HMM, MEMM and CRF

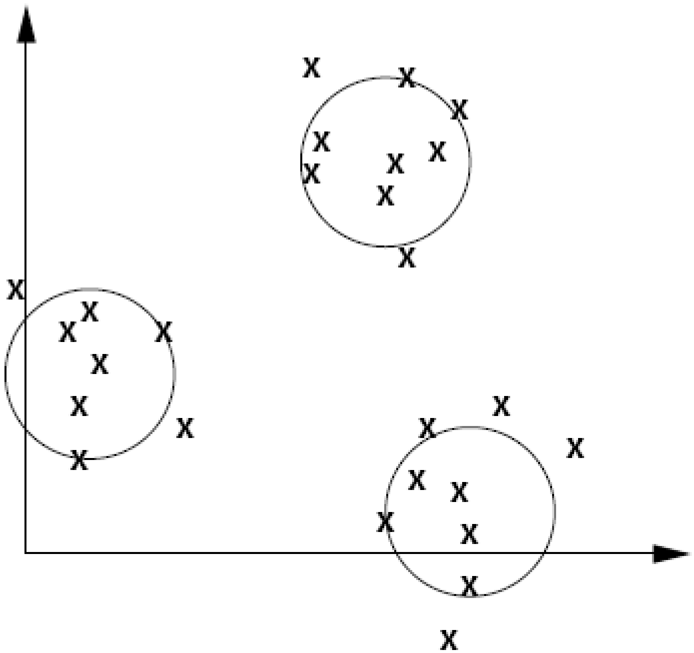
40-957 Special Topics in Artificial Intelligence:
Probabilistic Graphical Models
Sharif University of Technology

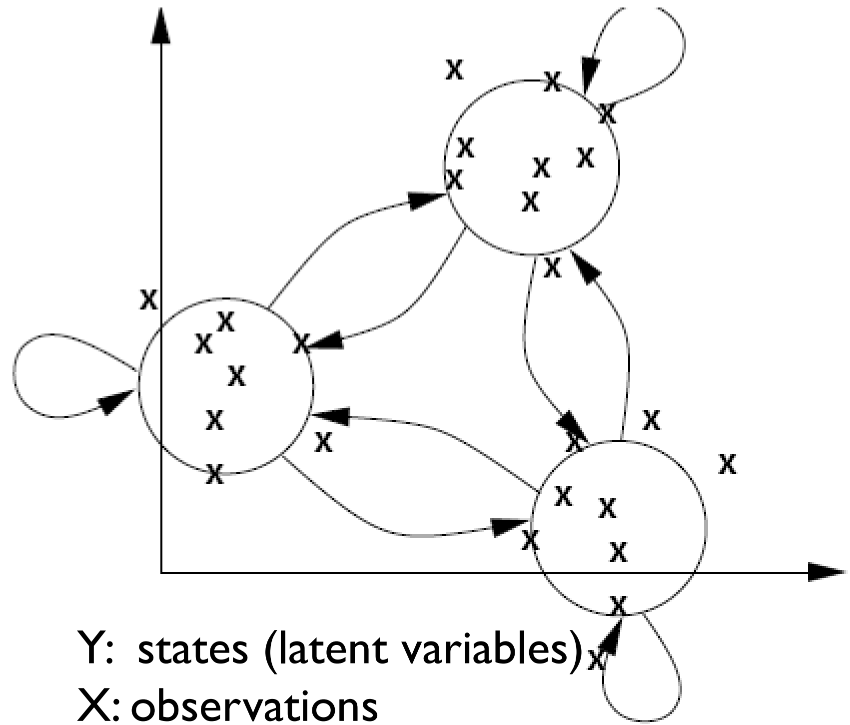
Soleymani
Spring 2014

Sequence labeling

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

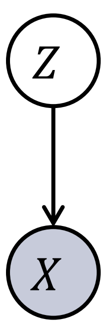
Generalization of mixture models for sequential data

