Introduction to NLP

Session 8

Forward Backward Algorithm

- Forward quantity

$$f(i, x, c) := P(X_1 = x_1, ..., X_i = x_i, Y_i = c)$$

- Backward quantity

$$b(i, x, c) := P(X_{i+1} = x_{i+1}, ..., X_N = x_N | Y_i = c)$$

Efficient Forward Quantity Computations

- If we define $f(i, x, c) := P(X_1 = x_1, ..., X_i = x_i, Y_i = c)$ then, the forward quantity at position i can be written as:

$$f(i, x, c) = P_{\text{emiss}}(x_i|c) \cdot \left(\sum_{\tilde{c} \in \Lambda} P_{\text{trans}}(c|\tilde{c}) \cdot f(i-1, x, \tilde{c}) \right)$$

Forward Quantities Definition

- Let us consider a sequence x with N words
- For every state c let us define:

$$f(1, x, c) = P_{\mathrm{init}}(c_k|\mathrm{start}) \cdot P_{\mathrm{emiss}}(x_1|c_k)$$

- For every position $i = 2$ up to N

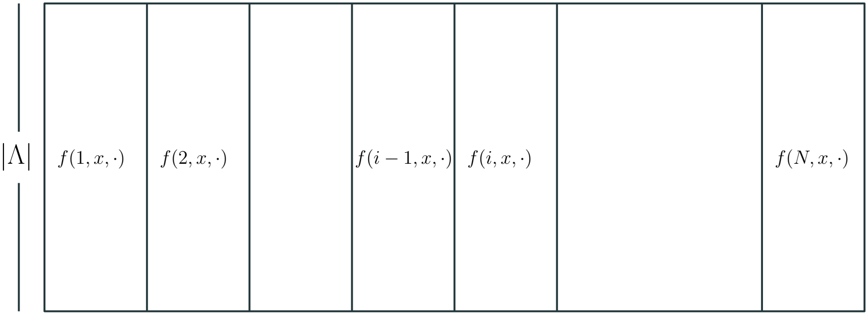
$$f(i, x, c) = P_{\mathrm{emiss}}(x_i|c) \cdot \left(\sum_{\tilde{c} \in \Lambda} P_{\mathrm{trans}}(c|\tilde{c}) \cdot f(i-1, x, \tilde{c}) \right)$$

- For position $N + 1$

$$f(N+1, \mathrm{stop}) = \sum_{c_l \in \Lambda} P_{\mathrm{final}}(\mathrm{stop}|c) \cdot f(N, x, c)$$

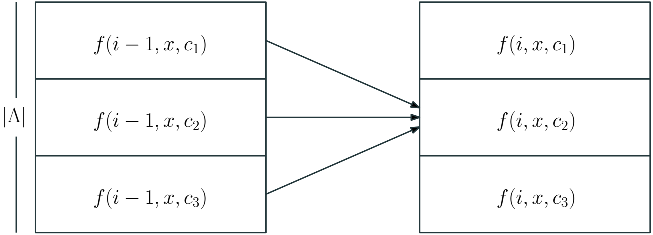
Forward Quantities

$$f(i, x, c) = P_{\text{emiss}}(x_i | c) \cdot \left(\sum_{\tilde{c} \in \Lambda} P_{\text{trans}}(c | \tilde{c}) \cdot f(i - 1, x, \tilde{c}) \right)$$

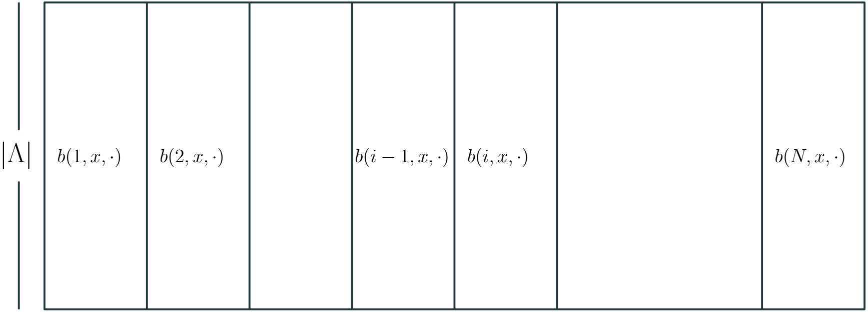


Forward Quantities

$$f(i, x, c) = P_{\text{emiss}}(x_i|c) \cdot \left(\sum_{\tilde{c} \in \Lambda} P_{\text{trans}}(c|\tilde{c}) \cdot f(i-1, x, \tilde{c}) \right)$$

