Introduction to Hidden Markov models

Mark Johnson

Macquarie University

September 17, 2014
Outline

Sequence labelling

Hidden Markov Models

Finding the most probable label sequence

Higher-order HMMs

Summary
What is sequence labelling?

- A sequence labelling problem is one where:
  - the input consists of a sequence $X = (X_1, \ldots, X_n)$, and
  - the output consists of a sequence $Y = (Y_1, \ldots, Y_n)$ of labels, where:
    - $Y_i$ is the label for element $X_i$

- Example: Part-of-speech tagging

$$
\begin{pmatrix}
Y \\
X
\end{pmatrix} =
\begin{pmatrix}
\text{Verb,} & \text{Determiner,} & \text{Noun} \\
\text{spread,} & \text{the,} & \text{butter}
\end{pmatrix}
$$

- Example: Spelling correction

$$
\begin{pmatrix}
Y \\
X
\end{pmatrix} =
\begin{pmatrix}
\text{write,} & \text{a,} & \text{book} \\
\text{rite,} & \text{a,} & \text{buk}
\end{pmatrix}
$$
Named entity extraction with IOB labels

• **Named entity recognition and classification** (NER) involves finding the named entities in a text and identifying what type of entity they are (e.g., person, location, corporation, dates, etc.)

• NER can be formulated as a sequence labelling problem

• **Inside-Outside-Begin** (IOB) labelling scheme indicates the **beginning and span** of each named entity

  B-ORG  I-ORG  O  O  O  O  B-LOC  I-LOC  I-LOC  O
  Macquarie University is located in New South Wales.

• The IOB labelling scheme lets us identify **adjacent named entities**

  B-LOC  I-LOC  I-LOC  B-LOC  I-LOC  O  B-LOC  O  ...
  New South Wales Northern Territory and Queensland are ...

• This technology can extract information from:
  ▶ news stories
  ▶ financial reports
  ▶ classified ads
Other applications of sequence labelling

- Speech transcription as a sequence labelling task
  - The input $X = (X_1, \ldots, X_n)$ is a sequence of *acoustic frames* $X_i$, where $X_i$ is a set of features extracted from a 50msec window of the speech signal
  - The output $Y$ is a sequence of words (the transcript of the speech signal)
- Financial applications of sequence labelling
  - identifying trends in price movements
- Biological applications of sequence labelling
  - gene-finding in DNA or RNA sequences
A first (bad) approach to sequence labelling

- Idea: train a supervised classifier to *predict entire label sequence at once*

  B-ORG    I-ORG    O    O    O    O    B-LOC    I-LOC    I-LOC    O
  Macquarie University is located in New South Wales.

- Problem: *the number of possible label sequences grows exponentially with the length of the sequence*
  - with *binary labels*, there are $2^n$ different label sequences of a sequence of length $n$ ($2^{32} = 4$ billion)

  ⇒ most labels won’t be observed even in very large training data sets

- This approach fails because it has massive *sparse data problems*
A better approach to sequence labelling

- Idea: train a supervised classifier to *predict the label of one word at a time*

  
  Western Australia is the largest state in Australia.

- Avoids sparse data problems in label space

- As well as current word, classifiers can use *previous and following words as features*

- But this approach can produce *inconsistent label sequences*

  
  The New York Times is a newspaper.

⇒ Track *dependencies between adjacent labels*

  ▶ “chicken-and-egg” problem that Hidden Markov Models solve!
Outline

Sequence labelling

Hidden Markov Models

Finding the most probable label sequence

Higher-order HMMs

Summary
The big picture

- Optimal classifier: \( \hat{y}(x) = \arg\max_y P(Y=y \mid X=x) \)
- How can we avoid sparse data problems when estimating \( P(Y \mid X) \)
  \( \Rightarrow \) Decompose \( P(Y \mid X) \) into a product of distributions that don’t have sparse data problems
- How can we compute the \( \arg\max \) over all possible label sequences \( y \)?
  \( \Rightarrow \) Use the Viterbi algorithm (dynamic programming over a trellis) to search over an exponential number of sequences \( y \) in linear time
Introduction to Hidden Markov models

- Hidden Markov models (HMMs) are a simple sequence labelling model
- HMMs are *noisy channel models* generating

\[ P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{X} \mid \mathbf{Y})P(\mathbf{Y}) \]

- the *source model* \( P(\mathbf{Y}) \) is a Markov model (e.g., a bigram language model)

\[ P(\mathbf{Y}) = \prod_{i=1}^{n+1} P(Y_i \mid Y_{i-1}) \]