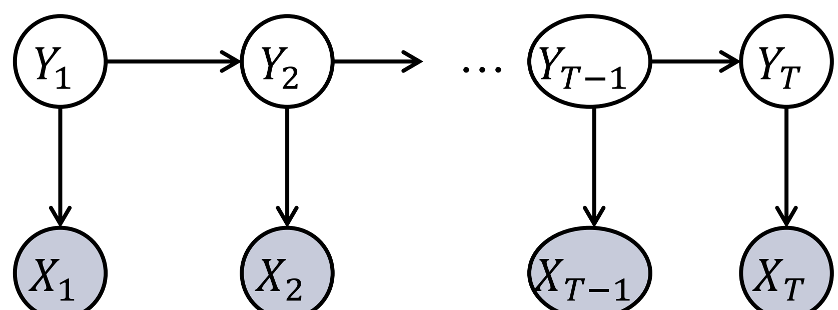




[Jordan]

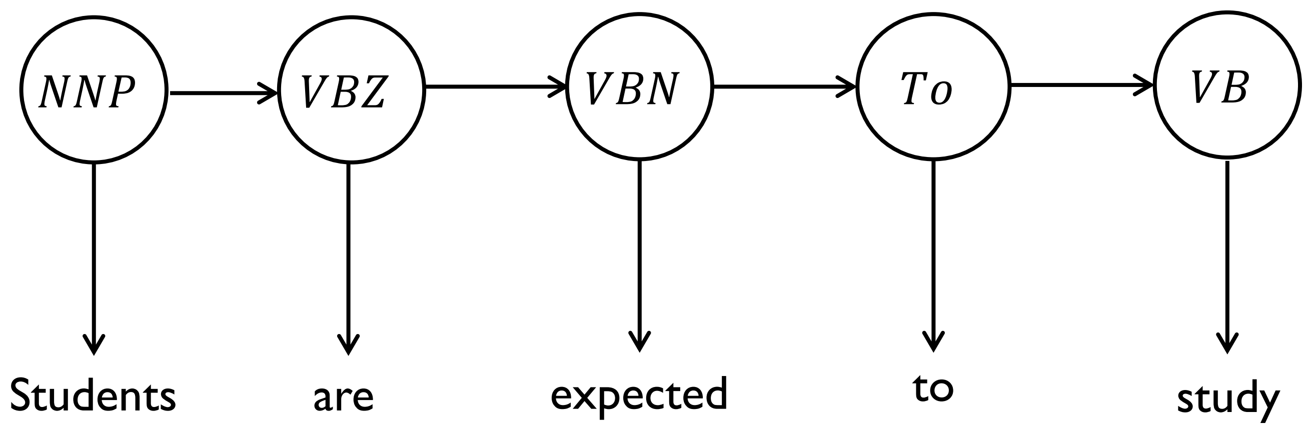
Y: states (latent variables)
X: observations





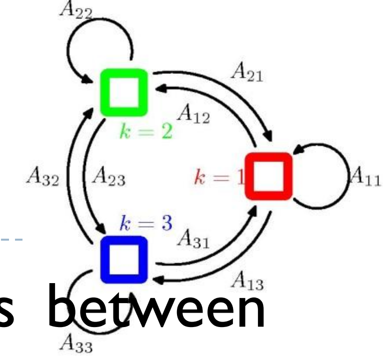
HMM examples

- 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
  - $P(X_t | Y_t, \mathbf{\Phi})$

