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

MACQUARIE UNIVERSITY

What is sequence labelling?

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

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

- Example: Spelling correction

$$\begin{pmatrix} \mathbf{Y} \\ \mathbf{X} \end{pmatrix} = \begin{pmatrix} \text{write,} & \text{a,} & \text{book} \\ \text{rite,} & \text{a,} & \text{buk} \end{pmatrix}$$

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	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

MACQUARIE UNIVERSITY

Other applications of sequence labelling

- Speech transcription as a sequence labelling task
  - The input $\mathbf{X} = (X_1, \ldots, X_n)$ 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 $\mathbf{Y}$ 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

MACQUARIE UNIVERSITY

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	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)

$\Rightarrow$ 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*

B-LOC	I-LOC	O	O	O	O	O	B-LOC	O
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*

O	B-LOC	I-ORG	I-ORG	O	O	O	O
The	New	York	Times	is	a	newspaper	.

⇒ Track *dependencies between adjacent labels*

- "chicken-and-egg" problem that Hidden Markov Models solve!

MACQUARIE UNIVERSITY

Outline

Sequence labelling

Hidden Markov Models

Finding the most probable label sequence

Higher-order HMMs

Summary

The big picture

- Optimal classifier: $\widehat{\mathbf{y}}(\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}} \mathrm{P}(\mathbf{Y}{=}\mathbf{y} \mid \mathbf{X}{=}\mathbf{x})$
- How can we avoid *sparse data* problems when estimating $\mathrm{P}(\mathbf{Y} \mid \mathbf{X})$

$\Rightarrow$ *Decompose* $\mathrm{P}(\mathbf{Y} \mid \mathbf{X})$ *into a product of distributions that don't have sparse data problems*

- How can we *compute the* argmax *over all possible label sequences* $\mathbf{y}$*?*

$\Rightarrow$ Use the *Viterbi algorithm* (*dynamic programming* over a *trellis*) to search over *an exponential number of sequences* $\mathbf{y}$ *in linear time*

Introduction to Hidden Markov models

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

$$\mathrm{P}(\mathbf{X}, \mathbf{Y}) \quad = \quad \mathrm{P}(\mathbf{X} \mid \mathbf{Y})\mathrm{P}(\mathbf{Y})$$

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

$$\mathrm{P}(\mathbf{Y}) \quad = \quad \prod_{i=1}^{n+1} \mathrm{P}(Y_i \mid Y_{i-1})$$

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

$$\mathrm{P}(\mathbf{X} \mid \mathbf{Y}) \quad = \quad \prod_{i=1}^{n} \mathrm{P}(X_i \mid Y_i)$$

- 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 $(\mathbf{x}, \mathbf{y})$
- The sequence $\mathbf{x}$ is called:
  - the *input sequence*, or
  - the *observations*, or
  - the *visible data*

  because $\mathbf{x}$ is given when an HMM is used for sequence labelling
- The sequence $\mathbf{y}$ is called:
  - the *label sequence*, or
  - the *tag sequence*, or
  - the *hidden data*

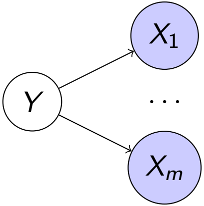
  because $\mathbf{y}$ is unknown when an HMM is used for sequence labelling
- A $y \in \mathcal{Y}$ is sometimes called a *hidden state* because an HMM can be viewed as a *stochastic automaton*
  - each different $y \in \mathcal{Y}$ is a state in the automaton
  - the x are *emissions* from the automaton

MACQUARIE UNIVERSITY

Review: Naive Bayes models

- The naive Bayes model:

$$\mathrm{P}(Y, X_1, \ldots, X_m) \quad = \quad \mathrm{P}(Y)\,\mathrm{P}(X_1 \mid Y) \ldots \mathrm{P}(X_m \mid Y)$$



- In a Naive Bayes classifier, the X_i are *features* and Y is the class label we want to predict
  - $\mathrm{P}(X_i \mid Y)$ can be Bernoulli, multinomial, Gaussian etc.

