10-701
Machine Learning
Hidden Markov models (HMMs)
What’s wrong with Bayesian networks

- Bayesian networks are very useful for modeling joint distributions.
- But they have their limitations:
  - Cannot account for temporal / sequence models
  - DAG’s (no self or any other loops)

This is not a valid Bayesian network!
Hidden Markov models

- Model a set of observation with a set of hidden states
  - Robot movement
    - Observations: range sensor, visual sensor
    - Hidden states: location (on a map)
  - Speech processing
    - Observations: sound signals
    - Hidden states: parts of speech, words
  - Biology
    - Observations: DNA base pairs
    - Hidden states: Genes
Hidden Markov models

- Model a set of observation with a set of hidden states
  - Robot movement
    - **Observations:** range sensor, visual sensor
    - **Hidden states:** location (on a map)
  1. Hidden states generate observations
  2. Hidden states transition to other hidden states
Examples: Speech processing
Example: Biological data

ATGAAGCTACTGTCTTCTATCGAACAAGCATGCG
ATATTTGCCGACTTTAAAAAAGCTCAAG
TGCTCCAAAGAAAAACCGAAGTGCGCCAAGTG
CTGAAGAACAACTGGAGGAGTGTCGCTAC
CTCCTCCAAAACCAAAAAGGTCTCCGCTGACTAGG
GCACATCTGACAGAAGTGGAATCAAGG
CTAGAAAGACTGGAACAGCTATTTTCTACTGATT
TCCTCGAGAAGACCTTGACATGATT
## Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>page ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Sequence similarity, homology, and alignment</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Overview of the book</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Probabilities and probabilistic models</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Further reading</td>
<td>10</td>
</tr>
<tr>
<td>2 Pairwise alignment</td>
<td>12</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>12</td>
</tr>
<tr>
<td>2.2 The scoring model</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Alignment algorithms</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Dynamic programming with more complex models</td>
<td>28</td>
</tr>
<tr>
<td>2.5 Heuristic alignment algorithms</td>
<td>32</td>
</tr>
<tr>
<td>2.6 Linear space alignments</td>
<td>34</td>
</tr>
<tr>
<td>2.7 Significance of scores</td>
<td>36</td>
</tr>
<tr>
<td>2.8 Deriving score parameters from alignment data</td>
<td>41</td>
</tr>
<tr>
<td>2.9 Further reading</td>
<td>45</td>
</tr>
<tr>
<td>3 Markov chains and hidden Markov models</td>
<td>46</td>
</tr>
<tr>
<td>3.1 Markov chains</td>
<td>48</td>
</tr>
<tr>
<td>3.2 Hidden Markov models</td>
<td>51</td>
</tr>
<tr>
<td>3.3 Parameter estimation for HMMs</td>
<td>62</td>
</tr>
<tr>
<td>3.4 HMM model structure</td>
<td>68</td>
</tr>
<tr>
<td>3.5 More complex Markov chains</td>
<td>72</td>
</tr>
<tr>
<td>3.6 Numerical stability of HMM algorithms</td>
<td>77</td>
</tr>
<tr>
<td>3.7 Further reading</td>
<td>79</td>
</tr>
<tr>
<td>4 Pairwise alignment using HMMs</td>
<td>80</td>
</tr>
<tr>
<td>4.1 Pair HMMs</td>
<td>81</td>
</tr>
<tr>
<td>4.2 The full probability of x and y, summing over all paths</td>
<td>87</td>
</tr>
<tr>
<td>4.3 Suboptimal alignment</td>
<td>89</td>
</tr>
<tr>
<td>4.4 The posterior probability that x_i is aligned to y_j</td>
<td>91</td>
</tr>
<tr>
<td>4.5 Pair HMMs versus FSAs for searching</td>
<td>95</td>
</tr>
<tr>
<td>4.6</td>
<td>Further reading</td>
</tr>
<tr>
<td>5</td>
<td>Profile HMMs for sequence families</td>
</tr>
<tr>
<td>5.1</td>
<td>Ungapped score matrices</td>
</tr>
<tr>
<td>5.2</td>
<td>Adding insert and delete states to obtain profile HMMs</td>
</tr>
<tr>
<td>5.3</td>
<td>Deriving profile HMMs from multiple alignments</td>
</tr>
<tr>
<td>5.4</td>
<td>Searching with profile HMMs</td>
</tr>
<tr>
<td>5.5</td>
<td>Profile HMM variants for non-global alignments</td>
</tr>
<tr>
<td>5.6</td>
<td>More on estimation of probabilities</td>
</tr>
<tr>
<td>5.7</td>
<td>Optimal model construction</td>
</tr>
<tr>
<td>5.8</td>
<td>Weighting training sequences</td>
</tr>
<tr>
<td>5.9</td>
<td>Further reading</td>
</tr>
<tr>
<td>6</td>
<td>Multiple sequence alignment methods</td>
</tr>
<tr>
<td>6.1</td>
<td>What a multiple alignment means</td>
</tr>
<tr>
<td>6.2</td>
<td>Scoring a multiple alignment</td>
</tr>
<tr>
<td>6.3</td>
<td>Multidimensional dynamic programming</td>
</tr>
<tr>
<td>6.4</td>
<td>Progressive alignment methods</td>
</tr>
<tr>
<td>6.5</td>
<td>Multiple alignment by profile HMM training</td>
</tr>
<tr>
<td>6.6</td>
<td>Further reading</td>
</tr>
<tr>
<td>7</td>
<td>Building phylogenetic trees</td>
</tr>
<tr>
<td>7.1</td>
<td>The tree of life</td>
</tr>
<tr>
<td>7.2</td>
<td>Background on trees</td>
</tr>
<tr>
<td>7.3</td>
<td>Making a tree from pairwise distances</td>
</tr>
<tr>
<td>7.4</td>
<td>Parsimony</td>
</tr>
<tr>
<td>7.5</td>
<td>Assessing the trees: the bootstrap</td>
</tr>
<tr>
<td>7.6</td>
<td>Simultaneous alignment and phylogeny</td>
</tr>
<tr>
<td>7.7</td>
<td>Further reading</td>
</tr>
<tr>
<td>7.8</td>
<td>Appendix: proof of neighbour-joining theorem</td>
</tr>
<tr>
<td>8</td>
<td>Probabilistic approaches to phylogeny</td>
</tr>
<tr>
<td>8.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>8.2</td>
<td>Probabilistic models of evolution</td>
</tr>
<tr>
<td>8.3</td>
<td>Calculating the likelihood for ungapped alignments</td>
</tr>
<tr>
<td>8.4</td>
<td>Using the likelihood for inference</td>
</tr>
<tr>
<td>8.5</td>
<td>Towards more realistic evolutionary models</td>
</tr>
<tr>
<td>8.6</td>
<td>Comparison of probabilistic and non-probabilistic methods</td>
</tr>
<tr>
<td>8.7</td>
<td>Further reading</td>
</tr>
</tbody>
</table>
Example: Gambling on dice outcome

