Modeling and Predicting Sequences: HMM and (may be) CRF

Amr Ahmed

10701

Feb 25

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**

Predicting Sequences

- Example: POS

y	NN	VBD	DD	NN	JJ
x	Students	found	the	HW	easy

- Example NP chunking

y:	B	I	E	O	B	E	O	B	E
X:	Rockwell	International	Corp	signed	an	agreement	with	Boeing	Co.

Back To big Picture

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

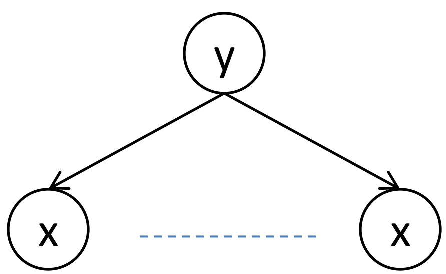
Back To big Picture

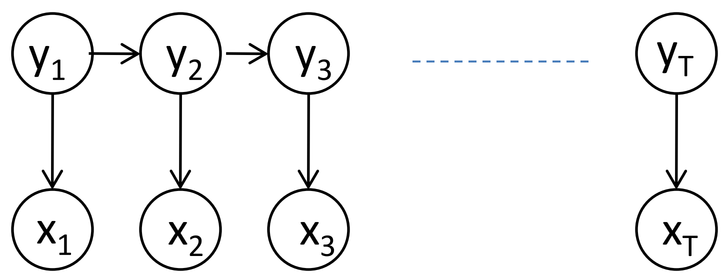
Sequence of Output

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

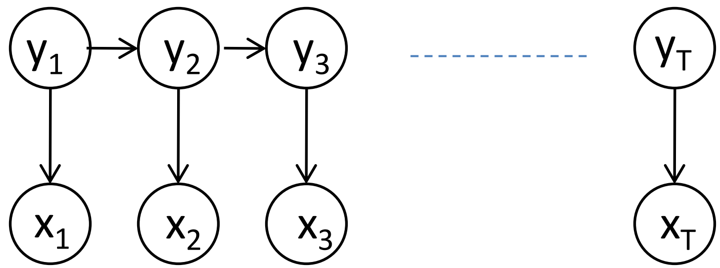
HMM

- 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



- What we need to define

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

- Factorization:

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

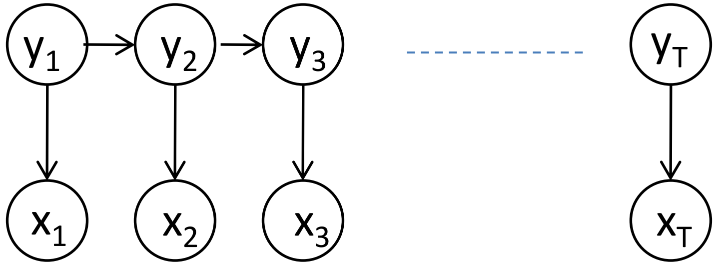
Tasks

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

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

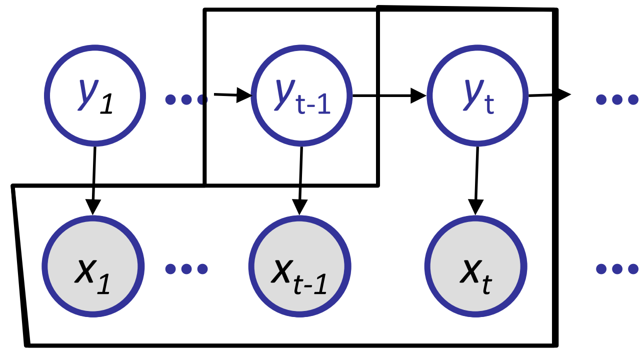


MPA

A trick that we will use often: add a variable and marginalize over to be able to apply recursion

- We need to do that for any *t*
  - $\alpha_1, \alpha_2, \ldots, \alpha_T$
  - Define a recursive program

$$\alpha_t^i = p(y_t = i, x_1, \ldots, x_t)$$
$$\alpha_{t-1}^j = p(y_{t-1} = j, x_1, \ldots, x_{t-1})$$



Divide variable into three sets: $\{X_1,..,x_{t-1},y_{t-1}\}$ (to be able to see α_{t-1}),$\{y_t\}$,$\{x_t\}$, then apply chain rule

