CSCI 544: Applied Natural Language Processing

Sequence Labeling-2

Xuezhe Ma (Max)

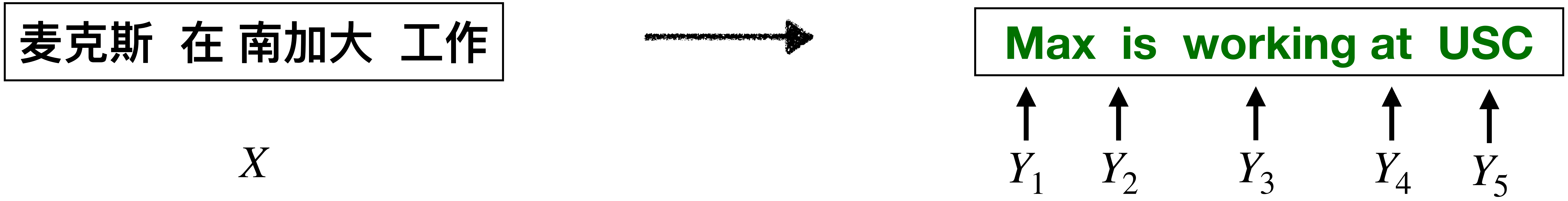
USC University of Southern California

Recap

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation: Maximum Likelihood Estimation (MLE)
  - The Viterbi algorithm

Recap: Structured Prediction

- Y consists of multiple components $Y = \{y_1, y_2, \ldots, y_n\}$
- (Strong) correlations between output components
- Exponential output space
  - Decoding: $y^* = \text{argmax}_{y \in \mathcal{Y}} p(y \mid x)$