HMM: probabilistic model

Y: states (latent variables)
X: observations

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

Inference problems in sequential data

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

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

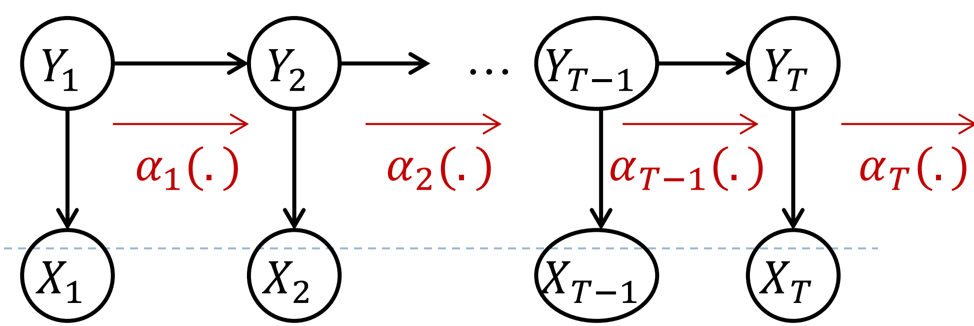
$$i, j = 1, \dots, M$$

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

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

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

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



Backward algorithm

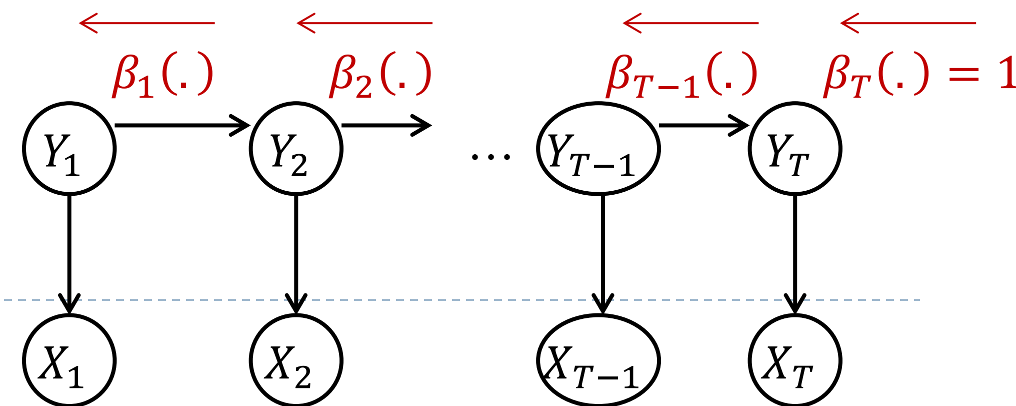
$$\beta_t(i) = m_{t\to t-1}(i) = P(x_{t+1}, \dots, x_T | Y_t = i)$$

$i, j \in states$

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

- Initialization:
  - $\beta_T(i) = 1$
- Iterations: $t = T$ down to 2
  - $\beta_{t-1}(i) = \sum_j \beta_t(j) P(Y_t = j | Y_{t-1} = i) P(x_t | Y_t = i)$

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



Forward-backward algorithm

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

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

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

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

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

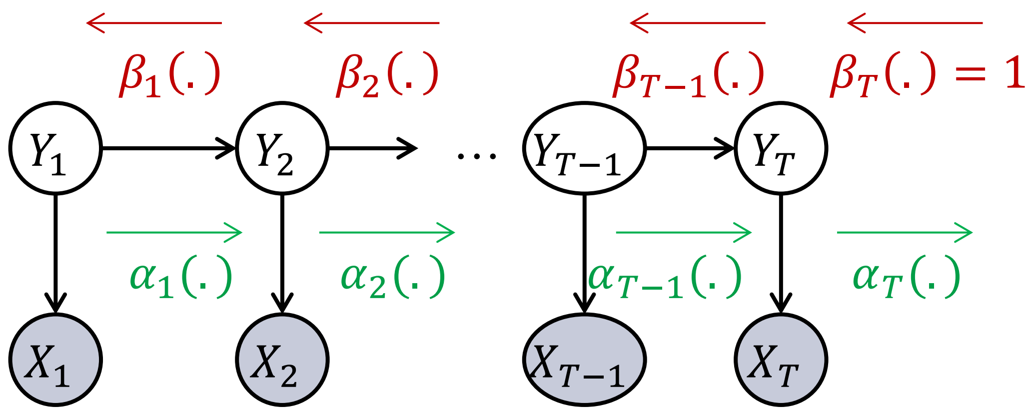
$$\beta_T(i) = 1$$

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

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

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:
  - $\underset{y_1,\ldots,y_t}{\operatorname{argmax}} P(y_1, \ldots, y_t | x_1, \ldots, x_t)$

