

# HMM, MEMM and CRF

*40-957 Special Topics in Artificial Intelligence:  
Probabilistic Graphical Models  
Sharif University of Technology*

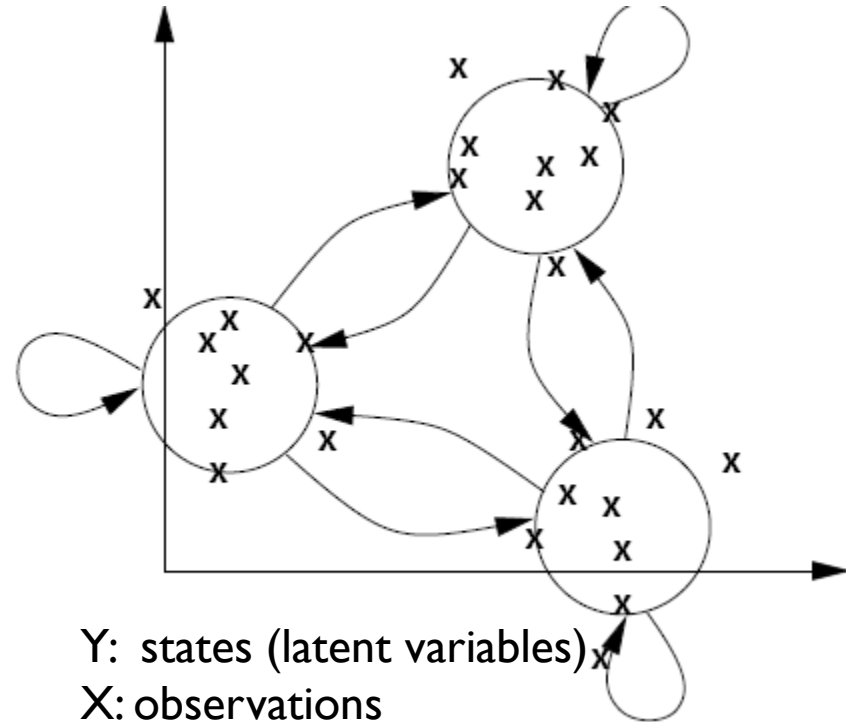
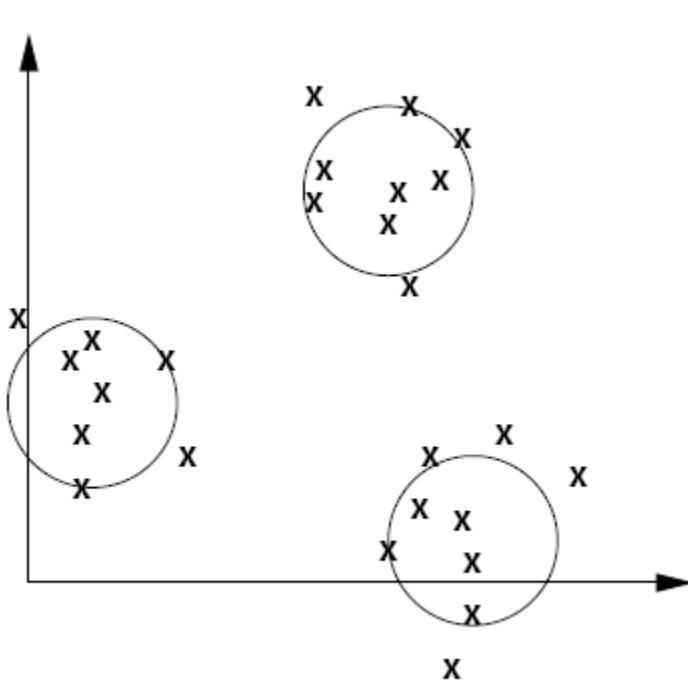
Soleymani  
*Spring 2014*

# Sequence labeling

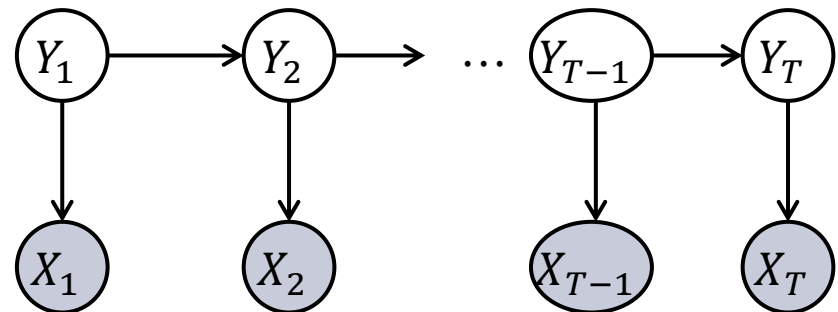
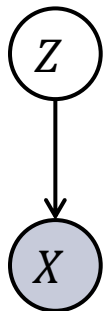
---

- ▶ Taking collectively a set of interrelated instances  $\mathbf{x}_1, \dots, \mathbf{x}_T$  and jointly labeling them
  - ▶ We get as input a sequence of observations  $\mathbf{X} = \mathbf{x}_{1:T}$  and need to label them with some joint label  $\mathbf{Y} = y_{1:T}$