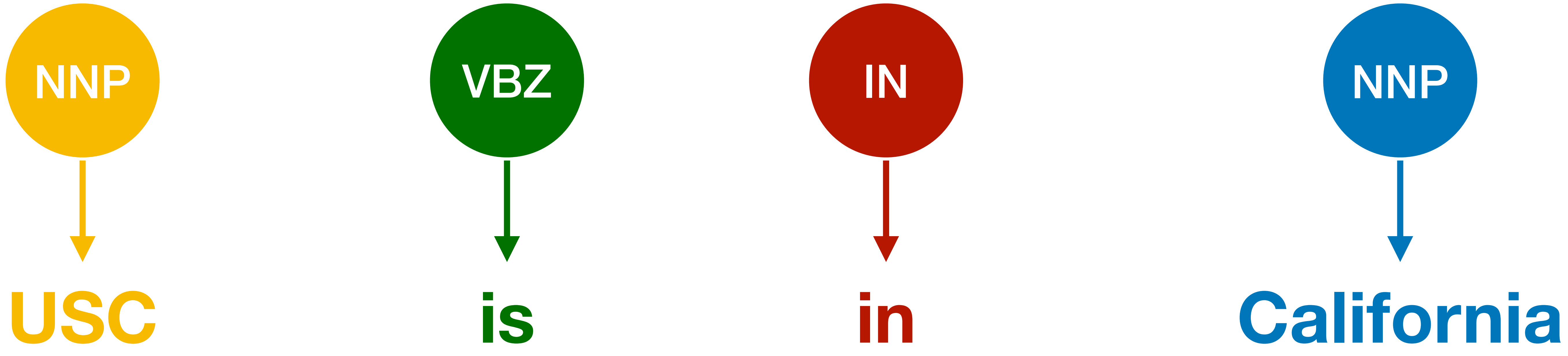


Recap: Sequence Labeling

A type of structured prediction tasks

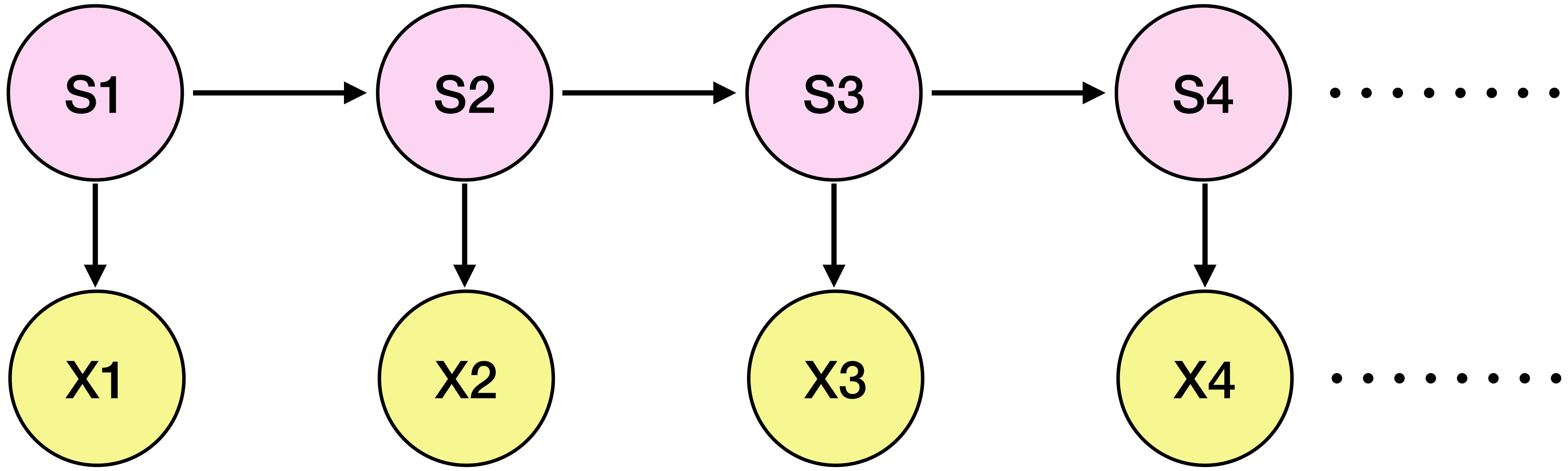
$$Y = < y_i, y_2, \ldots, y_n >$$

$$X = < x_i, x_2, \ldots, x_n >$$



Assigning each token of X, e.g. x_i a corresponding label y_i

Recap: Hidden Markov Models

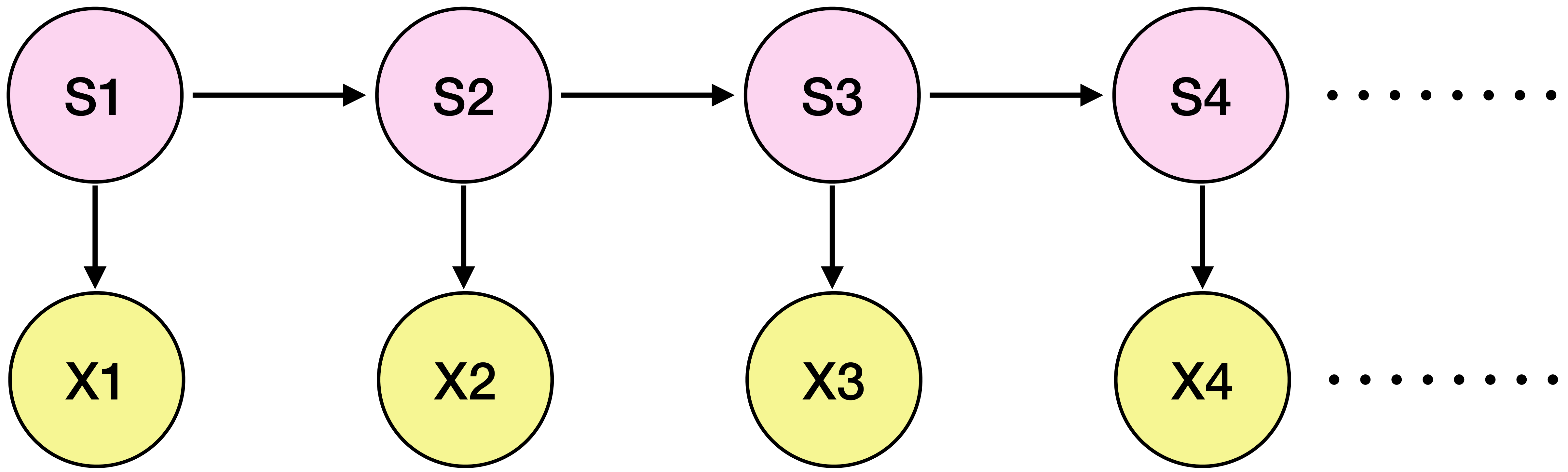


- **Set of states** $S \in \{1,2,\ldots,k\}$ **and set of observations (words)** X
- **Transition probabilities** $P(s_{j+1} \mid s_j) = t(s_{j+1} \mid s_j)$
- **Emission probabilities** $P(x_j \mid s_j) = e(x_j \mid s_j)$
- **Decoding with Viterbi algorithm**

$$p(x_1, \ldots, x_m, s_1, \ldots, s_m) = t(s_1) \prod_{j=2}^{m} t(s_j \mid s_{j-1}) \prod_{j=1}^{m} e(x_j \mid s_j)$$

Slide credit: Danqi Chen COS 484/584

Recap: Hidden Markov Models



- **A generative model with strong assumptions**
  - Is generative model necessary?
- **Simple to train**
  - Just need to compile counts from the training corpus
- **Performs relatively well**
  - 96% on POS tagging (92.3% of most frequent class)
- **Features only on *word type* and *tag***

$$p(x_1, \ldots, x_m, s_1, \ldots, s_m) = t(s_1) \prod_{j=2}^{m} \boxed{t(s_j \mid s_{j-1})} \prod_{j=1}^{m} \boxed{e(x_j \mid s_j)}$$

Slide credit: Danqi Chen COS 484/584

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Maximum Likelihood Estimation (MLE)
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Generative Models vs. Discriminative Models
  - General Form of Log-Linear Models
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Generative Models vs Discriminative Models