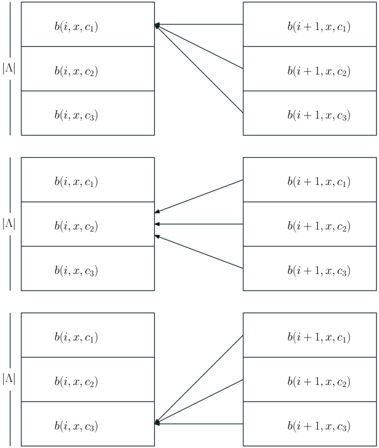


$$b(i, x, c) := \sum_{\tilde{c} \in \Lambda} P_{\text{trans}}(\tilde{c}|c) \cdot b(i+1, x, \tilde{c}) \cdot P_{\text{emiss}}(x_{i+1}|\tilde{c})$$



Backward Quantities

$$b(i, x, c) := \sum_{\tilde{c} \in \Lambda} P_{\text{trans}}(\tilde{c}|c) \cdot b(i+1, x, \tilde{c}) \cdot P_{\text{emiss}}(x_{i+1}|\tilde{c})$$



Forward Backward Algorithm

- Proposition I: $P(X = x, Y_i = c) = f(i, x, c) \cdot b(i, x, c)$
- Proposition II: $P(X = x) = \sum_{c \in \Lambda} f(i, x, c) \cdot b(i, x, c)$
- Proposition III: $P(Y_i = c | X = x) = \frac{f(i,x,c) \cdot b(i,x,c)}{P(X=x)}$
- Proposition IV (Posterior decoding):

$$y^* = \arg\max_{c \in \Lambda} P(Y_i = c | X_1 = x_1, ..., X_N = x_N) = \arg\max_{c \in \Lambda} f(i, x, c) \cdot b(i, x, c)$$

Proposition I

- Proposition I: $P(X = x, Y_i = c) = f(i, x, c) \cdot b(i, x, c)$
- Remember that we defined

$$b(i, x, c) := P(X_{i+1} = x_{i+1}, ..., X_N = x_N | Y_i = c)$$
$$f(i, x, c) := P(X_1 = x_1, ..., X_i = x_i, Y_i = c)$$

- Therefore

$$P(X = x, Y_i = c) = P(X_1 = x_1, ..., X_N = x_N, Y_i = c) =$$
$$P(X_1 = x_1, ..., X_i = x_i, ..., X_N = x_n, Y_i = c) =$$
$$P(X_1 = x_1, ..., X_i = x_i | X_{i+1} = x_{i+1}, ..., X_N = x_N, Y_i = c) \cdot P(X_{i+1} = x_{i+1}, ..., X_N = x_N, Y_i = c) =$$
$$P(X_1 = x_1, ..., X_i = x_i | Y_i = c) \cdot P(X_{i+1} = x_{i+1}, ..., X_N = x_1, Y_i = c) =$$
$$b(i, x, c) \cdot f(i, x, c)$$

Proposition II

- Proposition II: $P(X = x) = \sum_{c \in \Lambda} f(i, x, c) \cdot b(i, x, c)$

$$P(X = x) = \sum_{c \in \Lambda} P(X = x, Y_i = c) \underset{\text{(Prop.I)}}{=} \sum_{c \in \Lambda} f(i, x, c) \cdot b(i, x, c)$$

Proposition III

- Proposition III: $P(Y_i = c|X = x) = \frac{f(i,x,c) \cdot b(i,x,c)}{P(X=x)}$

$$P(Y_i = c|X = x) := \frac{P(Y_i = c, X = x)}{P(X = x)} \underset{\text{(Prop.I)}}{=} \frac{f(i,x,c) \cdot b(i,x,c)}{P(X = x)}$$

Proposition IV

- Proposition IV (Posterior decoding):

$$y^* = \arg\max_{c \in \Lambda} P(Y_i = c | X_1 = x_1, ..., X_N = x_N) = \arg\max_{c \in \Lambda} f(i, x, c) \cdot b(i, x, c)$$