Review: Markov models and *n*-gram language models

- An bigram language model is a *first-order Markov model* that *factorises the distribution over a sequence* $\mathbf{y}$ *into a product of conditional distributions*:

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

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

$$\$ \longrightarrow Y_1 \longrightarrow Y_2 \longrightarrow Y_3 \longrightarrow Y_4 \longrightarrow \$$$

MACQUARIE UNIVERSITY

Hidden Markov models

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

$$
\begin{aligned}
\mathrm{P}(\mathbf{X}, \mathbf{Y}) &= \mathrm{P}(\mathbf{X} \mid \mathbf{Y})\, \mathrm{P}(\mathbf{Y}) \\
\mathrm{P}(\mathbf{Y}{=}(\$, y_1, \ldots, y_n, \$)) &= \prod_{i=1}^{n+1} \mathrm{P}(Y_i{=}y_i \mid Y_{i-1}{=}y_{i-1}) \\
&= \prod_{i=1}^{n+1} s_{y_i, y_{i-1}} \qquad (i.e., \mathrm{P}(Y_i{=}y \mid Y_{i-1}{=}y') = s_{y,y'}) \\
\mathrm{P}(\mathbf{X}{=}(x_1, \ldots, x_n) \mid \mathbf{Y}{=}(\$, y_1, \ldots, y_n, \$)) & \\
&= \prod_{i=1}^{n} \mathrm{P}(X_i{=}x_i \mid Y_i{=}y_i) \\
&= \prod_{i=1}^{n} t_{x_i, y_i} \qquad (i.e., \mathrm{P}(X_i{=}x \mid Y_i{=}y) = t_{x,y})
\end{aligned}
$$

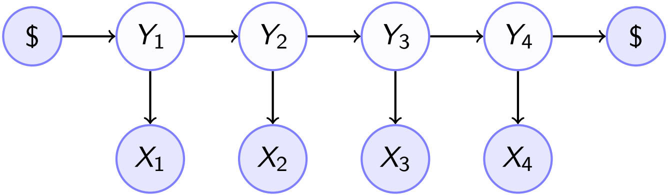
MACQUARIE UNIVERSITY

Hidden Markov models as Bayes nets

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

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

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



The parameters of an HMM

- An HMM is specified by two matrices:
  - a *matrix* $\mathbf{s}$, where $s_{y,y'} = \mathrm{P}(Y_i{=}y \mid Y_{i-1}{=}y')$ is the *probability of a label y following the label y'*
    - this is the same as in a bigram language model
    - remember to pad $\mathbf{y}$ with begin/end of sentence markers \$
  - a *matrix* $\mathbf{t}$, where $t_{x,y} = \mathrm{P}(X_i{=}x \mid Y_i = y)$ is the *probability of an item x given the label y*
    - similiar to the feature-class probability $\mathrm{P}(F_j \mid Y)$ in naive Bayes
- If the set of labels is $\mathcal{Y}$ and the vocabulary is $\mathcal{X}$, then
  - $\mathbf{s}$ is an $m \times m$ matrix, where $m = |\mathcal{Y}|$.
  - $\mathbf{t}$ is a $v \times m$ matrix, where $v = |\mathcal{X}|$.

HMM estimate of a labelled sequence's probability

$\mathbf{s} =$

$y_i \backslash y_{i-1}$	\$	*Verb*	*Det*	*Noun*
\$	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

$\mathbf{t} =$

$x_i \backslash y_i$	*Verb*	*Det*	*Noun*
spread	0.5	0.1	0.4
the	0.1	0.8	0.1
butter	0.4	0.1	0.5

$$
\begin{aligned}
& \mathrm{P}\left(\mathbf{X}{=}(\text{spread}, \text{the}, \text{butter}), \mathbf{Y}{=}(\text{Verb}, \text{Det}, \text{Noun})\right) \\
= \;& s_{\$,\text{Verb}} \cdot s_{\text{Det},\text{Verb}} \cdot s_{\text{Noun},\text{Det}} \cdot s_{\$,\text{Noun}} \\
& \cdot t_{\text{spread},\text{Verb}} \cdot t_{\text{the},\text{Det}} \cdot t_{\text{butter},\text{Noun}} \\
= \;& 0.4 \cdot 0.4 \cdot 0.7 \cdot 0.3 \cdot 0.5 \cdot 0.8 \cdot 0.5 \\
\approxeq \;& 0.009
\end{aligned}
$$

