Hidden Markov Models
Reminders

• Practice Problems for Exam 2
  – Out: Fri, Mar 20

• Midterm Exam 2
  – Thu, Apr 2 – evening exam, details announced on Piazza

• Homework 7: HMMs
  – Out: Thu, Apr 02
  – Due: Fri, Apr 10 at 11:59pm

• Today’s In-Class Poll
  – http://poll.mlcourse.org
HMMs: History

- Markov chains: Andrey Markov (1906)
  - Random walks and Brownian motion
- Used in Shannon’s work on information theory (1948)
- Baum-Welsh learning algorithm: late 60’s, early 70’s.
  - Used mainly for speech in 60s-70s.
- Late 80’s and 90’s: David Haussler (major player in learning theory in 80’s) began to use HMMs for modeling biological sequences
- Mid-late 1990’s: Dayne Freitag/Andrew McCallum
  - Freitag thesis with Tom Mitchell on IE from Web using logic programs, grammar induction, etc.
  - McCallum: multinomial Naïve Bayes for text
  - With McCallum, IE using HMMs on CORA
- ...
Higher-order HMMs

- **1st-order HMM (i.e. bigram HMM)**

- **2nd-order HMM (i.e. trigram HMM)**

- **3rd-order HMM**
Higher-order HMMs

- 1st-order HMM (i.e. bigram HMM)
- 2nd-order HMM (i.e. trigram HMM)
- 3rd-order HMM

Hidden States, $y$
Observations, $x$
BACKGROUND: MESSAGE PASSING
Great Ideas in ML: Message Passing

Count the soldiers

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Count the soldiers

Belief: Must be 2 + 1 + 3 = 6 of us

there's 1 of me

2 before you

only see my incoming messages

3 behind you

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Count the soldiers

There's 1 of me

1 before you

Only see my incoming messages

4 behind you

Belief:
Must be
\[ 1 + 1 + 4 = 6 \text{ of us} \]

Belief:
Must be
\[ 1 + 3 = 6 \text{ of us} \]

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of the tree

Belief: Must be 14 of us

wouldn't work correctly with a 'loopy' (cyclic) graph

adapted from MacKay (2003) textbook
THE FORWARD-BACKWARD ALGORITHM
Inference

Question:
True or False: The joint probability of the observations and the hidden states in an HMM is given by:

$$P(X = x, Y = y) = C_{y_1} \left[ \prod_{t=1}^{T} A_{y_t, x_t} \right] \left[ \prod_{t=1}^{T-1} B_{y_{t+1}, y_t} \right]$$

Recall:
Emission matrix, $A$, where $P(X_t = k | Y_t = j) = A_{j,k}, \forall t, k$
Transition matrix, $B$, where $P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k$
Initial probs, $C$, where $P(Y_1 = k) = C_k, \forall k$
Inference

Question:
True or False: The probability of the observations in an HMM is given by:

\[
P(X = x) = \prod_{t=1}^{T} A_{x_t, x_{t-1}} = \prod_{i} p(x_i | y_1, y_2, \ldots, y_T)
\]

Recall:
Emission matrix, \( A \), where \( P(X_t = k | Y_t = j) = A_{j,k}, \forall t, k \)
Transition matrix, \( B \), where \( P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k \)
Initial probs, \( C \), where \( P(Y_1 = k) = C_k, \forall k \)
Inference

Question:
True or False: Suppose each hidden state takes K values. The marginal probability of a hidden state $y_t$ given the observations $x$ is given by:

$$P(Y_t = y_t | X = x) = \sum_{j=1}^{K} B_{j,y_t}$$

Recall:

Emission matrix, $A$, where $P(X_t = k | Y_t = j) = A_{j,k}, \forall t, k$

Transition matrix, $B$, where $P(Y_t = k | Y_{t-1} = j) = B_{j,k}, \forall t, k$

Initial probs, $C$, where $P(Y_1 = k) = C_k, \forall k$
Inference for HMMs

Whiteboard

– Three Inference Problems for an HMM

1. Evaluation: Compute the probability of a given sequence of observations
2. Viterbi Decoding: Find the most-likely sequence of hidden states, given a sequence of observations
3. Marginals: Compute the marginal distribution for a hidden state, given a sequence of observations
Dataset for Supervised Part-of-Speech (POS) Tagging

Data: \( \mathcal{D} = \{ x^{(n)}, y^{(n)} \}_{n=1}^{N} \)

Sample 1:
- \( x^{(1)} = \text{time} \), \( y^{(1)} = \text{flies} \), \( x^{(1)} = \text{like} \), \( y^{(1)} = \text{an} \), \( x^{(1)} = \text{arrow} \)

