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

\[ y \quad \text{NN} \quad \text{VBD} \quad \text{DD} \quad \text{NN} \quad \text{JJ} \]

\[ x \quad \text{Students} \quad \text{found} \quad \text{the} \quad \text{HW} \quad \text{easy} \]

• Example NP chunking

\[ y:\]  
\[ B \quad I \quad E \quad O \quad B \quad E \quad O \quad B \quad E \]

\[ X:\]  
\[ \text{Rockwell International Corp} \quad \text{signed} \quad \text{an agreement} \quad \text{with} \quad \text{Boeing Co.} \]
• Generative:
  – Models $P(x,y)$
  – Predict using Bayes rule $\text{argmax } P(y|x)$
  – Naïve Bayes

• Discriminative:
  – Model $P(y|x)$
  – Predict using $\text{argmax } P(y|x)$
  – Logistic Regression
• Generative:
  – Models $P(x, y)$
  – Predict using Bayes rule $\arg\max_y P(y|x)$
  – HMM

• Discriminative:
  – Model $P(y|x)$
  – Predict using $\arg\max_y P(y|x)$
  – 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 $M^{|X|} \times K^{|Y|}$
• We need to add some conditional independence assumption to make things manageable
  – We have done that in Naïve Bayes
What we need to define

- Initial state: $P(y_1)$
- Transition: $P(y_t | y_{t-1})$
- Emission: $P(x_t | y_t)$

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

• Inference
  – Find $P(y|x)$
    • MPA: $P(y_t|x)$
    • Viterbi: $P(y|X)$
  – Learning
    • Learning model parameters using MLE
      – $\pi_i, a_{ij}, b_{ik}$
      – Fully Observed:
        » count and normalize
      – Unsupervised:
        » EM
Inference: MPA

- Find $\arg\max_i P(y_t=i|x)$
- We need to compute $P(y_t=i|x)$ first

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

(1)
• We need to do that for any $t$
  
  - $\alpha_1, \alpha_2, \ldots, \alpha_T$
  
  - Define a recursive program
    
    $\alpha^i_t = p(y_t = i, x_1, \ldots, x_t)$
    
    $\alpha^j_{t-1} = p(y_{t-1} = j, x_1, \ldots, x_{t-1})$

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

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

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

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

    $= P(x_t | y_t = k) \sum_i \alpha^i_{t-1} a_{i,k}$

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

Divide variable into three sets: $\{X_1, \ldots, X_{t-1}, y_{t-1}\}$ (to be able to see $\alpha_{t-1}, \{y_t\}, \{x_t\}$, then apply chain rule

Summing over $y_{t-1}$ is just summing Over $y_{t-1} = 1 \ldots K$
Forward Algorithm

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

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

\[ \alpha_2^1 = P(x_2 \mid y_2 = 1) \sum_i \alpha_i^1 a_{i,1} \]

\[ \alpha_2^k = P(x_2 \mid y_2 = k) \sum_i \alpha_i^1 a_{i,k} \]

\[ \alpha_T^1 \]

\[ \alpha_T^k \]
Inference: MPA

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

\[
p(y_t = i \mid x_1, \ldots, x_T) = \frac{p(y_t = i, x_1, \ldots, x_T)}{p(x_1, \ldots, x_T)}
\]

\[
= \frac{p(y_t = i, x_1, \ldots, x_t)p(x_{t+1}, \ldots, x_T \mid y_t = i, x_1, \ldots, x_t)}{p(x_1, \ldots, x_T)}
\]

\[
= \frac{\alpha_t^i \beta_t^i}{p(x_1, \ldots, x_T)}
\]