- Two dices, both skewed (output model).
- Can either stay with the same dice or switch to the second dice (transition mode).

![Diagram of dice gambling model](image)
A Hidden Markov model

- A set of states \( \{s_1 \ldots s_n\} \)
  - In each time point we are in exactly one of these states denoted by \( q_t \)
  - \( \Pi_i \), the probability that we start at state \( s_i \)
  - A transition probability model, \( P(q_t = s_i \mid q_{t-1} = s_j) \)
  - A set of possible outputs \( \Sigma \)
    - At time \( t \) we emit a symbol \( \sigma \in \Sigma \)
  - An emission probability model, \( p(o_t = \sigma \mid s_i) \)
The Markov property

- A set of states \{s_1 \ldots s_n\}
  - In each time point we are in exactly one of these states denoted by \(q_t\)
- \(\Pi_i\), the probability that we start at state \(s_i\)
- A transition probability model, \(P(q_t = s_i \mid q_{t-1} = s_j)\)
- A set of possible outputs \(\sum\)
- An emission probability model, \(p(o_t = o \mid s_i)\)

An important aspect of this definitions is the Markov property: \(q_{t+1}\) is conditionally independent of \(q_{t-1}\) (and any earlier time points) given \(q_t\)

More formally \(P(q_{t+1} = s_i \mid q_t = s_j) = P(q_{t+1} = s_i \mid q_t = s_j, q_{t-1} = s_j)\)
What can we ask when using a HMM?