- the *channel model* \( P(\mathbf{X} \mid \mathbf{Y}) \) generates each \( X_i \) independently, i.e.,

\[ P(\mathbf{X} \mid \mathbf{Y}) = \prod_{i=1}^{n} P(X_i \mid Y_i) \]

- At testing time we only know \( \mathbf{X} \), so \( \mathbf{Y} \) is unobserved or *hidden*
Terminology in Hidden Markov Models

- Hidden Markov models (HMMs) generate pairs of sequences \((x, y)\)
- The sequence \(x\) is called:
  - the *input sequence*, or
  - the *observations*, or
  - the *visible data*
  
  because \(x\) is given when an HMM is used for sequence labelling

- The sequence \(y\) is called:
  - the *label sequence*, or
  - the *tag sequence*, or
  - the *hidden data*
  
  because \(y\) is unknown when an HMM is used for sequence labelling

- A \(y \in \mathcal{Y}\) is sometimes called a *hidden state* because an HMM can be viewed as a *stochastic automaton*
  - each different \(y \in \mathcal{Y}\) is a state in the automaton
  - the \(x\) are *emissions* from the automaton
Review: Naive Bayes models

- The naive Bayes model:

\[ P(Y, X_1, \ldots, X_m) = P(Y) P(X_1 | Y) \ldots P(X_m | Y) \]

- In a Naive Bayes classifier, the \( X_i \) are features and \( Y \) is the class label we want to predict
  - \( P(X_i | Y) \) can be Bernoulli, multinomial, Gaussian etc.
An bigram language model is a first-order Markov model that factorises the distribution over a sequence $y$ into a product of conditional distributions:

$$ P(y) = \prod_{i=1}^{m} P(y_i | y_{i-n}, \ldots, y_{i-1}) $$

- *Pad* $y$ with end markers, i.e., $y = (\$, y_1, y_2, \ldots, y_m, \$)$
- In a bigram language model, $y$ is a sentence and the $y_i$ are words.
- First-order Markov model as a Bayes net:
Hidden Markov models

- A Hidden Markov Model \((s, t)\) defines a probability distribution over an item sequence \(X = (X_1, \ldots, X_n)\), where each \(X_i \in \mathcal{X}\), and a label sequence \(Y = (Y_1, \ldots, Y_n)\), where each \(Y_i \in \mathcal{Y}\), as:

\[
P(X, Y) = P(X | Y) P(Y)
\]

\[
P(Y=(\$, y_1, \ldots, y_n, \$)) = \prod_{i=1}^{n+1} P(Y_i=y_i | Y_{i-1}=y_{i-1})
\]

\[
= \prod_{i=1}^{n+1} s_{y_i, y_{i-1}} \quad (i.e., P(Y_i=y | Y_{i-1}=y') = s_{y, y'})
\]

\[
P(X=(x_1, \ldots, x_n) | Y=(\$, y_1, \ldots, y_n, \$))
\]

\[
= \prod_{i=1}^{n} P(X_i=x_i | Y_i=y_i)
\]

\[
= \prod_{i=1}^{n} t_{x_i, y_i} \quad (i.e., P(X_i=x | Y_i=y) = t_{x, y})
\]
Hidden Markov models as Bayes nets

- A **Hidden Markov Model** (HMM) defines a joint distribution $P(X, Y)$ over:
  - *item sequences* $X = (X_1, \ldots, X_n)$ and
  - *label sequences* $Y = (Y_0 = $, $Y_1, \ldots, Y_n$, $Y_{n+1} = $):

$$P(X, Y) = \left( \prod_{i=1}^{n} P(Y_i \mid Y_{i-1}) P(X_i \mid Y_i) \right) P(Y_{n+1} \mid Y_n)$$

