A gentle introduction to Hidden Markov Models

Mark Johnson
Brown University

November 2009

Outline

What is sequence labeling?

Markov models

Hidden Markov models

Finding the most likely state sequence

Estimating HMMs

Conclusion

Sequence labeling

- Input: a sequence $\boldsymbol{x} = (x_1, \ldots, x_n)$ (usually n may vary)
- Output: a *sequence* $\boldsymbol{y} = (x_1, \ldots, y_n)$, where *$y_i$ is a label for x_i*

Part of speech (POS) tagging:

$$\begin{array}{llllllll} \boldsymbol{y}: & \text{DT} & \text{JJ} & \text{NN} & \text{VBD} & \text{NNP} & . \\ \boldsymbol{x}: & \text{the} & \text{big} & \text{cat} & \text{bit} & \text{Sam} & . \end{array}$$

Noun-phrase chunking:

$$\begin{array}{lllllll} \boldsymbol{y}: & [\text{NP} & \text{NP} & \text{NP}] & \_ & [\text{NP}] & . \\ \boldsymbol{x}: & \text{the} & \text{big} & \text{cat} & \text{bit} & \text{Sam} & . \end{array}$$

Named entity detection:

$$\begin{array}{llllllll} \boldsymbol{y}: & [\text{CO} & \text{CO}] & \_ & [\text{LOC}] & \_ & [\text{PER}] & \_ \\ \boldsymbol{x}: & \text{XYZ} & \text{Corp.} & \text{of} & \text{Boston} & \text{announced} & \text{Spade's} & \text{resignation} \end{array}$$

Speech recognition: The $\boldsymbol{x}$ are 100 msec. time slices of acoustic input, and the $\boldsymbol{y}$ are the corresponding phonemes (i.e., y_i is the phoneme being uttered in time slice x_i)

Sequence labeling with probabilistic models

- General idea: develop a probabilistic model $\mathrm{P}(\boldsymbol{y}, \boldsymbol{x})$
  - use this to compute $\mathrm{P}(\boldsymbol{y}|\boldsymbol{x})$ and $\hat{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{argmax}_{\boldsymbol{y}}\, \mathrm{P}(\boldsymbol{y}|\boldsymbol{x})$
- A *Hidden Markov Model* (HMM) factors:

$$\mathrm{P}(\boldsymbol{y}, \boldsymbol{x}) \;\; = \;\; \mathrm{P}(\boldsymbol{y})\, \mathrm{P}(\boldsymbol{x}|\boldsymbol{y})$$

- $\mathrm{P}(\boldsymbol{y})$ is the probability of sequence $\boldsymbol{y}$
  - In an HMM, $\mathrm{P}(\boldsymbol{y})$ is a *Markov chain*
    - in applications, $\boldsymbol{y}$ is usually "hidden"
  - $\mathrm{P}(\boldsymbol{x}|\boldsymbol{y})$ is the *emission model*.

Outline

Markov models of sequences

- Let $\boldsymbol{y} = (y_1, \ldots, y_n)$, where $y_i \in \mathcal{Y}$ and n is fixed

$$\begin{aligned} \mathrm{P}(\boldsymbol{y}) &= \mathrm{P}(y_1)\,\mathrm{P}(y_2|y_1)\,\mathrm{P}(y_3|y_1, y_2)\,\ldots\,\mathrm{P}(y_n|y_1, \ldots, y_{n-1}) \\ &= \mathrm{P}(y_1) \prod_{i=2}^{n} \mathrm{P}(y_i|y_{1:i-1}) \end{aligned}$$

  where $y_{i:j}$ is the subsequence $(y_i, \ldots, y_j)$ of $\boldsymbol{y}$
- In a (first-order) *Markov chain*:

$$\mathrm{P}(y_i|y_{1:i-1}) = \mathrm{P}(y_i|y_{i-1}) \text{ for } i > 1$$

- In a *homogenous* Markov chain, $\mathrm{P}(y_i|y_{i-1})$ does not depend on i, so probability of a sequence is completely determined by:

$$\begin{aligned} \mathrm{P}(y_1) &= \iota(y_1) && \textit{(start probabilities)} \\ \mathrm{P}(y_i|y_{i-1}) &= \sigma(y_{i-1}, y_i) && \textit{(transition probabilities)}, \text{ so:} \\ \mathrm{P}(\boldsymbol{y}) &= \iota(y_1) \prod_{i=2}^{n} \sigma(y_{i-1}, y_i) \end{aligned}$$

