Markov Decision Processes: Making Decision in the Presence of Uncertainty

(some of) R&N 16.1-16.6
R&N 17.1-17.4
Different Aspects of “Machine Learning”

• Supervised learning
  – Classification - concept learning
  – Learning from labeled data
  – Function approximation

• Unsupervised learning
  – Data is not labeled
  – Data needs to be grouped, clustered
  – We need distance metric

• Control and action model learning
  – Learning to select actions efficiently
  – Feedback: goal achievement, failure, reward
  – Control learning, reinforcement learning
Decision Processes: General Description

- Suppose you have to make decisions that impact your future... You know the current state around you.
- You have a choice of several possible actions.
- You cannot predict with certainty the consequence of these actions given the current state, but you have a guess as to the likelihood of the possible outcomes.
- How can you define a policy that will guarantee that you always choose the action that maximizes expected future profits?

Note: Russel & Norvig, Chapter 17.
Decision Processes: General Description

• Decide what action to take next, given:
  – A probability to move to different states
  – A way to evaluate the reward of being in different states

Robot path planning
Travel route planning
Elevator scheduling
Aircraft navigation
Manufacturing processes
Network switching & routing
Example

• Assume that time is discretized into *discrete time steps*
• Suppose that your world can be in one of a finite number of states $s$
  – this is a major simplification, but let’s assume….
• Suppose that for every state $s$, we can anticipate a reward that you receive for being in that state $R(s)$.
• Assume also that $R(s)$ is bounded ($R(s) < M$ for all $s$) meaning that there is a threshold in reward.

• Question: What is the total value of the reward for a particular configuration of states $\{s_1, s_2, \ldots\}$ over time?
Example

• Question: What is the total value of the reward for a particular configuration of states \( \{s_1, s_2, \ldots \} \) over time?

• It is simply the sum of the rewards (possibly negative) that we will receive in the future:

\[
U(s_1, s_2, \ldots, s_n, \ldots) = R(s_1) + R(s_2) + \ldots + R(s_n) + \ldots
\]

What is wrong with this formula???
Horizon Problem

$U(s_0, \ldots, s_N) = R(s_0) + R(s_1) + \ldots + R(s_N)$

Need to know $N$, the length of the sequence (finite horizon)

The sum may be arbitrarily large depending on $N$
Horizon Problem

• The problem is that we did not put any limit on the “future”, so this sum can be infinite.

• For example: Consider the simple case of computing the total future reward if you remain forever in the same state:

$$U(s, s, .., s, ..) = R(s) + R(s) + .. + R(s) + ....$$

is clearly infinite in general!!

• This definition is useless unless we consider a finite time horizon.

• But, in general, we don’t have a good way to define such a time horizon.
Discounting

\[ U(s_0, \ldots) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^N R(s_N) + \ldots \]

Discount factor \( 0 < \gamma < 1 \)

The length of the sequence is arbitrary (infinite horizon)
Discounting

- $U(s_0,\ldots) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^N R(s_N) + \ldots$
- Always converges if $\gamma < 1$ and $R(.)$ is bounded
- $\gamma$ close to 0 $\Rightarrow$ instant gratification, don’t pay attention to future reward
- $\gamma$ close to 1 $\Rightarrow$ extremely conservative, big influence of the future
- The resulting model is the *discounted reward*
  - Prefers expedient solutions (models impatience)
  - Compensates for uncertainty in available time (models mortality)
- Economic example:
  - Being promised $10,000 next year is worth only 90% as much as receiving $10,000 right now.
  - Assuming payment $n$ years in future is worth only $(0.9)^n$ of payment now
Actions

• Assume that we also have a finite set of actions \( a \)

• An action \( a \) causes a transition from a state \( s \) to a state \( s' \)
The Basic Decision Problem

• Given:
  – Set of states \( S = \{s\} \)
  – Set of actions \( A = \{a\} \) \( a: S \rightarrow S \)
  – Reward function \( R(.) \)
  – Discount factor \( \gamma \)
  – Starting state \( s_1 \)

• Find a sequence of actions such that the resulting sequence of states maximizes the total discounted reward:

\[
U(s_0, \ldots) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^N R(s_N) + \ldots
\]
Maze Example: Utility