MACQUARIE UNIVERSITY

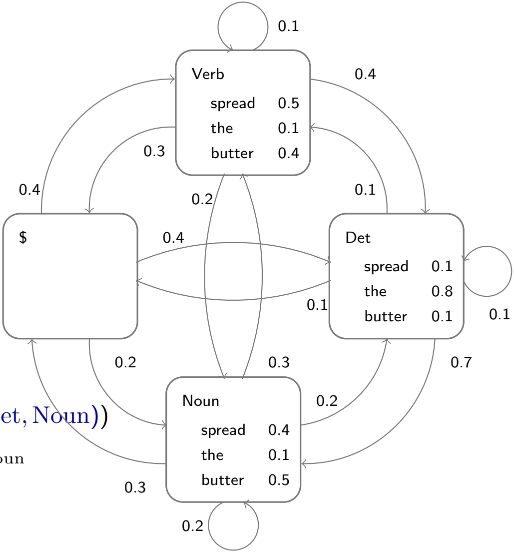
A HMM as a stochastic automaton

State to state transition probabilities

$y_i \backslash y_{i-1}$	\$	*Verb*	*Det*	*Noun*
\$	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

State to word emission probabilities

$x_i \backslash y_i$	*Verb*	*Det*	*Noun*
spread	0.5	0.1	0.4
the	0.1	0.8	0.1
butter	0.4	0.1	0.5



$$
\begin{aligned}
\mathrm{P}\left(\mathbf{X}{=}(\text{spread}, \text{the}, \text{butter}), \mathbf{Y}{=}(\text{Verb}, \text{Det}, \text{Noun})\right) \\
&= s_{\$,\text{Verb}} \cdot s_{\text{Det},\text{Verb}} \cdot s_{\text{Noun},\text{Det}} \cdot s_{\$,\text{Noun}} \\
&\quad \cdot t_{\text{spread},\text{Verb}} \cdot t_{\text{the},\text{Det}} \cdot t_{\text{butter},\text{Noun}} \\
&= 0.4 \cdot 0.4 \cdot 0.7 \cdot 0.3 \cdot 0.5 \cdot 0.8 \cdot 0.5 \\
&\approx 0.009
\end{aligned}
$$

Estimating an HMM from labelled data

- Training data consists of *labelled sequences of data items*
- Estimate $s_{y,y'} = \mathrm{P}(Y_i{=}y \mid Y_{i-1}{=}y')$ from label sequences $\mathbf{y}$ (just as for language models)
  - 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:

$$\hat{\hat{s}}_{y,y'} \quad = \quad \frac{n_{y,y'} + 1}{n_{y'} + |\mathcal{Y}|}$$

  - Be sure to *count the end-markers* $
- Estimate $t_{x,y} = \mathrm{P}(X_i = x \mid Y_i = y)$ from pairs of item sequences $\mathbf{x}$ and their corresponding label sequence $\mathbf{y}$
  - 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:

$$\hat{\hat{t}}_{x,y} \quad = \quad \frac{r_{x,y} + 1}{r_y + |\mathcal{X}|}$$

Estimating an HMM example

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

$$\mathbf{n} = \begin{array}{c|cccc} y_i \backslash y_{i-1} & \$ & \mathit{Verb} & \mathit{Det} & \mathit{Noun} \\ \hline \$ & 0 & 0 & 0 & 2 \\ \mathit{Verb} & 2 & 0 & 0 & 0 \\ \mathit{Det} & 0 & 1 & 0 & 0 \\ \mathit{Noun} & 0 & 1 & 1 & 0 \end{array} \qquad \hat{\hat{\mathbf{s}}} = \begin{array}{c|cccc} y_i \backslash y_{i-1} & \$ & \mathit{Verb} & \mathit{Det} & \mathit{Noun} \\ \hline \$ & 1/6 & 1/6 & 1/5 & 3/6 \\ \mathit{Verb} & 3/6 & 1/6 & 1/5 & 1/6 \\ \mathit{Det} & 1/6 & 2/6 & 1/5 & 1/6 \\ \mathit{Noun} & 1/6 & 2/6 & 2/5 & 1/6 \end{array}$$

$$\mathbf{r} = \begin{array}{c|ccc} x_i \backslash y_i & \mathit{Verb} & \mathit{Det} & \mathit{Noun} \\ \hline \mathit{spread} & 1 & 0 & 1 \\ \mathit{the} & 0 & 1 & 0 \\ \mathit{butter} & 1 & 0 & 1 \end{array} \qquad \hat{\hat{\mathbf{t}}} = \begin{array}{c|ccc} x_i \backslash y_i & \mathit{Verb} & \mathit{Det} & \mathit{Noun} \\ \hline \mathit{spread} & 2/5 & 1/4 & 2/5 \\ \mathit{the} & 1/5 & 2/4 & 1/5 \\ \mathit{butter} & 2/5 & 1/4 & 2/5 \end{array}$$