(1)
Backward Algorithm

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

\[
\beta_t^i = p(x_{t+1}, \ldots, x_T | y_t = i)
\]
\[
\beta_{t+1}^j = p(x_{t+2}, \ldots, x_T | y_{t+1} = j)
\]
\[
\beta_t^k = P(x_{t+1}, \ldots, x_T | y_t = k)
= \sum_{y_{t+1}} P(x_{t+1}, \ldots, x_T, y_{t+1} | y_t = k)
= \sum_i P(y_{t+1} = i | y_t = k) p(x_{t+1} | y_{t+1} = i, y_t = k) P(x_{t+2}, \ldots, x_T | x_{t+1}, y_{t+1} = i, y_t = k)
= \sum_i P(y_{t+1} = i | y_t = k) p(x_{t+1} | y_{t+1} = i) P(x_{t+2}, \ldots, x_T | y_{t+1} = i)
= \sum_i a_{k,i} p(x_{t+1} | y_{t+1} = i) \beta_{t+1}^i
\]
Backward Algorithm

\[
\beta_1^1 \quad \beta_1^k
\]

\[
\sum_i a_{1,i} p(x_T \mid y_T = i) \beta_T^i
\]

\[
\sum_i a_{k,i} p(x_T \mid y_T = i) \beta_T^i
\]

\[
\beta_{T-1}^1 \quad \beta_{T-1}^k
\]

\[
\beta_T^1 = 1
\]

\[
\beta_T^k = 1
\]
Inference: MPA

• Find $\arg\max_i P(y_t=i|x)$

• We need to compute $P(y_t=i|x)$ first

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

$$= \frac{p(y_t = i, x_1, \ldots, x_T)p(x_{t+1}, \ldots, x_T | y_t = i, x_1, \ldots, x_t)}{p(x_1, \ldots, x_T)}$$

$$= \frac{p(y_t = i, x_1, \ldots, x_T)p(x_{t+1}, \ldots, x_T | y_t = i)}{p(x_1, \ldots, x_T)}$$

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

(1)
Evaluation

\[
P(x_1, \ldots, x_T) = \sum_{y_T} P(x_1, \ldots, x_T, y_T) = \sum_{i=1}^{k} P(x_1, \ldots, x_T, y_T = i) = \sum_{i=1}^{k} \alpha_i^T
\]

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

• $\beta, \alpha$ are product of many terms
• Likely to run (and you will) into underflow for any sequence $> 10$
• Can we use logs?

$$\alpha^k_t = P(x_t \mid y^k_t = 1) \sum_i \alpha^i_{t-1} a_{i,k}$$

$$\log(\alpha^k_t) = \log(P(x_t \mid y^k_t = 1)) + \log(\sum_i \alpha^i_{t-1} a_{i,k})$$

• In general we didn’t get $\log(\alpha)$ 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 $\alpha$ after each step!
Scaling

- Normalize $\alpha$ 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 $\alpha$
- 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 $\alpha_{t-1}$

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

Subs. $\alpha_{t-1}$ from hypothesis

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

By definition of $c_t$
Scaling: Computation

- Can we still calculate \( P(x_1,\ldots,x_T) \)
- 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 $\beta$
• Now how to compute MPA

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

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

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

• Then finally normalize
• Note that the constant in the “hat” version of both $\alpha$ and $\beta$ is only function of $t$ (same for all $k$)
Tasks

• Inference
  – Find $P(y|\mathbf{x})$
    • MPA: $P(y_t|\mathbf{x})$
    • Viterbi: $P(y|X)$

• Learning
  • Learning model parameters using MLE
    – $\pi_i, a_{ij}, b_{ik}$
    – Fully Observed:
      » count and normalize
    – Unsupervised:
      » EM
Viterbi

- Find the globally maximal posterior sequence
  \[ \text{argmax}_{y_1, \ldots, y_T} P(y_1, \ldots, y_T | x_1, \ldots, x_T) \]
  - Same as \[ \text{argmax}_{y_1, \ldots, y_T} P(y_1, \ldots, y_T, x_1, \ldots, x_T) \] why?
  - Develop a dynamic (recursive) program
    - \[ \max_{y_1 \ldots y_{t-1}} P(y_1, \ldots, y_t, x_1, \ldots, x_t) \] and relate it to
    - \[ \max_{y_1 \ldots y_{t-2}} P(y_1, \ldots, y_{t-1}, x_1, \ldots, x_{t-1}) \]
  - 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 \( y_1, \ldots, y_{t-1} \)
Viterbi: the math