- Viterbi algorithm:
  - Define auxiliary variable δ:
    - $\delta_t(i) = \max_{y_1,\ldots,y_{t-1}} P(y_1, y_2, \ldots, y_{t-1}, Y_t = i, x_1, x_2, \ldots, x_t)$
    - $\delta_t(i)$: probability of the most probable path ending in state $Y_t = i$
  - Recursive relation:
    - $\delta_t(i) = \max_j(\delta_{t-1}(j) P(Y_t = i | Y_{t-1} = j))\, P(x_t | Y_t = i)$

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

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

Max-product algorithm

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

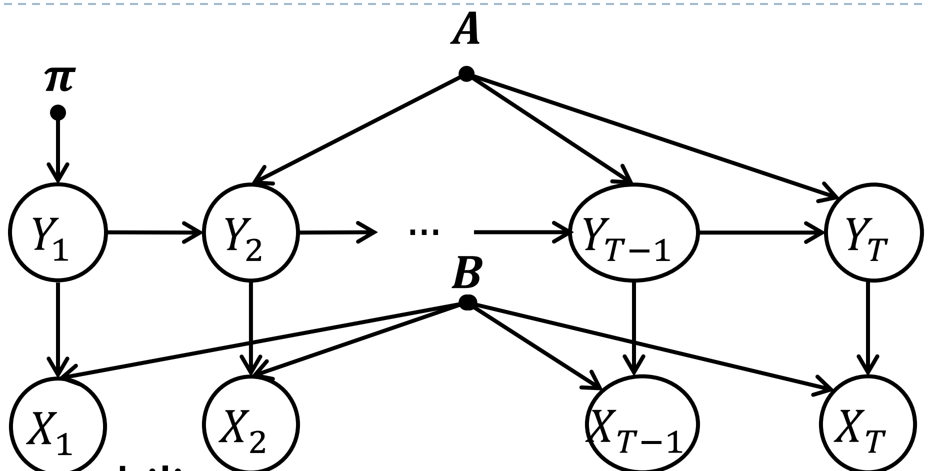
$$\delta_t(i) = m_{t-1,t}^{max} \times \phi(x_i)$$

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:

$$\pi_i = P(Y_1 = i), \qquad 1 \le i \le M$$

- State transition probability:

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

- State transition probability:

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

Discrete observations

HMM: supervised parameter learning by MLE

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

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

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

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

Discrete observations

Learning

- Problem: how to construct an HHM given only observations?

- Find $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$, maximizing $P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T | \boldsymbol{\theta})$
  - Incomplete data
    - EM algorithm

HMM learning by EM (Baum-Welch)

- $\boldsymbol{\theta}^{old} = [\boldsymbol{\pi}^{old}, \boldsymbol{A}^{old}, \boldsymbol{\Phi}^{old}]$
- E-Step:
  - $\gamma_{n,t}^{i} = P\left(Y_t^{(n)} = k \middle| \boldsymbol{X}^{(n)}; \boldsymbol{\theta}^{old}\right)$
  - $\xi_{n,t}^{i,j} = P\left(Y_{t-1}^{(n)} = i, Y_t^{(n)} = j \middle| \boldsymbol{x}^{(n)}; \boldsymbol{\theta}^{old}\right)$

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

- M-Step:
  - $\pi_i^{new} = \frac{\sum_{n=1}^{N} \gamma_{n,1}^{i}}{N}$
  - $A_{i,j}^{new} = \frac{\sum_{n=1}^{N} \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_{n=1}^{N} \sum_{t=1}^{T-1} \gamma_{n,t}^{i}}$
  - $B_{i,k}^{new} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \gamma_{n,t}^{i} I\left(X_t^{(n)} = i, \right)}{\sum_{n=1}^{N} \sum_{t=1}^{T} I\left(X_t^{(n)} = i\right)}$

