 probabilistic models for structured data

03: Hidden Markov Models

Instructor: Yizhou Sun

yzsun@cs.ucla.edu

January 17, 2019
• Preliminary: Markov Chains
• The Hidden Markov Model
• Inference
  • Likelihood Computation: The Forward Algorithm
  • Decoding: The Viterbi Algorithm
• Learning
  • The Forward-Backward Algorithm
• Summary
From I.I.D. Data to Sequence

- Dependency exists among data points, which form a sequence
- Examples
  - Speech recognition
  - Handwriting recognition

\[ \begin{array}{c}
Q \quad V \quad E \quad S \quad T \\
\text{x1} \quad \text{Is it a V or a U?} \quad \text{x5}
\end{array} \]

Note: this is a data model, not a graphical model

- Named-entity recognition (NER)
(Discrete) Markov Chain

• Sequence
  • Ordered elements or events
    • \(< x_1, x_2, \ldots, x_T >\)
  • Examples:
    • A word, where each element is a letter
    • A sequence of text, where each element is a word

• Markov chain (also called observed Markov chain)
  • A probabilistic graphical model
  • Model how an observed sequence is generated
Definition of a First-Order Markov Chain

- **N States:** $S = \{S_1, S_2, \ldots, S_N\}$
  - Sometimes, a start state $S_0$ and an end state $S_F$ are included
  - **Examples:**
    - For a word sequence, 26 states are 26 letters
    - For a text sequence, $|V|$ states are $|V|$ words in the dictionary

- **Transition probability matrix:** $A = \{a_{ij}\}_{i,j=1}^N$
  - First order: $P(x_t = S_j|x_{t-1} = S_i, x_{t-2} = S_k, \ldots) = P(x_t = S_j|x_{t-1} = S_i)$ (Markov Assumption)
  - $a_{ij} = P(x_t = S_j|x_{t-1} = S_i)$
    - $a_{ij} \geq 0$ and $\sum_j a_{ij} = 1$

- **Initial distribution:** $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$
  - $\pi_i = P(x_1 = S_i)$, the probability that the Markov chain starts with state $S_i$
    - $\pi_i \geq 0$ and $\sum_i \pi_i = 1$
Generation of a Sequence

• To generate an observed sequence: \( x = (x_1x_2 \ldots x_T) \)
  
  • For \( t = 1 \), sample \( x_1 \sim \pi \)
  
  • For \( t = 2:T \)
    
    • Sample a new state \( x_t | x_{t-1} \sim a_{x_{t-1}} \).
The Probability of a Sequence

• Under first-order Markov chain model

\[ P(x_1, x_2, \ldots, x_T) \]
\[ = P(x_1)P(x_2|x_1) \ldots P(x_T|x_{T-1}, x_{T-2}, \ldots, x_1) \]
\[ = P(x_1) \prod_{t=2}^{T} P(x_t|x_{t-1}) = \pi_{x_1} \prod_{t=2}^{T} a_{x_{t-1}x_t} \]
The Weather Example

• Three states: Sunny, Rainy, and Foggy
• Transition probability matrix

<table>
<thead>
<tr>
<th>Today’s Weather</th>
<th>Tomorrow’s Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sunny</td>
</tr>
<tr>
<td>Sunny</td>
<td>0.8</td>
</tr>
<tr>
<td>Rainy</td>
<td>0.2</td>
</tr>
<tr>
<td>Foggy</td>
<td>0.2</td>
</tr>
</tbody>
</table>

represented as a table

represented as a graph
The Weather Example (Continued)

- Question 1: What’s the probability that tomorrow is sunny and the day after is rainy, given today is sunny?

\[
P(x_2 = Sunny, x_3 = Rainy|x_1 = Sunny) = \\
P(x_2 = Sunny|x_1 = Sunny) \times \\
P(x_3 = Rainy|x_2 = Sunny) = 0.8 \times 0.05 = 0.04
\]