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

$$= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1}) P(y_t = k \mid y_{t-1},x_1,\ldots,x_{t-1}) P(x_t \mid y_t = k, x_1,\ldots,x_{t-1},y_{t-1})$$

$$= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1}) P(y_t = k \mid y_{t-1}) P(x_t \mid y_t = k)$$

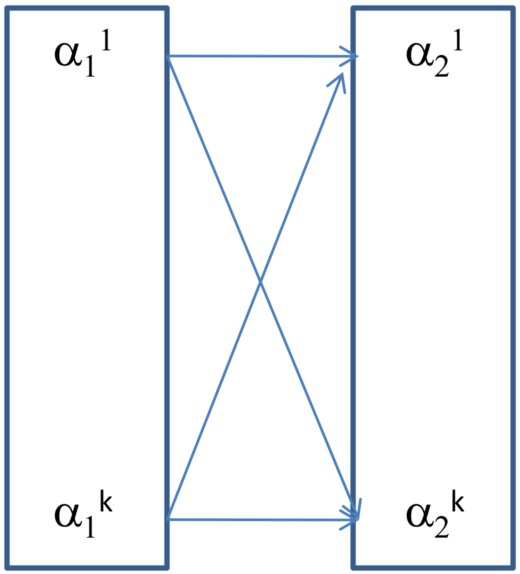
$$= P(x_t \mid y_t = k) \sum_i P(x_1,\ldots,x_{t-1},y_{t-1} = i) P(y_t = k \mid y_{t-1} = i)$$

Summing over y_{t-1} is just summing Over y_{t-1}=1…K

$$= P(x_t \mid y_t = k) \sum_i \alpha_{t-1}^i a_{i,k}$$

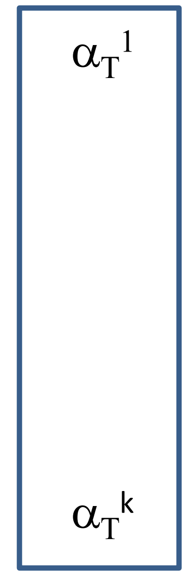
Forward Algorithm

$$\alpha_1^1 = P(x_1 \mid y_1 = 1)\pi_1$$

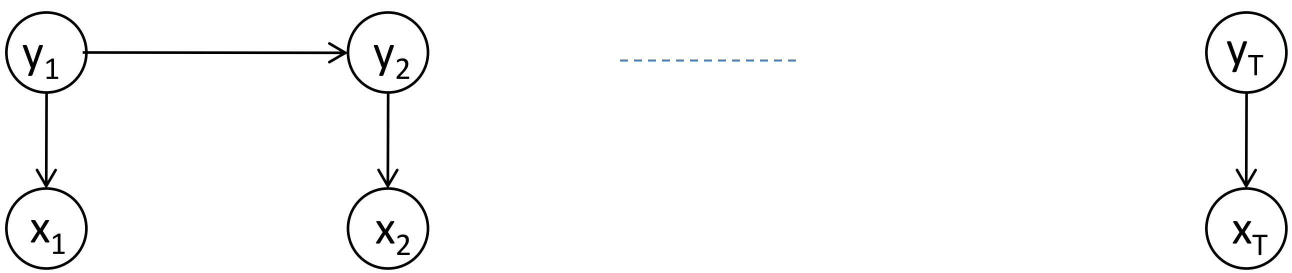


$$= P(x_2 \mid y_2 = 1)\sum_i \alpha_1^i a_{i,1}$$

$$= P(x_2 \mid y_2 = k)\sum_i \alpha_1^i a_{i,k}$$



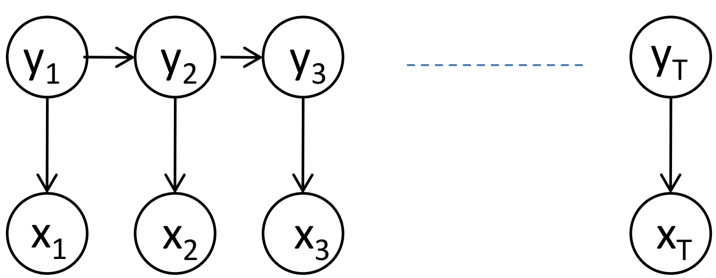
$$\alpha_1^k = P(x_1 \mid y_1 = k)\pi_k$$