• Define the reward of being in a state:
  – $R(s) = -0.04$ if $s$ is empty state
  – $R(4,3) = +1$ (maximum reward when goal is reached)
  – $R(4,2) = -1$ (avoid (4,2) as much as possible)

• Define the utility of a sequence of states:
  – $U(s_0,\ldots, s_N) = R(s_0) + R(s_1) + \ldots + R(s_N)$
Maze Example: Utility

- Define the reward of being in a state:
  - \( R(s) = -0.04 \) if \( s \) is an empty state
  - \( R(4,3) = +1 \) (maximum reward when goal is reached)
  - \( R(4,2) = -1 \) (avoid \((4,2)\) as much as possible)

- Define the utility of a sequence of states:
  - \( U(s_0, \ldots, s_N) = R(s_0) + R(s_1) + \ldots + R(s_N) \)

If no uncertainty:
Find the sequence of actions that maximizes the sum of the rewards of the traversed states.
Maze Example: No Uncertainty

- States: locations in maze grid
- Actions: Moves up/left left/right
- If no uncertainty: Find sequence of actions from current state to goal (+1) that maximizes utility → We know how to do this using earlier search techniques
What we are looking for: *Policy*

- **Policy** = Mapping from states to action $\pi(s) = a$
  - Which action should be taken in each state
- In the maze example, $\pi(s)$ associates a motion to a particular location on the grid
- For any state $s$, the utility $U(s)$ of $s$ is the sum of discounted rewards of the sequence of states starting at $s$ generated by using the policy $\pi$
  \[ U(s) = R(s) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots. \]
- Where we move from $s$ to $s_1$ by action $\pi(s)$
- We move from $s_1$ to $s_2$ by action $\pi(s_1)$, etc.
Optimal Decision Policy

• *Policy*
  – Mapping from states to action \( \pi(s) = a \)

• **Optimal Policy**
  – The policy \( \pi^* \) that maximizes the expected utility \( U(s) \) of the sequence of states generated by \( \pi^* \), starting at \( s \)

• In the maze example, \( \pi^*(s) \) tells us which motion to choose at every cell of the grid to bring us closer to the goal
Maze Example: No Uncertainty

- $\pi^*(((1,1))) = \text{UP}$
- $\pi^*(((1,3))) = \text{RIGHT}$
- $\pi^*(((4,1))) = \text{LEFT}$
Maze Example: With Uncertainty

The robot may not execute exactly the action that is commanded → The outcome of an action is no longer deterministic

Uncertainty:
- We know in which state we are (fully observable)
- But we are not sure that the commanded action will be executed exactly
Uncertainty

• No uncertainty:
  – An action \( a \) deterministically causes a transition from a state \( s \) to another state \( s' \)

• With uncertainty:
  – An action \( a \) causes a transition from a state \( s \) to another state \( s' \) with some probability \( T(s,a,s') \)
    – \( T(s,a,s') \) is called the transition probability from state \( s \) to state \( s' \) through action \( a \)
  – In general, we need \( |S|^2 \times |A| \) numbers to store all the transitions probabilities
Maze Example: With Uncertainty

- We can no longer find a unique sequence of actions, but
- Can we find a policy that tells us how to decide which action to take from each state except that now the policy maximizes the expected utility
Maze Example: Utility Revisited

\[ U(s) = \text{Expected reward of future states starting at } s \]

How to compute \( U \) after one step?

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
3 & & & +1 \\
2 & \text{black} & & -1 \\
1 & & & \\
\end{array} \]

Intended action \( a: \)

\[ \begin{array}{c}
\uparrow \\
T(s, a, s') \end{array} \]

\[ \begin{array}{c}
0.8 \\
0.1 \quad 0.1 \\
\end{array} \]
Maze Example: Utility Revisited

Suppose \( s = (1,1) \) and we choose action Up.

\[
U(1,1) = R(1,1) + \]

\[
\int \text{ended action } a: \quad T(s,a,s')
\]

\[
\begin{array}{c|c|c|c|c}
1 & 2 & 3 & 4 \\
\hline
3 & & & +1 \\
\hline
2 & & -1 & \\
\hline
1 & s & & \\
\end{array}
\]

\[
0.1 \quad 0.1 \\
0.8 \\
\]
Suppose $s = (1,1)$ and we choose action Up.