Sample 2:
- \( x^{(2)} = \text{time} \), \( y^{(2)} = \text{flies} \), \( x^{(2)} = \text{like} \), \( y^{(2)} = \text{an} \), \( x^{(2)} = \text{arrow} \)

Sample 3:
- \( x^{(3)} = \text{flies} \), \( y^{(3)} = \text{fly} \), \( x^{(3)} = \text{with} \), \( y^{(3)} = \text{their} \), \( x^{(3)} = \text{wings} \)

Sample 4:
- \( x^{(4)} = \text{with} \), \( y^{(4)} = \text{time} \), \( x^{(4)} = \text{you} \), \( y^{(4)} = \text{will} \), \( x^{(4)} = \text{see} \)
Inference for HMMs

Whiteboard

– Forward-backward search space
A Hidden Markov Model (HMM) provides a joint distribution over the sentence/tags with an assumption of dependence between adjacent tags.

\[ p(\text{n}, \text{v}, \text{p}, \text{d}, \text{n}, \text{time}, \text{flies}, \text{like}, \text{an}, \text{arrow}) = (0.3 * 0.8 * 0.2 * 0.5 * ...) \]
Forward-Backward Algorithm

$Y_1 \rightarrow Y_2 \rightarrow Y_3$

$X_1$: find
$X_2$: preferred
$X_3$: tags

Could be verb or noun
Could be adjective or verb
Could be noun or verb
Forward-Backward Algorithm

$Y_1 \xrightarrow{} X_1 \text{ find}$

$Y_2 \xrightarrow{} X_2 \text{ preferred}$

$Y_3 \xrightarrow{} X_3 \text{ tags}$
Forward-Backward Algorithm

• Let’s show the possible values for each variable
Forward-Backward Algorithm

- Let’s show the possible values for each variable
Forward-Backward Algorithm

- Let’s show the possible *values* for each variable
- One possible assignment
Let’s show the possible *values* for each variable
One possible assignment
And what the 7 transition / emission factors think of it ...
Let’s show the possible values for each variable
One possible assignment
And what the 7 transition / emission factors think of it ...
Viterbi Algorithm: Most Probable Assignment

- So \( p(v \ a \ n) = (1/Z) \times \text{product of 7 numbers} \)
- Numbers associated with edges and nodes of path
- Most probable assignment = path with highest product
Viterbi Algorithm: Most Probable Assignment

- So $p(v \ a \ n) = (1/Z) \times \text{product weight of one path}$
Forward-Backward Algorithm: Finds Marginals

- So $p(\mathbf{v} \ a \ \mathbf{n}) = (1/Z) \ \text{product weight of one path}$
- Marginal probability $p(Y_2 = a) = (1/Z) \ \text{total weight of all paths through}$

\[
\begin{align*}
S_{v \ a \ n} &= \text{product weight of one path} \\
S_{Y_2} &= \text{total weight of all paths through } a
\end{align*}
\]
Forward-Backward Algorithm: Finds Marginals

- So $p(v \ a \ n) = (1/Z) \times \text{product weight of one path}$
- Marginal probability $p(Y_2 = n) = (1/Z) \times \text{total weight of all paths through } n$
Forward-Backward Algorithm: Finds Marginals

- So $p(v \, a \, n) = (1/Z) \times \text{product weight of one path}$
- Marginal probability $p(Y_2 = v) = (1/Z) \times \text{total weight of all paths through v}$
Forward-Backward Algorithm: Finds Marginals

- So $p(v \ a \ n) = (1/Z) \times \text{product weight of one path}$
- Marginal probability $p(Y_2 = n) = (1/Z) \times \text{total weight of all paths through } \triangle n$
Forward-Backward Algorithm: Finds Marginals

\[ \alpha_2(n) = \text{total weight of these path prefixes } = a+b+c \]

(found by dynamic programming: matrix-vector products)
Forward-Backward Algorithm: Finds Marginals

\[ \beta_2(n) = \text{total weight of these path suffixes} = x + y + z \]

(found by dynamic programming: matrix-vector products)
Forward-Backward Algorithm: Finds Marginals

\[ \alpha_2(n) = \text{total weight of these path prefixes } (a + b + c) \]

\[ \beta_2(n) = \text{total weight of these path suffixes } (x + y + z) \]

Product gives:

\[ ax + ay + az + bx + by + bz + cx + cy + cz = \text{total weight of paths} \]
Forward-Backward Algorithm: Finds Marginals

Oops! The weight of a path through a state also includes a weight at that state. So $\alpha(n) \cdot \beta(n)$ isn’t enough.