# Generalization of mixture models for sequential data



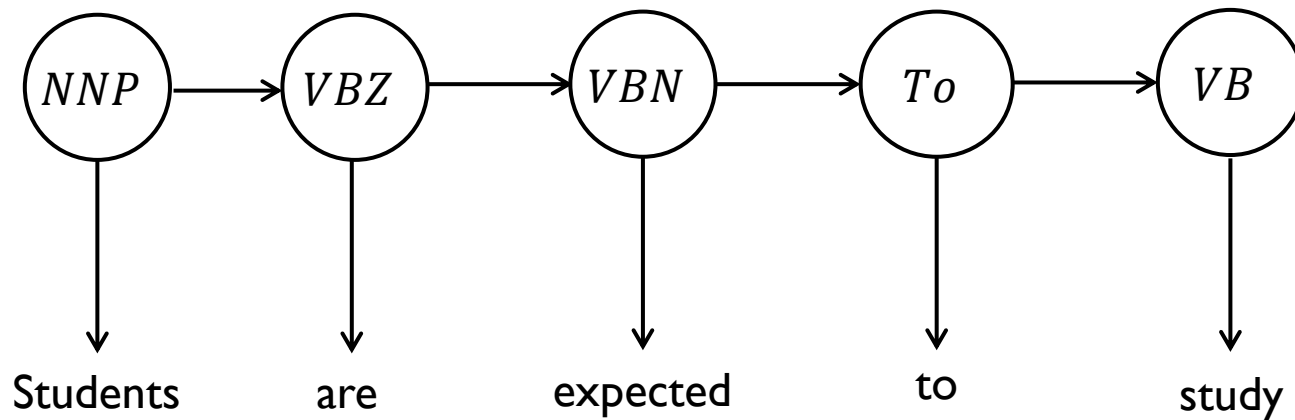
Y: states (latent variables)  
X: observations



# HMM examples

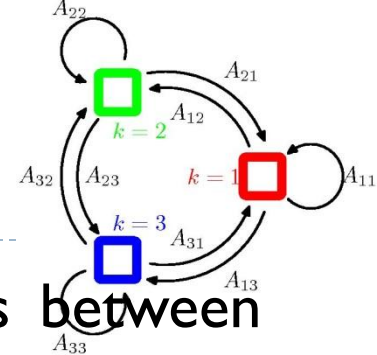
---

## ► Part-of-speech-tagging



## ► Speech recognition

# HMM: probabilistic model



- ▶ **Transitional probabilities:** transition probabilities between states
  - ▶  $A_{ij} \equiv P(Y_t = j | Y_{t-1} = i)$
- ▶ **Initial state distribution:** start probabilities in different states
  - ▶  $\pi_i \equiv P(Y_1 = i)$
- ▶ **Observation model:** Emission probabilities associated with each state
  - ▶  $P(X_t | Y_t, \Phi)$

# HMM: probabilistic model

---

$Y$ : states (latent variables)  
 $X$ : observations

- ▶ **Transitional probabilities:** transition probabilities between states
  - ▶  $P(Y_t | Y_{t-1} = i) \sim \text{Multi}(A_{i1}, \dots, A_{iM}) \forall i \in \text{states}$
- ▶ **Initial state distribution:** start probabilities in different states
  - ▶  $P(Y_1) \sim \text{Multi}(\pi_1, \dots, \pi_M)$
- ▶ **Observation model:** Emission probabilities associated with each state
  - ▶ Discrete observations:  $P(X_t | Y_t = i) \sim \text{Multi}(B_{i,1}, \dots, B_{i,K}) \forall i \in \text{states}$
  - ▶ General:  $P(X_t | Y_t = i) = f(\cdot | \theta_i)$

# Inference problems in sequential data

---

- ▶ **Decoding:**  $\operatorname{argmax}_{y_1, \dots, y_T} P(y_1, \dots, y_T | x_1, \dots, x_T)$
- ▶ **Evaluation**
  - ▶ **Filtering:**  $P(y_t | x_1, \dots, x_t)$
  - ▶ **Smoothing:**  $t' < t, P(y_{t'} | x_1, \dots, x_t)$
  - ▶ **Prediction:**  $t' > t, P(y_{t'} | x_1, \dots, x_t)$

# Some questions

---

- ▶  $P(y_t | x_1, \dots, x_t) = ?$ 
  - ▶ Forward algorithm
- ▶  $P(x_1, \dots, x_T) = ?$ 
  - ▶ Forward algorithm
- ▶  $P(y_t | x_1, \dots, x_T) = ?$ 
  - ▶ Forward-Backward algorithm
- ▶ How do we adjust the HMM parameters:
  - ▶ **Complete data:** each training data includes a state sequence and the corresponding observation sequence
  - ▶ **Incomplete data:** each training data includes only an observation sequence



# Forward algorithm

$$\alpha_t(i) = P(x_1, \dots, x_t, Y_t = i) \quad i, j = 1, \dots, M$$

$$\alpha_t(i) = \sum_j \alpha_{t-1}(j) P(Y_t = i | Y_{t-1} = j) P(x_t | Y_t = i)$$