- Proposition III tells us $P(Y_i = c | X = x) = \frac{f(i,x,c) \cdot b(i,x,c)}{P(X=x)}$
- Therefore

$$\begin{aligned} y^* = & \arg\max_{c \in \Lambda} P(Y_i = c | X_1 = x_1, ..., X_N = x_N) = \\ & \arg\max_{c \in \Lambda} \frac{f(i, x, c) \cdot b(i, x, c)}{P(X = x)} = \\ & \frac{1}{P(X = x)} \arg\max_{c \in \Lambda} f(i, x, c) \cdot b(i, x, c) \end{aligned}$$

Forward Backward Algorithm

```
forward_backward(x, Λ, P_init, P_trans, P_final, P_emiss)
    # Forward pass
    for c_k ∈ Λ:
        forward(1, x, c_k) = P_init(c_k|start) · P_emiss(x_1|c_k)
    for i = 1 to N:
        for c̃ ∈ Λ:
            forward(i, x, c̃) = (Σ_{c_k ∈ Λ} P_trans(c̃|c_k) · forward(i − 1, x, c_k)) P_emiss(x_i|c̃)

    # Backward pass
    for c_k ∈ Λ:
        backward(N, x, c_k) = P_final(stop|c_k)
    for i = N − 1 to 1:
        for c̃ ∈ Λ:
            backward(i, x, c̃) = (Σ_{c_k ∈ Λ} P_trans(c_k|c̃) · backward(i + 1, x, c_k)) P_emiss(x_{i+1}|c_k)
```

Viterbi Decoding

- So far we have seen how to perform posterior decoding:

$$y_i^* = \arg\max_{y_i \in \Lambda} P(Y_i = y_i | X_1 = x_1, ..., X_N = x_N)$$

- We will define Viterbi decoding as

$$y* = \arg\max_{y \in \Lambda} P(Y_1 = y_1, ..., Y_N = y_N | X_1 = x_1, ..., X_N = x_N)$$

- Viterbi decoding is very similar to the forward pass of the FB algorithm
- It makes use of the same trellis structure to efficiently represent the exponential number of sequences without prohibitive cost
- The only difference from the FB algorithm is the recursion, that takes the maximum instead of summing over all possible hidden states

$$\mathrm{viterbi}(i, x, c) = \max_{y_1, ..., y_i} P(Y_1 = y_1, ..., Y_i = y_i, X_1 = x_1, ..., X_i = x_i)$$

Viterbi Quantities

- For every state c let us define:

$$\mathrm{viterbi}(1, x, c) = P_{\mathrm{init}}(c|\mathrm{start}) \cdot P_{\mathrm{emiss}}(x_1|c)$$

- For every position $i = 2$ up to $N - 1$:

$$\mathrm{viterbi}(i, x, c) = P_{\mathrm{emiss}}(x_i|c) \cdot \max_{\tilde{c} \in \Lambda}(P_{\mathrm{trans}}(c|\tilde{c}) \cdot \mathrm{viterbi}(i-1, x, \tilde{c}))$$

- For every state at the last position

$$\mathrm{viterbi}(N, x, c) = P_{\mathrm{emiss}}(x_N|c) \cdot \max_{\tilde{c} \in \Lambda}(P_{\mathrm{trans}}(c|\tilde{c}) \cdot \mathrm{viterbi}(N-1, x, \tilde{c}))$$

Viterbi Quantities

- We can define the Viterbi quantity at the stop position as:

$$\mathrm{viterbi}(N+1, x, \mathrm{stop}) = \max_{c \in \Lambda} P_{\mathrm{final}}(\mathrm{stop}|c) \cdot \mathrm{viterbi}(N, x, c)$$

- Using the recurrence rule:

$$\max_{y \in \Lambda^N} P(X = x, Y = y) = \max_{c \in \Lambda} P_{\mathrm{final}}(\mathrm{stop}|c) \cdot \mathrm{viterbi}(N, x, c)$$

- The Viterbi algorithm tells us:

$$y* = \mathrm{viterbi}(N+1, x, y_i) = \max_{y_1, \ldots, y_N} P(Y_1 = y_1, ..., Y_N = y_N, X_1 = x_1, ..., X_i = x_N)$$