- HMMs can be expressed as Bayes nets, and standard message-passing inference algorithms work well with HMMs.

```
$ \rightarrow Y_1 \rightarrow Y_2 \rightarrow Y_3 \rightarrow Y_4 \rightarrow $ \\
| \downarrow \ | \downarrow \ | \downarrow \ | \downarrow \ |
\ X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow 
```

```
The parameters of an HMM

- An HMM is specified by two matrices:
  - a *matrix* $s$, where $s_{y,y'} = P(Y_i = y | Y_{i-1} = y')$ is the *probability of a label* $y$ *following the label* $y'$
    - this is the same as in a bigram language model
    - remember to pad $y$ with begin/end of sentence markers $\$
  - a *matrix* $t$, where $t_{x,y} = P(X_i = x | Y_i = y)$ is the *probability of an item* $x$ *given the label* $y$
    - similar to the feature-class probability $P(F_j | Y)$ in naive Bayes

- If the set of labels is $Y$ and the vocabulary is $X$, then
  - $s$ is an $m \times m$ matrix, where $m = |Y|$.
  - $t$ is a $v \times m$ matrix, where $v = |X|$.
HMM estimate of a labelled sequence’s probability

\[
P \left( X = (\text{spread, the, butter}), Y = (\text{Verb, Det, Noun}) \right) = s_s, \text{Verb} \cdot s_{\text{Det, Verb}} \cdot s_{\text{Noun, Det}} \cdot s_{\$, Noun} \cdot t_{\text{spread, Verb}} \cdot t_{\text{the, Det}} \cdot t_{\text{butter, Noun}}
\]

\[
= 0.4 \cdot 0.4 \cdot 0.7 \cdot 0.3 \cdot 0.5 \cdot 0.8 \cdot 0.5
\]

\[
\approx 0.009
\]
A HMM as a stochastic automaton

State to state transition probabilities

<table>
<thead>
<tr>
<th>$y_i \setminus y_{i-1}$</th>
<th>$$$</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$$</td>
<td>0.0</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>Verb</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>Det</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>Noun</td>
<td>0.2</td>
<td>0.2</td>
<td>0.7</td>
<td>0.2</td>
</tr>
</tbody>
</table>

State to word emission probabilities

<table>
<thead>
<tr>
<th>$x_i \setminus y_i$</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>spread</td>
<td>0.5</td>
<td>0.1</td>
<td>0.4</td>
</tr>
<tr>
<td>the</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>butter</td>
<td>0.4</td>
<td>0.1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

\[ P (X=(\text{spread, the, butter}), Y=(\text{Verb, Det, Noun})) = 0.4 \cdot 0.4 \cdot 0.7 \cdot 0.3 \cdot 0.5 \cdot 0.8 \cdot 0.5 \approx 0.009 \]
Estimating an HMM from labelled data

- Training data consists of \textit{labelled sequences of data items}

- Estimate $s_{y,y'} = P(Y_i = y \mid Y_{i-1} = y')$ from label sequences $y$ (just as for language models)
  
  \begin{itemize}
  \item If $n_{y,y'}$ is the number of times $y$ follows $y'$ in training label sequences and $n_{y'}$ is the number of times $y'$ is followed by anything, then:
  \end{itemize}

  \[ \hat{s}_{y,y'} = \frac{n_{y,y'} + 1}{n_{y'} + |Y|} \]

  \begin{itemize}
  \item Be sure to \textit{count the end-markers} $\$\$
  \end{itemize}

- Estimate $t_{x,y} = P(X_i = x \mid Y_i = y)$ from pairs of item sequences $x$ and their corresponding label sequence $y$

  \begin{itemize}
  \item If $r_{x,y}$ is the number of times a data item $x$ is labelled $y$, and $r_y$ is the number of times $y$ appears in a label sequence, then:
  \end{itemize}

  \[ \hat{t}_{x,y} = \frac{r_{x,y} + 1}{r_y + |X|} \]
Estimating an HMM example

\[ D = \begin{pmatrix} \text{Verb} & \text{Noun} \\ \text{spread} & \text{butter} \end{pmatrix}, \begin{pmatrix} \text{Verb} & \text{Det} & \text{Noun} \\ \text{butter} & \text{the} & \text{spread} \end{pmatrix} \]