- Question 2: What’s the probability it will be rainy two days from now, given today is foggy?

\[
P(x_3 = Rainy|x_1 = Foggy) = \\
\sum_i P(x_3 = Rainy, x_2 = S_i|x_1 = Foggy) = \\
P(x_3 = Rainy, x_2 = Sunny|x_1 = Foggy) + \\
P(x_3 = Rainy, x_2 = Rainy|x_1 = Foggy) + \\
P(x_3 = Rainy, x_2 = Foggy|x_1 = Foggy) = 0.34
\]
• Preliminary: Markov Chains

• The Hidden Markov Model

• Inference
  • Likelihood Computation: The Forward Algorithm
  • Decoding: The Viterbi Algorithm

• Learning
  • The Forward-Backward Algorithm

• Summary
Hidden Markov Model

- States are not observable, observations are generated from a hidden state
  - Hidden state sequence
    - $< y_1, y_2, \ldots, y_T >$
  - Observation sequence
    - $< x_1, x_2, \ldots, x_T >$

- Examples
  - Speech recognition
  - Handwriting recognition

- Named-entity recognition (NER)
Definition of a Discrete Hidden Markov Model

- **N** States: \( S = \{S_1, S_2, ..., S_N\} \)
- **M** observation symbols: \( V = \{v_1, v_2, ..., v_M\} \)
- Transition probability matrix: \( A = \{a_{ij}\}_{i,j=1}^{N} \)
  - \( a_{ij} = P(y_t = S_j | y_{t-1} = S_i) \)
  - \( a_{ij} \geq 0 \) and \( \sum_j a_{ij} = 1 \)
- Observation symbol probability distribution: \( B = \{b_{ik}\}, 1 \leq i \leq N, 1 \leq k \leq M \)
  - \( b_{ik} = P(x_t = v_k | y_t = S_i) \)
  - \( b_{ik} \geq 0 \) and \( \sum_k b_{ik} = 1 \)
- Initial distribution: \( \pi = \{\pi_1, \pi_2, ..., \pi_N\} \)
  - \( \pi_i = P(y_1 = S_i) \)
  - \( \pi_i \geq 0 \) and \( \sum_i \pi_i = 1 \)

Can be extended to numerical values

Multinoulli: Can be extended to other probabilistic distribution
To generate an observed sequence: \( x = (x_1 x_2 \ldots x_T) \)

- For \( t = 1 \), sample \( y_1 \sim \pi \), sample \( x_1 | y_1 \sim b_{y_1} \).

- For \( t = 2: T \)
  - Sample a new state \( y_t | y_{t-1} \sim a_{y_{t-1}} \).
  - Sample an observation \( x_t | y_t \sim b_{y_t} \).
Three Basic Questions

- **[Likelihood Computation]** How likely is a given sequence?
  - The Forward algorithm

- **[Decoding]** What is the most probable “path” for generating a given sequence?
  - The Viterbi algorithm

- **[Learning]** How can we learn the HMM parameters given a set of sequences?
  - The Forward-Backward (Baum-Welch) algorithm
Content

• Preliminary: Markov Chains

• The Hidden Markov Model

• Inference
  • Likelihood Computation: The Forward Algorithm
  • Decoding: The Viterbi Algorithm

• Learning
  • The Forward-Backward Algorithm

• Summary
Likelihood Computation

Given an HMM $\lambda = (A, B, \pi)$ and an observation sequence $x = (x_1 x_2 \ldots x_T)$, determine the likelihood $P(x|\lambda)$

- $P(x|\lambda) = \sum_y P(x, y|\lambda) = \sum_y P(x|y, \lambda) P(y|\lambda)$
  - $P(x|y, \lambda)$: the conditional probability of observation sequence when a state sequence known
    - $P(x|y, \lambda) = \prod_t P(x_t|y_t, \lambda) = \prod_t b_{y_t x_t}$
  - $P(y|\lambda)$: the probability of a state sequence $y$
    - $P(y|\lambda) = P(y_1) \prod_{t=2} P(y_t|y_{t-1}) = \pi_{y_1} \prod_{t=2} a_{y_{t-1} y_t}$
Challenge of Brute-force Computation