$$U(1,1) = R(1,1) + 0.8 \times U(1,2) +$$
Suppose $s = (1,1)$ and we choose action Up.

$$U(1,1) = R(1,1) + 0.8 \times U(1,2) + 0.1 \times U(2,1) +$$
Maze Example: Utility Revisited

Suppose $s = (1,1)$ and we choose action Up.

\[
U(1,1) = R(1,1) + 0.8 \times U(1,2) + 0.1 \times U(2,1) + 0.1 \times R(1,1)
\]
Suppose $s = (1,1)$ and we choose action Up.

\[ U(1,1) = R(1,1) + 0.8 \times U(1,2) + 0.1 \times U(2,1) + 0.1 \times R(1,1) \]

Move up with prob. 0.8

Move left with prob. 0.1 (notice the wall!)

Move right with prob. 0.1
Suppose $s = (1,1)$ and we choose action Up.

$$U(1,1) = R(1,1) + \gamma (0.8 \times U(1,2) + 0.1 \times U(2,1) + 0.1 \times R(1,1))$$
More General Expression

• If we choose action $a$ at state $s$, expected future rewards are:

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s') U(s')$$
More General Expression

- If we choose action $a$ at state $s$:

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s') U(s')$$

- Reward at current state $s$
- Expected sum of future discounted rewards starting at state $s'$
- Expected sum of future discounted rewards starting at state $s$
- Probability of moving from state $s$ to state $s'$ with action $a$
More General Expression

- If we are using policy $\pi$, we choose action $a=\pi(s)$ at state $s$, expected future rewards are:

$$U_\pi(s) = R(s) + \gamma \sum_{s',T(s,\pi(s),s')} U_\pi(s')$$
Formal Definitions

- Finite set of states: \( S \)
- Finite set of allowed actions: \( A \)
- Reward function \( R(s) \)

- Transitions probabilities: \( T(s,a,s') = P(s'|a,s) \)
- Utility = sum of discounted rewards:
  \[- U(s_0, \ldots) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^N R(s_N) + \ldots \]

- Policy: \( \pi : S \rightarrow A \)
- Optimal policy: \( \pi^*(s) = \) action that maximizes the expected sum of rewards from state \( s \)
Markov Decision Process (MDP)

- Key property (Markov):
  \[ P(s_{t+1} \mid a, s_0, \ldots, s_t) = P(s_{t+1} \mid a, s_t) \]

- In words: The new state reached after applying an action depends only on the previous state and it does not depend on the previous history of the states visited in the past

  → Markov Process
Markov Example

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>s₁</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>s₀</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- When applying the action “Right” from state \(s_2 = (1,3)\), the new state depends only on the previous state \(s_2\), not the entire history \(\{s_1, s_0\}\).
Graphical Notations

- Nodes are states
- Each arc corresponds to a possible transition between two states given an action
- Arcs are labeled by the transition probabilities

$T(s, a_1, s') = 0.8$
$T(s', a_2, s) = 0.6$
$T(s, a_2, s) = 0.2$

$S \xrightarrow{a_1 \text{ Prob. } 0.8} S'$
$S' \xrightarrow{a_1 \text{ Prob. } 0.4} S$

$S \xrightarrow{a_2 \text{ Prob. } 0.2} S$
$S \xrightarrow{a_2 \text{ Prob. } 0.6} S'$
\[ T(s, a_1, s') = 0.8 \]
\[ T(s', a_2, s) = 0.6 \]
\[ T(s, a_2, s) = 0.2 \]
Intended action $a$: $T(s, a, s')$

Warning: The transitions are NOT all shown in this example!
Example

• I run a company
• I can choose to either save money or spend money on advertising
• If I advertise, I may become famous (50% prob.) but will spend money so I may become poor
• If I save money, I may become rich (50% prob.), but I may also become unknown because I don’t advertise
• What should I do?
### Policy Number 1:

<table>
<thead>
<tr>
<th>STATE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>S</td>
</tr>
<tr>
<td>PF</td>
<td>A</td>
</tr>
<tr>
<td>RU</td>
<td>S</td>
</tr>
<tr>
<td>RF</td>
<td>A</td>
</tr>
</tbody>
</table>

![Diagram of state transitions and actions]
Policy Number 2:

<table>
<thead>
<tr>
<th>STATE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>A</td>
</tr>
<tr>
<td>PF</td>
<td>A</td>
</tr>
<tr>
<td>RU</td>
<td>A</td>
</tr>
<tr>
<td>RF</td>
<td>A</td>
</tr>
</tbody>
</table>
Example Policies

- Many policies
- The best policy?
- How to compute the optimal policy?
Key Result

- For every MDP, there exists an optimal policy
- There is no better option (in terms of expected sum of rewards) than to follow this policy

- How to compute the optimal policy? → We cannot evaluate all possible policies (in real problems, the number of states is very large)
Bellman’s Equation

If we choose an action $a$:

$$U(s) = R(s) + \gamma \sum_{s'} T(s, a, s') U(s')$$
Bellman’s Equation

If we choose an action $a$:

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s') U(s')$$

In particular, if we always choose the action $a$ that maximizes future rewards (optimal policy), $U(s)$ is the maximum $U^*(s)$ we can get over all possible choices of actions:

$$U^*(s) = R(s) + \gamma \max_a (\sum_{s'} T(s,a,s') U^*(s'))$$
Bellman’s Equation

\[ U^*(s) = R(s) + \gamma \max_a (\sum_{s'} T(s, a, s') U^*(s')) \]

- The optimal policy (choice of \( a \) that maximizes \( U \)) is:

\[ \pi^*(s) = \arg\max_a (\sum_{s'} T(s, a, s') U^*(s')) \]
Why it cannot be solved directly

\[ U^*(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s,a,s') U^*(s') \right) \]

The optimal policy (choice of \(a\) that maximizes \(U^*)\) is:

\[ \pi^*(s) = \text{argmax}_a \left( \sum_{s'} T(s,a,s') U^*(s') \right) \]

- Set of \(|S|\) equations. Non-linear because of the “max”: Cannot be solved directly!
- Expected sum of rewards using policy \(\pi^*\) → The right-hand depends on the unknown. Cannot solve directly
First Solution: Value Iteration

• Define $U_1(s) = \text{best value after one step} \quad U_1(s) = R(s)$

• Define $U_2(s) = \text{best possible value after two steps}$

\[ U_2(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s,a,s') U_1(s') \right) \]

• Define $U_k(s) = \text{best possible value after } k \text{ steps}$

\[ U_k(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s,a,s') U_{k-1}(s') \right) \]
First Solution: Value Iteration

- Define $U_1(s) = \text{best value after one step}$
  \[ U_1(s) = R(s) \]
- Define $U_2(s) = \text{best value after two steps}$
  \[ U_2(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s,a,s') U_1(s') \right) \]
- Define $U_k(s) = \text{best value after k steps}$
  \[ U_k(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s,a,s') U_{k-1}(s') \right) \]

Maximum possible expected sum of discounted rewards that I can get if I start at state $s$ and I survive for $k$ time steps.
3-State Example – Value Iteration Computation for Markov chain (no policy)
3-State Example: Values $\gamma = 0.5$

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SUN</th>
<th>WIND</th>
<th>HAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>2</td>
<td>5.0</td>
<td>-1.0</td>
<td>-10.0</td>
</tr>
<tr>
<td>3</td>
<td>5.0</td>
<td>-1.25</td>
<td>-10.75</td>
</tr>
<tr>
<td>4</td>
<td>4.9375</td>
<td>-1.4375</td>
<td>-11.0</td>
</tr>
<tr>
<td>5</td>
<td>4.875</td>
<td>-1.515625</td>
<td>-11.109375</td>
</tr>
<tr>
<td>6</td>
<td>4.8398437</td>
<td>-1.5585937</td>
<td>-11.15625</td>
</tr>
<tr>
<td>7</td>
<td>4.8203125</td>
<td>-1.5791016</td>
<td>-11.178711</td>
</tr>
<tr>
<td>8</td>
<td>4.8103027</td>
<td>-1.5895996</td>
<td>-11.189453</td>
</tr>
<tr>
<td>9</td>
<td>4.805176</td>
<td>-1.5947876</td>
<td>-11.194763</td>
</tr>
<tr>
<td>10</td>
<td>4.802597</td>
<td>-1.5973969</td>
<td>-11.197388</td>
</tr>
<tr>
<td>11</td>
<td>4.8013</td>
<td>-1.5986977</td>
<td>-11.198696</td>
</tr>
<tr>
<td>12</td>
<td>4.8006506</td>
<td>-1.599349</td>
<td>-11.199348</td>
</tr>
<tr>
<td>13</td>
<td>4.8003254</td>
<td>-1.5996745</td>
<td>-11.199675</td>
</tr>
<tr>
<td>14</td>
<td>4.800163</td>
<td>-1.5998373</td>
<td>-11.199837</td>
</tr>
<tr>
<td>15</td>
<td>4.8000813</td>
<td>-1.5999185</td>
<td>-11.199919</td>
</tr>
</tbody>
</table>
### 3-State Example: Values $\gamma = 0.9$