Demonstration Viterbi at Pos. N+1 is the Max. Probability

Show that $\max_{\boldsymbol{y}\in\Lambda^{\boldsymbol{N}}} P(X=x, Y=y) = \max_{\boldsymbol{c}\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c}) \cdot \text{viterbi}(\boldsymbol{N}, x, \boldsymbol{c}) := \text{viterbi}(\boldsymbol{N}+\mathbf{1}, x, \text{stop})$

$$\text{viterbi}(\boldsymbol{N}+\mathbf{1}, x, \boldsymbol{c}) = \max_{\boldsymbol{c}\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c}) \cdot \text{viterbi}(\boldsymbol{N}, x, \boldsymbol{c}) =$$

$$\max_{\boldsymbol{c}\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c}) \cdot \left(P_{\text{emiss}}(x_{\boldsymbol{N}}|\boldsymbol{c}) \cdot \max_{\boldsymbol{c}'\in\Lambda}(P_{\text{trans}}(\boldsymbol{c}|\boldsymbol{c}') \cdot \text{viterbi}(\boldsymbol{N}-\mathbf{1}, x, \boldsymbol{c}'))\right) =$$

$$\max_{\boldsymbol{c},\boldsymbol{c}'\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c}) \cdot \left(P_{\text{emiss}}(x_{\boldsymbol{N}}|\boldsymbol{c}) \cdot P_{\text{trans}}(\boldsymbol{c}|\boldsymbol{c}') \cdot \text{viterbi}(\boldsymbol{N}-\mathbf{1}, x, \boldsymbol{c}')\right) =$$

$$\max_{\boldsymbol{c},\boldsymbol{c}'\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c}) \cdot (P_{\text{emiss}}(x_{\boldsymbol{N}}|\boldsymbol{c}) \cdot P_{\text{trans}}(\boldsymbol{c}|\boldsymbol{c}') \cdot P_{\text{emiss}}(x_{\boldsymbol{N}-\mathbf{1}}|\boldsymbol{c}) \cdot \max_{\boldsymbol{c}''\in\Lambda}(P_{\text{trans}}(\boldsymbol{c}'|\boldsymbol{c}'') \cdot \text{viterbi}(\boldsymbol{N}-\mathbf{2}, x, \boldsymbol{c}'')) =$$

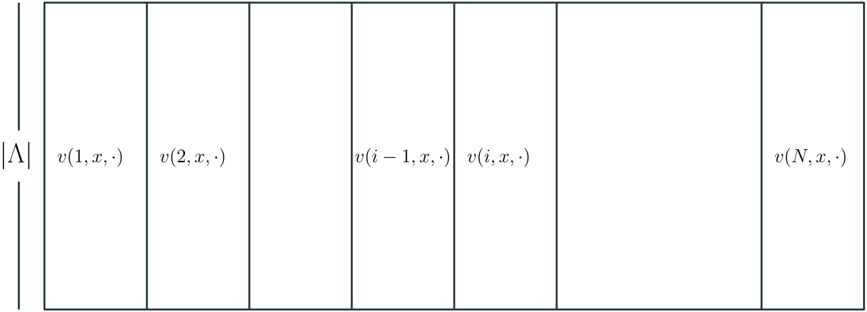
$$\max_{\boldsymbol{c},\boldsymbol{c}',\boldsymbol{c}''\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c}) \cdot (P_{\text{emiss}}(x_{\boldsymbol{N}}|\boldsymbol{c}) \cdot P_{\text{trans}}(\boldsymbol{c}|\boldsymbol{c}') \cdot P_{\text{emiss}}(x_{\boldsymbol{N}-\mathbf{1}}|\boldsymbol{c}) \cdot P_{\text{trans}}(\boldsymbol{c}'|\boldsymbol{c}'') \cdot \text{viterbi}(\boldsymbol{N}-\mathbf{2}, x, \boldsymbol{c}'')) =$$