- \( P(x|\lambda) = \sum_y P(x, y|\lambda) = \sum_y P(x|y, \lambda) P(y|\lambda) \)
  - Sum over all the possible state sequences
  - How many of them?
    - \( N^T \) (why?)
The Forward Algorithm

- A dynamic programming algorithm
  - Use table to store intermediate results
- Define forward variable $\alpha_t(j)$ as the probability of seeing the first $t$ observations and the $t$-th state is $j$
  - $\alpha_t(j) = P(x_1, x_2, ..., x_t, y_t = j | \lambda)$
  - $= \sum_{y_1, y_2, ..., y_{t-1}} P(y_1, y_2, ..., y_{t-1}, y_t = j, x_1, x_2, ..., x_t | \lambda)$
- $\alpha_t(j)$ can be recursively defined as
  - $\alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} b_j x_t$ (when $1 < t \leq T$)

$P(x_1, x_2, ..., x_{t-1}, y_{t-1} = i)$  $P(y_t = j | y_{t-1} = i)$  $P(x_t | y_t = j)$
**Major Steps**

1. Initialization (*when* $t = 1$):
   - $\alpha_1(j) = \pi_j b_{jx_1}$, *for* $1 \leq j \leq N$

2. Recursion (*when* $1 < t \leq T$):
   - $\alpha_t(j) = \sum_i \alpha_{t-1}(i)a_{ij}b_{jx_t}$, *for* $1 \leq j \leq N$

3. Termination:
   - $P(x|\lambda) = \sum_j \alpha_T(j)$

- Time complexity
  - $O(N^2T)$
• Operations for computing $\alpha_{t+1}(j)$ from $\alpha_t(j)$

• Computing $\alpha_t(j)$ as a lattice
The Decoding Problem

- Given an HMM $\lambda = (A, B, \pi)$ and an observation sequence $x = (x_1 x_2 \ldots x_T)$, find the most probable sequence of states $y = (y_1 y_2 \ldots y_T)$
  - $y = \underset{y}{\text{argmax}} P(y|x, \lambda) = \underset{y}{\text{argmax}} P(x, y|\lambda)$
- Brute-force computation
  - Enumerate all the possible state sequences, and pick the one with the maximum likelihood
  - Challenge: the same as before, $N^T$ possible sequences!
The Viterbi Algorithm

- Also a dynamic programming algorithm
- Define \( \nu_t(j) \) as the probability of the most probable path accounting for the first \( t \) observations and ending in state \( j \)
  - \( \nu_t(j) = \max_i \nu_{t-1}(i) a_{ij} b_j x_t \) (1 < \( t \) ≤ \( T \))
- Backtracking the best path
  - Keep the maximizing argument for each \( t \) and \( j \)

A special case of max product

Almost identical to the forward algorithm, except replacing sum with max
Major Steps

1. Initialization (when $t = 1$):
   - $v_1(j) = \pi_j b_{jx_1}$, for $1 \leq j \leq N$
2. Recursion (when $1 < t \leq T$):
   - $v_t(j) = \max_i v_{t-1}(i)a_{ij}b_{jx_t}$, for $1 \leq j \leq N$
   - $\text{ptr}_t(j) = \text{argmax}_i v_{t-1}(i)a_{ij}b_{jx_t}$, for $1 \leq j \leq N$
   - Termination:
     - $P^* = \max_P(y|x, \lambda) = \max_j v_T(j)$
     - Backtracking from $\text{argmax}_j v_T(j)$
   - Time complexity
     - $O(N^2T)$
Preliminary: Markov Chains

The Hidden Markov Model

Inference
- Likelihood Computation: The Forward Algorithm
- Decoding: The Viterbi Algorithm

Learning
- The Forward-Backward Algorithm

Summary
Learning the Parameters of an HMM

• Given an observation sequence $x$ and the set of possible states in the HMM, learn the HMM parameters $\lambda = (A, B, \pi)$
  • Find $\lambda = (A, B, \pi)$ that locally maximizes $P(x|\lambda)$ (Maximum Likelihood Estimation, MLE)
    • Gradient techniques
    • Forward-backward algorithm (Baum-Welch algorithm)
      • A special case of EM (Expectation-Maximization) algorithm
How to Estimate Parameters for an Observed Markov Chain Model?