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

Discrete observations

Baum-Welch algorithm (Baum, 1972)

HMM learning by EM (Baum-Welch)

- $\boldsymbol{\theta}^{old} = [\boldsymbol{\pi}^{old}, \boldsymbol{A}^{old}, \boldsymbol{\Phi}^{old}]$
- E-Step:

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

  - $\gamma_{n,t}^{i} = P\left(Y_t^{(n)} = k \middle| \boldsymbol{X}^{(n)}; \boldsymbol{\theta}^{old}\right)$
  - $\xi_{n,t}^{i,j} = P\left(Y_{t-1}^{(n)} = i, Y_t^{(n)} = j \middle| \boldsymbol{x}^{(n)}; \boldsymbol{\theta}^{old}\right)$
- M-Step:

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

  - $\pi_i^{new} = \frac{\sum_{n=1}^{N} \gamma_{n,1}^{i}}{N}$
  - $A_{i,j}^{new} = \frac{\sum_{n=1}^{N} \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_{n=1}^{N} \sum_{t=1}^{T-1} \gamma_{n,t}^{i}}$
  - $\boldsymbol{\mu}_i^{new} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \gamma_{n,t}^{i} \boldsymbol{x}_t^{(n)}}{\sum_{n=1}^{N} \sum_{t=1}^{T} \gamma_{n,t}^{i}}$

Assumption: Gaussian emission probabilities

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

Baum-Welch algorithm (Baum, 1972)

Forward-backward algorithm for E-step

$$P(y_{t-1}, y_t | \boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$$

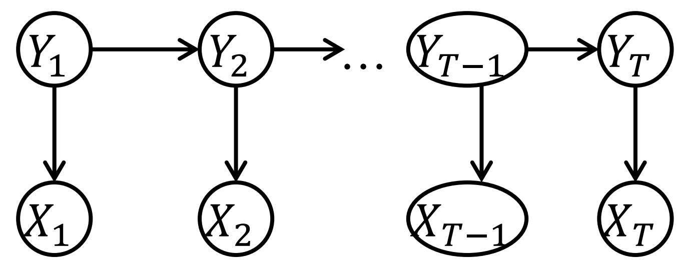
$$= \frac{P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T | y_{t-1}, y_t) P(y_{t-1}, y_t)}{P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)}$$

$$= \frac{P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_{t-1} | y_{t-1}) P(\boldsymbol{x}_t | y_t) P(\boldsymbol{x}_{t+1}, \dots, \boldsymbol{x}_T | y_t) P(y_t | y_{t-1}) P(y_{t-1})}{P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)}$$

$$= \frac{\alpha_{t-1}(y_{t-1}) P(\boldsymbol{x}_t | y_t) P(y_t | y_{t-1}) \beta_t(y_t)}{\sum_{i=1}^{M} \alpha_T(i)}$$

