Week 5, Lecture 1

Reinforcement Learning

Announcements:
HW 3 due on 11/5 at NOON

Suggested reading:
RL survey paper (Kaelbing)
Chapters 1-3 in Reinforcement Learning (Sutton and Barto)

(Lecture notes based on “Reinforcement Learning”, Sutton and Barto, 1998)
- The Reinforcement Learning Concept
- N-armed Bandit Problem
- Value Functions
- Markov Decision Processes

- Monte Carlo Methods
- Sarsa Learning
- Q-Learning
Reinforcement Learning Concept

- Learn from interactions with the environment
  - Take action
  - Receive feedback from the environment
  - Modify your behavior
  - Achieve some goal

- Examples:
  - Baby playing
    o Connection to the environment guides sensory input/output
  - Driving a car
    o Learn from interaction
• What if reward is not immediate?
  – Reward/value for sequences of actions
  – How do I assign/blame to sequences of actions

• Do I choose best reward?
  – What about potential for better reward of unexplored action
  – Exploration/Exploitation
    ○ Explore: try a new action
    ○ Exploit: repeat (so far) best action
Agent Environment Interaction in RL

\[ s_t \xrightarrow{a_t} s_{t+1}, r_t \]

Kagan Tumer  
Oregon State University
Agent Environment Interaction in RL

\[ S_t \times a_t \rightarrow S_{t+1}, r_t \]

Policy, \( \pi_t \): map states to probabilities of taking actions:

\[ \pi_t(s,a) = P(a_t = a \mid s_t = s) \]

N-Armed Bandit

- Think multiple armed slot machine

  - N actions
  - Each slot has a reward (randomly chosen from a distribution)
  - Aim: maximize long term reward
    - Which arm to pull?
• Think multiple armed slot machine

- N actions
- Each slot has a reward (randomly chosen from a distribution)
- Aim: maximize long term reward
  - Which arm to pull?

- Actions: pick an arm
- Action-value: Long term payoff of a particular action
- Immediate reward: reward of this time step

Let’s say the actual value of an action is $Q'(a)$
If action $a$ has been sampled $k_a$ times
  - The estimate at time step $t$ is $Q_t(a)$:
• Let’s say the actual value of an action is $Q^*(a)$
• If action $a$ has been sampled $k_a$ times
  — The estimate at time step $t$ is $Q_t(a)$:

$$Q_t(a) = \frac{r_1 + r_2 + r_2 + \cdots + r_{k_a}}{k_a}$$

— By the law of large numbers

$$\lim_{t \to \infty} Q_t(a) \to Q^*(a)$$
Computing Q values

• For large values of $t$:
  – Need to keep all values in memory.
  – How about we do this incrementally?

$$Q_{k+1} = \frac{r_1 + r_2 + r_3 + \cdots + r_{k+1}}{k+1}$$
$$= \frac{1}{k+1} \left( r_{k+1} + \sum_{i=1}^{k} r_i \right)$$

$$Q_{k+1} = \frac{r_1 + r_2 + r_3 + \cdots + r_{k+1}}{k+1}$$
$$= \frac{1}{k+1} \left( r_{k+1} + \sum_{i=1}^{k} r_i \right)$$

$$= \frac{1}{k+1} \left( r_{k+1} + kQ_k \right)$$
Computing Q values

• For large values of $t$:
  – Need to keep all values in memory.
  – How about we do this incrementally?
  
  – $Q_{k+1}$: average at k+1
  – $Q_k$: average after k

\[ Q_{k+1} = \frac{r_1 + r_2 + r_3 + \cdots + r_{k+1}}{k+1} \]

\[ = \frac{1}{k+1} \left( r_{k+1} + \sum_{i=1}^{k} r_i \right) \]

\[ = \frac{1}{k+1} \left( r_{k+1} + kQ_k \right) \]

\[ = \frac{1}{k+1} \left( r_{k+1} + kQ_k + Q_k - Q_k \right) \]

\[ = \frac{1}{k+1} \left( r_{k+1} + (k + 1)Q_k - Q_k \right) \]
Summary

- **Result:**

\[
Q_{k+1} = Q_k + \frac{1}{k+1} (r_{k+1} - Q_k)
\]

- **Key Reinforcement Learning Concept:**

\[
NewEstimate \leftarrow OldEstimate + \text{Stepsize} \ (Target - OldEstimate)
\]

What if process is non-stationary?