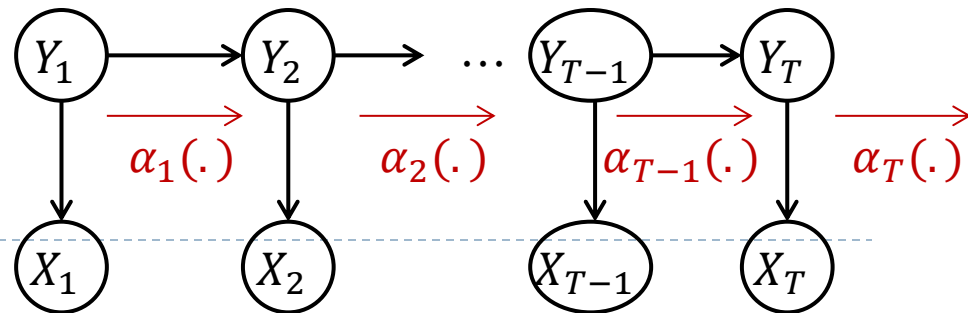
## ► Initialization:

$$\alpha_1(i) = P(x_1, Y_1 = i) = P(x_1 | Y_1 = i) P(Y_1 = i)$$

## ► Iterations: $t = 2$ to $T$

$$\alpha_t(i) = \sum_j \alpha_{t-1}(j) P(Y_t = i | Y_{t-1} = j) P(x_t | Y_t = i)$$

$$\alpha_t(i) = m_{t-1 \rightarrow t}(i)$$



# Backward algorithm

$$\beta_t(i) = m_{t \rightarrow t-1}(i) = P(x_{t+1}, \dots, x_T | Y_t = i) \quad i, j \in \text{states}$$

$$\beta_{t-1}(i) = \sum_j \beta_t(j) P(Y_t = j | Y_{t-1} = i) P(x_t | Y_t = i)$$

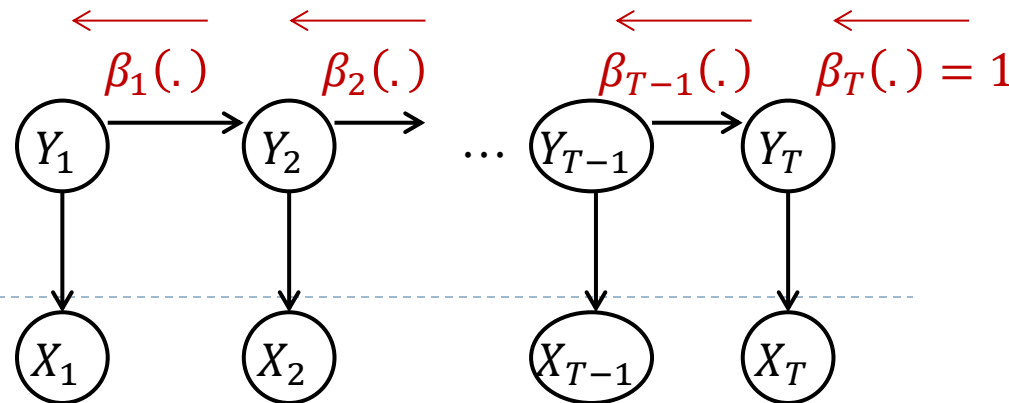
## ► Initialization:

►  $\beta_T(i) = 1$

## ► Iterations: $t = T$ down to 2

►  $\beta_{t-1}(i) = \sum_j \beta_t(j) P(Y_t = j | Y_{t-1} = i) P(x_t | Y_t = i)$

$$\beta_t(i) = m_{t \rightarrow t-1}(i)$$



# Forward-backward algorithm

---

$$\alpha_t(i) \equiv P(x_1, x_2, \dots, x_t, Y_t = i)$$

$$\beta_t(i) \equiv P(x_{t+1}, x_{t+2}, \dots, x_T | Y_t = i)$$

$$\alpha_t(i) = \sum_j \alpha_{t-1}(j) P(Y_t = i | Y_{t-1} = j) P(x_t | Y_t = i)$$

$$\alpha_1(i) = P(x_1, Y_1 = i) = P(x_1 | Y_1 = i) P(Y_1 = i)$$

$$\beta_{t-1}(i) = \sum_j \beta_t(j) P(Y_t = j | Y_{t-1} = i) P(x_t | Y_t = i)$$

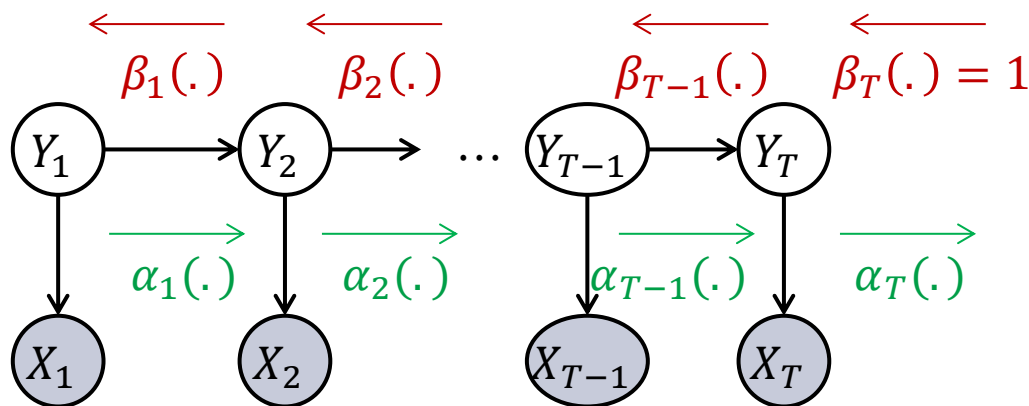
$$\beta_T(i) = 1$$

$$P(x_1, x_2, \dots, x_T) = \sum_i \alpha_T(i) \beta_T(i) = \sum_i \alpha_T(i)$$

$$P(Y_t = i | x_1, x_2, \dots, x_T) = \frac{\alpha_t(i) \beta_t(i)}{\sum_i \alpha_T(i)}$$