Inference: MPA

- Find argmax$_i$ P(y_t=i|**x**)
- We need to compute P(y_t=i|**x**) first

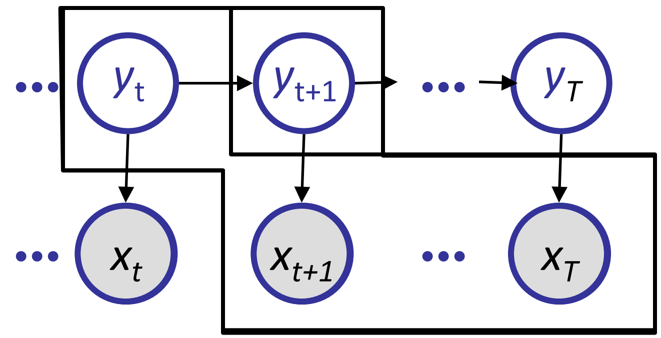
$$
\begin{aligned}
p(y_t = i | x_1, \ldots, x_T) &= \frac{p(y_t = i, x_1, \ldots, x_T)}{p(x_1, \ldots, x_T)} \\
&= \frac{p(y_t = i, x_1, \ldots, x_t) p(x_{t+1}, \ldots, x_T | y_t = i, x_1, \ldots, x_t)}{p(x_1, \ldots, x_T)} \\
&= \frac{\boxed{p(y_t = i, x_1, \ldots, x_t)}\boxed{p(x_{t+1}, \ldots, x_T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



Backward Algorithm

- We need to do that for any *t*
  - $\beta_1, \beta_2, \ldots, \beta_T$
  - Define a recursive program

$$\beta_t^i = p(x_{t+1}, \ldots, x_T | y_t = i)$$
$$\beta_{t+1}^j = p(x_{t+2}, \ldots, x_T | y_{t+1} = j)$$