The extra weight is the opinion of the emission probability at this variable.

“belief that $Y_2 = n$”

The total weight of all paths through

$$= \alpha_2(n) \cdot A(\text{pref.}, n) \cdot \beta_2(n)$$
Forward-Backward Algorithm: Finds Marginals

Total weight of all paths through

\[
\text{total weight of } all \text{ paths through } v = \alpha_2(v) \cdot A(\text{pref.}, v) \cdot \beta_2(v)
\]

"belief that \( Y_2 = v \)"

"belief that \( Y_2 = n \)"
Forward-Backward Algorithm: Finds Marginals

Total weight of all paths through

\[ \alpha_2(a) \cdot A(\text{pref.}, a) \cdot \beta_2(a) \]

“belief that \( Y_2 = v \)”

“belief that \( Y_2 = n \)”

“belief that \( Y_2 = a \)”

Sum = \( Z \) (total weight of all paths)

\[ p(\bar{x}) = \frac{\sum p(x, \bar{y})}{Z} \]
Forward-Backward Algorithm

Could be verb or noun

Could be adjective or verb

Could be noun or verb
Inference for HMMs

Whiteboard

– Derivation of Forward algorithm
– Forward-backward algorithm
– Viterbi algorithm
Forward-Backward Algorithm

Define:

\[
\alpha_t(k) = p(x_1, \ldots, x_t, y_t = k) \\
\beta_t(k) = p(x_{t+1}, \ldots, x_T | y_t = k)
\]

Assume:

\[
y_0 = \text{START} \\
y_{T+1} = \text{END}
\]

1. Initialize:

\[
\begin{align*}
\alpha_0(\text{START}) &= 1 \\
\alpha_0(k) &= 0 \quad \forall k \neq \text{START} \\
\beta_{T+1}(\text{END}) &= 1 \\
\beta_{T+1}(k) &= 0 \quad \forall k \neq \text{END}
\end{align*}
\]

2. For \( t = 1, \ldots, T \):

   For \( k = 1, \ldots, K \):

   \[
   \alpha_t(k) = p(x_t | y_t = k) \prod_{j=1}^{k} \alpha_{t-1}(j) p(y_t = k | y_{t-1} = j)
   \]

3. For \( t = T, \ldots, 1 \):

   For \( k = 1, \ldots, K \):

   \[
   \beta_t(k) = \sum_{j=1}^{K} p(x_{t+1} | y_{t+1} = j) \beta_{t+1}(j) p(y_{t+1} = j | y_t = k)
   \]

4. Compute:

   \[
   p(\tilde{x}) = \alpha_{T+1}(\text{END})
   \]

   [Evaluation]

5. Compute:

   \[
   p(y_t = k | \tilde{x}) = \frac{\alpha_t(k) \beta_t(k)}{p(\tilde{x})}
   \]

   [Marginals]
Derivation of Forward Algorithm

Definition: \( \alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k) \)

Derivation:

\[
\alpha_T(END) = p(x_1, \ldots, x_T, y_T = END) \\
= p(x_1, \ldots, x_T, y_T) p(y_T) \quad \downarrow \quad \text{by def of joint} \\
= p(x_T | y_T) p(x_1, \ldots, x_{T-1}, y_T) p(y_T) \quad \downarrow \quad \text{by cond. indep. of HMM} \\
= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}, y_T) p(y_{T-1}) \quad \downarrow \quad \text{by def. of marginal} \\
= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_T | y_{T-1}) p(y_{T-1}) \quad \downarrow \quad \text{by def. of joint} \\
= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}) p(y_T | y_{T-1}) p(y_{T-1}) \quad \downarrow \quad \text{by cond. indep. of HMM} \\
= p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}) \alpha_{T-1}(y_{T-1}) p(y_T | y_{T-1}) \quad \downarrow \quad \text{by def. of joint} \\
= p(x_T | y_T) \sum_{y_{T-1}} \alpha_{T-1}(y_{T-1}) p(y_T | y_{T-1}) \quad \downarrow \quad \text{by def. of } \alpha_t(k)
\]
Viterbi Algorithm

\[
\text{Define:} \quad \omega_t(k) = \max_{y_1, \ldots, y_{t-1}} p(x_1, \ldots, x_t, y_1, \ldots, y_{t-1}, y_t = k) \\
\text{"backpoints"} \quad b_t(k) = \arg\max_{y_1, \ldots, y_{t-1}} p(x_1, \ldots, x_t, y_1, \ldots, y_{t-1}, y_t = k)
\]

