (next class...)