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

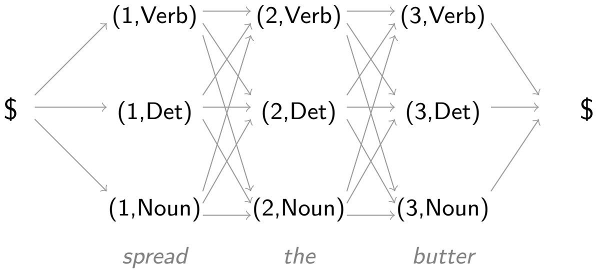
- When we use an HMM, we're given data items $\mathbf{x}$ and want to return *the most probable label sequence*:

$$\widehat{\mathbf{y}}(\mathbf{x}) \quad = \quad \underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \, \mathrm{P}(\mathbf{X}{=}\mathbf{x} \mid \mathbf{Y}{=}\mathbf{y}) \, \mathrm{P}(\mathbf{Y}{=}\mathbf{y})$$

- If $\mathbf{x}$ 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 $\widehat{\mathbf{y}}(\mathbf{x})$ 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 $\mathbf{x}$ 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 $\mathbf{y}$ is *a path through the trellis*



Using a trellis to find the most probable label sequence $\mathbf{y}$

- One-to-one correspondence between *paths from start to finish in the trellis* and *label sequences* $\mathbf{y}$
- 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

$$\mathrm{P}(\mathbf{X}{=}\mathbf{x}, \mathbf{Y}{=}\mathbf{y})$$

  - use *dynamic programming* to find highest scoring path from start to finish
  - return the corresponding label sequence $\widehat{\mathbf{y}}(\mathbf{x})$

Conditioned on the input $\mathbf{X}$, the distribution $\mathrm{P}(\mathbf{Y} \mid \mathbf{X})$ is an *inhomogenous (i.e., time-varying) Markov chain*

- the transition probabilities $\mathrm{P}(Y_i \mid Y_{i-1}, X_i)$ 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

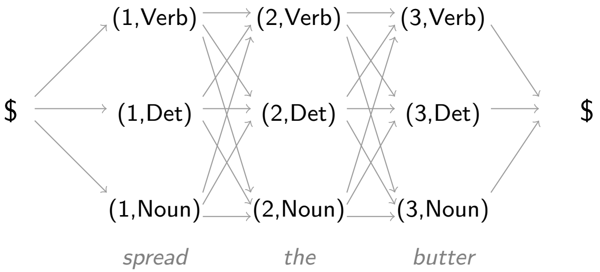
$$
\begin{aligned}
\mathrm{P}(\mathbf{X}, \mathbf{Y}) &= \mathrm{P}(\mathbf{Y})\,\mathrm{P}(\mathbf{X} \mid \mathbf{Y}) \\
&= s_{y_1,\$} \cdot s_{y_2,y_1} \cdot \ldots \cdot s_{y_n,y_{n-1}} \cdot s_{\$,y_n} \cdot t_{x_1,y_1} \cdot t_{x_2,y_2} \cdot \ldots \cdot t_{x_n,y_n} \\
&= s_{y_1,\$} \cdot t_{x_1,y_1} \cdot s_{y_2,y_1} \cdot t_{x_2,y_2} \cdot \ldots \cdot s_{y_n,y_{n-1}} \cdot t_{x_n,y_n} \cdot s_{\$,y_n} \\
&= \left(\prod_{i=1}^{n} s_{y_i,y_{i-1}} \cdot t_{x_i,y_i} \right) \cdot s_{\$,y_n}
\end{aligned}
$$

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

$\Rightarrow$ product of weights on edges of path $\mathbf{y}$ in trellis is $\mathrm{P}(\mathbf{X}{=}\mathbf{x}, \mathbf{Y}{=}\mathbf{y})$