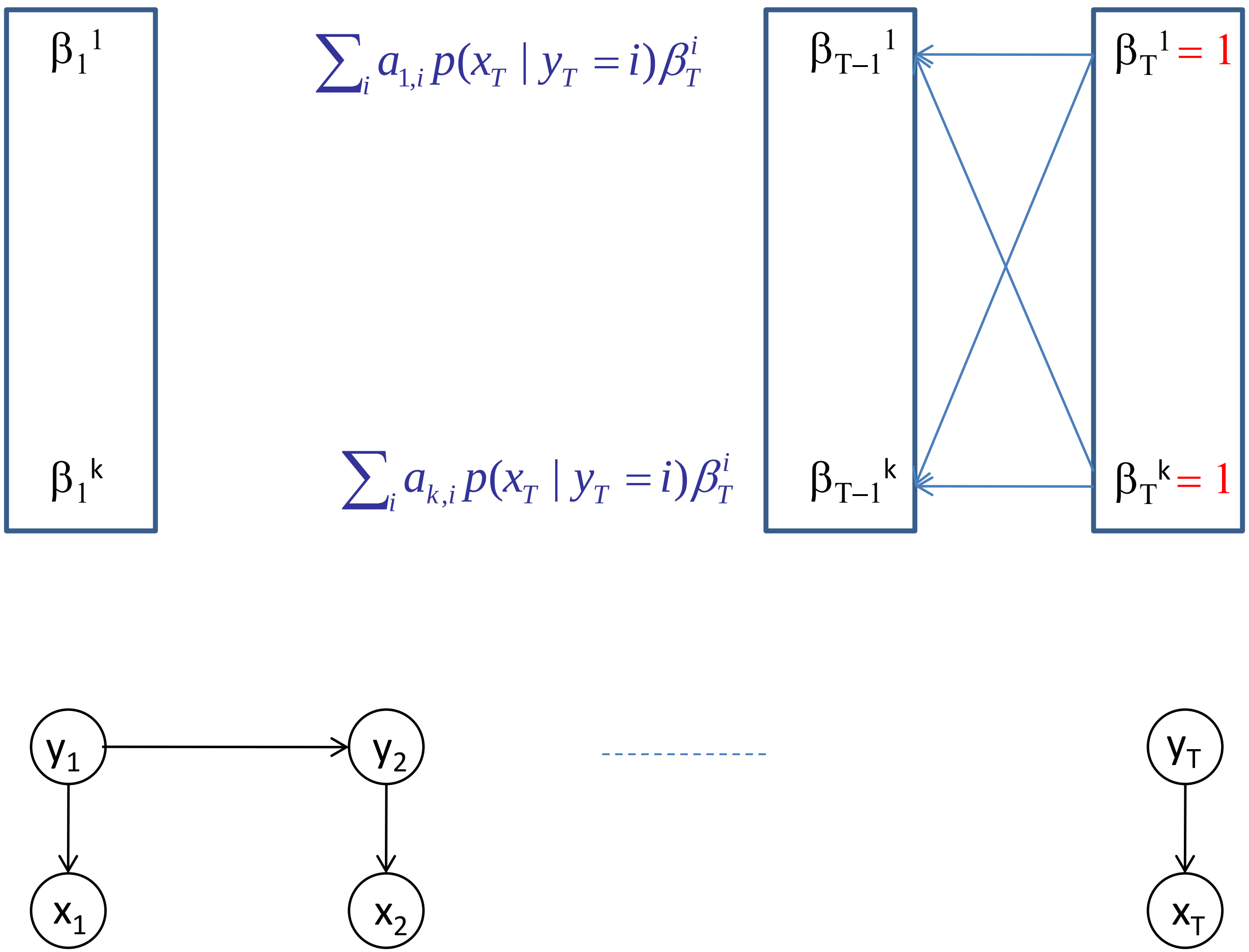


add and marginalize trick

Divide variable into three sets: $\{y_{t+1}\}$, $\{x_{t+1}\}$,$\{X_{t+2},..,x_T\}$ (to be able to see β_{t+1}) then apply chain rule

$$\beta_t^k = P(x_{t+1},\ldots,x_T \mid y_t = k)$$
$$= \sum_{y_{t+1}} P(x_{t+1},\ldots,x_T, y_{t+1} \mid y_t = k)$$
$$= \sum_i P(y_{t+1} = i \mid y_t = k) p(x_{t+1} \mid y_{t+1} = i, y_t = k) P(x_{t+2},\ldots,x_T \mid x_{t+1}, y_{t+1} = i, y_t = k)$$
$$= \sum_i P(y_{t+1} = i \mid y_t = k) p(x_{t+1} \mid y_{t+1} = i) \boxed{P(x_{t+2},\ldots,x_T \mid y_{t+1} = i)}$$
$$= \sum_i a_{k,i}\, p(x_{t+1} \mid y_{t+1} = i) \beta_{t+1}^i$$

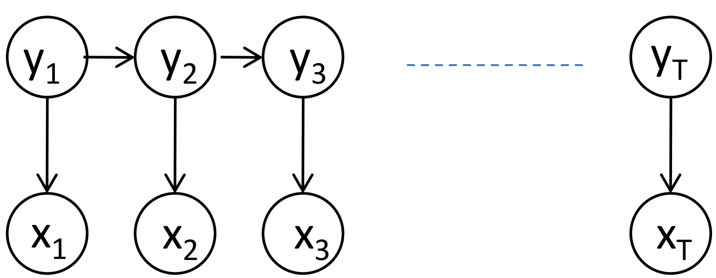
Backward Algorithm



Inference: MPA

- Find argmax$_i$ P(y_t=i|**x**)
- We need to compute P(y_t=i|**x**) first

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



Evaluation

$$
\begin{aligned}
P(x_1, \ldots, x_T) &= \sum_{y_T} P(x_1, \ldots, x_T, y_T) \\
&= \sum_{i=1}^{k} P(x_1, \ldots, x_T, y_T = i) \\
&= \sum_{i=1}^{k} \alpha_T^i
\end{aligned}
$$

Now we have everything to compute:

$$
p(y_t = i | x_1, \ldots, x_T) = \frac{\alpha_t^i \beta_t^i}{p(x_1, \ldots, x_T)}
$$

Practical Consideration

- β, α are product of many terms
- Likely to run (and you will) into underflow for any sequence > 10
- Can we use logs?

$$\alpha_t^k = P(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\log\left(\alpha_t^k\right) = \log\left(P(x_t \mid y_t^k = 1)\right) + \log\left(\sum_i \alpha_{t-1}^i a_{i,k}\right)$$

- In general we didn't get log(α) on the right hand side, but you can use a technique called (log add) that I didn't discuss in the recitation.
- Solution: rescaling --- normalize α after each step!