A few examples:

• “What dice is currently being used?”
• “What is the probability of a 6 in the next role?”
• “What is the probability of 6 in any of the next 3 roles?”
Inference in HMMs

- Computing $P(Q)$ and $P(q_t = s_i)$
  - If we cannot look at observations
- Computing $P(Q \mid O)$ and $P(q_t = s_i \mid O)$
  - When we have observation and care about the last state only
- Computing $\arg\max_Q P(Q \mid O)$
  - When we care about the entire path
What dice is currently being used?

- We played $t$ rounds so far
- We want to determine $P(q_t = A)$
- Let's assume for now that we cannot observe any outputs (we are blind folded)
- How can we compute this?
Simple answer:

\[ P(q_t = A) = \]

Let's determine \( P(Q) \) where \( Q \) is any path that ends in \( A \):

\[ Q = q_1, \ldots, q_{t-1}, A \]

\[ P(Q) = P(q_1, \ldots, q_{t-1}, A) = P(A \mid q_1, \ldots, q_{t-1}) \cdot P(q_1, \ldots, q_{t-1}) = \]

\[ P(A \mid q_{t-1}) \cdot P(q_1, \ldots, q_{t-1}) = \ldots = P(A \mid q_{t-1}) \cdot \ldots \cdot P(q_2 \mid q_1) \cdot P(q_1) \]

**Markov property!**

**Initial probability**
\[ P(q_t = A) \]?

- Simple answer:
  1. Let’s determine \( P(Q) \) where \( Q \) is any path that ends in \( A \)

\[
Q = q_1, \ldots, q_{t-1}, A
\]

\[
P(Q) = P(q_1, \ldots, q_{t-1}, A) = P(A | q_1, \ldots, q_{t-1}) \cdot P(q_1, \ldots, q_{t-1}) = P(A | q_{t-1}) \cdot P(q_1, \ldots, q_{t-1}) = \ldots = P(A | q_{t-1}) \ldots P(q_2 | q_1) \cdot P(q_1)
\]

2. \( P(q_t = A) = \sum P(Q) \)
   
   where the sum is over all sets of \( t \) states that end in \( A \)
\[ \text{P}(q_t = A)? \]

- **Simple answer:**
  1. Let's determine \( \text{P}(Q) \) where \( Q \) is any path that ends in \( A \)

\[
Q = q_1, \ldots q_{t-1}, A
\]

\[
\text{P}(Q) = \text{P}(q_1, \ldots q_{t-1}, A) = \text{P}(A \mid q_1, \ldots q_{t-1}) \text{P}(q_1, \ldots q_{t-1}) = \text{P}(A \mid q_{t-1}) \text{P}(q_1, \ldots q_{t-1}) = \ldots = \text{P}(A \mid q_{t-1}) \ldots \text{P}(q_2 \mid q_1) \text{P}(q_1)
\]

2. \( \text{P}(q_t = A) = \sum \text{P}(Q) \)

   where the sum is over all sets of states that end in \( A \)

**Q:** How many sets \( Q \) are there?

**A:** A lot! \((2^{t-1})\)

Not a feasible solution
\( P(q_t = A) \), the smart way

- Lets define \( p_t(i) \) as the probability of being in state \( i \) at time \( t \):
  \[ p_t(i) = p(q_t = s_i) \]
- We can determine \( p_t(i) \) by induction:
  1. \( p_1(i) = \prod_i \)
  2. \( p_t(i) = ? \)
\( P(q_t = A), \text{ the smart way} \)

- Lets define \( p_t(i) = \text{probability state i at time } t = p(q_t = s_i) \)
- We can determine \( p_t(i) \) by induction
  1. \( p_1(i) = \Pi_i \)
  2. \( p_t(i) = \sum_j p(q_t = s_i \mid q_{t-1} = s_j)p_{t-1}(j) \)
$P(q_t = A)$, the smart way

- Lets define $p_t(i) = \text{probability state } i \text{ at time } t = p(q_t = s_i)$
- We can determine $p_t(i)$ by induction
  1. $p_1(i) = \Pi_i$
  2. $p_t(i) = \sum_j p(q_t = s_i | q_{t-1} = s_j)p_{t-1}(j)$

This type of computation is called dynamic programming

Complexity: $O(n^2 \times t)$

<table>
<thead>
<tr>
<th>Time / state</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Inference in HMMs

- Computing $P(Q)$ and $P(q_t = s_i)$
- Computing $P(Q \mid O)$ and $P(q_t = s_i \mid O)$
- Computing $\text{argmax}_Q P(Q)$
But what if we observe outputs?