• MLE estimation
  
  • Find $\lambda = (A, \pi)$ that locally maximizes $P(y|\lambda)$
    
    • $L(\lambda; y) = P(y|\lambda) = \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1}y_t}$
  
  • Constraints:
    
    • $\pi_i \geq 0$ and $\sum_i \pi_i = 1$
    
    • $a_{ij} \geq 0$ and $\sum_j a_{ij} = 1$
  
  • Lagrange multiplier method
    
    • $\hat{\pi}_i = 1$ if $y_1 = i$
    
    • $\hat{a}_{ij} = \frac{C(i\rightarrow j)}{\sum_{k\in S} C(i\rightarrow k)}$, where $C(i \rightarrow j)$ is the number of times state $i$ transits to state $j$
How to Estimate Parameters $B$ If Both $y$ and $x$ are observed?

• MLE estimation
  • Find $B$ that maximizes $P(x, y|B)$
  • Equivalently find $B$ that maximizes $P(x|y, B)$
Backward Procedure

• Define backward probability $\beta_t(i)$ as the probability of seeing the observations from time $t + 1$ to the end, given state at time $t$ is $i$
  
  $\beta_t(i) = P(x_{t+1}, x_{t+2}, \ldots, x_T | y_t = i, \lambda)$

• $\beta_t(i)$ can be recursively defined as
  
  $\beta_t(i) = \sum_j a_{ij} b_{jx_{t+1}} \beta_{t+1}(j)$

  $P(x_{t+1}, x_{t+2}, \ldots, x_T | y_t = i, \lambda) =$
  
  $\sum_j P(x_{t+1}, x_{t+2}, \ldots, x_T, y_{t+1} = j | y_t = i, \lambda) =$
  
  $\sum_j P(x_{t+1}, x_{t+2}, \ldots, x_T | y_t = i, y_{t+1} = j, \lambda) P(y_{t+1} = j | y_t = i, \lambda) =$
  
  $= \sum_j P(x_{t+1} | y_{t+1} = j, \lambda) P(x_{t+2}, \ldots, x_T | x_{t+1}, y_{t+1} = j, \lambda) P(y_{t+1} = j | y_t = i, \lambda)$
Major Steps

1. Initialization \((\text{when } t = T)\):
   \(\beta_T(j) = 1, \text{ for } 1 \leq j \leq N\)

2. Recursion \((\text{when } 1 < t \leq T)\):
   \(\beta_t(i) = \sum_j a_{ij} b_{tx_{t+1}} \beta_{t+1}(j), \text{ for } 1 \leq j \leq N\)

3. Termination:
   \(P(x|\lambda) = \sum_j \pi_j b_{jx_1} \beta_1(j)\)

   - Time complexity
     \(O(N^2T)\)
The Forward-Backward Algorithm

- Also called Baum-Welch algorithm
- A special case of EM algorithm
- **Repeat until converge**
  - **E-step:**
    - Expected state occupancy count \( \gamma_t(j) = P(y_t = j|x, \lambda) \)
      - Probability of being in state \( j \) at time \( t \)
    - Expected state transition count \( \xi_t(i, j) = P(y_t = i, y_{t+1} = j|x, \lambda) \)
      - Probability of being in state \( i \) at time \( t \) and in state \( j \) at time \( t+1 \)
  - **M-step:**
    - Estimate \( \pi_i, a_{ij}, b_{ik} \)
E-Step: Compute $\gamma_t(j)$

