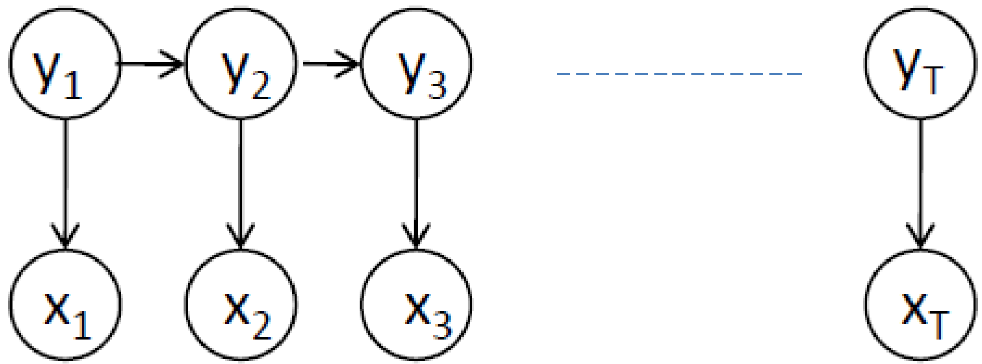
HMM and Neural Network

Xi Chen
(HMM Modified Based
on Amr's Recitation)

HMM



$y_t \in \{1, \ldots, K\}$

$x_t \in \{1, \ldots, M\}$

Factorization

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

		Short hand	# of Paramters
Initial State	$p(y_1)$	$\pi_i = p(y_1 = i)$	K-1
Transition	$p(y_{t+1}\|y_t)$	$a_{ij} = p(y_{t+1} = j\|y_t = i)$	K*(K-1)
Emission	$p(x_t\|y_t)$	$b_{io} = p(x_t = o\|y_t = i)$	K*(M-1)

Tasks

- Inference (known parameters):
- – MAP: $\mathrm{argmax}_i P(y_t = i|\mathbf{x})$
- – Veterbi: $\mathrm{argmax}_{y_1,\ldots,y_T} p(y_1,\ldots,y_T|x_1,\ldots,x_T)$

- Learning (learn parameters):
- – Fully Observed Data: Count and Normalize
- – Partially Observed Data: EM

Inference MAP (given the parameters)

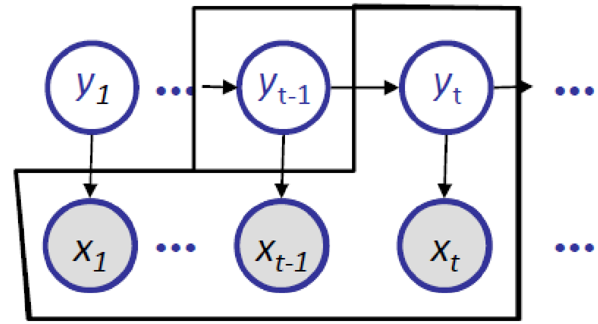
- Find: $\text{argmax}_i P(y_t = i|\mathbf{x})$