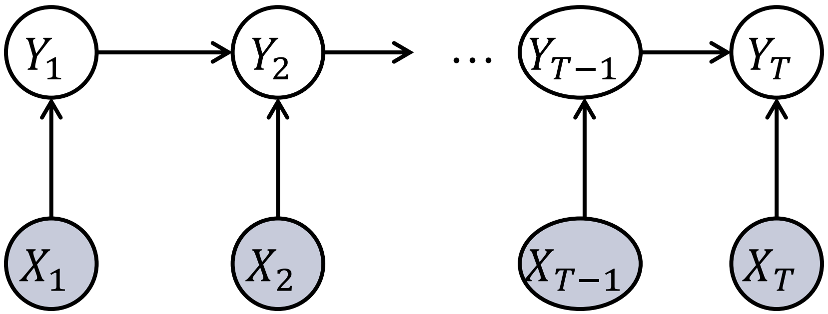
HMM shortcomings

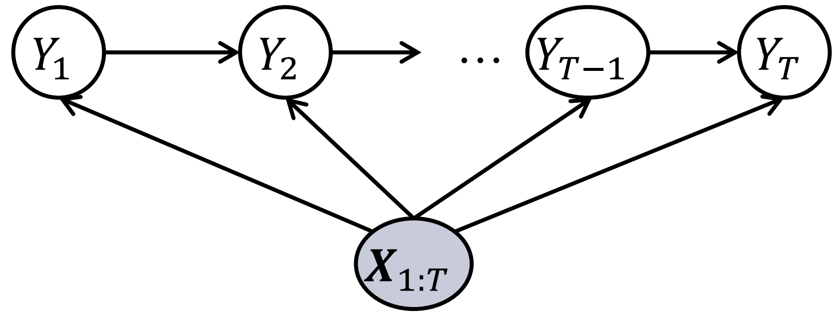
- In modeling the joint distribution $P(\boldsymbol{Y}, \boldsymbol{X})$, HMM ignores many dependencies between observations $X_1, \dots, X_T$ (similar to most generative models that need to simplify the structure)



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

Maximum Entropy Markov Model (MEMM)





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

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

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

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

Feature function

- Feature function $\boldsymbol{f}(y_t, y_{t-1}, \boldsymbol{x}_{1:T})$ 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 $f(y_t, y_{t-1}, \boldsymbol{x}_{1:T})$ 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

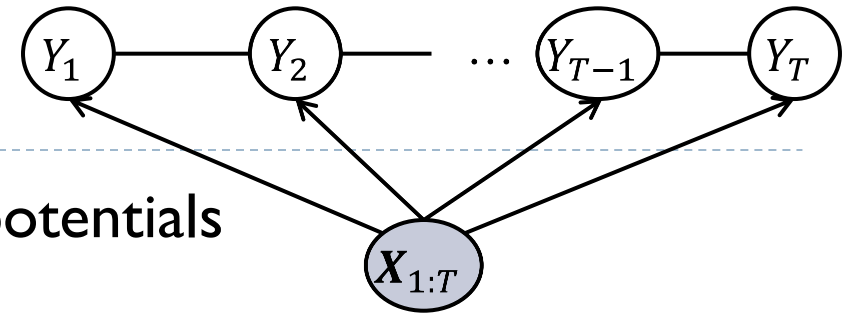
Label bias problem in MEMM

- **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

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})}\prod_{t=1}^{T}\phi(y_{t-1}, y_t, \boldsymbol{X})$$

$$= \frac{1}{Z(\boldsymbol{X}, \boldsymbol{w})}\prod_{t=1}^{T}\exp\{\boldsymbol{w}^T f(y_t, y_{t-1}, \boldsymbol{x})\}$$

$$\boldsymbol{X} \equiv \boldsymbol{x}_{1:T}$$
$$\boldsymbol{Y} \equiv y_{1:T}$$

- CRF is a discriminative model (like MEMM)
  - can dependence between each state and the entire observation sequence
  - uses global normalizer $Z(\boldsymbol{X}, \boldsymbol{w})$ 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

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

$$= \frac{1}{Z(\boldsymbol{x},\boldsymbol{\lambda})}\exp\left\{\sum_{i=1}^{T}\sum_{k}\lambda_k f_k(y_i, y_{i-1}, \boldsymbol{x})\right\}$$