# Forward-backward algorithm

- ▶ This will be used for expectation maximization to train a HMM



- ▶ 
$$P(Y_t = i | x_1, \dots, x_T) = \frac{P(x_1, \dots, x_T, Y_t = i)}{P(x_1, \dots, x_T)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^N \alpha_T(j)}$$

# Decoding Problem

---

- ▶ Choose state sequence to maximize the observations:

- ▶  $\operatorname{argmax}_{y_1, \dots, y_t} P(y_1, \dots, y_t | x_1, \dots, x_t)$

- ▶ Viterbi algorithm:

- ▶ Define auxiliary variable  $\delta$ :

- ▶  $\delta_t(i) = \max_{y_1, \dots, y_{t-1}} P(y_1, y_2, \dots, y_{t-1}, Y_t = i, x_1, x_2, \dots, x_t)$

- ▶  $\delta_t(i)$ : probability of the most probable path ending in state  $Y_t = i$

- ▶ Recursive relation:

- ▶  $\delta_t(i) = \max_j (\delta_{t-1}(j) P(Y_t = i | Y_{t-1} = j)) P(x_t | Y_t = i)$

$$\delta_t(i) = \left( \max_{j=1, \dots, N} \delta_{t-1}(j) P(Y_t = i | Y_{t-1} = j) \right) P(x_t | Y_t = i)$$

# Decoding Problem: Viterbi algorithm

---

- ▶ **Initialization**  $i = 1, \dots, M$ 
  - ▶  $\delta_1(i) = P(x_1|Y_1 = i)P(Y_1 = i)$
  - ▶  $\psi_1(i) = 0$
- ▶ **Iterations:**  $t = 2, \dots, T$   $i = 1, \dots, M$ 
  - ▶  $\delta_t(i) = \left( \max_j \delta_{t-1}(j) P(Y_t = i|Y_{t-1} = j) \right) P(x_t|Y_t = i)$
  - ▶  $\psi_t(i) = \underset{j}{\operatorname{argmax}} (\delta_{t-1}(j) P(Y_t = i|Y_{t-1} = j))$
- ▶ **Final computation:**
  - ▶  $P^* = \max_{j=1, \dots, N} \delta_T(j)$
  - ▶  $y_T^* = \underset{j=1, \dots, N}{\operatorname{argmax}} \delta_T(j)$
- ▶ **Traceback state sequence:**  $t = T - 1$  down to  $1$ 
  - ▶  $y_t^* = \psi_{t+1}(y_{t+1}^*)$

# Max-product algorithm

---

$$m_{ji}^{max}(x_i) = \max_{x_j} \left( \phi(x_j) \phi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{kj}^{max}(x_j) \right)$$

$$\delta_t(i) = m_{t-1,t}^{max} \times \phi(x_i)$$

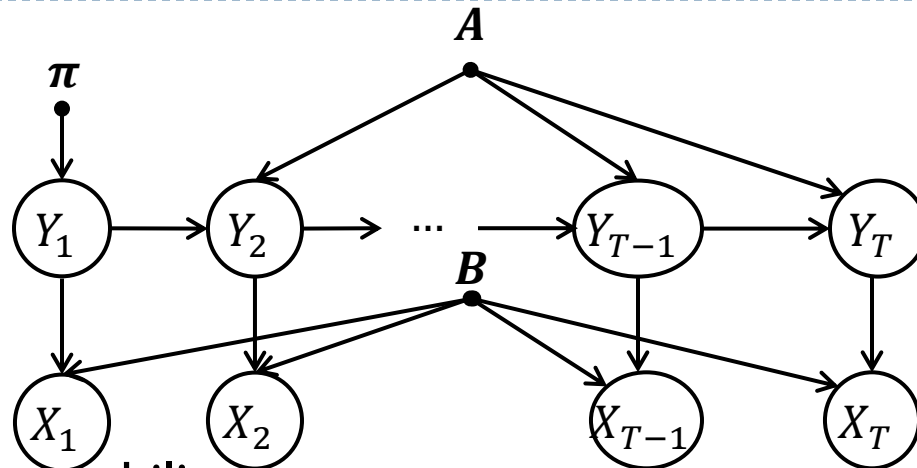
# HMM Learning

---

- ▶ **Supervised learning:** When we have a set of data samples, each of them containing a pair of sequences (one is the observation sequence and the other is the state sequence)
- ▶ **Unsupervised learning:** When we have a set of data samples, each of them containing a sequence of observations