- **How about a fixed “stepsize”:**

\[
Q_{k+1} = Q_k + \alpha (r_{k+1} - Q_k)
\]
What if process is non-stationary?

• How about a fixed “stepsize”:

\[ Q_{k+1} = Q_k + \alpha(r_{k+1} - Q_k) \]

• Impact:
  – New observations count more
  – Slowly forget old info
  – Track non-stationary process

Example: N-Armed Bandit

• Given Q-values at time t:

| 0.1 | 0.4 | 0 | 0.05 | 0.1 | 0 | 0.05 | 0.1 | 0 | 0.2 |

  – Greedy selection:
    o Action 2 (highest Q value)

  – \( \varepsilon \)-greedy selection:
    o Action 2 with probability \((1-\varepsilon)\)
    o Another action with probability \(\varepsilon\)

  – Softmax:
    o Each action with probability based on it’s Q-value
Example: N-Armed Bandit, Softmax

- Given Q-values at time $t$:

| 0.1 | 0.4 | 0 | 0.05 | 0.1 | 0 | 0.05 | 0.1 | 0 | 0.2 |

- Softmax:
  - Select each action with probability $p_t(a)$ based on:
    - Temperature $\tau$
    - The Q-value of that action at that time step
    $$p_t(a) = \frac{\exp(Q_t(a)/\tau)}{\sum_b \exp(Q_t(b)/\tau)}$$
    - If $\tau \rightarrow 0$, then greedy selection (highest Q value has probability 1)
    - If $\tau \rightarrow \infty$, then $p(a) = 1/n$ for all actions

---

Initial Values?

- How do we start?
- What are initial Q values?
  - Random
  - Zero
  - Optimistic
    - Encourage exploration of actions not yet taken

---

Kagan Tumer  
Oregon State University
Rewards and Time Horizons

- **Maximize reward**
  - What does this really mean?
  - Maximize expected reward
    - Simplest case: sum of rewards:
      \[ R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \]
      - Makes sense when there is a natural final step \( T \).
    - Episodic task:
      - Break learning into episodes of \( T \) steps.
      - Example: Play a game

Rewards and time horizons

- **What if there is no final time \( T \)?**
  - Maximize discounted sum of rewards
    \[ R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T \]
    \[ = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]
    - A reward received \( k \) step in the future is worth only \( \lambda^{k-1} \) times what it would be worth now
    - \( \lambda < 1 \) means the infinite sum has a finite value as long each reward is bounded
    - \( \lambda = 0 \) means agent is myopic: care only about immediate rewards

Kagan Tumer  
Oregon State University
Markov property:

- A state that retains all the information relevant to future actions and rewards is said to be Markov, or to have the Markov property.

- Key: How you got to a state doesn’t matter.

- Example:
  - Chess board: all relevant info is there

- Important in Reinforcement Learning because decisions and values are based on state

Markov property depends on representation of problem, not the problem

- Example:
  - Pole balancing
    - State: location and velocity of pole tip

    - State: Right, left, middle right, middle left
Markov Property

- Markov property depends on representation of problem, not the problem

- Example:
  - Pole balancing
    - State: location and velocity of pole tip
    - Markov (as an approximation)
  - State: Right, left, middle right, middle left
  - Not Markov

Reinforcement Learning (part 2)

Week 5, Lecture 2

Reinforcement Learning

Announcements:
HW 3 due on 11/5 at NOON
Midterm Exam 11/8

Suggested reading:
Chapters 5-6 in Reinforcement Learning (Sutton and Barto)

(Lecture notes based on “Reinforcement Learning”, Sutton and Barto, 1998)
Markov Decision Processes

- **Markov Decision Process (MDP)**

- **Finite MDP (finite number of states) defined by:**
  - Set of states $S$
  - Set of actions $A$
  - Transition probabilities
    $$P_{ss'}^a = \Pr \{ s_{t+1} = s' | s_t = s ; a_t = a \}$$
  - Rewards
    $$R_{ss'}^a = \mathbb{E} [ r_t | s_t = s ; a_t = a ; s_{t+1} = s' ]$$
  - Transition probabilities and Rewards fully specify an MDP

Policy vs. Value Functions

- We have a chicken and egg problem:
  - Need policy to evaluate value function
  - Need values to refine policy
  - Let’s focus on value functions first
• Value Function for policy $\pi$:

$$V^\pi(s) = E^\pi(R_t \mid s_t = s)$$

• Action-Value Function for policy $\pi$:

$$Q^\pi(s,a) = E^\pi(R_t \mid s_t = s, a_t = a)$$
• Value Function can be estimated from experience
  — Average rewards received so far

• Called *Monte Carlo methods*:
  — Averaging over random samples.
  — If there are many states, you may not be able to keep averages for all the states.
  — Instead use parametrized functions $V(s)$ and $Q(s)$, and update the parameters of those functions.

