HMM & Common Mistakes in Midterm

Bin Zhao

Outline

- HMM
  - Big Picture
  - Conditional Independence
  - Inference
  - Learning
  - Application: Named Entity Recognition
- Common Mistakes in Midterm

Big Picture

- Predicting a Single Label
  - Input (**x**): A set of features:
    - Bag of words in a document
  - Output (*y*): Class label
    - Topic of the document
- Predicting Sequence of Labels
  - Input (**x**): A set of features (with order/structure among them)
    - Sequence of words in a sentence
  - Output (**y**)
    - Part of speech (POS) tag of each word

Notation Note:

I use normal face letters for scalar as in y and bold face letters for vectors like **x** and **y**

Big Picture – Single Output

Single Output

- Generative:
  - Models P(**x**,y)
  - Predict using Bayes rule argmax P(y|**x**)
  - Naïve Bayes
- Discriminative:
  - Model P(y|**x**)
  - Predict using argmax P(y|**x**)
  - Logistic Regression

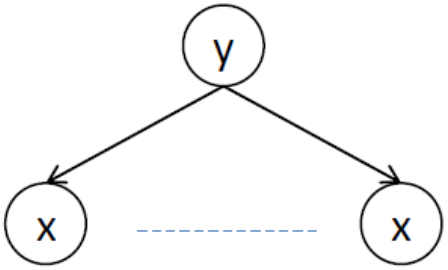
Big Picture – Sequence of Output

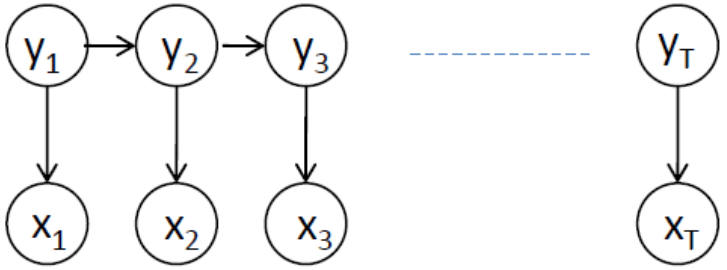
Sequence of Output

- Generative:
  - Models P(**x**,**y**)
  - Predict using Bayes rule $\text{argmax}_{\mathbf{y}}$ P(**y**|**x**)
  - HMM
- Discriminative:
  - Model P(**y**|**x**)
  - Predict using $\text{argmax}_{\mathbf{y}}$ P(**y**|**x**)
  - CRF

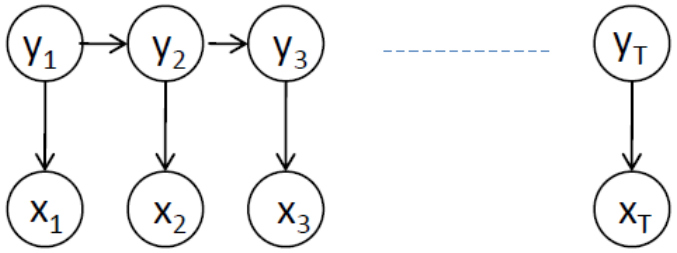
HMM: Motivation

- Defines a generative model over P(**x,y**)
- Each x has M options and each y has K options
- You need a big table of size $M^{|X|}$ $K^{|Y|}$
- We need to add some conditional independence assumption to make things manageable
  - We have done that in Naïve Bayes





HMM: The Model



- What we need to define

	#par.	shorthand
– Initial state: $P(y_1)$	K-1	$\pi_i = P(y_1=i)$
– Transitoin: $P(y_t \mid y_{t-1})$	K*(K-1)	$a_{ij} = P(y_{t+1}=j \mid y_t=i)$
– Emission: $P(x_t \mid y_t)$	K*(M-1)	$b_{ik} = P(x_t=k \mid y_t=i)$