\text{Assume} \quad y_0 = \text{START}

\text{1. Initialize} \quad \omega_0(\text{START}) = 1, \quad \omega_0(k) = 0 \quad \forall k \neq \text{START}

\text{2. For } t = 1, \ldots, T:

\text{For } k = 1, \ldots, K:

\omega_t(k) = \max_{j \in \{1, \ldots, K\}} p(x_t | y_t = k) \omega_{t-1}(j) p(y_t = k | y_{t-1} = j)

\text{Compute Most Probable Assignment} \quad [\text{Decoding}]

\hat{y}_T = b_{T+1}(\text{END})

\text{For } t = T-1, \ldots, 1

\hat{y}_t = b_{t+1}(\hat{y}_{t+1}) \quad \text{follow the "backpoints"}
Inference in HMMs

What is the computational complexity of inference for HMMs?

- The naïve (brute force) computations for Evaluation, Decoding, and Marginals take exponential time, $O(K^T)$

- The forward-backward algorithm and Viterbi algorithm run in polynomial time, $O(T*K^2)$
  - Thanks to dynamic programming!
Shortcomings of Hidden Markov Models

• HMM models capture dependences between each state and only its corresponding observation
  – NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line length, indentation, amount of white space, etc.

• Mismatch between learning objective function and prediction objective function
  – HMM learns a joint distribution of states and observations $P(Y, X)$, but in a prediction task, we need the conditional probability $P(Y|X)$
MBR DECODING
Inference for HMMs

– Three Inference Problems for an HMM

1. Evaluation: Compute the probability of a given sequence of observations
2. Viterbi Decoding: Find the most-likely sequence of hidden states, given a sequence of observations
3. Marginals: Compute the marginal distribution for a hidden state, given a sequence of observations
4. MBR Decoding: Find the lowest loss sequence of hidden states, given a sequence of observations (Viterbi decoding is a special case)
Minimum Bayes Risk Decoding

• Suppose we are given a loss function \( l(y', y) \) and are asked for a single tagging
• How should we choose just one from our probability distribution \( p(y|x) \)?
• A minimum Bayes risk (MBR) decoder \( h(x) \) returns the variable assignment with minimum expected loss under the model’s distribution

\[
h_\theta(x) = \arg\min_{\hat{y}} \mathbb{E}_{y \sim p_\theta(\cdot|x)}[l(\hat{y}, y)]
\]

\[
= \arg\min_{\hat{y}} \sum_{y} p_\theta(y \mid x) l(\hat{y}, y)
\]
Minimum Bayes Risk Decoding

\[ h_\theta(x) = \arg\min_{\hat{y}} \mathbb{E}_{y \sim p_\theta(. | x)}[\ell(\hat{y}, y)] \]

Consider some example loss functions:

The **0-1 loss function** returns 1 only if the two assignments are identical and 0 otherwise:

\[ \ell(\hat{y}, y) = 1 - \mathbb{I}(\hat{y}, y) \]

The MBR decoder is:

\[ h_\theta(x) = \arg\min_{\hat{y}} \sum_y p_\theta(y | x)(1 - \mathbb{I}(\hat{y}, y)) \]

\[ = \arg\max_{\hat{y}} p_\theta(\hat{y} | x) \]

which is exactly the Viterbi decoding problem!
Minimum Bayes Risk Decoding

\[ h_\theta(x) = \arg\min_{\hat{y}} \mathbb{E}_{y \sim p_\theta} \left[ \ell(\hat{y}, y) \right] \]

Consider some example loss functions:

The **Hamming loss** corresponds to accuracy and returns the number of incorrect variable assignments:

\[ \ell(\hat{y}, y) = \sum_{i=1}^{V} (1 - \mathbb{I}(\hat{y}_i, y_i)) \]

The MBR decoder is:

\[ \hat{y}_i = h_\theta(x)_i = \arg\max_{\hat{y}_i} p_\theta(\hat{y}_i \mid x) \]

This decomposes across variables and requires the variable marginals.
Learning Objectives

Hidden Markov Models

You should be able to...
1. Show that structured prediction problems yield high-computation inference problems
2. Define the first order Markov assumption
3. Draw a Finite State Machine depicting a first order Markov assumption
4. Derive the MLE parameters of an HMM
5. Define the three key problems for an HMM: evaluation, decoding, and marginal computation
6. Derive a dynamic programming algorithm for computing the marginal probabilities of an HMM
7. Interpret the forward-backward algorithm as a message passing algorithm
8. Implement supervised learning for an HMM
9. Implement the forward-backward algorithm for an HMM
10. Implement the Viterbi algorithm for an HMM
11. Implement a minimum Bayes risk decoder with Hamming loss for an HMM