HMMs
+
Bayesian Networks
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
THE FORWARD-BACKWARD ALGORITHM
Forward-Backward Algorithm

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

Assume \( y_0 = \text{START} \)
\( y_{T+1} = \text{END} \)

1. Initialize \( \alpha_0(\text{START}) = 1 \)
\( \alpha_0(k) = 0 \quad \forall k \neq \text{START} \)
\( \beta_{T+1}(\text{END}) = 1 \)
\( \beta_{T+1}(k) = 0 \quad \forall k \neq \text{END} \)

2. For \( t = 1, \ldots, T : \)
   For \( k = 1, \ldots, K : \)
   \[ \alpha_t(k) = p(x_t | y_t = k) \sum_{j=1}^{K} \alpha_{t-1}(j) p(y_t = k | y_{t-1} = j) \]

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

5. Compute \( p(y_t = k | \tilde{x}) = \frac{\alpha_t(k) \beta_t(k)}{p(\tilde{x})} \)

Brute force algorithm would be \( O(K^T) \)

the alphas include the emission probabilities so we don’t multiply them in separately
Inference for HMMs

Whiteboard

– Forward-backward algorithm (edge weights version)
– Viterbi algorithm (edge weights version)
Forward-Backward Algorithm

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

Assume \( y_0 = \text{START} \)
\( y_{T+1} = \text{END} \)

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

1. Initialize \( \alpha_0(\text{START}) = 1 \) \( \alpha_0(k) = 0 \quad \forall k \neq \text{START} \)
\( \beta_{T+1}(\text{END}) = 1 \)
\( \beta_{T+1}(k) = 0 \quad \forall k \neq \text{END} \)

2. For \( t = 1, \ldots, T \):
   - For \( k = 1, \ldots, K \):
     \[ \alpha_t(k) = p(x_t | y_t = k) \sum_{j=1}^{K} \alpha_{t-1}(j) p(y_t = k | y_{t-1} = j) \]

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

5. Compute \( p(y_t = k | \hat{x}) = \frac{\alpha_t(k) \beta_t(k)}{p(\hat{x})} \) [Marginals]

Brute force algorithm would be \( O(K^T) \)
Derivation of Forward Algorithm

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

Derivation:

\[ \alpha_T(\text{END}) = p(x_1, \ldots, x_T, y_T = \text{END}) \]
\[ = p(x_T | y_T) p(x_1, \ldots, x_{T-1} | y_T) p(y_T) \]
\[ = p(x_T | y_T) p(x_1, \ldots, x_{T-1}, y_T) \]
\[ = p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}, y_T) \]
\[ = p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1} | y_T) p(y_{T-1}) \]
\[ = p(x_T | y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}) y_{T-1} p(y_{T-1} | y_{T-1}) p(y_{T-1}) \]
\[ = p(x_T | y_T) \sum_{y_{T-1}} \alpha_{T-1}(y_{T-1}) p(y_T | y_{T-1}) \]

Herein using "y_T" as shorthand for "y_T = END".
Viterbi Algorithm

Define: \( \omega_t(k) \triangleq \max_{y_1, \ldots, y_{t-1}} p(x_1, \ldots, x_t, y_1, \ldots, y_{t-1}, y_t = k) \)

"backpoints": \( b_t(k) \triangleq \arg\max_{y_1, \ldots, y_{t-1}} p(x_1, \ldots, x_t, y_1, \ldots, y_{t-1}, y_t = k) \)

Assume \( y_0 = \text{START} \)

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

2. For \( t = 1, \ldots, T \):
   For \( k = 1, \ldots, K \):
   \[ \omega_t(k) = \max_{j \in \{1, \ldots, K\}} \omega_{t-1}(j) p(y_t = k | y_{t-1} = j) \]
   \[ b_t(k) = \arg\max_{j \in \{1, \ldots, K\}} p(x_t | y_t = k) \omega_{t-1}(j) p(y_t = k | y_{t-1} = j) \]

3. Compute Most Probable Assignment
   \( \hat{y}_T = b_{T+1}(\text{END}) \)
   For \( t = T-1, \ldots, 1 \)
   \( \hat{y}_t = b_{t+1}(\hat{y}_{t+1}) \)
   follow the "backpoints"

[Decoding]
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 \( P(\mathbf{Y}, \mathbf{X}) \), but in a prediction task, we need the conditional probability \( P(\mathbf{Y}|\mathbf{X}) \).
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.