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

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

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

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

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

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

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

- Also keep track of the maximizing \( i \)
Viterbi Algorithm

\[ \alpha_1^1 = \alpha_1^K = P(x_2 \mid y_2 = 1) \max_i a_{i,1} V_1^1 \]

\[ P(x_2 \mid y_2 = K) \max_i a_{i,K} V_1^K \]

\[ j = \arg \max_i P(x_T \mid y_T = i) a_{i,z} V_{T-1}^z \]
**Viterbi Algorithm**

\[
\alpha_1^1 = P(x_2 \mid y_2 = 1) \max_{i} a_{i,1} V_1^1 \\
= \alpha_1^1 \\
= P(x_2 \mid y_2 = K) \max_{i} a_{i,K} V_1^K \\
= \alpha_1^K \\
\]

\[
j = \arg \max_{i} P(x_T \mid y_T = z) a_{i,z} V_{T-1}^z \\
\]

Diagram:

- \(y_1\) connected to \(y_2\)
- \(x_1\) connected to \(x_2\)
- \(y_T\) connected to \(x_T\)
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 \( P(y|x) \)
    • MPA: \( P(y_t|x) \)
    • Veterbi: \( P(y|X) \)

– Learning
  • Learning model parameters using MLE
    – \( \pi_i, a_{ij}, b_{ik} \)
    – Fully Observed:
      » count and normalize
    – Unsupervised:
      » EM
Learning

• For fully observed data $D = \{(x_n, y_n)\} \ n=1:N$

• $LL$ is log-lik

$$LL(\theta) = \log p(X, Y)$$

• For partially observed data (missing $Y$)

$$LL(\theta) = \sum_n \log p(x_n | \theta)$$

$$= \sum_n \log \sum_{y_n} p(x_n, y_n | \theta)$$

• Using Jensen

$$Q(\theta, \theta^{old}) = \sum_n \sum_{y_n} p(y_n | x_n, \theta^{old}) \log p(x_n, y_n | \theta)$$
Learning: Observed

• For a given sequence

\[ LL(\theta) = \log p(X, Y) \]

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

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

• \( C_{i,n} \): number of times first state was \( i \) in \( x_n \) (0 or 1)

• \( B_{io,n} \): number of times state \( i \) emits \( o \) in \( (x_n,y_n) \)

• \( A_{ij,n} \): number of times state \( i \) moves to state \( j \) in \( (x_n,y_n) \)
digression

- Take $y=1,2,3,1,2$  $x=1,3,5,1,1$
- Then

$$p(x,y) = p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t})$$
- Which

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

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

$$= \prod_{i=1}^{K} \pi_{i,n} \prod_{i,j=1}^{K} a_{ij,n} \prod_{i=1,o=1}^{i,K,o=M} b_{io,n}$$

- 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

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

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

\[ \alpha_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n A_{ij,n}}{\sum_n \sum_{j'} A_{ij',n}} \]
Learning: unsupervised

• Recall  \( LL(\theta) = \sum_n \log \sum_{y_n} p(x_n, y_n | \theta) \)

\[
Q(\theta, \theta^{old}) = \sum_n \sum_y p(y | x_n, \theta^{old}) \log p(x_n, y | \theta)
\]

\[
\log p(x_n, y_n) = \sum_{i=1}^K C_{i,n} \log \pi_i + \sum_{i,j=1}^K A_{i,j,n} \log a_{ij} + \sum_{i=1,o=1}^{K,M} B_{i,o,n} \log b_{io}
\]

• So we have:

\[
Q(\theta, \theta^{old}) = \sum_n \sum_{y_n} P(y_n | x_n, \theta^{old}) \left( \sum_{i=1}^K C_{i,n} \log \pi_i + \sum_{i,j=1}^K A_{i,j,n} \log a_{ij} + \sum_{i=1,o=1}^{K,M} B_{i,o,n} \log b_{io} \right)
\]
\[ Q(\theta, \theta^{old}) = \]
\[ \sum_{n} \left( \sum_{i=1}^{K} \langle C_{i,n} \rangle \log \pi_i + \sum_{i,j=1}^{K} \langle A_{ij,n} \rangle \log a_{ij} + \sum_{i=1, o=1}^{K, M} \langle B_{io,n} \rangle \log b_{io} \right) \]