Markov models of varying-length sequences

- To define a model of sequences $\boldsymbol{y} = (y_1, \ldots, y_n)$ of varying lengths n, need to define $\mathrm{P}(n)$
- An easy way to do this:
  - add a new *end marker* symbol '$\triangleleft$' to $\mathcal{Y}$
  - pad all sequences with this end-marker

$$\boldsymbol{y} \quad = \quad y_1, y_2, \ldots, y_n, \triangleleft, \triangleleft, \ldots$$

  - set $\sigma(\triangleleft, \triangleleft) = 1$, i.e., $\triangleleft$ is a *sink state*
  - so $\sigma(y, \triangleleft)$ is probability of sequence ending after state y
- We can also incorporate the start probabilities $\boldsymbol{\iota}$ into $\boldsymbol{\sigma}$ by introducing a new *begin marker* $y_0 = \triangleright$

$$\begin{aligned}\boldsymbol{y} \quad &= \quad \triangleright, y_1, y_2, \ldots, y_n, \triangleleft, \triangleleft, \ldots, \text{ so:} \\ \mathrm{P}(\boldsymbol{y}) \quad &= \quad \prod_{i=1}^{\infty} \sigma(y_{i-1}, y_i) \quad \text{(only compute to } i = n+1\text{)}\end{aligned}$$

Outline

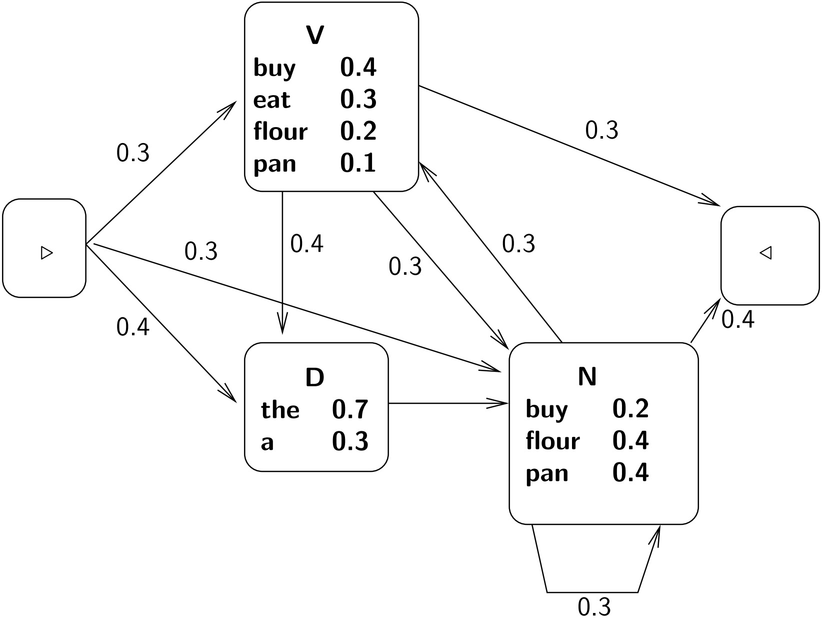
Hidden Markov models

- An HMM is specified by:
  - A *state-to-state transition matrix* $\boldsymbol{\sigma}$, where $\sigma(y, y')$ is the probability of moving from state y to state y'
  - A *state-to-observation emission matrix* $\boldsymbol{\tau}$, where $\tau(y, x)$ is the probability of emitting x in state y

$$
\begin{aligned}
\mathrm{P}(\boldsymbol{y}, \boldsymbol{x}) &= \mathrm{P}(\boldsymbol{y})\,\mathrm{P}(\boldsymbol{x}|\boldsymbol{y}) \\
\mathrm{P}(\boldsymbol{y}) &= \prod_{i=1}^{n+1} \mathrm{P}(y_i|y_{i-1}) \;=\; \prod_{i=1}^{n+1} \sigma(y_{i-1}, y_i) \\
\mathrm{P}(\boldsymbol{x}|\boldsymbol{y}) &= \prod_{i=1}^{n} \mathrm{P}(x_i|y_i) \;=\; \prod_{i=1}^{n} \tau(y_i, x_i)
\end{aligned}
$$

- I.e., each x_i is generated conditioned on y_i with probability $\tau(y_i, x_i)$