Scaling

- Normalize α after each step!
- c_t is a normalization constant
- Keep track of c_t for all *t*

$$\hat{\alpha}_t^k = \frac{P(x_t \mid y_t = k)\sum_i \hat{\alpha}_{t-1}^i a_{i,k}}{\sum_j P(x_t \mid y_t = j)\sum_i \hat{\alpha}_{t-1}^i a_{i,j}}$$

$$c_t = \sum_j P(x_t \mid y_t = j)\sum_i \hat{\alpha}_{t-1}^i a_{i,j}$$

Scaling: Interpretation

- How to interpret c_t and the normalized α
- Claim: $\hat{\alpha}_t^k = \alpha_t^k \prod_{i=1}^{t} \frac{1}{c_i}$ remember $c_t = \sum_j P(x_t \mid y_t = j) \sum_i \hat{\alpha}_{t-1}^i a_{i,j}$
- Proof by induction: assume it is true for α_{t-1}

$$\hat{\alpha}_t^k = \frac{P(x_t \mid y_t = k) \sum_i \hat{\alpha}_{t-1}^i a_{i,k}}{\sum_j P(x_t \mid y_t = j) \sum_i \hat{\alpha}_{t-1}^i a_{i,j}}$$

Subs. α_{t-1} from hypothesis

$$= \frac{P(x_t \mid y_t = k) \sum_i \prod_{t'=1}^{t-1} \frac{1}{c_{t'}} \alpha_{t-1}^i a_{i,k}}{c_t}$$

By definition of c_t

$$= \prod_{t'=1}^{t-1} \frac{1}{c_{t'}} \frac{P(x_t \mid y_t = k) \sum_i \alpha_{t-1}^i a_{i,k}}{c_t} = \prod_{t'=1}^{t} \frac{1}{c_{t'}} \alpha_t^k$$

Scaling: Computation

- Can we still calculate $P(x_1,..,x_T)$
- Yes!

$$\sum_{i=1}^{K} \hat{\alpha}_T^i = 1$$

$$\sum_{i=1}^{K} \alpha_T^i \prod_{t=1}^{T} \frac{1}{c_t} = 1$$

$$\sum_{i=1}^{K} \alpha_T^i = \prod_{t=1}^{T} c_t = P(x_1,\ldots,x_T)$$

- But you really need to do it in log space:

$$\log P(x_1,\ldots,x_T) = \sum_{t=1}^{T} \log(c_t)$$

Scaling: Backward

- You can use the same trick with β
- Now how to compute MPA

$$P(y_t = k \mid \mathbf{x}) = \frac{P(y_t = k, \mathbf{x})}{P(\mathbf{x})}$$

$$\propto \alpha_t^k \beta_t^k$$

$$\propto \hat{\alpha}_t^k \hat{\beta}_t^k$$

- Then finally normalize
- Note that the constant in the “hat” version of both α and β is only function of *t* (same for all k*)*

Tasks

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

Viterbi

- Find the globally maximal posterior sequence
  - $\text{argmax}_{y1,..,yT}\ P(y_1,..,y_T|x_1,...x_T)$
  - Same as $\text{argmax}_{y1,..,yT}\ P(y_1,..,y_T,x_1,...x_T)$ 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:

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

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

Viterbi: the math

- You should be bored of that by now?
- No, this is a different trick (pushing max in)