MACQUARIE UNIVERSITY

Trellis for *spread the butter*



s =

$y_i \backslash y_{i-1}$	\$	*Verb*	*Det*	*Noun*
\$	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

t =

$x_i \backslash y_i$	*Verb*	*Det*	*Noun*
spread	0.5	0.1	0.4
the	0.1	0.8	0.1
butter	0.4	0.1	0.5

$\mathbf{w}_1$ =

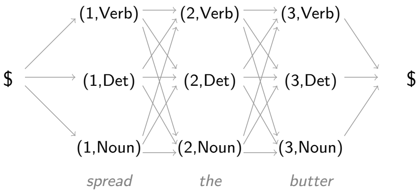
$y_1 \backslash y_0$	\$
Verb	0.2
Det	0.04
Noun	0.08

$\mathbf{w}_2$ =

$y_2 \backslash y_1$	*Verb*	*Det*	*Noun*
Verb	0.01	0.01	0.03
Det	0.32	0.08	0.16
Noun	0.02	0.07	0.02

MACQUARIE UNIVERSITY

Finding the highest-scoring path

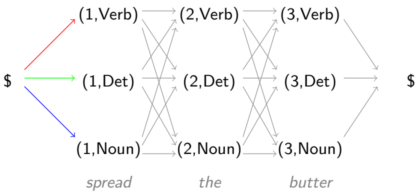


- The *score* of a path is the *product of the weights of the edges* along that path
- Key insight: the highest-scoring path to a node (i, y) must begin with a highest-scoring path to some node $(i-1, y')$
- Let $\mathrm{MaxScore}(i, y)$ be the *score of the highest scoring path to node (i, y)*. Then:

$$
\begin{aligned}
\mathrm{MaxScore}(1, y) &= w_{1,y,\$} \\
\mathrm{MaxScore}(i, y) &= \max_{y' \in \mathcal{Y}} \mathrm{MaxScore}(i-1, y') \cdot w_{i,y,y'} \quad \text{if } 1 < 2 \leq n \\
\mathrm{MaxScore}(n+1, \$) &= \max_{y' \in \mathcal{Y}} \mathrm{MaxScore}(n, y) \cdot w_{n+1,\$,y}
\end{aligned}
$$

- To find the highest scoring path, compute $\mathrm{MaxScore}(i, y)$ for $i = 1, 2, \ldots, n+1$ in turn

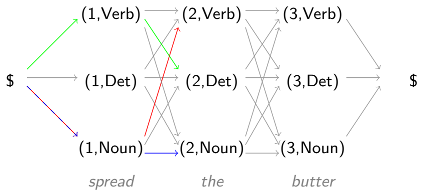
Finding the highest-scoring path example (1)



$$\mathbf{w}_1 = \begin{array}{c|c} y_1 \backslash y_0 & \$ \\ \hline \textit{Verb} & 0.2 \\ \textit{Det} & 0.04 \\ \textit{Noun} & 0.08 \end{array}$$

$$\begin{aligned} \mathrm{MaxScore}(1, \mathrm{Verb}) &= 0.2 \\ \mathrm{MaxScore}(1, \mathrm{Det}) &= 0.04 \\ \mathrm{MaxScore}(1, \mathrm{Noun}) &= 0.08 \end{aligned}$$

Finding the highest-scoring path example (2)



$$\begin{aligned} \mathrm{MaxScore}(1,\mathrm{Verb}) &= 0.2 \\ \mathrm{MaxScore}(1,\mathrm{Det}) &= 0.04 \\ \mathrm{MaxScore}(1,\mathrm{Noun}) &= 0.08 \end{aligned}$$

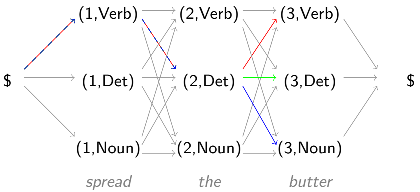
$\mathbf{w}_2 =$

$y_2 \backslash y_1$	*Verb*	*Det*	*Noun*
Verb	0.01	0.01	0.03
Det	0.32	0.08	0.16
Noun	0.02	0.07	0.02

$$\begin{aligned} \mathrm{MaxScore}(2,\mathrm{Verb}) &= 0.0024 && \textit{via Noun} \\ \mathrm{MaxScore}(2,\mathrm{Det}) &= 0.064 && \textit{via Verb} \\ \mathrm{MaxScore}(2,\mathrm{Noun}) &= 0.004 && \textit{via Noun} \end{aligned}$$