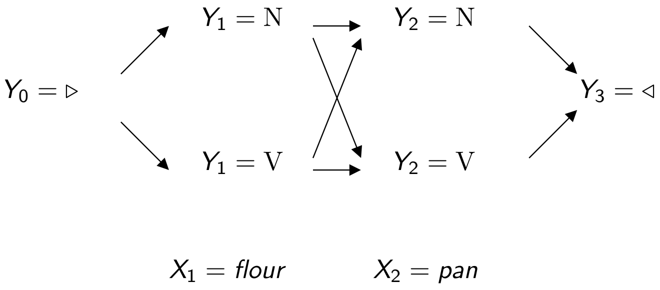
HMMs as stochastic automata



Bayes net representation of an HMM

$$\begin{tikzcd}
Y_0 = \triangleleft \arrow[r] & Y_1 = \mathrm{V} \arrow[r] \arrow[d] & Y_2 = \mathrm{D} \arrow[r] \arrow[d] & Y_3 = \mathrm{N} \arrow[r] \arrow[d] & Y_4 = \triangleleft \\
 & X_1 = \text{flour} & X_2 = \text{the} & X_3 = \text{pan} &
\end{tikzcd}$$

Trellis representation of HMM



- The trellis representation represents the possible state sequences $\boldsymbol{y}$ for *a single observation sequence* $\boldsymbol{x}$
- A trellis is a DAG with *nodes* (i, y), where $i \in 0, \ldots, n+1$ and $y \in \mathcal{Y}$ and *edges* from $(i-1, y')$ to (i, y)
- It plays a crucial role in *dynamic programming algorithms* for HMMs

Outline

Two important inference problems

- Given an HMM $\boldsymbol{\sigma}, \boldsymbol{\tau}$ and an observation sequence $\boldsymbol{x}$:
  - what is *the most likely state sequence* $\hat{\boldsymbol{y}} = \mathrm{argmax}_{\boldsymbol{y}} \mathrm{P}(\boldsymbol{x}, \boldsymbol{y})$?
  - what is the *probability of a sequence* $\mathrm{P}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \mathrm{P}(\boldsymbol{x}, \boldsymbol{y})$?
- If the sequences are short enough, can be solved by exhaustively enumerating $\boldsymbol{y}$
- Given a sequence of length n and an HMM with m states, can be solved using dynamic programming in $O(n\, m^2)$ time

Key observation: $\mathrm{P}(\boldsymbol{x}, \boldsymbol{y})$ factorizes!

$$\begin{aligned}
\mathrm{P}(\boldsymbol{x}, \boldsymbol{y}) &= \left(\prod_{i=1}^{n} \sigma(y_{i-1}, y_i)\, \tau(y_i, x_i)\right) \sigma(y_n, \triangleleft), \text{ so:} \\
-\log \mathrm{P}(\boldsymbol{x}, \boldsymbol{y}) &= \left(\sum_{i=1}^{n} -\log(\sigma(y_{i-1}, y_i)\, \tau(y_i, x_i))\right) - \log \sigma(y_n, \triangleleft)
\end{aligned}$$

- Given observation sequence $\boldsymbol{x}$, define a trellis with *edge weights*

$$\begin{aligned}
w((i-1, y'), (i, y)) &= -\log(\sigma(y_{i-1}, y_i)\, \tau(y_i, x_i)) \\
w((n, y), (n+1, \triangleleft)) &= -\log(\sigma(y, \triangleleft))
\end{aligned}$$

- Then the $-\log$ probability of any state sequence $\boldsymbol{y}$ is the sum of the weights of the corresponding path in the trellis

Finding the most likely state sequence $\hat{\boldsymbol{y}}$

- Goal: find $\hat{\boldsymbol{y}} = \operatorname{argmax}_{\boldsymbol{y}} \mathrm{P}(\boldsymbol{x}, \boldsymbol{y})$
- Algorithm:
  - Construct a trellis with edge weights

$$
\begin{aligned}
w((i-1, y'), (i, y)) &= -\log(\sigma(y_{i-1}, y_i)\,\tau(y_i, x_i)) \\
w((n, y), (n+1, \triangleleft)) &= -\log(\sigma(y, \triangleleft))
\end{aligned}
$$

  - Run a *shortest path algorithm* on this trellis, which returns a state sequence with lowest $-\log$ probability

The Viterbi algorithm for finding $\hat{\boldsymbol{y}}$

- Idea: given $\boldsymbol{x} = (x_1, \ldots, x_n)$, let $\mu(k, u)$ be the maximum probability of any state sequence $y_1, \ldots, y_k$ ending in $y_k = u \in \mathcal{Y}$.