# HMM supervised learning by MLE



- ▶ Initial state probability:

$$\pi_i = P(Y_1 = i), \quad 1 \leq i \leq M$$

- ▶ State transition probability:

$$A_{ji} = P(Y_{t+1} = i | Y_t = j), \quad 1 \leq i, j \leq M$$

- ▶ State transition probability:

$$B_{ik} = P(X_t = k | Y_t = i), \quad 1 \leq k \leq K$$

Discrete  
observations

# HMM: supervised parameter learning by MLE

---

$$P(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^N \left[ P(y_1^{(n)}|\boldsymbol{\pi}) \prod_{t=2}^T P(y_t^{(n)}|y_{t-1}^{(n)}, \mathbf{A}) \prod_{t=1}^T P(\mathbf{x}_t^{(n)}|y_t^{(n)}, \mathbf{B}) \right]$$

$$\hat{A}_{ij} = \frac{\sum_{n=1}^N \sum_{t=2}^T I(y_{t-1}^{(n)} = i, y_t^{(n)} = j)}{\sum_{n=1}^N \sum_{t=2}^T I(y_{t-1}^{(n)} = i)}$$

$$\hat{\pi}_i = \frac{\sum_{n=1}^N I(y_1^{(n)} = i)}{N}$$

$$\hat{B}_{ik} = \frac{\sum_{n=1}^N \sum_{t=1}^T I(y_t^{(n)} = i, x_t^{(n)} = k)}{\sum_{n=1}^N \sum_{t=1}^T I(y_t^{(n)} = i)}$$

Discrete  
observations

# Learning

---

- ▶ Problem: how to construct an HHM given only observations?
- ▶ Find  $\theta = (A, B, \pi)$ , maximizing  $P(x_1, \dots, x_T | \theta)$ 
  - ▶ Incomplete data
    - ▶ EM algorithm

# HMM learning by EM (Baum-Welch)

---

▶  $\boldsymbol{\theta}^{old} = [\boldsymbol{\pi}^{old}, \mathbf{A}^{old}, \boldsymbol{\Phi}^{old}]$

▶ E-Step:

$$i, j = 1, \dots, M$$

▶  $\gamma_{n,t}^i = P(Y_t^{(n)} = i | \mathbf{X}^{(n)}; \boldsymbol{\theta}^{old})$

▶  $\xi_{n,t}^{i,j} = P(Y_{t-1}^{(n)} = i, Y_t^{(n)} = j | \mathbf{x}^{(n)}; \boldsymbol{\theta}^{old})$

▶ M-Step:

$$i, j = 1, \dots, M$$

▶  $\pi_i^{new} = \frac{\sum_{n=1}^N \gamma_{n,1}^i}{N}$

▶  $A_{i,j}^{new} = \frac{\sum_{n=1}^N \sum_{t=2}^T \xi_{n,t}^{i,j}}{\sum_{n=1}^N \sum_{t=1}^{T-1} \gamma_{n,t}^i}$

▶  $B_{i,k}^{new} = \frac{\sum_{n=1}^N \sum_{t=1}^T \gamma_{n,t}^i I(X_t^{(n)} = k)}{\sum_{n=1}^N \sum_{t=1}^T I(X_t^{(n)} = i)}$

Discrete observations

# HMM learning by EM (Baum-Welch)

▶  $\boldsymbol{\theta}^{old} = [\boldsymbol{\pi}^{old}, \mathbf{A}^{old}, \boldsymbol{\Phi}^{old}]$

▶ E-Step:

$$i, j = 1, \dots, M$$

▶  $\gamma_{n,t}^i = P\left(Y_t^{(n)} = k \mid \mathbf{X}^{(n)}; \boldsymbol{\theta}^{old}\right)$

▶  $\xi_{n,t}^{i,j} = P\left(Y_{t-1}^{(n)} = i, Y_t^{(n)} = j \mid \mathbf{x}^{(n)}; \boldsymbol{\theta}^{old}\right)$

▶ M-Step:

$$i, j = 1, \dots, M$$

▶  $\pi_i^{new} = \frac{\sum_{n=1}^N \gamma_{n,1}^i}{N}$

▶  $A_{i,j}^{new} = \frac{\sum_{n=1}^N \sum_{t=2}^T \xi_{n,t}^{i,j}}{\sum_{n=1}^N \sum_{t=1}^{T-1} \gamma_{n,t}^i}$

▶  $\boldsymbol{\mu}_i^{new} = \frac{\sum_{n=1}^N \sum_{t=1}^T \gamma_{n,t}^i \mathbf{x}_t^{(n)}}{\sum_{n=1}^N \sum_{t=1}^T \gamma_{n,t}^i}$

Assumption: Gaussian emission probabilities

$$\boldsymbol{\Sigma}_i^{new} = \frac{\sum_{n=1}^N \sum_{t=1}^T \gamma_{n,t}^i \left(\mathbf{x}_t^{(n)} - \boldsymbol{\mu}_i^{new}\right) \left(\mathbf{x}_t^{(n)} - \boldsymbol{\mu}_i^{new}\right)^T}{\sum_{n=1}^N \sum_{t=1}^T \gamma_{n,t}^i}$$