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

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

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

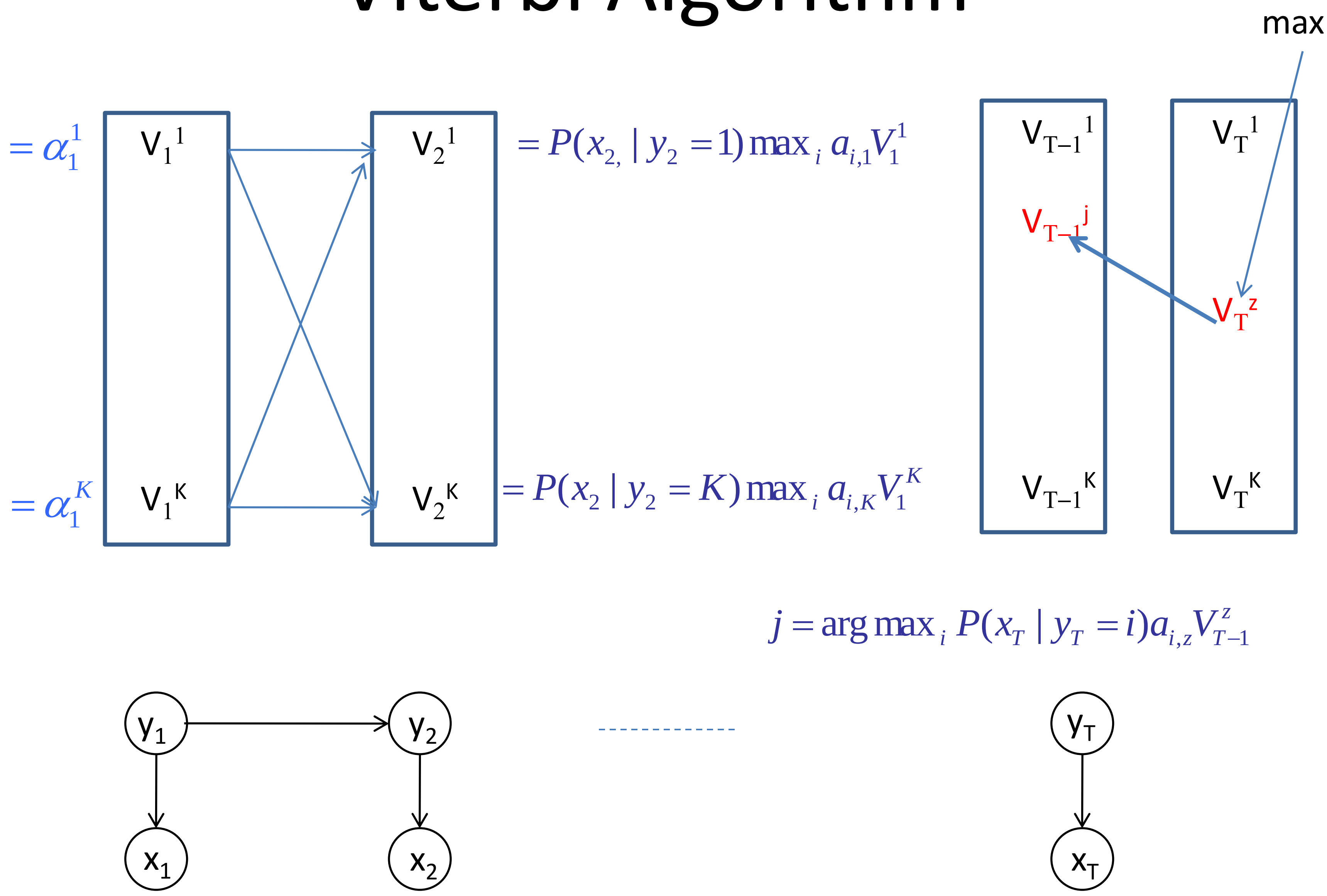
$$= \max_i P(x_{t+1,}y_{t+1}=k \mid y_t=i)\max_{\{y_1,\dots y_{t-1}\}} P(x_1,\dots,x_{t-1},y_1,\dots,y_{t-1},x_t,y_t=i)$$