---

**Optimal Value Functions**

• Value Functions depend on policy.

• Optimal value function is based on optimal policy

• Policy $\pi$ is optimal if its expected return is higher than all other policies $\pi'$:
  $$\pi \succeq \pi' \iff V^\pi(s) \geq V^{\pi'}(s) \quad \forall s \in S$$

• Denote optimal policy by $\pi^*$
Optimal Value Functions

- Optimal state value function:
  \[
  V^*(s) = \max_{\pi} V^\pi(s)
  \]

- Optimal action-value function
  \[
  Q^*(s,a) = \max_{\pi} Q^\pi(s,a)
  \]

Policy Iteration

- Policy evaluation
  - Find \( V^\pi \) for policy \( \pi \)
    (compute value of each state)

- Policy improvement
  - Pick new action \( a \) (not in policy) and then follow policy
  - This is new policy \( \pi' \)

- Back to policy evaluation
  - Find \( V^{\pi'} \) for policy \( \pi' \)
• Policy evaluation requires multiple sweeps through the state space and can be expensive/slow

• Value Iteration:
  – Policy improvement
    o Pick new action a (not in policy) and then follow policy
    o This is new policy $\pi'$
  – Policy evaluation for one sweep
    o Find $V^{\pi'}$ for policy $\pi'$

Policy Iteration Example: Grid world

– Policy:
  o If you are next to goal, go to goal
  o If you are not next to goal, go up
  o If you can’t go up, go right.
Policy Iteration Example

- Policy:
  - If you are next to goal, go to goal
  - If you are not next to goal, go up
  - if you can’t go up, go right.

- Policy

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G</td>
</tr>
</tbody>
</table>

Policy Evaluation (Values):

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
Policy Iteration Example

Is this policy optimal?

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>→</td>
<td>→</td>
<td>→</td>
<td>→</td>
<td>↓</td>
</tr>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>→</td>
</tr>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
</tbody>
</table>
```

```
4 5 6 7 8 9
3 4 5 6 9 10
2 3 4 5 8 9
```

Not an optimal policy

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>→</td>
<td>→</td>
<td>→</td>
<td>→</td>
<td>↓</td>
</tr>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>→</td>
</tr>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
</tbody>
</table>
```

```
4 5 6 7 8 9
3 4 5 6 9 10
2 3 4 5 8 9
```

Policy improvement
- Middle box: Go right

Kagan Tumer
Oregon State University
Policy Iteration Example

- Not an optimal policy

- Policy improvement
  - Middle box: Go right

Kagan Tumer  
Oregon State University

Policy Evaluation:

Kagan Tumer  
Oregon State University
Monte Carlo Policy Evaluation

Initialize:

\( \pi : \) Policy to be evaluated
\( V : \) a state value function

Repeat:

Use \( \pi \) for an episode (\( t \) in an interval \( (0, T) \))
For each state \( s \) observed in episode:

Compute Rewards \( R_t(s) \) for each state

\[ V(s) = \text{average}(R_t(s)) \]

Concerns with Monte Carlo Updates

- Recall Value update:

\[ V(s_t) \leftarrow V(s_t) + \alpha(r(s_t) - V(s_t)) \]

- Need to wait till you get \( r \)

- What if you have a long sequence of actions?
Concerns with Monte Carlo Updates

- Recall Value update:

\[ V(s_t) \leftarrow V(s_t) + \alpha(r(s_t) - V(s_t)) \]

- Need to wait till you get \( r \)

- What if you have a long sequence of actions?

- Replace \( r \) with an estimate

---

Temporal Difference Learning

- Recall Value update:

\[ V(s_t) \leftarrow V(s_t) + \alpha(r(s_t) - V(s_t)) \]
Temporal Difference Learning

- Recall Value update:

\[ V(s_t) \leftarrow V(s_t) + \alpha (r(s_t) - V(s_t)) \]

\[ r(s_t) + \gamma V(s_{t+1}) \]

Actual Reward \hspace{1cm} Estimate of Reward

- Temporal difference learning

\[ V(s_t) \leftarrow V(s_t) + \alpha ((r(s_t) + \gamma V(s_{t+1})) - V(s_t)) \]

Kagan Tumer
Oregon State University
Example of Temporal Difference Learning

- Two state system
  - A and B
  - Observed sequence of actions, rewards and transitions:
    - A, 0, B, 0
    - B, 1
    - B, 1
    - B, 1
    - B, 0
    - B, 1
    - B, 1
    - B, 1
Example of Temporal Difference Learning

- What are the Values of states A and B?