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

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

Compute: $\{\alpha_1, \alpha_2, \ldots, \alpha_T\}$

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



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

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

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

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

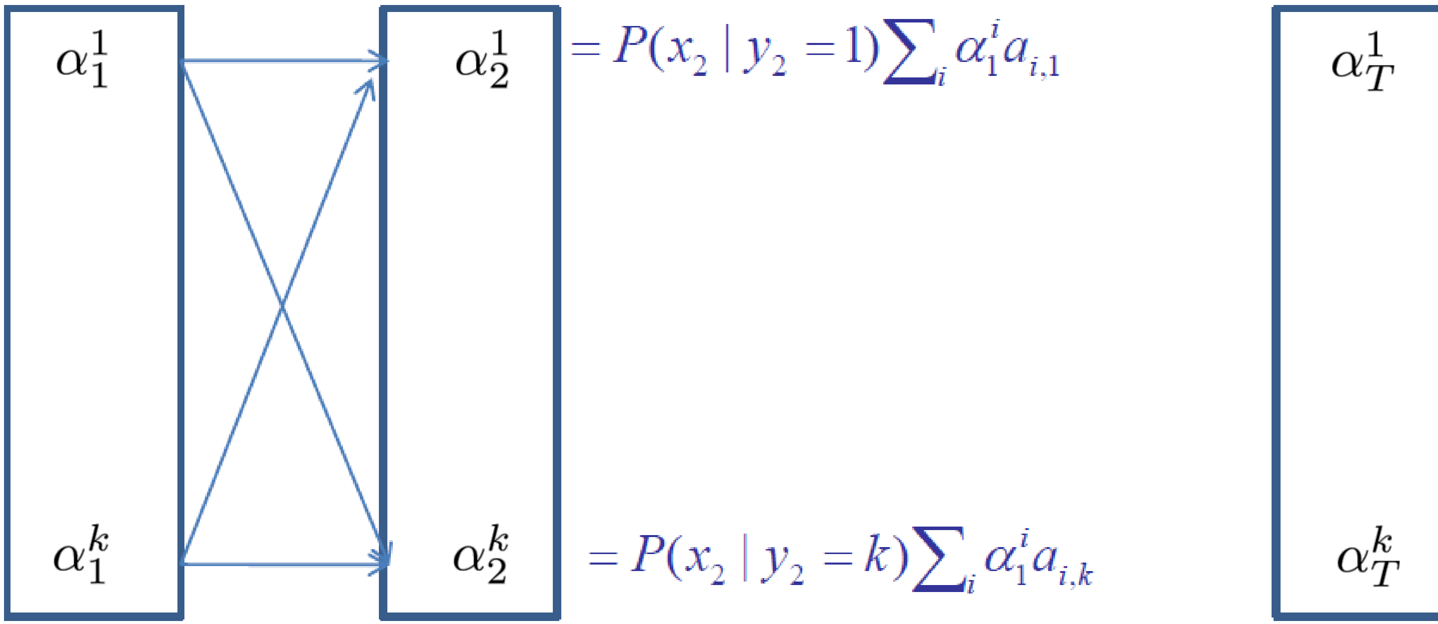
$$= P(x_t \mid y_t = k) \sum_i P(x_1,\ldots,x_{t-1},y_{t-1} = i) P(y_t = k \mid y_{t-1} = i)$$

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

Trick: add a variable and marginalize over it to enable the recursion

Forward Algorithm

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

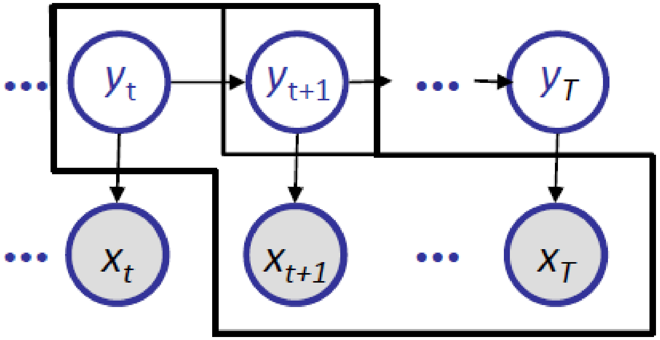


$$\alpha_1^k = P(x_1 \mid y_1 = k)\pi_k$$

$$p(x_1, \ldots, x_T) = \sum_{y_T} p(x_1, \ldots, x_T, y_T) = \sum_{i=1}^{K} p(x_1, \ldots, x_T, y_T = i) = \sum_{i=1}^{K} \alpha_T^i$$