$$\mu(k, u) \quad = \quad \max_{\boldsymbol{y}_{1:k} \,:\, y_k = u} \mathrm{P}(\boldsymbol{x}_{1:k}, \boldsymbol{y}_{1:k})$$

  - Base case: $\mu(0, \triangleright) = 1$
  - Recursive case:

$$\mu(k, u) \quad = \quad \max_{u'} \mu(k-1, u')\, \sigma(u', u)\, \tau(u, x_k)$$

  - Final case: $\mu(n+1, \triangleleft) \;=\; \max_{u'} \mu(n, u')\, \sigma(u', \triangleleft)$
- To find $\hat{\boldsymbol{y}}$, for each trellis node (k, u) keep a *back-pointer* to *most likely predecessor*

$$\rho(k, u) \quad = \quad \operatorname*{argmax}_{u' \in \mathcal{Y}} \mu(k-1, u')\, \sigma(u', u)\, \tau(u, x_k)$$

Why does the Viterbi algorithm work?

$$
\begin{aligned}
\mu(k,u) &= \max_{\boldsymbol{y}_{1:k}\,:\,y_k=u} \mathrm{P}(\boldsymbol{x}_{1:k},\boldsymbol{y}_{1:k}) \\
&= \max_{\boldsymbol{y}_{1:k}\,:\,y_k=u} \prod_{i=1}^{k} \sigma(y_{i-1},y_i)\,\tau(y_i,x_i) \\
&= \max_{\boldsymbol{y}_{1:k-1}} \left(\prod_{i=1}^{k-1} \sigma(y_{i-1},y_i)\,\tau(y_i,x_i) \right) \sigma(y_{k-1},u)\,\tau(u,x_k) \\
&= \max_{u'} \left(\max_{\boldsymbol{y}_{1:k-1}\,:\,y_{k-1}=u'} \prod_{i=1}^{k-1} \sigma(y_{i-1},y_i)\,\tau(y_i,x_i) \right) \sigma(u',u)\,\tau(u,x_k) \\
&= \max_{u'} \mu(k-1,u')\,\sigma(u',u)\,\tau(u,x_k)
\end{aligned}
$$

The sum-product algorithm for computing $\mathrm{P}(\boldsymbol{x})$

- Goal: compute $\mathrm{P}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}^n} \mathrm{P}(\boldsymbol{x}, \boldsymbol{y})$
- Idea: given $\boldsymbol{x} = (x_1, \ldots, x_n)$, let $\alpha(k, u)$ be the *sum of the probabilities* of $x_1, \ldots, x_k$ and all state sequences $y_1, \ldots, y_k$ ending in $y_k = u$.

$$\alpha(k, u) \;=\; \sum_{y_{1:k} \,:\, y_k = u} \mathrm{P}(\boldsymbol{x}_{1:k}, \boldsymbol{y}_{1:k})$$

  - Base case: $\alpha(0, \triangleright) = 1$
  - Recursive case: for $k = 1, \ldots, n$:

$$\alpha(k, u) \;=\; \sum_{u'} \alpha(k-1, u')\, \sigma(u', u)\, \tau(u, x_i)$$

  - Final case: $\alpha(n+1, \triangleleft) \;=\; \sum_{u'} \alpha(n, u')\, \sigma(u', \triangleleft)$
- The $\boldsymbol{\alpha}$ are called *forward probabilities*

Outline

Estimating HMMs from *visible data*

- Training data: $D = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n))$, where the $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are *sequences*
  - for conceptual simplicity, assume that the data comes as two long sequences $D = (\boldsymbol{x}, \boldsymbol{y})$
- Then the MLE is:

$$
\begin{aligned}
\hat{\sigma}(y', y) &= \frac{c(y', y)}{\sum_{y \in \mathcal{Y}} c(y', y)} \\
\hat{\tau}(y, x) &= \frac{c(y, x)}{\sum_{x \in \mathcal{X}} c(y, x)}, \text{ where:} \\
c(y', y) &= \text{the number of times state } y' \text{ precedes state } y \text{ in } \boldsymbol{y} \\
c(y, x) &= \text{the number of times state } y \text{ emits } x \text{ in } (\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