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

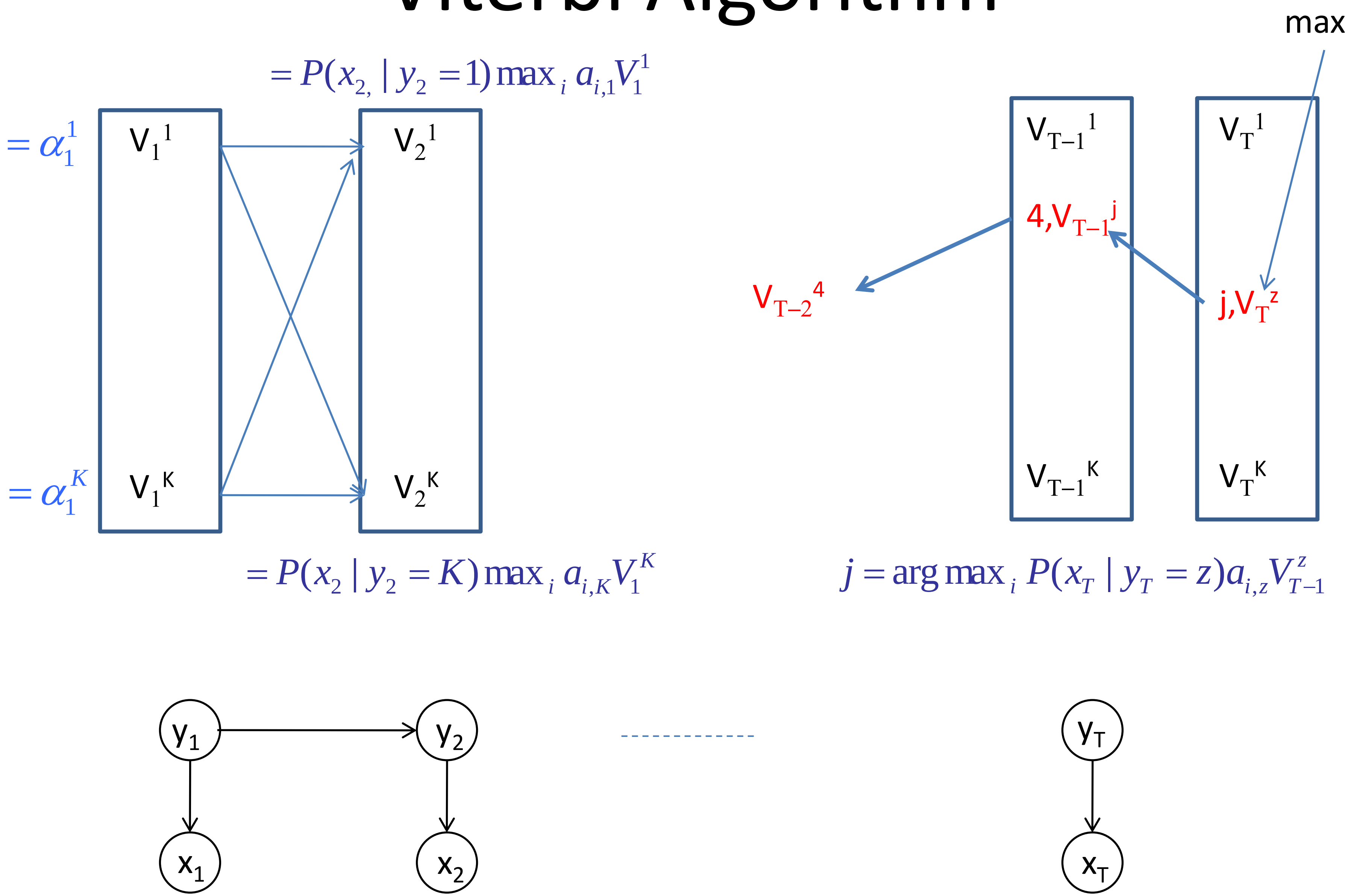
$$= P(x_{t+1,} \mid y_{t+1}=k)\max_i a_{i,k}V_t^i$$

- Also keep track of the maximizing *i*

Viterbi Algorithm



Viterbi Algorithm



Viterbi: scaling

- You can use log here

$$V_t^k = P(x_{t,} \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

$$\log V_t^k = \log p(x_t \mid y_t^k = 1) + \max_i \left(\log\left(a_{i,k}\right) + \log V_{t-1}^i\right)$$

Tasks

- Inference
  - Find $P(\mathbf{y}|\mathbf{x})$
    - 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

Learning

- For fully observed dataD = {($\mathbf{x}_n$,$\mathbf{y}_n$)} n=1:N
- *LL* is log-lik

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

- For partially observed data (missing Y)

$$LL(\boldsymbol{\theta}) = \sum_n \log p(\mathbf{x}_n | \boldsymbol{\theta})$$
$$= \sum_n \log \sum_{\mathbf{y}_n} p(\mathbf{x}_n, \mathbf{y}_n | \boldsymbol{\theta})$$

- Using Jensen

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_n \sum_{y_n} p(\mathbf{y}_n | \mathbf{x}_n, \boldsymbol{\theta}^{old}) \log p(\mathbf{x}_n, \mathbf{y}_n | \boldsymbol{\theta})$$

Learning: Observed

- For a give sequence

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

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

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

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

On the board
(see next page)

- $C_{i,n}$: 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 ($\mathbf{x}_n$,$\mathbf{y}_n$)

digression

- Take **y=**1,2,3,1,2   **x=**1,3,5,1,1
- Then $$p(\mathbf{x},\mathbf{y}) = p(y_{n,1})\prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1})\prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t})$$
- Which

$$= \pi_1 * a_{12} * a_{23} * a_{32} * a_{12} * b_{11} * b_{23} * b_{35} * b_{11} * b_{21}$$

$$= \pi_1 * (a_{12})^2 * a_{23} * a_{31} * (b_{11})^2 * b_{23} * b_{35} * b_{21}$$

$$= \prod_{i=1}^{K} \pi_i^{C_{i,n}} \prod_{i,j=1}^{K} a_{ij}^{A_{ij,n}} \prod_{i=1,o=1}^{i=K,o=M} b_{io}^{B_{io,n}}$$

-Note that if the count of any item in C or A or B is zero then simply the term that involves it will be 1.