\[ n = \begin{array}{c|cccc} y_i \backslash y_{i-1} & $ & \text{Verb} & \text{Det} & \text{Noun} \\ \hline $ & 0 & 0 & 0 & 2 \\ \text{Verb} & 2 & 0 & 0 & 0 \\ \text{Det} & 0 & 1 & 0 & 0 \\ \text{Noun} & 0 & 1 & 1 & 0 \end{array} \]

\[ \hat{s} = \begin{array}{c|cccc} y_i \backslash y_{i-1} & $ & \text{Verb} & \text{Det} & \text{Noun} \\ \hline $ & 1/6 & 1/6 & 1/5 & 3/6 \\ \text{Verb} & 3/6 & 1/6 & 1/5 & 1/6 \\ \text{Det} & 1/6 & 2/6 & 1/5 & 1/6 \\ \text{Noun} & 1/6 & 2/6 & 2/5 & 1/6 \end{array} \]

\[ r = \begin{array}{c|ccc} x_i \backslash y_i & \text{Verb} & \text{Det} & \text{Noun} \\ \hline \text{spread} & 1 & 0 & 1 \\ \text{the} & 0 & 1 & 0 \\ \text{butter} & 1 & 0 & 1 \end{array} \]

\[ \hat{t} = \begin{array}{c|ccc} x_i \backslash y_i & \text{Verb} & \text{Det} & \text{Noun} \\ \hline \text{spread} & 2/5 & 1/4 & 2/5 \\ \text{the} & 1/5 & 2/4 & 1/5 \\ \text{butter} & 2/5 & 1/4 & 2/5 \end{array} \]
Outline

Sequence labelling

Hidden Markov Models

Finding the most probable label sequence

Higher-order HMMs

Summary
Why is finding the most probable label sequence hard

• When we use an HMM, we’re given data items $x$ and want to return the most probable label sequence:

$$\hat{y}(x) = \arg\max_{y \in \mathcal{Y}^n} P(X=x \mid Y=y) P(Y=y)$$

• If $x$ has $n$ elements and each item has $m = |\mathcal{Y}|$ possible labels, the number of possible label sequences is $m^n$
  ▶ the number of possible label sequences grows exponentially with the length of the string
  ⇒ exhaustive search for the optimal label sequence become impossible once $n$ is large

• But the Viterbi algorithm finds the most probable label sequence $\hat{y}(x)$ in $O(n)$ (linear) time using dynamic programming over a trellis
  ▶ the Viterbi algorithm is actually just the shortest path algorithm on the trellis
The trellis

- Given input $x$ of length $n$, the **trellis** is a **directed acyclic graph** where:
  - the nodes are all pairs $(i, y)$ for $y \in \mathcal{Y}$ and $i \in 1, \ldots, n$, plus a starting node and a final node
  - there are edges from the starting node to all nodes $(1, y)$ for each $y \in \mathcal{Y}$
  - for each $y', y \in \mathcal{Y}$ and each $i \in 2, \ldots, n$ there is an edge from $(i - 1, y')$ to $(i, y)$
  - for each $y \in \mathcal{Y}$ there is an edge from $(n, y)$ to the final node
- Every possible $y$ is a **path through the trellis**

$\overset{(1,\text{Verb})}{\text{spread}} \overset{(3,\text{Verb})}{\text{the}} \overset{(2,\text{Verb})}{\text{butter}}$
Using a trellis to find the most probable label sequence $y$

- One-to-one correspondence between *paths from start to finish in the trellis* and *label sequences* $y$
- High level description of algorithm:
  - associate each edge in the trellis with a weight (a number)
  - the product of the weights along a path from start to finish will be
    \[ P(X=x, Y=y) \]
  - use *dynamic programming* to find highest scoring path from start to finish
  - return the corresponding label sequence $\hat{y}(x)$

Conditioned on the input $X$, the distribution $P(Y | X)$ is an inhomogenous (i.e., time-varying) Markov chain
- the transition probabilities $P(Y_i | Y_{i-1}, X_i)$ depend on $X_i$ and hence $i$
Rearranging the terms in the HMM formula

- Rearrange terms so all terms associated with a time \( i \) are together

\[
P(X, Y) = P(Y) P(X \mid Y)
\]

\[
= s_{y_1} \cdot s_{y_2, y_1} \cdot \ldots \cdot s_{y_n, y_{n-1}} \cdot s_{y_n} \cdot t_{x_1, y_1} \cdot t_{x_2, y_2} \cdot \ldots \cdot t_{x_n, y_n}
\]

\[