Compute: $\beta_1, \beta_2, \ldots, \beta_T$

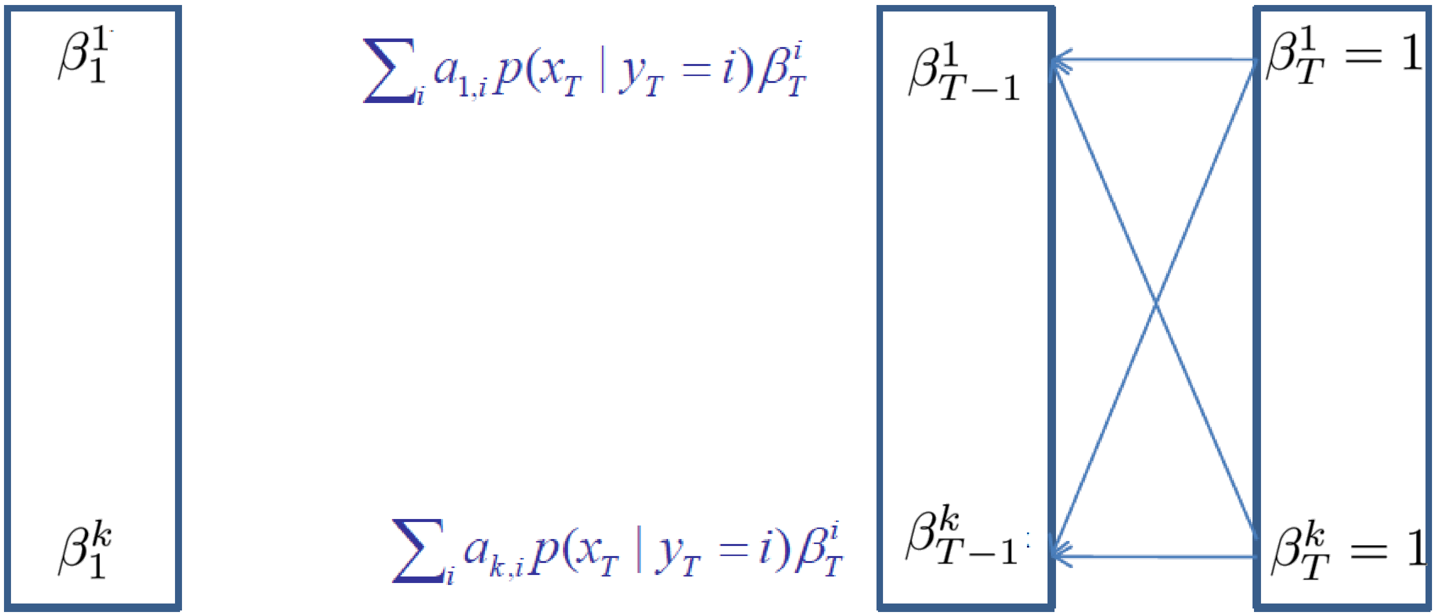
$$\beta_t^i = p(x_{t+1}, \ldots, x_T | y_t = i)$$
$$\beta_{t+1}^j = p(x_{t+2}, \ldots, x_T | y_{t+1} = j)$$



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

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

Backward Algorithm



Viterbi Algorithm

- Find the globally maximal posterior sequence:
- Goal:

$$\text{argmax}_{y_1,\ldots,y_T} p(y_1,\ldots,y_T|x_1,\ldots,x_T) \propto p(y_1,\ldots,y_T,x_1,\ldots,x_T)$$

$$V_t^k = \text{argmax}_{\{y_1,\ldots,y_{t-1}\}} p(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t=k)$$

$$V_1^k = p(y_1=k,x_1) = \alpha_1^k$$

- Maximal Probability of ending in *state k* at time *t* where we maximize over $\{y_1,\ldots,y_{t-1}\}$