$\ldots$

$$\max_{\boldsymbol{c_1},\ldots,\boldsymbol{c_N}\in\Lambda} P_{\text{final}}(\text{stop}|\boldsymbol{c_N}) \cdot \prod_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}} P_{\text{emiss}}(x_{\boldsymbol{i}}|\boldsymbol{c_i}) \cdot \prod_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}-\mathbf{1}} P_{\text{trans}}(\boldsymbol{c_{i+1}}|\boldsymbol{c_i}) \cdot P_{\text{init}}(y_{\mathbf{1}}|\text{start}) =$$

$$\max_{\boldsymbol{c_1},\ldots,\boldsymbol{c_N}\in\Lambda} P(X=x, Y=y)$$

Viterbi Algorithm Diagram



Viterbi Algorithm Forwards

- Previous recurrence finds the hidden state sequence y^* with highest probability $P(X = x, Y = y^*)$
- To sum up, we have seen that if we consider the quantities:

$$\mathrm{viterbi}(i, x, y_i) = \max_{y_1,\ldots,y_{i-1}} P(Y_1 = y_1, ..., Y_i = y_i, X_1 = x_1, ..., X_i = x_i)$$

- We can compute $\mathrm{viterbi}(i, x, c)$ using $\mathrm{viterbi}(i-1, x, \cdot)$

$$\mathrm{viterbi}(i, x, c) = P_{\mathrm{emiss}}(x_i|c) \cdot \max_{\tilde{c}\in\Lambda}(P_{\mathrm{trans}}(c|\tilde{c}) \cdot \mathrm{viterbi}(i-1, x, \tilde{c}))$$

- Moreover

$$\mathrm{viterbi}(N+1, x, \mathrm{stop}) = \max_{c_l\in\Lambda} P_{\mathrm{final}}(\mathrm{stop}|c_l) \cdot \mathrm{viterbi}(N, x, c_l) = \max_{c_1,\ldots,c_N\in\Lambda} P(X = x, Y = y)$$

Viterbi Algorithm Backwards

- Once the Viterbi value at position N is computed the algorithm can backtrack using the following recurrence:

$$\mathrm{backtrack}(N+1, x, \mathrm{stop}) = \arg\max_{c_l \in \Lambda} P_{\mathrm{final}}(\mathrm{stop}|c_l) \cdot \mathrm{viterbi}(N, x, c_l)$$

$$\mathrm{backtrack(i, x, c)} = \arg\max_{\tilde{c} \in \Lambda}(P_{\mathrm{trans}}(c|\tilde{c}) \cdot \mathrm{viterbi}(i-1, x, \tilde{c}))$$

- To do this we need to keep track of the backtrack quantities when we compute the viterbi quantities

Decoding with the Viterbi Quantities

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4e-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24

Decoding with the Viterbi Quantities

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4e-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24

Full Viterbi Algorithm

```
viterbi(x,Λ,P_init,P_trans,P_final,P_emiss)
    # Forward pass
    for c_k ∈ Λ
        viterbi(1, x, c_k) = P_init(c_k|start) · P_emiss(x_1|c_k)
    for i = 2 to N:
        for c_k ∈ Λ:
            viterbi(i, x, c_k) = (max_{c_l∈Λ} P_trans(c_k|c_l) · viterbi(i − 1, x, c_l)) · P_emiss(x_i|c_k)
            backtrack(i, x, c_k) = (arg max_{c_l∈Λ} P_trans(c_k|c_l) · viterbi(i − 1, x, c_l))
    viterbi(N + 1, x, stop) := max_{c_l∈Λ} P_final(stop|c_l) · viterbi(N, x, c_l)
    backtrack(N + 1, x, stop) = arg max_{c_l∈Λ} P_final(stop|c_l) · viterbi(N, x, c_l)

    # Backward pass
    ŷ_N = backtrack(N + 1, x, stop)
    for i = N − 1 to 1:
        ŷ_i = backtrack(i + 1, x, ŷ_{i+1})
```

Viterbi Summary

- Let N be the length of the sequence and m the number of states
- Memory:
    - The Viterbi table contains $N \times m$ numbers
    - The Backtrack table contains $N \times m$ numbers
- Runtime:
    - Each cell in the table requires $O(m)$ operations
    - Total runtime is $O(N \cdot m^2)$

Viterbi vs Forward Backward

- Is Viterbi decoding better than posterior decoding?
- Imagine the game: given a sequence x and an HMM
- G1) All or nothing Cost (Viterbi decoding)
  - Pay 10$ if you make a single error
  - You care a lot about the whole label sequence coherency
- G2) Hamming Cost (Posterior decoding)
  - Pay 1$ for every error in your decoding
  - You don't care much about the whole label sequence coherency