= s_{y_1} \cdot t_{x_1, y_1} \cdot s_{y_2, y_1} \cdot t_{x_2, y_2} \cdot \ldots \cdot s_{y_n, y_{n-1}} \cdot t_{x_n, y_n} \cdot s_{y_n}
\]

\[
= \prod_{i=1}^{n} s_{y_i, y_{i-1}} \cdot t_{x_i, y_i} \cdot s_{y_n}
\]

- Trellis edge weights:
  - weight on edge from start node to node (1, \( y \)) is \( w_{1, y, s} = s_{y, s} \cdot t_{x_1, y} \)
  - weight on edge from node (\( i-1, y' \)) to node (\( i, y \)) is \( w_{i, y, y'} = s_{y, y'} \cdot t_{x_i, y} \)
  - weight on edge from node (\( n, y \)) to final node is \( w_{n+1, s, y} = s_{y, y} \)

\[\Rightarrow\] product of weights on edges of path \( y \) in trellis is \( P(X=x, Y=y) \)
Trellis for *spread the butter*

$\begin{array}{c|cccc}
 y_i \backslash y_{i-1} & $ & Verb & Det & Noun \\
\hline
$ & 0 & 0.3 & 0.1 & 0.3 \\
Verb & 0.4 & 0.1 & 0.1 & 0.3 \\
Det & 0.4 & 0.4 & 0.1 & 0.2 \\
Noun & 0.2 & 0.2 & 0.7 & 0.2 \\
\end{array}$

$\begin{array}{c|cccc}
 x_i \backslash y_i & Verb & Det & Noun \\
\hline
spread & 0.5 & 0.1 & 0.4 \\
the & 0.1 & 0.8 & 0.1 \\
butter & 0.4 & 0.1 & 0.5 \\
\end{array}$

$\begin{array}{c|c}
 y_1 \backslash y_0 & $ \\
\hline
Verb & 0.2 \\
Det & 0.04 \\
Noun & 0.08 \\
\end{array}$

$\begin{array}{c|ccc}
 y_2 \backslash y_1 & Verb & Det & Noun \\
\hline
Verb & 0.01 & 0.01 & 0.03 \\
Det & 0.32 & 0.08 & 0.16 \\
Noun & 0.02 & 0.07 & 0.02 \\
\end{array}$
Finding the highest-scoring path

- The score of a path is the product of the weights of the edges along that path.
- Key insight: the highest-scoring path to a node \((i, y)\) must begin with a highest-scoring path to some node \((i - 1, y')\).
- Let \(\text{MaxScore}(i, y)\) be the score of the highest scoring path to node \((i, y)\). Then:
  \[
  \begin{align*}
  \text{MaxScore}(1, y) &= w_{1,y,$$} \\
  \text{MaxScore}(i, y) &= \max_{y' \in Y} \text{MaxScore}(i - 1, y') \cdot w_{i,y,y'} \quad \text{if } 1 < 2 \leq n \\
  \text{MaxScore}(n + 1, $) &= \max_{y' \in Y} \text{MaxScore}(n, y) \cdot w_{n+1,$,y}
  \end{align*}
  \]
- To find the highest scoring path, compute \(\text{MaxScore}(i, y)\) for \(i = 1, 2, \ldots, n + 1\) in turn.
Finding the highest-scoring path example (1)

\[
\begin{array}{c|c|c}
\text{w}_1 & y_1 \backslash y_0 & \$ \\
\hline
\text{Verb} & 0.2 & \\
\text{Det} & 0.04 & \\
\text{Noun} & 0.08 & \\
\end{array}
\]

MaxScore(1, Verb) = 0.2
MaxScore(1, Det) = 0.04
MaxScore(1, Noun) = 0.08
Finding the highest-scoring path example (2)

\[ \begin{align*}
\text{MaxScore}(1, \text{Verb}) &= 0.2 \\
\text{MaxScore}(1, \text{Det}) &= 0.04 \\
\text{MaxScore}(1, \text{Noun}) &= 0.08 \\
\end{align*} \]

\[
\begin{array}{c|ccc}
 y_2 \backslash y_1 & \text{Verb} & \text{Det} & \text{Noun} \\
\hline
\text{Verb} & 0.01 & 0.01 & 0.03 \\
\text{Det} & 0.32 & 0.08 & 0.16 \\
\text{Noun} & 0.02 & 0.07 & 0.02 \\
\end{array}
\]

\[ \begin{align*}
\text{MaxScore}(2, \text{Verb}) &= 0.0024 \quad \text{via Noun} \\
\text{MaxScore}(2, \text{Det}) &= 0.064 \quad \text{via Verb} \\
\text{MaxScore}(2, \text{Noun}) &= 0.004 \quad \text{via Noun} \\
\end{align*} \]
Finding the highest-scoring path example (3)

MaxScore(2, Verb) = 0.0024
MaxScore(2, Det) = 0.064
MaxScore(2, Noun) = 0.004

\[
\begin{array}{c|ccc}
\toprule
y_3 \\ y_2 & \text{Verb} & \text{Det} & \text{Noun} \\
\hline
\text{Verb} & 0.04 & 0.04 & 0.12 \\
\text{Det} & 0.04 & 0.01 & 0.02 \\
\text{Noun} & 0.1 & 0.35 & 0.1 \\
\bottomrule
\end{array}
\]

MaxScore(3, Verb) = 0.00256 via Det
MaxScore(3, Det) = 0.00064 via Det
MaxScore(3, Noun) = 0.0224 via Det
Finding the highest-scoring path example (4)

MaxScore(3, Verb) = 0.00256