• So far, we assumed that we could not observe the outputs
• In reality, we almost always can.

| v | $P(v | A)$ | $P(v | B)$ |
|---|---|---|
| 1 | .3 | .1 |
| 2 | .2 | .1 |
| 3 | .2 | .1 |
| 4 | .1 | .2 |
| 5 | .1 | .2 |
| 6 | .1 | .3 |
But what if we observe outputs?

• So far, we assumed that we could not observe the outputs.
• In reality, we almost always can.

\[
\begin{array}{c|cc}
\text{v} & P(v | A) & P(v | B) \\
1 & .3 & .1 \\
2 & .2 & .1 \\
3 & .2 & .1 \\
4 & .1 & .2 \\
5 & .1 & .2 \\
6 & .1 & .3 \\
\end{array}
\]

Does observing the sequence 5, 6, 4, 5, 6, 6 change our belief about the state?
P(q_t = A) when outputs are observed

- We want to compute P(q_t = A | O_1 … O_t)
- For ease of writing we will use the following notations (commonly used in the literature)
  - \( a_{j,i} = P(q_t = s_i | q_{t-1} = s_j) \)
  - \( b_i(o_t) = P(o_t | s_i) \)

Transition probability

Emission probability
P(q_t = A) when outputs are observed

- We want to compute P(q_t = A | O_1 \ldots O_t)
- Let’s start with a simpler question. Given a sequence of states Q, what is P(Q | O_1 \ldots O_t) = P(Q | O)?
  - It is pretty simple to move from P(Q) to P(q_t = A)
  - In some cases P(Q) is the more important question
    - Speech processing
    - NLP
We can use Bayes rule:

\[
P(Q | O) = \frac{P(O | Q)P(Q)}{P(O)}
\]

Easy, \( P(O | Q) = P(o_1 | q_1) \cdot P(o_2 | q_2) \cdots P(o_t | q_t) \)
We can use Bayes rule:

$$P(Q | O) = \frac{P(O | Q)P(Q)}{P(O)}$$

Easy, \(P(Q) = P(q_1) P(q_2 | q_1) \ldots P(q_t | q_{t-1})\)
\[ P(\mathbf{Q} | \mathbf{O}) \]

- We can use Bayes rule:

\[
P(\mathbf{Q}|\mathbf{O}) = \frac{P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q})}{P(\mathbf{O})}
\]

Hard!
What is the probability of seeing a set of observations:
- An important question in its own rights, for example classification using two HMMs

Define $\alpha_t(i) = P(o_1, o_2 \ldots, o_t \land q_t = s_i)$

- $\alpha_t(i)$ is the probability that we:
  1. Observe $o_1, o_2 \ldots, o_t$
  2. End up at state $i$

How do we compute $\alpha_t(i)$?
Computing $\alpha_t(i)$

- $\alpha_1(i) = P(o_1 \land q_t = i) = P(o_1 | q_t = s_i) \Pi_t$

  We must be at a state in time $t$

  chain rule

  Markov property
Example: Computing $\alpha_3(B)$

- We observed 2,3,6

$\alpha_1(A) = P(2 \land q_1 = A) = P(2 \mid q_1 = A) \Pi_A = .2 \cdot .7 = .14, \quad \alpha_1(B) = .1 \cdot .3 = .03$

$\alpha_2(A) = \sum_{j=A,B} b_A(3) a_{j,A} \quad \alpha_1(j) = .2 \cdot .8 \cdot .14 + .2 \cdot .2 \cdot .03 = 0.0236, \quad \alpha_2(B) = 0.0052$

$\alpha_3(B) = \sum_{j=A,B} b_B(6) a_{j,B} \quad \alpha_2(j) = .3 \cdot .2 \cdot .0236 + .3 \cdot .8 \cdot .0052 = 0.00264$
Where we are

- We want to compute $P(Q \mid O)$
- For this, we only need to compute $P(O)$
- We know how to compute $\alpha_t(i)$

From now its easy

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t \land q_t = s_i)$$

so

$$P(O) = P(o_1, o_2, \ldots, o_t) = \sum_i P(o_1, o_2, \ldots, o_t \land q_t = s_i) = \sum_i \alpha_t(i)$$

note that

$$p(q_t=s_i \mid o_1, o_2, \ldots, o_t) \equiv \frac{\alpha_t(i)}{\sum_j \alpha_t(j)}$$

