Markov decision processes (MDPs)
The weeks ahead

- Applications of HMM to biology
- Dimensionality reduction
- SVM
- Boosting
- Model and feature selection
Markov decision processes (MDPs)
What’s missing in HMMs

• HMMs cannot model important aspects of agent interactions:
  - No model for rewards
  - No model for actions which can affect these rewards
• These are actually issues that are faced by many applications:
  - Agents negotiating deals on the web
  - A robot which interacts with its environment
Example: No actions

Graduate student
20

Asst. prof
40

Tenured prof.
100

Google
200

On the street
0

Dead
0

0.6
0.2
0.1
0.1
0.6
0.3
0.1
0.1
0.1
0.3
0.1
0.1
0.6
0.7
0.9
1

0.1
Example: Actions

Action A: Leave to Google
Action B: Stay in academia
Formal definition of MDPs

- A set of states \( \{s_1 \ldots s_n\} \)
- A set of rewards \( \{r_1 \ldots r_n\} \)
- A set of actions \( \{a_1 \ldots a_m\} \)
- Transition probability

\[
P^k_{i,j} = P(q_{t+1} = s_j \mid q_t = i \& h_t = a_k)
\]

One reward for each state

Number of actions could be larger than number of states
Questions

- What is my expected pay if I am in state $i$?
- What is my expected pay if I am in state $i$ and perform action $a$?
Solving MDPs

• No actions: Value iterations

• With actions: Value iteration, Policy iteration
Value computation

• An obvious question for such models is what is the combined expected value for each state
• What can we expect to earn over our life time if we become Asst. prof.?
• What if we go to industry?

Before we answer this question, we need to define a model for future rewards:

• The value of a current award is higher than the value of future awards
  - Inflation, confidence
  - Example: Lottery
Discounted rewards

• The discounted rewards model is specified using a parameter $\gamma$

• Total rewards = current reward +

  $\gamma$ (reward at time $t+1$) +
  $\gamma^2$ (reward at time $t+2$) +
  ................
  $\gamma^k$ (reward at time $t+k$) +

  infinite sum
Discounted awards

- The discounted award model is specified using a parameter $\gamma$
- Total awards = current award +
  $\gamma$ (award at time t+1) +
  $\gamma^2$ (award at time t+2) +
  $\ldots$+
  $\gamma^k$ (award at time t+k) +

  infinite sum

Converges if $0 < \gamma < 1$
Determining the total rewards in a state

• Define \( J^*(s_i) \) = expected discounted sum of rewards when starting at state \( s_i \)
• How do we compute \( J^*(s_i) \)?

\[
J^*(s_i) = r_i + \gamma X
= r_i + \gamma (p_{i1}J^*(s_1) + p_{i2}J^*(s_2) + \cdots p_{in}J^*(s_n))
\]

Factors expected pay for all possible transitions for step \( i \)

How can we solve this?
Computing $j^*(s_i)$

\[
J^*(s_1) = r_1 + \gamma(p_{11}J^*(s_1) + p_{12}J^*(s_2) + \cdots p_{1n}J^*(s_n))
\]

\[
J^*(s_2) = r_2 + \gamma(p_{21}J^*(s_1) + p_{22}J^*(s_2) + \cdots p_{2n}J^*(s_n))
\]

\[
J^*(s_n) = r_n + \gamma(p_{n1}J^*(s_1) + p_{n2}J^*(s_2) + \cdots p_{nn}J^*(s_n))
\]

- We have $n$ equations with $n$ unknowns
- Can be solved in close form
Iterative approaches

- Solving in closed form is possible, but may be time consuming.
- It also doesn’t generalize to non-linear models.
- Alternatively, this problem can be solved in an iterative manner.
- Let's define $J^t(s_i)$ as the expected discounted rewards after $t$ steps.
- How can we compute $J^t(s_i)$?

\[
J^1(S_i) = r_i
\]

\[
J^2(S_i) = r_i + \gamma \left( \sum_k p_{i,k} J^1(s_k) \right)
\]

\[
J^{t+1}(S_i) = r_i + \gamma \left( \sum_k p_{i,k} J^t(s_k) \right)
\]
Iterative approaches

- We know how to solve this!

Let's fill the dynamic programming table

Let's define $J^k(s_i)$ as the expected discounted awards after $k$ steps

But wait …

This is a never ending task!

\[
J^2(S_i) = r_i + \gamma \left( \sum_k p_{i,k} J^1(s_k) \right)
\]

\[
J^{t+1}(S_i) = r_i + \gamma \left( \sum_k p_{i,k} J^t(s_k) \right)
\]
When do we stop?

\[
J^1(S_i) = r_i
\]

\[
J^2(S_i) = r_i + \gamma \left( \sum_k p_{i,k} J^1(s_k) \right)
\]

\[
J^{t+1}(S_i) = r_i + \gamma \left( \sum_k p_{i,k} J^t(s_k) \right)
\]

Remember, we have a converging function

We can stop when \(|J^{t-1}(s_i) - J^t(s_i)|_\infty < \varepsilon\)

Infinity norm selects maximal element
Example for $\gamma=0.9$

\[ J^2(\text{Gr}) = 20 + 0.9 \times (0.6 \times 20 + 0.2 \times 40 + 0.2 \times 200) \]