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

CRF: MAP inference

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

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

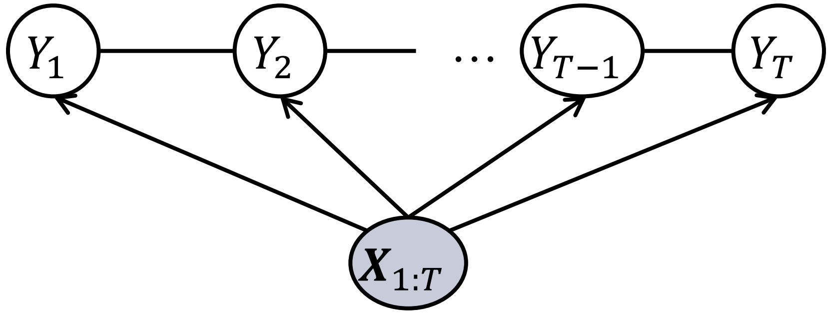
- $Z(\boldsymbol{x})$ is not a function of $\boldsymbol{y}$ and so has been ignored

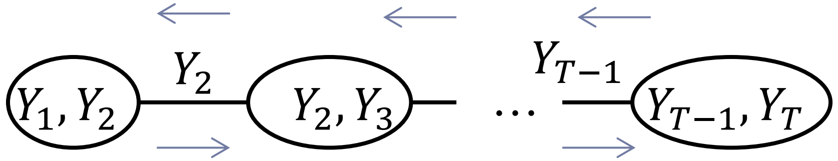
- Max-product algorithm can be used for this MAP inference problem
  - Same as Viterbi decoding used in HMMs

CRF: inference

Exact inference for 1-D chain CRFs

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





CRF: learning

Maximum Conditional Likelihood

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(\mathcal{D}^Y | \mathcal{D}^X, \boldsymbol{\theta})$$

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{n=1}^{N} P\left(\boldsymbol{y}^{(n)} \middle| \boldsymbol{x}^{(n)}, \boldsymbol{\lambda}\right)$$

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

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

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

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

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

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

CRF: learning

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

$$= \sum_{i=1}^{T} \sum_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}^{(n)}, \boldsymbol{\lambda}) \boldsymbol{f}\big(y_i, y_{i-1}, \boldsymbol{x}^{(n)}\big)$$

$$= \sum_{i=1}^{T} \sum_{y_i, y_{i-1}} P(y_i, y_{i-1}|\boldsymbol{x}^{(n)}, \boldsymbol{\lambda}) \boldsymbol{f}(y_i, y_{i-1}, \boldsymbol{x})$$

How do we find the above expectations?

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

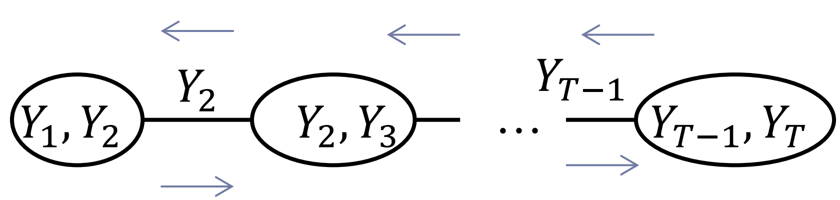
CRF: learning
Inference to find $P(y_i, y_{i-1}|\boldsymbol{x}^{(n)}, \boldsymbol{\lambda})$

- Junction tree algorithm
  - Initialization of clique potentials:

$$\psi(y_i, y_{i-1}) = \exp\{\boldsymbol{\lambda}^T \boldsymbol{f}\left(y_i, y_{i-1}, \boldsymbol{x}^{(n)}\right)\}$$
$$\phi(y_i) = 1$$

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

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



CRF learning: gradient descent

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

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

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

$$\nabla_{\boldsymbol{\lambda}} \ln Z(\boldsymbol{x}^{(n)}, \boldsymbol{\lambda}) = E_{P(\boldsymbol{y}|\boldsymbol{x}^{(n)}, \boldsymbol{\lambda}^t)}\left[\boldsymbol{f}\left(y_i, y_{i-1}, \boldsymbol{x}^{(n)}\right)\right]$$
$$= \sum_{y_i, y_{i-1}} P(y_i, y_{i-1} | \boldsymbol{x}^{(n)}, \boldsymbol{\lambda}) \boldsymbol{f}\left(y_i, y_{i-1}, \boldsymbol{x}^{(n)}\right)$$

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.