MaxScore(3, Det) = 0.00064
MaxScore(3, Noun) = 0.0224

MaxScore(4, $) = 0.00672 via Noun
Forward-backward algorithms

- This dynamic programming algorithm is an instance of a general family of algorithms known as *forward-backward algorithms*.
- The *forward pass* computes a probability of a *prefix* of the input.
- The *backward pass* computes a probability of a *suffix* of the input.
- With these it is possible to compute:
  - the marginal probability of any state given the input $P(Y_i=y \mid X = x)$
  - the marginal probability of any adjacent pair of states $P(Y_{i-1}=y, Y_i=y' \mid X = x)$
  - the expected number of times any pair of states is seen in the input $E[n_{y,y'} \mid x]$.
- These are required for estimating HMMs from unlabelled data using the *Expectation-Maximisation Algorithm*.
Outline

Sequence labelling

Hidden Markov Models

Finding the most probable label sequence

Higher-order HMMs

Summary
The “order” of an HMM

- The order of an HMM is the number of previous labels used to predict the current label $Y_i$.
- A first-order HMM uses the previous label $Y_{i-1}$ to predict $Y_i$:
  \[
  P(Y) = \prod_{i=1}^{n+1} P(Y_i | Y_{i-1})
  \]
  (the HMMs we’ve seen so far are all first-order HMMs)
- A second-order HMM uses the previous two labels $Y_{i-2}$ and $Y_{i-1}$ to predict $Y_i$:
  \[
  P(Y) = \prod_{i=1}^{n+1} P(Y_i | Y_{i-1}, Y_{i-2})
  \]
  - the parameters used in a second-order HMM are more complex
  - $s$ is a three-dimensional array of size $m \times m \times m$, where $m = |\mathcal{Y}|$
  \[
  P(Y_i=y | Y_{i-1}=y', Y_{i-2}=y'') = s_{y,y',y''}
  \]
Bayes net representation of a 2nd-order HMM

\[
P(X, Y) = \left( \prod_{i=1}^{n} P(Y_i \mid Y_{i-1}) P(X_i \mid Y_i) \right) P(Y_{n+1} \mid Y_n)
\]
The trellis in higher-order HMMs

- The nodes in the trellis encode the information required from “the past” in order to predict the next label $Y_i$
  
  - a first-order HMM tracks one past label $\Rightarrow$ trellis states are pairs $(i, y')$, where $Y_{i-1} = y'$
  
  - a second-order HMM tracks two past labels $\Rightarrow$ trellis states are triples $(i, y', y'')$, where $Y_{i-1} = y'$ and $Y_{i-2} = y''$
  
  - in a $k$-th order HMM, trellis states consist of a position index $i$ and $k$ $Y$-values for $Y_{i-1}, Y_{i-2}, \ldots, Y_{i-k}$

- This means that for a sequence $X$ of length $n$ and where the number of states $|Y| = m$, a $k$-th order HMM has $O(n \cdot m^k)$ nodes in the trellis

- Since every edge is visited a constant number of times, the computational complexity of the Viterbi algorithm is also $O(n \cdot m^{k+1})$. 
Outline

Sequence labelling

Hidden Markov Models

Finding the most probable label sequence

Higher-order HMMs

Summary
Summary

- **Sequence labelling** is an important kind of *structured prediction problem*.
- **Hidden Markov Models** (HMMs) can be viewed as a generalisation of Naive Bayes models where the label $Y$ is generated by a Markov model.
  - HMMs can also be viewed as stochastic automata with a hidden state.
- **Forward-backward algorithms** are dynamic programming algorithms that can compute:
  - the Viterbi (i.e., most likely) label sequence for a string.
  - the expected number of times each state-output or state-state transition is used in the analysis of a string.
- HMMs make the same independence assumptions as Naive Bayes.
  - **Conditional Random Fields** (CRFs) are a generalisation of HMMs that relax these independence assumptions.
  - CRFs are the sequence labelling generalisation of logistic regression.