Viterbi Algorithm

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

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

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

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

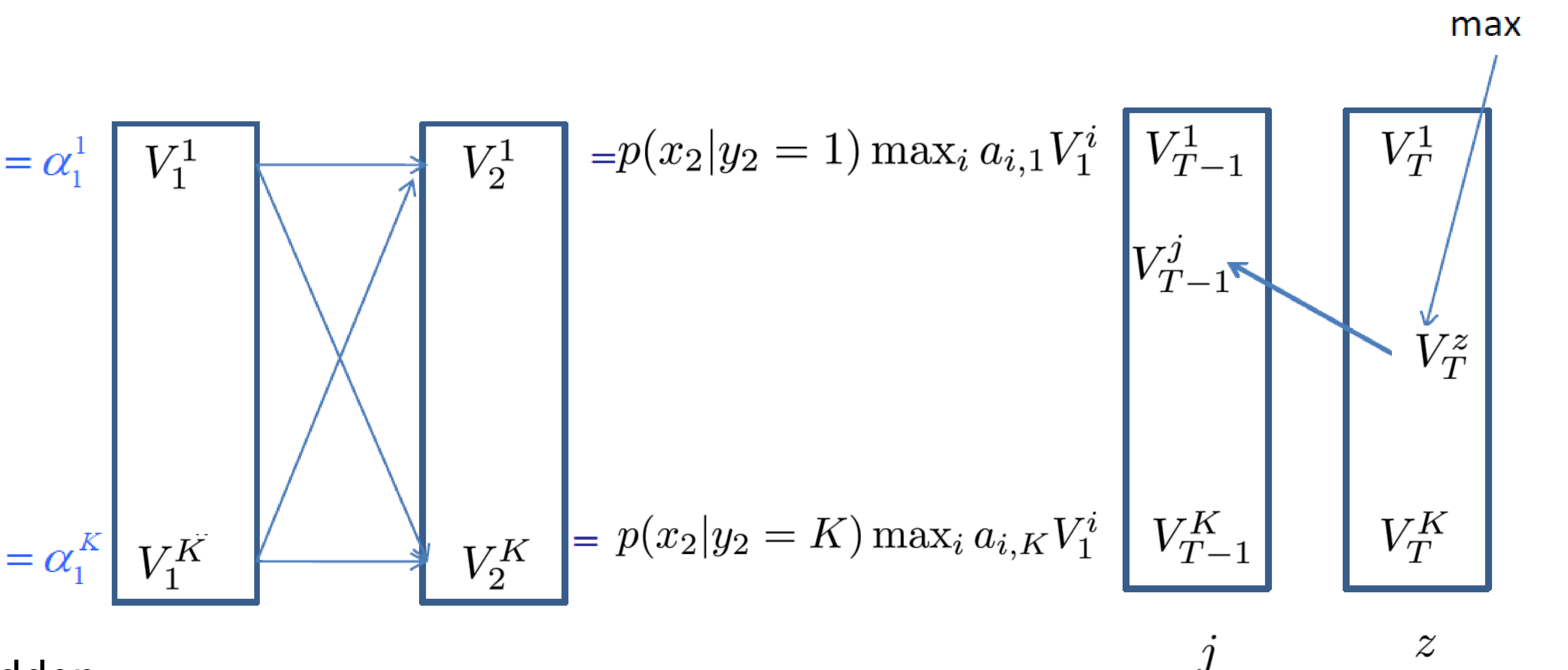
$$= \max_{i} P(x_{t+1,} \mid y_{t+1}=k)a_{i,k}V_t^i$$

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

Scaling: $\log(V_{t+1}^k) = \log p(x_{t+1}|y_{t+1}=k) + \max_i \left(\log(a_{i,k}) + \log(V_{t-1}^i)\right)$

Keeping track of i

Viterbi Algorithm



Hidden State:

$$V_T^z = p(x_T|y_T = z)\max_i a_{i,z}V_{T-1}^i$$

$$j = \text{argmax}_i a_{i,z}V_{T-1}^i$$

Tasks

- Inference (known parameters):
- – MAP: $\text{argmax}_i P(y_t = i|\mathbf{x})$
- – Veterbi: $\text{argmax}_{y_1,\ldots,y_T} p(y_1,\ldots,y_T|x_1,\ldots,x_T)$

- Learning (learn parameters):
- – Fully Observed Model: count and normalize
- – No Observed Hidden State: EM

Learning

- Fully Observed Data $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$
- Initial State: $p(y_{n1}) \sim \text{Multinomial}(\pi_1, \ldots, \pi_K)$

$$p(y_{n1}) = \prod_{i=1}^{K} (\pi_i)^{C_{i,n}}, \quad C_{i,n} \in \{0, 1\}, \quad \sum_{i=1}^{K} C_{i,n} = 1$$

- Transition: $a_{ij} = p(y_{t+1} = j | y_t = i)$
- Emission: $b_{io} = p(x_t = o | y_t = i)$