Learning: observed

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

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

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

Learning: unsupervised

- Recall $LL(\boldsymbol{\theta}) = \sum_n \log \sum_{\mathbf{y}_n} p(\mathbf{x_n}, \mathbf{y}_n | \boldsymbol{\theta})$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_n \sum_y p(\mathbf{y} | \mathbf{x_n}, \boldsymbol{\theta}^{old}) \log p(\mathbf{x_n}, \mathbf{y} | \boldsymbol{\theta})$$

$$\log p(\mathbf{x}_n, \mathbf{y}_n) = \sum_{i=1}^{K} C_{i,n} \log \pi_i + \sum_{i,j=1}^{K} A_{ij,n} \log a_{ij} + \sum_{i=1,o=1}^{K,M} B_{io,n} \log b_{io}$$

- So we have:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) =$$

$$\sum_n \sum_{y_n} P(\mathbf{y}_n | \mathbf{x_n}, \boldsymbol{\theta}^{old}) \left(\sum_{i=1}^{K} C_{i,n} \log \pi_i + \sum_{i,j=1}^{K} A_{ij,n} \log a_{ij} + \sum_{i=1,o=1}^{K,M} B_{io,n} \log b_{io} \right)$$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) =$$

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

- All expectations are under $P(\mathbf{y}_n \mid \mathbf{x}_\mathbf{n}, \boldsymbol{\theta}^{old})$

$$\langle C_{i,n} \rangle = \sum_{y_n} p(\mathbf{y}_n \mid \mathbf{x}_\mathbf{n}, \boldsymbol{\theta}^{old}) C_{i,n}$$

$$= P\left(y_{n,1} = i \mid \mathbf{x}_\mathbf{n}, \boldsymbol{\theta}^{old} \right)$$

- We know how to compute that (F-B)

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) =$$

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

- Where

$$\langle B_{io,n} \rangle = \sum_y p(\mathbf{y}_n \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}) B_{io,n}$$
$$= \sum_{t:x_{n,t}=o} P\left(y_{n,t} = i \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}\right)$$

- We also know how to compute that (F-B)

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) =$$

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

- Where

$$\langle A_{ij,n} \rangle = \sum_{y_n} p(\mathbf{y}_n \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}) A_{ij,n}$$

$$= \sum_{t=1:T-1} P\left(y_{n,t} = i, y_{n,t+1} = j \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}\right)$$

- Do we know how to compute that ? Sort of

- Recall $\langle A_{ij,n} \rangle = \sum_{y_n} p(\mathbf{y}_n \mid \mathbf{x_n}, \boldsymbol{\theta}^{t-1}) A_{ij,n}$

$$= \sum_{t=1:T-1} P\left(y_{n,t} = i, y_{n,t+1} = j \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}\right)$$

- But: $$P\left(y_{n,t} = i, y_{n,t+1} = j \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}\right) = \frac{P\left(y_{n,t} = i, y_{n,t+1} = j, \mathbf{x_n} \mid \boldsymbol{\theta}^{old}\right)}{P\left(\mathbf{x_n} \mid \boldsymbol{\theta}^{old}\right)}$$

$$= \frac{\alpha_t^i P(x_{n,t+1} \mid y_{n,t+1} = j) a_{ij} \beta_{t+1}^j}{P\left(\mathbf{x_n} \mid \boldsymbol{\theta}^{old}\right)}$$

You should be able to Prove the above step

- Which we now how to compute

M-Step

- Now we have all what we need

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) =$$

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

- Just as before solve for MLE, for ex:

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

EM Summary for HMM

- 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

Final Note about Rescaling

- Recall

$$P\left(y_{n,t}=i, y_{n,t+1}=j \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}\right)=\frac{\alpha_t^i P(x_{n,t+1} \mid y_{n,t+1}=j) a_{ij} \beta_{t+1}^j}{P\left(\mathbf{x_n} \mid \boldsymbol{\theta}^{old}\right)}$$

$$\propto \hat{\alpha}_t^i P(x_{n,t+1} \mid y_{n,t+1}=j) a_{ij} \hat{\beta}_{t+1}^j$$

- Remember the underflow solution
- Same thing here, compute using normalized vectors and then finally normalize P