• All expectations are under \( P(y_n | x_n, \theta^{old}) \)

\[ \langle C_{i,n} \rangle = \sum_{y_n} P(y_n | x_n, \theta^{old}) C_{i,n} \]
\[ = P(y_{n,1} = i | x_n, \theta^{old}) \]

• We know how to compute that (F-B)
\[ Q(\theta, \theta^{old}) = \sum_n \left( \sum_{i=1}^{K} \langle C_{i,n} \rangle \log \pi_i + \sum_{i,j=1}^{K} \langle A_{ij,n} \rangle \log a_{ij} + \sum_{i=1, o=1}^{K, M} \langle B_{i o, n} \rangle \log b_{i o} \right) \]

• Where

\[ \langle B_{i o, n} \rangle = \sum_{y} p(y_n \mid x_n, \theta^{old}) B_{i o, n} \]

\[ = \sum_{t: x_{n,t} = o} P(y_{n,t} = i \mid x_n, \theta^{old}) \]

• We also know how to compute that (F-B)
\[ Q(\theta, \theta^{old}) = \]
\[
\sum_n \left( \sum_{i=1}^{K} \langle C_{i,n} \rangle \log \pi_i + \sum_{i,j=1}^{K} \langle A_{ij,n} \rangle \log a_{ij} + \sum_{i=1, o=1}^{K, M} \langle B_{io,n} \rangle \log b_{io} \right) \]

• Where

\[
\langle A_{ij,n} \rangle = \sum_{y_n} p(y_n \mid x_n, \theta^{old}) A_{ij,n} \]
\[
= \sum_{t=1:T-1} P(y_{n,t} = i, y_{n,t+1} = j \mid x_n, \theta^{old}) \]

• Do we know how to compute that? Sort of
• Recall \[ \langle A_{ij,n} \rangle = \sum_{y_n} p(y_n | x_n, \theta^{t-1}) A_{ij,n} \]
  \[ = \sum_{t=1:T-1} P(y_{n,t} = i, y_{n,t+1} = j | x_n, \theta^{old}) \]

• But:
  \[ P(y_{n,t} = i, y_{n,t+1} = j | x_n, \theta^{old}) = \frac{P(y_{n,t} = i, y_{n,t+1} = j, x_n | \theta^{old})}{P(x_n | \theta^{old})} \]
  \[ = \frac{\alpha_t^i P(x_{n,t+1} | y_{n,t+1} = j) a_{ij} \beta_{t+1}^j}{P(x_n | \theta^{old})} \]

• Which we now how to compute

You should be able to prove the above step.
M-Step

• Now we have all what we need

\[ Q(\theta, \theta^{old}) = \]

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

• Just as before solve for MLE, for ex:

\[ a_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \langle A_{ij,n} \rangle}{\sum_n \sum_{j'} \langle A_{ij',n} \rangle} \]
EM Summary for HMM

• Initialize HMM model parameters

• Repeat
  – E-Step
    • Run forward-backward over every sequence \( (x_n) \)
    • Compute necessary expectations using \( \alpha \) and \( \beta \) (or their normalized versions)
  – M–Step
    • Re-estimate model parameters
      – Simply count and normalize
Final Note about Rescaling

• Recall

\[
P(y_{n,t} = i, y_{n,t+1} = j \mid \mathbf{x}_n, \theta^{old}) = \frac{\alpha^i_t P(x_{n,t+1} \mid y_{n,t+1} = j)a_{ij}\beta^j_{t+1}}{P(x_n \mid \theta^{old})}
\]

\[
\propto \hat{\alpha}^i_t P(x_{n,t+1} \mid y_{n,t+1} = j)a_{ij}\hat{\beta}^j_{t+1}
\]

• Remember the underflow solution

• Same thing here, compute using normalized vectors and then finally normalize P