Learning: Fully Observed

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

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

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

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

$C_{i,n} = \delta(y_{n,1} = i)$: if the first hidden state is i

$A_{ij,n} = \sum_{t=1}^{T-1} \delta(y_{t+1} = j, y_t = i)$: counts that state i moves to j in $(\mathbf{x}_n, \mathbf{y}_n)$

$B_{io,n} = \sum_{t=1}^{T} \delta(x_t = o, y_t = i)$: counts that state i emits o in $(\mathbf{x}_n, \mathbf{y}_n)$

Example

Take **y=**1,2,3,1,2   **x=**1,3,5,1,1

Then

Which

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

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

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

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

Learning: Fully Observed

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

- All parameters are decoupled
- Take the gradient w.r.t each parameter and set it to zero
- Simple count and normalization

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

Learning: partially observed

- EM: E-Step

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) = \sum_{n=1}^{N}\sum_{\mathbf{y}_n} p(\mathbf{y}_n \mid \mathbf{x}_n, \boldsymbol{\theta}^{old}) \log p(\mathbf{x}_n, \mathbf{y}_n \mid \boldsymbol{\theta})$$

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

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

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

Learning: partially observed

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

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

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

$$\langle C_{i,n} \rangle = \sum_{y_n} p(\mathbf{y}_n \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}) C_{i,n} = P\left(y_{n,1} = i \mid \mathbf{x_n}, \boldsymbol{\theta}^{old}\right)$$

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

Forward-Backward Algorithm

Learning: partially observed

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

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

$$\alpha_t^i = p(y_t = i, x_1, \ldots, x_t)$$

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

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

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

$$\begin{aligned}
& p(y_t = i, y_{t+1} = j, x_1, \ldots, x_n) \\
= \; & p(y_t = i, x_1, \ldots, x_t) p(y_{t+1} = j, x_{t+1} | y_t = i, x_1, \ldots, x_t) \cdot \\
& p(x_{t+2}, \ldots, x_T | y_{t+1} = j, x_{t+1}, y_t = i, x_1, \ldots, x_t) \\
= \; & \alpha_t^i p(y_{t+1} = j, x_{t+1} | y_t = i) p(x_{t+2}, \ldots, x_T | y_{t+1} = j) \\
= \; & \alpha_t^i p(x_{t+1} | y_{t+1} = j, y_t = i) p(y_{t+1} = j | y_t = i) \beta_{t+1}^j \\
= \; & \alpha_t^i p(x_{t+1} | y_{t+1} = j) a_{ij} \beta_{t+1}^j
\end{aligned}$$

Learning: partially observed

- EM: M-Step

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

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

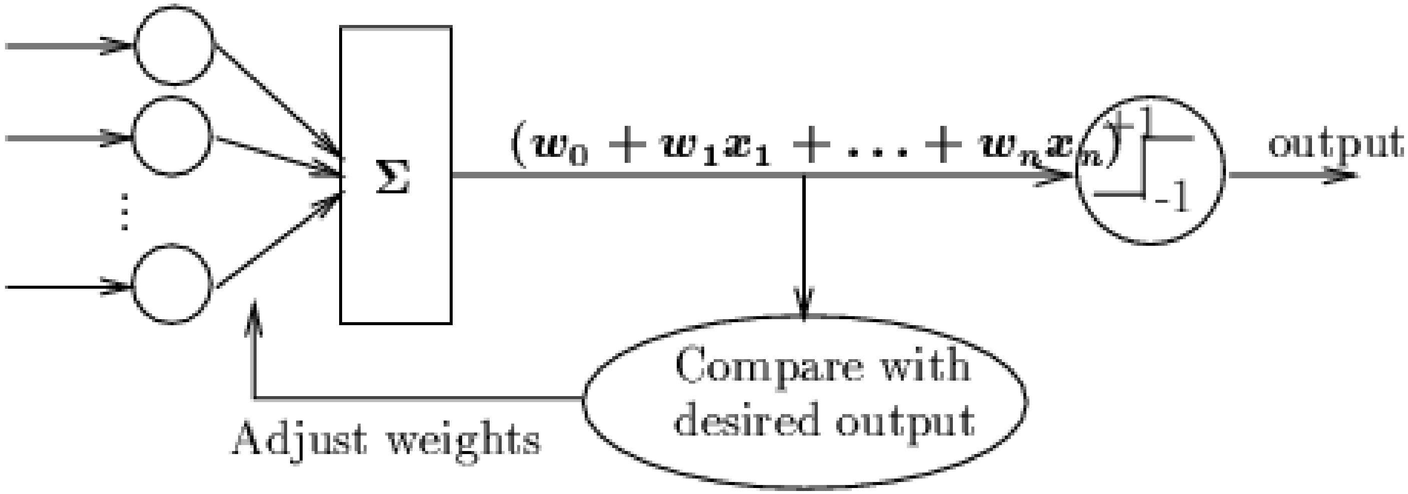
Solve MLE as in fully observed case:

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

EM Summary for HMM Learning

- 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

Neural Network



$$\text{Output } o(\mathbf{x}) = f(w_0 + \sum_{i=1}^{n} w_i x_i)$$

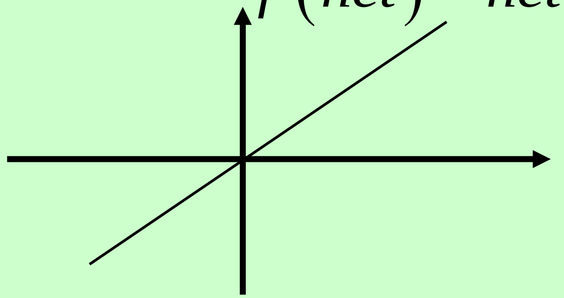
$$net = w_0 + \sum_{i=1}^{n} w_i x_i$$

$$o(\mathbf{x}) = f(net)$$

Activation Function

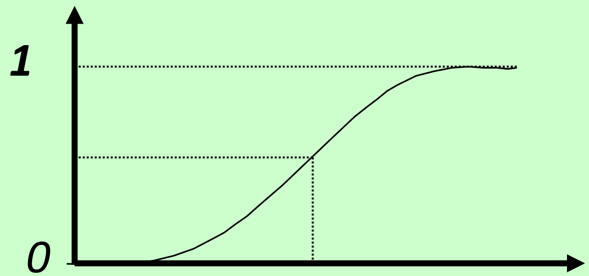
Linear activation

$$f\left(net\right)=net$$



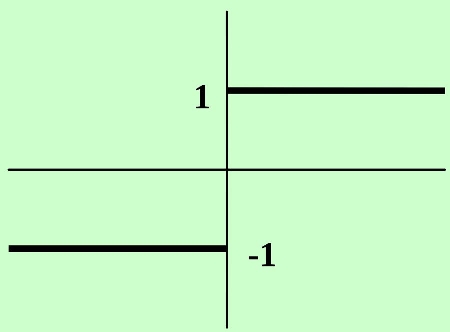
Sigmoid activation

$$f\left(net\right)=\frac{1}{1+e^{-net}}$$



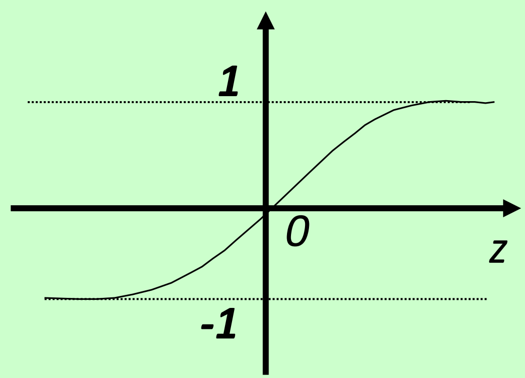
Threshold activation

$$f\left(net\right)=\operatorname{sign}(net)=\begin{cases}1, & if \quad net \ge 0,\\ -1, & if \quad net < 0.\end{cases}$$