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SUN</th>
<th>WIND</th>
<th>HAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>2</td>
<td>5.8</td>
<td>-1.8</td>
<td>-11.6</td>
</tr>
<tr>
<td>3</td>
<td>5.8</td>
<td>-2.6100001</td>
<td>-14.030001</td>
</tr>
<tr>
<td>4</td>
<td>5.4355</td>
<td>-3.7035</td>
<td>-15.488001</td>
</tr>
<tr>
<td>5</td>
<td>4.7794</td>
<td>-4.5236254</td>
<td>-16.636175</td>
</tr>
<tr>
<td>6</td>
<td>4.1150985</td>
<td>-5.335549</td>
<td>-17.521912</td>
</tr>
<tr>
<td>7</td>
<td>3.4507973</td>
<td>-6.0330653</td>
<td>-18.285858</td>
</tr>
<tr>
<td>8</td>
<td>2.8379793</td>
<td>-6.6757774</td>
<td>-18.943516</td>
</tr>
<tr>
<td>9</td>
<td>2.272991</td>
<td>-7.247492</td>
<td>-19.528683</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>50</td>
<td>-2.8152928</td>
<td>-12.345073</td>
<td>-24.633476</td>
</tr>
<tr>
<td>51</td>
<td>-2.8221645</td>
<td>-12.351946</td>
<td>-24.640347</td>
</tr>
<tr>
<td>52</td>
<td>-2.8283496</td>
<td>-12.3581295</td>
<td>-24.646532</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>86</td>
<td>-2.882461</td>
<td>-12.412242</td>
<td>-24.700644</td>
</tr>
<tr>
<td>87</td>
<td>-2.882616</td>
<td>-12.412397</td>
<td>-24.700798</td>
</tr>
<tr>
<td>88</td>
<td>-2.8827558</td>
<td>-12.412536</td>
<td>-24.70094</td>
</tr>
</tbody>
</table>
### 3-State Example: Values $\gamma = 0.2$

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SUN</th>
<th>WIND</th>
<th>HAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>2</td>
<td>4.4</td>
<td>-0.4</td>
<td>-8.8</td>
</tr>
<tr>
<td>3</td>
<td>4.4</td>
<td>-0.44000003</td>
<td>-8.92</td>
</tr>
<tr>
<td>4</td>
<td>4.396</td>
<td>-0.452</td>
<td>-8.936</td>
</tr>
<tr>
<td>5</td>
<td>4.3944</td>
<td>-0.454</td>
<td>-8.9388</td>
</tr>
<tr>
<td>6</td>
<td>4.39404</td>
<td>-0.45443997</td>
<td>-8.93928</td>
</tr>
<tr>
<td>7</td>
<td>4.39396</td>
<td>-0.45452395</td>
<td>-8.939372</td>
</tr>
<tr>
<td>8</td>
<td>4.393944</td>
<td>-0.4545412</td>
<td>-8.939389</td>
</tr>
<tr>
<td>9</td>
<td>4.3939404</td>
<td>-0.45454454</td>
<td>-8.939393</td>
</tr>
<tr>
<td>10</td>
<td>4.3939395</td>
<td>-0.45454526</td>
<td>-8.939394</td>
</tr>
<tr>
<td>11</td>
<td>4.3939395</td>
<td>-0.45454547</td>
<td>-8.939394</td>
</tr>
<tr>
<td>12</td>
<td>4.3939395</td>
<td>-0.45454547</td>
<td>-8.939394</td>
</tr>
</tbody>
</table>
Next

• More value iteration
• Policy iteration