  - Monte Carlo estimate:

Kagan Tumer  
Oregon State University
Example of Temporal Difference Learning

- What are the Values of states A and B?
  - Monte Carlo estimate:
    - $V(A) = 0$
    - $V(B) = (0 + 1 + 1 + 0 + 1 + 1 + 1 + 1) / 8 = .75$

  TD estimate:
Example of Temporal Difference Learning

- What are the Values of states A and B?
  - Monte Carlo estimate:
    - \( V(A) = 0 \)
    - \( V(B) = (0 + 1 + 1 + 0 + 1 + 1 + 1 + 1 + 1) / 8 = .75 \)
  - TD estimate:
    - \( V(B) = .75 \)
    - \( V(A) = .75 \) (A leads to B EVERY TIME)

Example of Temporal Difference Learning

- What are the Values of states A and B?
  - Monte Carlo estimate:
    - \( V(A) = 0 \)
    - \( V(B) = (0 + 1 + 1 + 0 + 1 + 1 + 1 + 1 + 1) / 8 = .75 \)
  - TD estimate:
    - \( V(B) = .75 \)
    - \( V(A) = .75 \) (A leads to B EVERYTIME)
  - TD(\( \lambda \)) estimate (take \( \lambda = .9 \)):
Example of Temporal Difference Learning

- What are the Values of states A and B?

  - Monte Carlo estimate:
    - \( V(A) = 0 \)
    - \( V(B) = \frac{0 + 1 + 1 + 1 + 0 + 1 + 1 + 1 + 1}{8} = .75 \)

  - TD estimate:
    - \( V(B) = .75 \)
    - \( V(A) = .75 \) (A leads to B EVERYTIME)

  - TD(\( \lambda \)) estimate (take \( \lambda = .9 \)):
    - \( V(B) = .75 \)
    - \( V(A) = .675 \) (discount reward because it is still uncertain)

Temporal Difference vs. Monte Carlo

- TD learns from guesses
  - Bootstrap learning

- No need for model of:
  - Environment
  - Reward
  - Transition probability

- Online implementation
  - No need to wait till the end
Sarsa Learning

- Idea: TD with (state,action) pairs rather than state alone

- Why:
  - Some states may have good and bad actions.
  - Let’s not discard good actions because of potential to take a bad action.

- Consider the following landscape:

  ![Diagram of a landscape with rewards: 10, -100, and 20.]

  If you start in region with reward of 10, you will not get to region with reward of 20.

  Why?
  - The penalty of falling off is too high
  - Each state on that bridge will have a negative value

Kagan Tumer  
Oregon State University
• If you start in region with reward of 10, you will not get to region with reward of 20.

• Why?
  – The penalty of falling off is too high
  – Each state on that bridge will have a negative value
• If you start in region with reward of 10, you will not get to region with reward of 20.

• Why?
  – The penalty of falling off is too high
  – Each state on that bridge will have a negative value

• Value $V(s)$ is low.
• What about $s,a$ pairs?

• $Q(s,a_1) = -100$
• $Q(s,a_6) = -100$
• $Q(s,a_3) = positive$

• $Q$ values for state-action pairs:

• Sarsa learning is temporal difference extended to $s,a$:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left( r(s_t,a_t) + \gamma \, Q(s_{t+1},a_{t+1}) - Q(s_t,a_t) \right)$$
Sarsa Learning

- Q values for state-action pairs:

- Sarsa learning is temporal difference extended to s,a:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right)
\]

NewEstimate ← OldEstimate + Stepsize (Target – OldEstimate)
Q-Learning

- What if we update Q values without using policy?

- Q-Learning:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

Actual Reward \hspace{1cm} Estimate of reward

Kagan Tumer Oregon State University
Q-Learning

- What if we update Q values without using policy?

- Q-Learning:

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \]

- Policy independent
  - Q update is based on best possible move
  - Q update does not depend on action taken

Kagan Tumer  
Oregon State University

Questions?