<table>
<thead>
<tr>
<th>t</th>
<th>$J^t(\text{Gr})$</th>
<th>$J^t(\text{P})$</th>
<th>$J^t(\text{Goo})$</th>
<th>$J^t(\text{D})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>40</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>74</td>
<td>87</td>
<td>362</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>141</td>
<td>135</td>
<td>493</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>209</td>
<td>182</td>
<td>600</td>
<td>0</td>
</tr>
</tbody>
</table>
Solving MDPs

• No actions: Value iterations √

• With actions: Value iteration, Policy iteration
Adding actions

A Markov Decision Process:
- A set of states \( \{s_1 \ldots s_n\} \)
- A set of rewards \( \{r_1 \ldots r_n\} \)
- A set of actions \( \{a_1 \ldots a_m\} \)
- Transition probability

\[
P_{i,j}^k = P(q_{t+1} = s_j \mid q_t = i \& h_t = a_k)
\]
Action A: Leave to Google
Action B: Stay in academia
Questions for MDPs

• Now we have actions
• The question changes as follows:

Given our current state and the possible actions, what is the best action for us in terms of long term payment?
Example: Actions

Action A: Leave to Google
Action B: Stay in academia

So should you leave now (right after class) or should you stay in the PhD program?
Policy

- A policy maps states to actions
- An optimal policy leads to the highest expected returns
- Note that this does not depend on the start state

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gr</td>
<td>B</td>
</tr>
<tr>
<td>Go</td>
<td>A</td>
</tr>
<tr>
<td>Asst. Pr.</td>
<td>A</td>
</tr>
<tr>
<td>Ten. Pr.</td>
<td>B</td>
</tr>
</tbody>
</table>
Solving MDPs with actions

• It could be shown that for every MDP there exists an optimal policy (we won’t discuss the proof).
• Such policy guarantees that there is no other action that is expected to yield a higher payoff
Computing the optimal policy:

1. Modified value iteration

- We can compute it by modifying the value iteration method we discussed.
- Define $p_{ij}^k$ as the probability of transitioning from state $i$ to state $j$ when using action $k$.
- Then we compute:

$$
J^{t+1}(S_i) = \max_k r_i + \gamma \left( \sum_j p_{i,j}^k J^t(s_j) \right)
$$

Also known as Bellman’s equation

Use probabilities associated with action $k$
Computing the optimal policy:

1. Modified value iteration

- We can compute it by modifying the value iteration method we discussed.
- Define $p^k_{ij}$ as the probability of transitioning from state $i$ to state $j$ when using action $k$.
- Then we compute:

$$J^{t+1}(S_i) = \max_k r_i + \gamma \left( \sum_j p^k_{i,j} J^t(s_j) \right)$$

Run until convergences
Computing the optimal policy: 1. Modified value iteration

- We can compute it by modifying the value iteration method we discussed.
- Define \( p^k_{ij} \) as the probability of transitioning from state \( i \) to state \( j \) when using action \( k \).
- Then we compute:

\[
J^{t+1}(S_i) = \max_k \left( r_i + \gamma \sum_j p^k_{i,j} J^t(s_j) \right)
\]

- When the algorithm converges, we have computed the best outcome for each state.
- We associate states with the actions that maximize their return.
Value iteration for $\gamma=0.9$

$J^2(Gr) = 20 + 0.9 \times \max \{0.2 \times 20 + 0.8 \times 200, 0.7 \times 20 + 0.3 \times 40\}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$J^t(Gr)$</th>
<th>$J^t(P)$</th>
<th>$J^t(Goo)$</th>
<th>$J^t(D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>40</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>168(A)</td>
<td>87</td>
<td>362</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>51(B)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>311(A)</td>
<td>135</td>
<td>493</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>149(B)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>431(A)</td>
<td>182</td>
<td>600</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>252(B)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Computing the optimal policy: 2. Policy iteration

- We can also compute optimal policies by revising an existing policy.
- We initially select a policy at random (mapping from states to actions).
- We re-compute the expected long term reward at each state using the selected policy.
- We select a new policy using the expected rewards and iterate until convergence.
Policy iteration: algorithm

- Let $\pi_t(s_i)$ be the selected policy at time $t$

1. Randomly chose $\pi_0$; set $t = 0$

2. For each state $s_i$ compute $J^*(s_i)$, the long term expected reward using policy $\pi_t$.

3. Set $\pi_t(s_i) = \max_k r_i + \gamma \left( \sum_j p_{i,j} J^*(s_j) \right)$

Policy iteration: algorithm

1. Randomly choose $\pi_0$; set $t = 0$
2. For each state $s_i$ compute $J^*(s_i)$, the long term expected reward using policy $\pi_t$
3. Set $\pi_t(s_i) = \max_k r_i + \gamma \left( \sum_j p_{i,j} J^*(s_j) \right)$

Once the policy is fixed we are back to rewards only models, so this can be computed using value iteration.

Can be computed using $J^*(s_i)$ for all states.
Value iteration vs. policy iteration

- Depending on the model and the information at hand:
  - If you have a good guess regarding the optimal policy then policy iteration would converge much faster
  - Similarly, if there are many possible actions, policy iteration might be faster
  - Otherwise, value iteration is a safer way
Demo
Reinforcement learning (RL)
MDP with actions: How do we learn the model?
From MDPs to Reinforcement Learning (RL)

1. We do not assume we know the Markov model
2. We learn the model from observations (online)

• Examples:
  - Game playing
  - Robot interacting with environment
  - Agents
What you should know

- Models that include rewards and actions
- Value iteration for solving MDPs
- Policy iteration