Estimating HMMs from *hidden data* using EM

- EM iteration at time t:

$$\sigma(y', y)^{(t)} = \frac{\mathrm{E}[c(y', y)]}{\sum_{y \in \mathcal{Y}} \mathrm{E}[c(y', y)]}$$

$$\tau(y, x)^{(t)} = \frac{\mathrm{E}[c(y, x)]}{\sum_{x \in \mathcal{X}} \mathrm{E}[c(y, x)]}, \text{ where:}$$

$$\mathrm{E}[c(y', y)] = \sum_{\boldsymbol{y} \in \mathcal{Y}^n} c(y', y) \, \mathrm{P}(\boldsymbol{y} | \boldsymbol{x}, \boldsymbol{\sigma}^{(t-1)}, \boldsymbol{\tau}^{(t-1)})$$

$$\mathrm{E}[c(y, x)] = \sum_{\boldsymbol{y} \in \mathcal{Y}^n} c(y, x) \, \mathrm{P}(\boldsymbol{y} | \boldsymbol{x}, \boldsymbol{\sigma}^{(t-1)}, \boldsymbol{\tau}^{(t-1)})$$

- Key computational problem: *computing these expectations efficiently*
- Dynamic programming to the rescue!

Expectations require marginal probabilities

- Suppose we could efficiently compute the *marginal probabilities*:

$$\mathrm{P}(Y_i = y' | \boldsymbol{x}) \;\;=\;\; \frac{\sum_{\boldsymbol{y} \in \mathcal{Y}^n \,:\, y_i = y'} \mathrm{P}(\boldsymbol{y}, \boldsymbol{x})}{\mathrm{P}(\boldsymbol{x})}$$

- Then we could efficiently compute:

$$\mathrm{E}[c(y', x')] \;\;=\;\; \sum_{i \,:\, x_i = x'} \mathrm{P}(Y_i = y' | \boldsymbol{x})$$

- Efficiently computing $\mathrm{E}[c(y', y)]$ requires computation of a different marginal probability

Backward probabilities

- Idea: given $\boldsymbol{x} = (x_1, \ldots, x_n)$, let $\beta(k, v)$ be the *sum of the probabilities* of $x_{k+1}, \ldots, x_n$ and all state sequences $y_{k+1}, \ldots, y_n$ conditioned on $y_k = v$.

$$\beta(k, v) \quad = \quad \sum_{y_{k:n} \,:\, y_k = v} \mathrm{P}(\boldsymbol{x}_{k+1:n}, \boldsymbol{y}_{k+1:n} | Y_k = v)$$

  - Base case: $\beta(n, v) = \sigma(v, \triangleleft)$ for each $v \in \mathcal{Y}$
  - Recursive case: for $k = n-1, \ldots, 0$:

$$\beta(k, v) \quad = \quad \sum_{v'} \sigma(v, v')\, \tau(v', x_{k+1})\, \beta(k+1, v')$$

- The $\boldsymbol{\beta}$ are called *backward probabilities*

Computing marginals from forward and backward probabilities

$$
\begin{aligned}
\mathrm{P}(Y_i = u|\boldsymbol{x}) &= \frac{1}{\mathrm{P}(\boldsymbol{x})} \sum_{\boldsymbol{y}\in\mathcal{Y}^n \,:\, y_i=u} \mathrm{P}(\boldsymbol{y},\boldsymbol{x}) \\
&= \frac{1}{\mathrm{P}(\boldsymbol{x})} \sum_{y_{1:k} \,:\, y_i=u} \; \sum_{y_{i:n} \,:\, y_i=u} \mathrm{P}(\boldsymbol{x}_{1:i},\boldsymbol{y}_{1:i})\mathrm{P}(\boldsymbol{x}_{i+1:n},\boldsymbol{y}_{i:n}) \\
&= \frac{1}{\mathrm{P}(\boldsymbol{x})} \left(\sum_{y_{1:k} \,:\, y_i=u} \mathrm{P}(\boldsymbol{x}_{1:i},\boldsymbol{y}_{1:i}) \right) \left(\sum_{y_{k:n} \,:\, y_i=u} \mathrm{P}(\boldsymbol{x}_{i+1:n},\boldsymbol{y}_{i:n}) \right) \\
&= \frac{1}{\mathrm{P}(\boldsymbol{x})} \, \alpha(i,u)\, \beta(i,u)
\end{aligned}
$$

- So the marginal $\mathrm{P}(Y_i|\boldsymbol{x})$ can be efficiently computed from $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$

Outline

Conclusion

- Hidden Markov Models (HMMs) can be used to label a sequence of observations
- There are efficient dynamic programming algorithms to find the most likely label sequence and the marginal probability of a label
- The expectation-maximization algorithm can be used to learn an HMM from unlabeled data
- The expected values required for EM can be efficiently computed