# Forward-backward algorithm for E-step

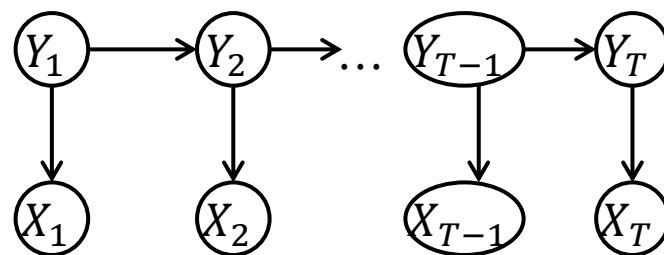
---

$$\begin{aligned} & P(y_{t-1}, y_t | \mathbf{x}_1, \dots, \mathbf{x}_T) \\ &= \frac{P(\mathbf{x}_1, \dots, \mathbf{x}_T | y_{t-1}, y_t) P(y_{t-1}, y_t)}{P(\mathbf{x}_1, \dots, \mathbf{x}_T)} \\ &= \frac{P(\mathbf{x}_1, \dots, \mathbf{x}_{t-1} | y_{t-1}) P(\mathbf{x}_t | y_t) P(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | y_t) P(y_t | y_{t-1}) P(y_{t-1})}{P(\mathbf{x}_1, \dots, \mathbf{x}_T)} \\ &= \frac{\alpha_{t-1}(y_{t-1}) P(\mathbf{x}_t | y_t) P(y_t | y_{t-1}) \beta_t(y_t)}{\sum_{i=1}^M \alpha_T(i)} \end{aligned}$$

# HMM shortcomings

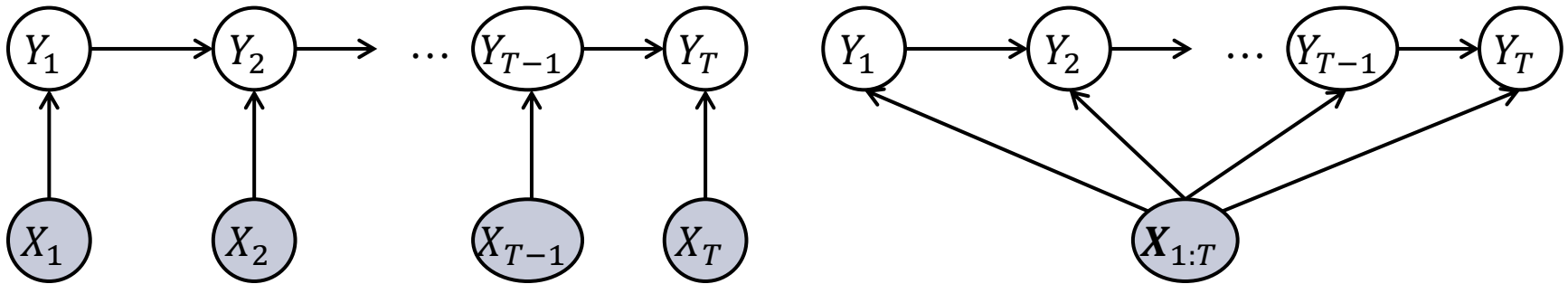
---

- ▶ In modeling the joint distribution  $P(\mathbf{Y}, \mathbf{X})$ , HMM ignores many dependencies between observations  $X_1, \dots, X_T$  (similar to most generative models that need to simplify the structure)



- ▶ In the sequence labeling task, we need to classify an observation sequence using the conditional probability  $P(\mathbf{Y}|\mathbf{X})$ 
  - ▶ However, HMM learns a joint distribution  $P(\mathbf{Y}, \mathbf{X})$  while uses only  $P(\mathbf{Y}|\mathbf{X})$  for labeling

# Maximum Entropy Markov Model (MEMM)



$$P(\mathbf{y}_{1:T}|\mathbf{x}_{1:T}) = P(y_1|\mathbf{x}_{1:T}) \prod_{t=2}^T P(y_t|y_{t-1}, \mathbf{x}_{1:T})$$

$$P(y_t|y_{t-1}, \mathbf{x}_{1:T}) = \frac{\exp\{\mathbf{w}^T \mathbf{f}(y_t, y_{t-1}, \mathbf{x}_{1:T})\}}{\sum_{y_t} \exp\{\mathbf{w}^T \mathbf{f}(y_t, y_{t-1}, \mathbf{x}_{1:T})\}}$$

## ► Discriminative model

- Only models  $P(\mathbf{Y}|\mathbf{X})$  and completely ignores modeling  $P(\mathbf{X})$
- Maximizes the conditional likelihood  $P(\mathcal{D}^Y|\mathcal{D}^X, \boldsymbol{\theta})$

$$Z(y_{t-1}, \mathbf{x}_{1:T}, \mathbf{w}) = \sum_{y_t} \exp\{\mathbf{w}^T \mathbf{f}(y_t, y_{t-1}, \mathbf{x}_{1:T})\}$$



# Feature function

---

- ▶ Feature function  $f(y_t, y_{t-1}, \mathbf{x}_{1:T})$  can take account of relations between both data and label space
  - ▶ However, they are often indicator functions showing absence or presence of a feature
- ▶  $w_i$  captures how closely  $f(y_t, y_{t-1}, \mathbf{x}_{1:T})$  is related with the label