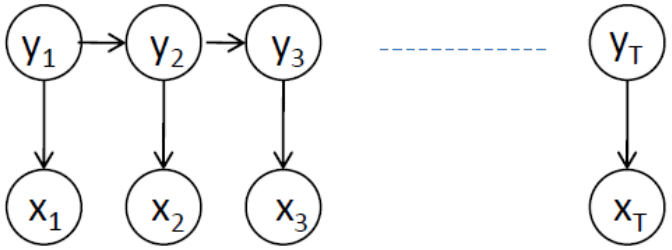
- Factorization:

$$P(x_1,\ldots,x_T,y_1,\ldots,y_T) = P(y_1)p(x_1|y_1)p(y_2|y_1)\ldots P(y_2|y_1)P(x_T|y_T)$$
$$= P(y_1)\prod_t P(y_t|y_{t-1})P(x_t|y_t)$$

Inference & Parameter Learning

- Inference
  - MPA: $P(y_t|\mathbf{x})$
  - Veterbi: $P(\mathbf{y}|\mathbf{X})$
- Learning
  - Learning model parameters using MLE
    - π_i, a_{ij}, b_{ik}
    - Fully Observed:
      - » count and normalize
    - Unsupervised:
      - » EM

Inference: MPA



- Find $\text{argmax}_i\ P(y_t=i|\mathbf{x})$
- We need to compute $P(y_t=i|\mathbf{x})$ first

$$
\begin{aligned}
p(y_t = i|x_1, \ldots, x_T) &= \frac{p(y_t = i, x_1, \ldots, x_T)}{p(x_1, \ldots, x_T)} \\
&= \frac{p(y_t = i, x_1, \ldots, x_t)p(x_{t+1}, \ldots, x_T|y_t = i, x_1, \ldots, x_t)}{p(x_1, \ldots, x_T)} \\
&= \frac{p(y_t = i, x_1, \ldots, x_t)p(x_{t+1}, \ldots, x_T|y_t = i)}{p(x_1, \ldots, x_T)} \\
&= \frac{\alpha_t^i \beta_t^i}{p(x_1, \ldots, x_T)}
\end{aligned}
$$

Inference: Viterbi

- Find the globally maximal posterior sequence
  - argmax$_{y1,..,yT}$ P(y_1,..,y_T | x_1,...x_T)
  - Same as argmax$_{y1,..,yT}$ P(y_1,..,y_T,x_1,...x_T) why?
  - Develop a dynamic (recursive) program
  - max$_{y1..yt-1}$P(y_1,..,y_t,x_1,...x_t) and relate it to
  - max$_{y1..yt-2}$P(y_1,..,y_{t-1},x_1,...x_{t-1})
  - We call this quantity V_t which is a vector:

$$V_t^k = \max_{\{y_1,\ldots y_{t-1}\}} P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t = k)$$

  - It means the maximal prob of ending in state k at time *t* where we are maximizing over $y_1..y_{t-1}$

Inference: Viterbi (Cont.)

- The math

$$V_{t+1}^{k} = \max_{\{y_1,\dots y_t\}} P(x_1,\dots,x_t,y_1,\dots,y_t,x_{t+1},y_{t+1}=k)$$

$$= \max_{\{y_1,\dots y_t\}} P(x_1,\dots,x_t,y_1,\dots,y_t)P(x_{t+1},y_{t+1}=k \mid x_1,\dots,x_t,y_1,\dots,y_t)$$

$$= \max_{\{y_1,\dots y_t\}} P(x_{t+1},y_{t+1}=k \mid y_t)P(x_1,\dots,x_{t-1},y_1,\dots,y_{t-1},x_t,y_t)$$

$$= \max_{i} P(x_{t+1},y_{t+1}=k \mid y_t=i) \max_{\{y_1,\dots y_{t-1}\}} P(x_1,\dots,x_{t-1},y_1,\dots,y_{t-1},x_t,y_t=i)$$

$$= \max_{i} P(x_{t+1}, \mid y_{t+1}=k)a_{i,k}V_t^i$$

$$= P(x_{t+1}, \mid y_{t+1}=k)\max_{i} a_{i,k}V_t^i$$

- Also: keep track of the maximizing *i*

Learning: Observed

- For a give sequence

$$LL(\boldsymbol{\theta}) = \log p(\mathbf{X}, \mathbf{Y})$$

$$= \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t}) \right)$$

$$= \log \prod_n \left(\prod_{i=1}^{K} \pi_i^{C_{i,n}} \prod_{i,j=1}^{K} a_{ij}^{A_{ij,n}} \prod_{i=1,o=1}^{i=K,o=M} b_{io}^{B_{io,n}} \right)$$

$$= \sum_n \left(\sum_{i=1}^{K} C_{i,n} \log \pi_i + \sum_{i,j=1}^{K} A_{ij,n} \log a_{ij} + \sum_{i=1,o=1}^{K,M} B_{io,n} \log b_{io} \right)$$