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

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

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

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:

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

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

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

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

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:

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

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
Bayes Nets Outline

• **Motivation**
  – Structured Prediction

• **Background**
  – Conditional Independence
  – Chain Rule of Probability

• **Directed Graphical Models**
  – Writing Joint Distributions
  – Definition: Bayesian Network
  – Qualitative Specification
  – Quantitative Specification
  – Familiar Models as Bayes Nets

• **Conditional Independence in Bayes Nets**
  – Three case studies
  – D-separation
  – Markov blanket

• **Learning**
  – Fully Observed Bayes Net
  – (Partially Observed Bayes Net)

• **Inference**
  – Background: Marginal Probability
  – Sampling directly from the joint distribution
  – Gibbs Sampling
Bayesian Networks

DIRECTED GRAPHICAL MODELS
Example: Ryan Reynolds’ Voicemail

From https://www.adweek.com/brand-marketing/ryan-reynolds-left-voicemails-for-all-mint-mobile-subscribers/
Example: Ryan Reynolds Voicemail

Images from imdb.com
Example: Ryan Reynolds’ Voicemail

From https://www.adweek.com/brand-marketing/ryan-reynolds-left-voicemails-for-all-mint-mobile-subscribers/
Directed Graphical Models (Bayes Nets)

Whiteboard

– Example: Ryan Reynolds’ Voicemail
– Writing Joint Distributions
  • Idea #1: Giant Table
  • Idea #2: Rewrite using chain rule
  • Idea #3: Assume full independence
  • Idea #4: Drop variables from RHS of conditionals
– Definition: Bayesian Network
Bayesian Network

\[ p(X_1, X_2, X_3, X_4, X_5) = p(X_5 | X_3)p(X_4 | X_2, X_3)p(X_3)p(X_2 | X_1)p(X_1) \]
Bayesian Network

Definition:

A Bayesian Network is a **directed graphical model**
It consists of a graph $G$ and the conditional probabilities $P$
These two parts full specify the distribution:
  - Qualitative Specification: $G$
  - Quantitative Specification: $P$

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i))$$
Qualitative Specification

• Where does the qualitative specification come from?
  – Prior knowledge of causal relationships
  – Prior knowledge of modular relationships
  – Assessment from experts
  – Learning from data (i.e. structure learning)
  – We simply prefer a certain architecture (e.g. a layered graph)
  – ...

© Eric Xing @ CMU, 2006-2011
Quantitative Specification

Example: Conditional probability tables (CPTs) for discrete random variables

\[ P(a, b, c, d) = P(a)P(b)P(c|a, b)P(d|c) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a₀</td>
<td>0.75</td>
</tr>
<tr>
<td>B</td>
<td>a₁</td>
<td>0.25</td>
</tr>
<tr>
<td>B</td>
<td>b₀</td>
<td>0.33</td>
</tr>
<tr>
<td>B</td>
<td>b₁</td>
<td>0.67</td>
</tr>
<tr>
<td>C</td>
<td>c₀</td>
<td>0.45</td>
</tr>
<tr>
<td>C</td>
<td>c₁</td>
<td>0.55</td>
</tr>
<tr>
<td>D</td>
<td>d₀</td>
<td>0.30</td>
</tr>
<tr>
<td>D</td>
<td>d₁</td>
<td>0.70</td>
</tr>
</tbody>
</table>

© Eric Xing @ CMU, 2006-2011
Example: Conditional probability density functions (CPDs) for continuous random variables

\[ A \sim N(\mu_a, \Sigma_a) \quad B \sim N(\mu_b, \Sigma_b) \]

\[ C \sim N(A+B, \Sigma_c) \]

\[ D \sim N(\mu_d + C, \Sigma_d) \]

\[ P(a,b,c,d) = P(a)P(b)P(c|a,b)P(d|c) \]
Quantitative Specification

Example: Combination of CPTs and CPDs for a mix of discrete and continuous variables

\[ P(a,b,c,d) = P(a)P(b)P(c|a,b)P(d|c) \]

\[ C \sim N(A + B, \Sigma_c) \]

\[ D \sim N(\mu_d + C, \Sigma_d) \]
Observed Variables

• In a graphical model, **shaded nodes** are “**observed**”, i.e. their values are given

Example:

\[ P(X_2, X_5 \mid X_1 = 0, X_3 = 1, X_4 = 1) \]
Familiar Models as Bayesian Networks

Question:
Match the model name to the corresponding Bayesian Network
1. Logistic Regression
2. Linear Regression
3. Bernoulli Naïve Bayes
4. Gaussian Naïve Bayes
5. 1D Gaussian

Answer:

1. Logistic Regression
2. Linear Regression
3. Bernoulli Naïve Bayes
4. Gaussian Naïve Bayes
5. 1D Gaussian

\[
p(y|x_1, \ldots, x_m) = p(y)p(x_1|y) \cdots p(x_m|y)
\]