# MEMM disadvantages

---

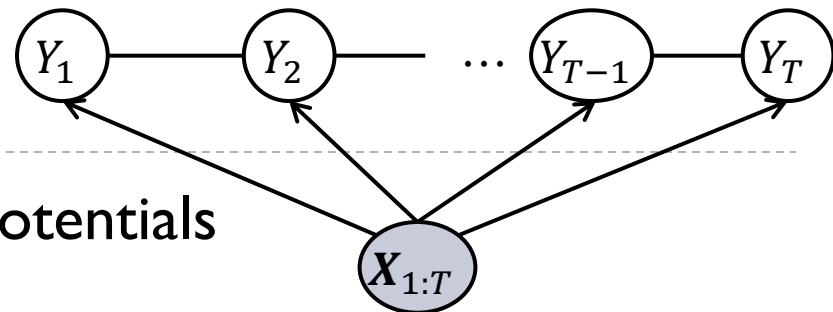
- ▶ The later observation in the sequence has absolutely no effect on the posterior probability of the current state
  - ▶ model does not allow for any smoothing.
  - ▶ The model is incapable of going back and changing its prediction about the earlier observations.
- ▶ The label bias problem
  - ▶ there are cases that a given observation is not useful in predicting the next state of the model.
    - ▶ if a state has a unique out-going transition, the given observation is useless

# Label bias problem in MEMM

---

- ▶ **Label bias problem:** states with fewer arcs are preferred
  - ▶ Preference of states with lower entropy of transitions over others in decoding
  - ▶ MEMMs should probably be avoided in cases where many transitions are close to deterministic
    - ▶ Extreme case: When there is only one outgoing arc, it does not matter what the observation is
- ▶ The source of this problem: Probabilities of outgoing arcs normalized separately for each state
  - ▶ sum of transition probability for any state has to sum to 1
- ▶ Solution: Do not normalize probabilities locally

# From MEMM to CRF



- ▶ From local probabilities to local potentials

$$P(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_{t=1}^T \phi(y_{t-1}, y_t, \mathbf{X})$$
$$= \frac{1}{Z(\mathbf{X}, \mathbf{w})} \prod_{t=1}^T \exp\{\mathbf{w}^T f(y_t, y_{t-1}, \mathbf{x})\}$$

$\mathbf{X} \equiv \mathbf{x}_{1:T}$   
 $\mathbf{Y} \equiv \mathbf{y}_{1:T}$

- ▶ CRF is a discriminative model (like MEMM)
  - ▶ can dependence between each state and the entire observation sequence
  - ▶ uses global normalizer  $Z(\mathbf{X}, \mathbf{w})$  that overcomes the label bias problem of MEMM
- ▶ MEMM use an exponential model for each state while CRF have a single exponential model for the joint probability of the entire label sequence

# CRF: conditional distribution

---

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp \left\{ \sum_{i=1}^T \boldsymbol{\lambda}^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) \right\} \\ &= \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp \left\{ \sum_{i=1}^T \sum_k \lambda_k f_k(y_i, y_{i-1}, \mathbf{x}) \right\} \end{aligned}$$

$$Z(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{\mathbf{y}} \exp \left\{ \sum_{i=1}^T \boldsymbol{\lambda}^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) \right\}$$

# CRF: MAP inference

---

- ▶ Given CRF parameters  $\lambda$ , find the  $\mathbf{y}^*$  that maximizes  $P(\mathbf{y}|\mathbf{x})$ :

$$\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} \exp \left\{ \sum_{i=1}^T \lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) \right\}$$

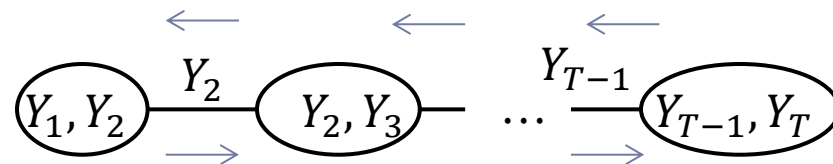
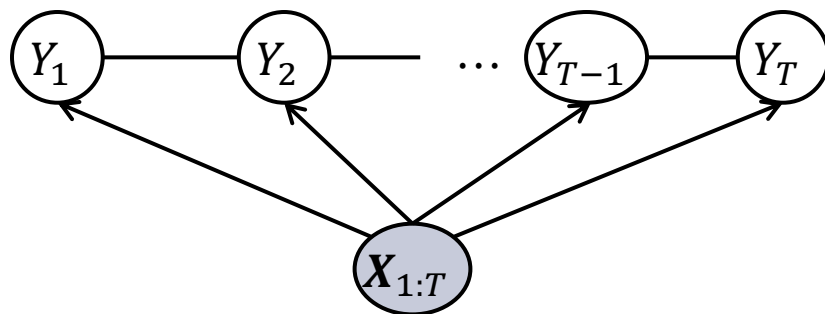
- ▶  $Z(\mathbf{x})$  is not a function of  $\mathbf{y}$  and so has been ignored
- ▶ Max-product algorithm can be used for this MAP inference problem
  - ▶ Same as Viterbi decoding used in HMMs

# CRF: inference

---

## Exact inference for I-D chain CRFs

$$\phi(y_i, y_{i-1}) = \exp\{\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x})\}$$



$$\hat{\theta} = \operatorname{argmax}_{\theta} P(\mathcal{D}^Y | \mathcal{D}^X, \theta)$$