- $C_{i,n}$: number of times first state was i in $\mathbf{x}_n$ (0 or 1)
- $B_{io,n}$: number of times state i emits *o* in $(\mathbf{x}_n, \mathbf{y}_n)$
- $A_{ij,n}$: number of time state *i* moves to state *j* in $(\mathbf{x}_n, \mathbf{y}_n)$

Learning: Observed (Cont.)

$$LL(\theta) = \sum_n \left(\sum_{i=1}^{K} C_{i,n} \log \pi_i + \sum_{i,j=1}^{K} A_{ij,n} \log a_{ij} + \sum_{i=1,o=1}^{K,M} B_{io,n} \log b_{io} \right)$$

- Note that all parameters are decoupled
- Take gradient and solve for every one separately
- Simply count and normalize, for example:

$$a_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n A_{ij,n}}{\sum_n \sum_{j'} A_{ij',n}}$$

Learning: Observed (Cont.)

- Solution for small training sets:
  - Add pseudocounts

    A_{ij} = # times state transition $i \to j$ occurs in $\mathbf{y}$ + R_{ij}

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$ + S_{ik}

  - R_{ij}, S_{ij} are pseudocounts representing our prior belief
  - Total pseudocounts: $R_i = \Sigma_j R_{ij}$, $S_i = \Sigma_k S_{ik}$,
    - --- "strength" of prior belief,
    - --- total number of imaginary instances in the prior
- Larger total pseudocounts $\Rightarrow$ strong prior belief
- Small total pseudocounts: just to avoid 0 probabilities --- smoothing

Learning: Hidden State

- Initialize HMM model parameters
- Repeat
  - E-Step
    - Run forward-backward over every sequence ($\mathbf{x}_n$)
    - Compute necessary expectations using α and β (or their normalized versions)
  - M–Step
    - Re-estimate model parameters
      - Simply count and normalize

Application: Named entity Recognition

- CoNLL 2002 named entity recognition task
  - 3144 sentences from a Spanish news wire archive
  - 9 label types
    - B-PER, I-PER, B-ORG, I-ORG, B-LOC, I-LOC, B-MISC, I-MISC, O
  - approximately 22000 tokens in the dictionary

Wolff	B-PER
,	O
currently	O
a	O
journalist	O
in	O
Argentina	B-LOC
,	O
played	O
with	O
Del	B-PER
Bosque	I-PER

Application: Named entity Recognition

- Features
  - Word itself
- Learning
  - Observed: counting
- Inference
  - Viterbi decoding

	Naïve Bayes	HMM
Accuracy	70.77%	91.44%

Resources for HMM & CRF

- HMM toolbox for MATLAB
  - Written by Kevin Murphy
  - http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
- Conditional Random Fields
  - One of the most popular machine learning algorithms
  - **Test of Time Award**, ICML 2011
  - http://www.inference.phy.cam.ac.uk/hmw26/crf/

Outline

- HMM
  - Big Picture
  - Conditional Independence
  - Inference
  - Learning
  - Application: Named Entity Recognition
- Common Mistakes in Midterm

Short Questions

6. **[2 points]** For any model with latent variables z and parameters θ, the EM algorithm alternates between the following two steps (1) replacing each occurrence of z in the likelihood function with its expectation $\langle z \rangle$, and (2) maximizing this likelihood function with respect to the model parameters θ.