$P(A \mid B) = P(A \land B) / P(B)$
Complexity

- How long does it take to compute $P(Q \mid O)$?
  - $P(Q)$: $O(n)$
  - $P(O\mid Q)$: $O(n)$
  - $P(O)$: $O(n^2t)$
Inference in HMMs

- Computing $P(Q)$ and $P(q_t = s_i)$  

- Computing $P(Q \mid O)$ and $P(q_t = s_i \mid O)$  

- Computing $\arg\max_Q P(Q)$
Most probable path

• We are almost done …
• One final question remains
  How do we find the most probable path, that is Q* such that

  \[ P(Q^* \mid O) = \arg\max_Q P(Q \mid O) \]?

• This is an important path
  - The words in speech processing
  - The set of genes in the genome
  - etc.
Example

- What is the most probable set of states leading to the sequence: 

\[1,2,2,5,6,5,1,2,3\] ?

\[
\begin{array}{|c|c|c|}
\hline
v & P(v |A) & P(v |B) \\
\hline
1 & .3 & .1 \\
2 & .2 & .1 \\
3 & .2 & .1 \\
4 & .1 & .2 \\
5 & .1 & .2 \\
6 & .1 & .3 \\
\hline
\end{array}
\]

\[\Pi_A=0.7\]
\[\Pi_b=0.3\]
Most probable path

\[ \arg \max_Q P(Q \mid O) = \arg \max_Q \frac{P(O \mid Q)P(Q)}{P(O)} = \arg \max_Q P(O \mid Q)P(Q) \]

We will use the following definition:

\[ \delta_t(i) = \max_{q_1 \ldots q_{t-1}} p(q_1 \ldots q_{t-1} \land q_t = s_i \land O_1 \ldots O_t) \]

In other words we are interested in the most likely path from 1 to \( t \) that:

1. Ends in \( S_i \)

2. Produces outputs \( O_1 \ldots O_t \)
Computing $\delta_t(i)$

$$
\delta_1(i) = p(q_1 = s_i \land O_1) \\
= p(q_1 = s_i)p(O_1 \mid q_1 = s_i) \\
= \pi_i b_i(O_1)
$$

$$
\delta_t(i) = \max_{q_1 \ldots q_{t-1}} p(q_1 \ldots q_{t-1} \land q_t = s_i \land O_1 \ldots O_t)
$$

Q: Given $\delta_t(i)$, how can we compute $\delta_{t+1}(i)$?

A: To get from $\delta_t(i)$ to $\delta_{t+1}(i)$ we need to

1. Add an emission for time $t+1$ ($O_{t+1}$)
2. Transition to state $s_i$

$$
\delta_{t+1}(i) = \max_{q_1 \ldots q_t} p(q_1 \ldots q_t \land q_{t+1} = s_i \land O_1 \ldots O_{t+1})
= \max_j \delta_t(j)p(q_{t+1} = s_i \mid q_t = s_j)p(O_{t+1} \mid q_{t+1} = s_i)
= \max_j \delta_t(j)a_{j,i}b_i(O_{t+1})
$$
The Viterbi algorithm

\[ \delta_{t+1}(i) = \max_{q_1 \ldots q_t} p(q_1 \ldots q_t \land q_{t+1} = s_i \land O_1 \ldots O_{t+1}) \]

\[ = \max_j \delta_t(j)p(q_{t+1} = s_i \mid q_t = s_j)p(O_{t+1} \mid q_{t+1} = s_i) \]

\[ = \max_j \delta_t(j)a_{j,i}b_i(O_{t+1}) \]

- Once again we use dynamic programming for solving \( \delta_t(i) \)
- Once we have \( \delta_t(i) \), we can solve for our \( P(Q^* \mid O) \)

By:

\[ P(Q^* \mid O) = \arg\max_Q P(Q \mid O) = P(Q^* \mid O) = \]

path defined by \( \arg\max_j \delta_t(j) \),
Inference in HMMs

• Computing $P(Q)$ and $P(q_t = s_i)$

• Computing $P(Q | O)$ and $P(q_t = s_i | O)$

• Computing $\arg\max_Q P(Q)$
What you should know

• Why HMMs? Which applications are suitable?
• Inference in HMMs
  - No observations
  - Probability of next state w. observations
  - Maximum scoring path (Viterbi)