# CRF: learning

$$\lambda^* = \operatorname{argmax}_{\lambda} \prod_{n=1}^N P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, \lambda)$$

$$\prod_{n=1}^N P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, \lambda) = \prod_{n=1}^N \frac{1}{Z(\mathbf{x}^{(n)}, \lambda)} \exp \left\{ \sum_{i=1}^T \lambda^T \mathbf{f}(y_i^{(n)}, y_{i-1}^{(n)}, \mathbf{x}^{(n)}) \right\}$$

$$= \exp \left\{ \sum_{n=1}^N \left( \sum_{i=1}^T \lambda^T \mathbf{f}(y_i^{(n)}, y_{i-1}^{(n)}, \mathbf{x}^{(n)}) - \ln Z(\mathbf{x}^{(n)}, \lambda) \right) \right\}$$

$$L(\lambda) = \ln \prod_{n=1}^N P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, \lambda)$$

$$\nabla_{\lambda} L(\lambda) = \sum_{n=1}^N \sum_{i=1}^T \left( \mathbf{f}(y_i^{(n)}, y_{i-1}^{(n)}, \mathbf{x}^{(n)}) - \nabla_{\lambda} \ln Z(\mathbf{x}^{(n)}, \lambda) \right)$$

$$\nabla_{\lambda} \ln Z(\mathbf{x}^{(n)}, \lambda) = \sum_{\mathbf{y}} \left( P(\mathbf{y} | \mathbf{x}^{(n)}, \lambda) \sum_{i=1}^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}^{(n)}) \right)$$

Gradient of the log-partition function in an exponential family is the expectation of the sufficient statistics.



# CRF: learning

---

$$\begin{aligned}\nabla_{\lambda} \ln Z(\mathbf{x}^{(n)}, \lambda) &= \sum_{\mathbf{y}} \left( P(\mathbf{y} | \mathbf{x}^{(n)}, \lambda) \sum_{i=1}^T f(y_i, y_{i-1}, \mathbf{x}^{(n)}) \right) \\ &= \sum_{i=1}^T \sum_{\mathbf{y}} P(\mathbf{y} | \mathbf{x}^{(n)}, \lambda) f(y_i, y_{i-1}, \mathbf{x}^{(n)}) \\ &= \sum_{i=1}^T \sum_{y_i, y_{i-1}} P(y_i, y_{i-1} | \mathbf{x}^{(n)}, \lambda) f(y_i, y_{i-1}, \mathbf{x})\end{aligned}$$

How do we find the above expectations?

$P(y_i, y_{i-1} | \mathbf{x}^{(n)}, \lambda)$  must be computed for all  $i = 2, \dots, T$

# CRF: learning

Inference to find  $P(y_i, y_{i-1} | \mathbf{x}^{(n)}, \boldsymbol{\lambda})$

---

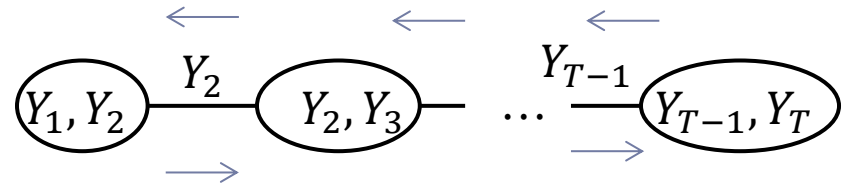
## ► Junction tree algorithm

### ► Initialization of clique potentials:

$$\begin{aligned}\psi(y_i, y_{i-1}) &= \exp\{\boldsymbol{\lambda}^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}^{(n)})\} \\ \phi(y_i) &= 1\end{aligned}$$

### ► After calibration: $\psi^*(y_i, y_{i-1})$

$$P(y_i, y_{i-1} | \mathbf{x}^{(n)}, \boldsymbol{\lambda}) = \frac{\psi^*(y_i, y_{i-1})}{\sum_{y_i, y_{i-1}} \psi^*(y_i, y_{i-1})}$$



# CRF learning: gradient descent

---

$$\lambda^t = \lambda^t + \eta \nabla_{\lambda} L(\lambda^t)$$

$$\nabla_{\lambda} L(\lambda) = \sum_{n=1}^N \sum_{i=1}^T \left( f(y_i^{(n)}, y_{i-1}^{(n)}, \mathbf{x}^{(n)}) - \nabla_{\lambda} \ln Z(\mathbf{x}^{(n)}, \lambda) \right)$$

- ▶ In each gradient step, for each data sample, we use inference to find  $P(y_i, y_{i-1} | \mathbf{x}^{(n)}, \lambda)$  required for computing feature expectation:

$$\begin{aligned} \nabla_{\lambda} \ln Z(\mathbf{x}^{(n)}, \lambda) &= E_{P(y|\mathbf{x}^{(n)}, \lambda^t)} [f(y_i, y_{i-1}, \mathbf{x}^{(n)})] \\ &= \sum_{y_i, y_{i-1}} P(y_i, y_{i-1} | \mathbf{x}^{(n)}, \lambda) f(y_i, y_{i-1}, \mathbf{x}^{(n)}) \end{aligned}$$

# Summary

---

## ▶ Discriminative vs. generative

- ▶ In cases where we have many correlated features, discriminative models (MEMM and CRF) are often better
  - ▶ avoid the challenge of explicitly modeling the distribution over features.
- ▶ but, if only limited training data are available, the stronger bias of the generative model may dominate and these models may be preferred.

## ▶ Learning

- ▶ HMMs and MEMMs are much more easily learned.
- ▶ CRF requires an iterative gradient-based approach, which is considerably more expensive
  - ▶ inference must be run separately for every training sequence

## ▶ MEMM vs. CRF (Label bias problem of MEMM)

- ▶ In many cases, CRFs are likely to be a safer choice (particularly in cases where many transitions are close to deterministic), but the computational cost may be prohibitive for large data sets.