- False
- E-step (slide #24, lecture9): computing the expected value of the sufficient statistics of the hidden variables (i.e., z) given current est. of the parameters
- A statistic $T(X)$ is **sufficient for underlying parameter ϑ** if the conditional probability distribution of the data X, given the statistic $T(X)$, does not depend on the parameter ϑ

$$\Pr(X = x | T(X) = t, \theta) = \Pr(X = x | T(X) = t),$$

- *Another example: z^2 should be replaced with <z^2>, not <z>^2*

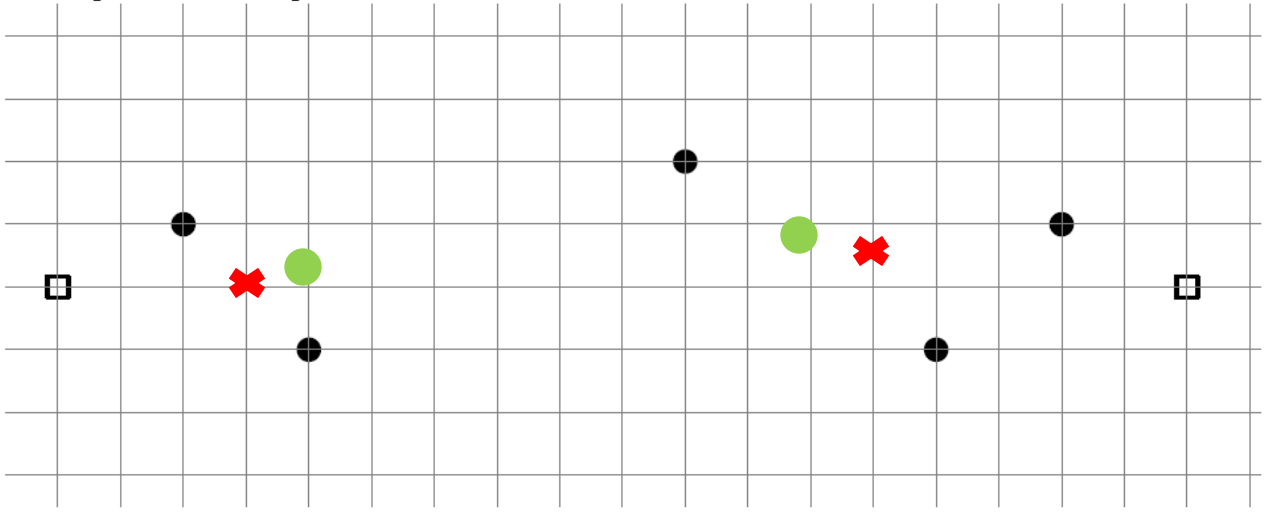
Short Questions

9. **[2 points]** With K-Gaussians EM, we can select the best K as follows: compute the log-likelihood $P(x|\theta)$ for each K, and pick the K that gives the highest value.

- False
- You can always increase the log-likelihood by having a higher K; thus, will have K=N

K-Means and EM

Consider the dataset depicted on the grid below. The data points are represented by filled black circles. Hollow shapes do not represent data points.



- **1. [3 points]** Suppose we initialize the K-means algorithm (with $K = 2$) to the two hollow squares, after which we run the algorithm. On the diagram, mark the final positions of the two centers with an 'x'. You only need to draw the positions qualitatively; there is no need to compute the exact locations.
- **2. [3 points]** Suppose we initialize a K-Gaussian mixture model (with $K = 2$) to the two hollow squares, after which we run the EM algorithm. On the diagram, mark the final centers of the two Gaussians with an 'o'. Again, do this qualitatively.

- K-Means: hard assignment
- GMM: 'soft' assignment, will consider the effect of all points

Classification

1. [**3 points**] Hence, Bob changes the objective function from problem 3.2.1 a little to prefer sparsity. Please write down the modified objective function. Note that if you would like to introduce new parameters in the objective function, please explicitly specify how the value of the parameter could affect sparsity (e.g. the larger the parameter is, the more sparsity the modified objective function prefers).

- Answer: $L(\theta)-\lambda|\theta|$
- Common mistake:
  - $L(\theta)+\lambda|\theta|$
  - $L(\theta)-\lambda|\theta|\_2$

Neural Network

[2 points] Assume $N = 1000$ and $d = 10$, what is an obvious problem with the above "non-linear" formulation?

- Answer: # features > # data → overfitting
- Insight:
  - Representation power of NN
  - Another popular approach to model non-linearity: support vector machine → implicit feature mapping using kernels