- $\gamma_t(j) = P(y_t = j|x, \lambda) = \frac{P(y_t=j,x|\lambda)}{P(x|\lambda)}$

  - $P(y_t = j, x|\lambda) = P(x_1, ..., x_t, y_t = j|\lambda)P(x_{t+1}, ..., x_T|x_1, ..., x_t, y_t = j, \lambda) = \alpha_t(j)\beta_t(j)$

  - $P(x|\lambda) = \sum_j \alpha_t(j)\beta_t(j)$

- Therefore, $\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_i \alpha_t(i)\beta_t(i)}$
E-Step: Compute $\xi_t(i, j)$

\[ \xi_t(i, j) = P(y_t = i, y_{t+1} = j | x, \lambda) = \frac{P(y_t = i, y_{t+1} = j, x | \lambda)}{P(x | \lambda)} \]

- $P(y_t = i, y_{t+1} = j, x | \lambda) = P(x_1, \ldots, x_T, y_t = i | \lambda)P(y_{t+1} = j | x_{t+1}, \ldots, x_T, y_t = i, \lambda)$
  - $= P(x_1, \ldots, x_t, y_t = i | \lambda)P(y_{t+1} = j | y_t = i, \lambda)P(x_{t+1}, \ldots, x_T | y_{t+1} = j)$
  - $= \alpha_t(i)a_{ij}b_{j|x_{t+1}}\beta_{t+1}(j)$

- $P(x | \lambda) = \sum_i \sum_j \alpha_t(i)a_{ij}b_{j|x_{t+1}}\beta_{t+1}(j)$

Therefore, $\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_{j|x_{t+1}}\beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i)a_{ij}b_{j|x_{t+1}}\beta_{t+1}(j)}$
M-Step

- $\pi_i$
  - $\hat{\pi}_i = \gamma_1(i)$

- $a_{ij}$
  - $\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_k \sum_{t=1}^{T-1} \xi_t(i,k)}$

- $b_{ik}$
  - $\hat{b}_{ik} = \frac{\sum_{x_t=v_k} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$
More on EM Algorithm

- **E-Step:** computing a **tight** lower bound $L$ of the original objective function $l$ at $\theta_{old}$
- **M-Step:** find $\theta_{new}$ to maximize the lower bound
- $l(\theta_{new}) \geq L(\theta_{new}) \geq L(\theta_{old}) = l(\theta_{old})$
How to Find Tight Lower Bound?

\[ \ell(\theta) = \log \sum_h p(d, h; \theta) \]

\[ = \log \sum_h \frac{q(h)}{q(h)} p(d, h; \theta) \]

\[ = \log \sum_h q(h) \frac{p(d, h; \theta)}{q(h)} \]

• Jensen’s inequality

\[ \log \sum_h q(h) \frac{p(d, h; \theta)}{q(h)} \geq \sum_h q(h) \log \frac{p(d, h; \theta)}{q(h)} \]

• When “=” holds to get a tight lower bound?
  • \( q(h) = p(h|d, \theta) \) (why?)

\( q(h) \): a distribution function over \( h \), the key to tight lower bound we want to get
Content

• Preliminary: Markov Chains

• The Hidden Markov Model

• Inference
  • Likelihood Computation: The Forward Algorithm
  • Decoding: The Viterbi Algorithm

• Learning
  • The Forward-Backward Algorithm

• Summary
Summary

• Preliminary: Markov Chains
  • Generative model for observed state sequence

• The Hidden Markov Model
  • Generative model for sequence where states are unseen

• Inference
  • Likelihood Computation: The Forward Algorithm
    • Dynamic programming; Forward variable: $\alpha_t(i)$
  • Decoding: The Viterbi Algorithm
    • $v_t(j)$

• Learning
  • The Forward-Backward Algorithm
    • Backward variable: $\beta_t(i)$
• Daniel Jurafsky & James H. Martin. Speech and Language Processing, Chapter 9. 2017