MACQUARIE UNIVERSITY

Finding the highest-scoring path example (3)



$$
\begin{aligned}
\mathrm{MaxScore}(2, \mathrm{Verb}) &= 0.0024 \\
\mathrm{MaxScore}(2, \mathrm{Det}) &= 0.064 \\
\mathrm{MaxScore}(2, \mathrm{Noun}) &= 0.004
\end{aligned}
$$

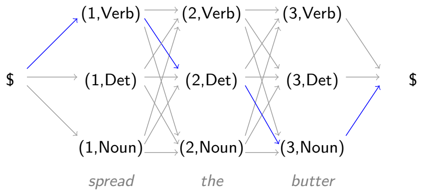
$\mathbf{w}_3 =$

$y_3 \backslash y_2$	*Verb*	*Det*	*Noun*
Verb	0.04	0.04	0.12
Det	0.04	0.01	0.02
Noun	0.1	0.35	0.1

$$
\begin{aligned}
\mathrm{MaxScore}(3, \mathrm{Verb}) &= 0.00256 \quad \textit{via Det} \\
\mathrm{MaxScore}(3, \mathrm{Det}) &= 0.00064 \quad \textit{via Det} \\
\mathrm{MaxScore}(3, \mathrm{Noun}) &= 0.0224 \quad \textit{via Det}
\end{aligned}
$$

MACQUARIE UNIVERSITY

Finding the highest-scoring path example (4)



$$\begin{aligned}\mathrm{MaxScore}(3,\mathrm{Verb}) &= 0.00256\\ \mathrm{MaxScore}(3,\mathrm{Det}) &= 0.00064\\ \mathrm{MaxScore}(3,\mathrm{Noun}) &= 0.0224\end{aligned}$$

$\mathbf{w}_4 =$

$\$ \backslash y_3$	*Verb*	*Det*	*Noun*
\$	0.3	0.1	0.3

$$\mathrm{MaxScore}(4,\$) = 0.00672 \quad \textit{via Noun}$$

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 compute:
  - the marginal probability of any state given the input $\mathrm{P}(Y_i{=}y \mid \mathbf{X} = x)$
  - the marginal probability of any adjacent pair of states $\mathrm{P}(Y_{i-1}{=}y, Y_i{=}y' \mid \mathbf{X} = \mathbf{x})$
  - the *expected number of times any pair of states is seen in the input* $\mathrm{E}[n_{y,y'} \mid \mathbf{x}]$
- These are required for estimating HMMs from unlabelled data using the *Expectation-Maximisation Algorithm*

MACQUARIE UNIVERSITY

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:

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

  (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:

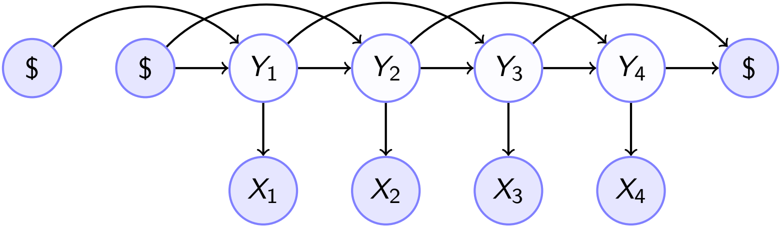
$$\mathrm{P}(\mathbf{Y}) = \prod_{i=1}^{n+1} \mathrm{P}(Y_i \mid Y_{i-1}, Y_{i-2})$$

- 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 $m = |\mathcal{Y}|$

$$\mathrm{P}(Y_i{=}y \mid Y_{i-1}{=}y', Y_{i-2}{=}y'') = s_{y,y',y''}$$

Bayes net representation of a 2nd-order HMM

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



MACQUARIE UNIVERSITY

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 $\mathbf{X}$ of length n and where the number of states $|\mathcal{Y}| = m$, *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 $O(n \cdot m^{k+1})$*.

MACQUARIE UNIVERSITY

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