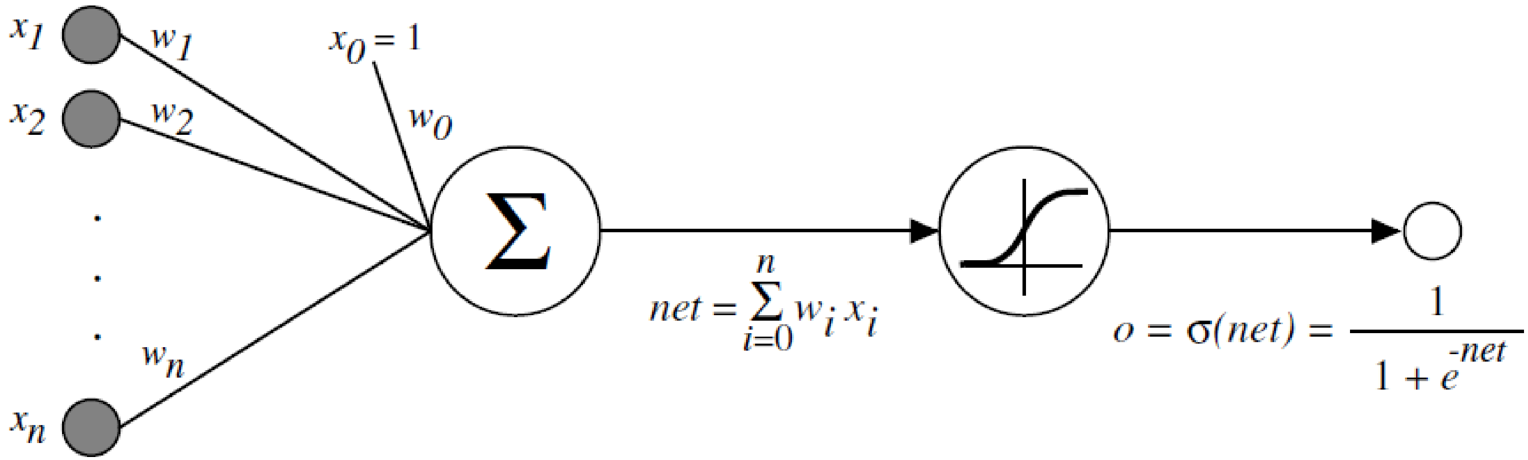


Hyperbolic tangent activation

$$f\left(net\right)=tanh\left(net\right)=\frac{1-e^{-2net}}{1+e^{-2net}}$$



Sigmoid Unit

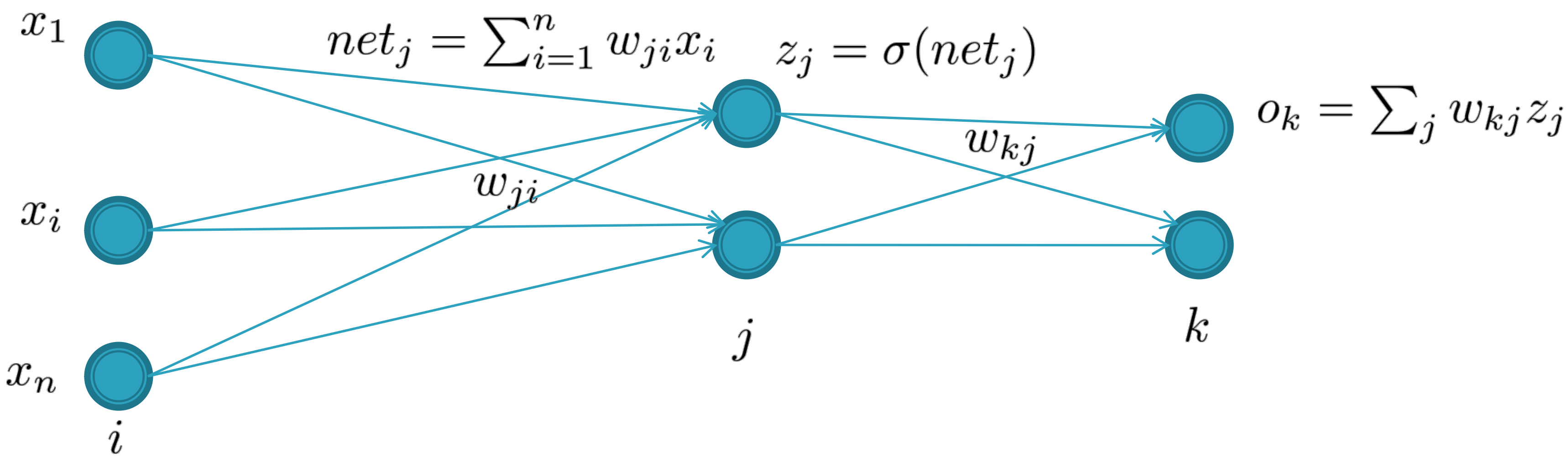


$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

Multilayer Multiple Output NN



$$E_d = \frac{1}{2} \sum_{k=1}^{K} (o_k - t_k)^2$$

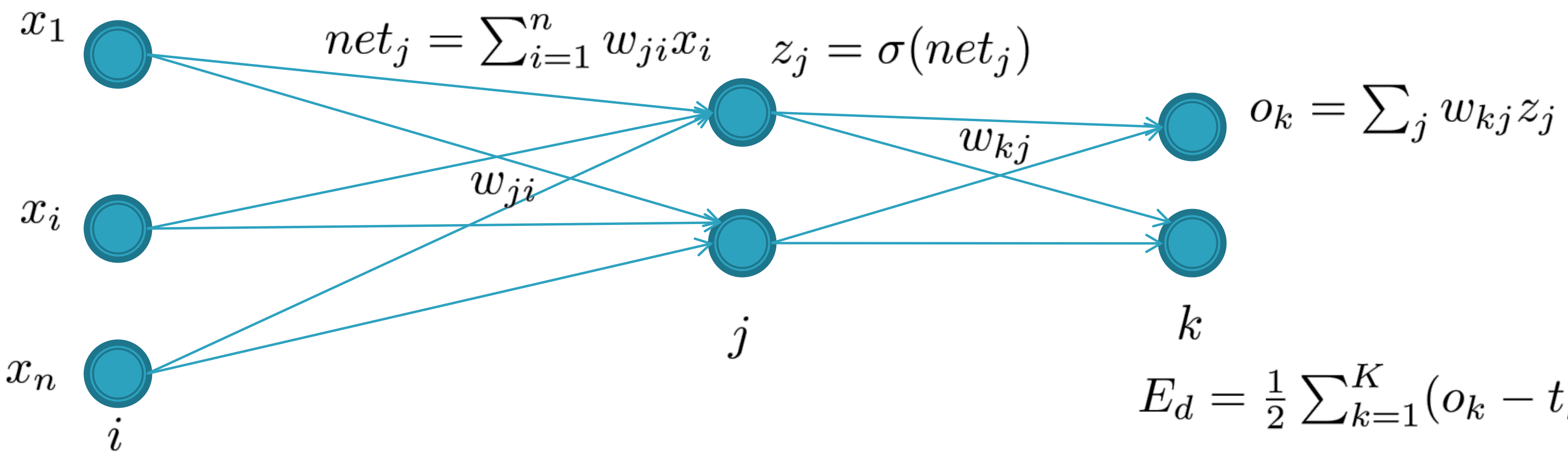
Forward Propagation

$$net_j = \sum_{i=1}^{n} w_{ji} x_i$$

$$z_j = \sigma(net_j)$$

$$o_k = \sum_j w_{kj} z_j$$

Back Propagation



$$E_d = \frac{1}{2} \sum_{k=1}^{K} (o_k - t_k)^2$$

$$\begin{aligned} \frac{\partial E_d}{\partial w_{kj}} &= (o_k - t_k) \frac{\partial o_k}{w_{kj}} \\ &= (o_k - t_k) z_j \end{aligned}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial z_j} \frac{\partial z_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$\frac{\partial net_j}{\partial w_{ji}} = x_i \qquad \frac{\partial z_j}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j))$$

$$\frac{\partial E_d}{\partial z_j} = \sum_k (o_k - t_k) w_{kj}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \sum_k (o_k - t_k) w_{kj} \sigma(net_j)(1 - \sigma(net_j)) x_i$$