Generative vs Discriminative

- **Generative Models**
  - Modeling the joint distribution: $P(X, S)$
- **Discriminative Models**
  - Modeling $P(S \mid X)$ directly?

	Generative	**Discriminative**
Classification	Naive Bayes: $P(y)P(x \mid y)$	Logistic Regression: $P(y \mid x)$
Sequence Labeling	HMM: $P(s_1, \ldots, s_n)P(x_1, \ldots, x_n \mid s_1, \ldots, s_n)$	MEMM/CRF: $P(s_1, \ldots, s_n \mid x_1, \ldots, x_n)$

Log-Linear Models

Log-Linear Models

- **The General Problem**

  - We have some **input domain** $\mathcal{X}$
  - Have a finite **label set** $\mathcal{Y}$
  - Aim is to provide a **conditional probability** $p(y \mid x)$ for any x, y where $x \in \mathcal{X}$, $y \in \mathcal{Y}$

Slide credit: Michael Collins COMS W4705

Log-Linear Models

- We have some input domain $\mathcal{X}$, and a finite label set $\mathcal{Y}$. Aim is to provide a conditional probability $p(y \mid x)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- A feature is a function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$
  (Often binary features or indicator functions $f_k : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$).
- Say we have m features f_k for $k = 1 \ldots m$
  $\Rightarrow$ A feature vector $f(x, y) \in \mathbb{R}^m$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- We also have a **parameter vector** $v \in \mathbb{R}^m$
- We define

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}$$

Slide credit: Michael Collins COMS W4705

Why the name?

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}$$

$$\log p(y \mid x; v) = \underbrace{v \cdot f(x,y)}_{\text{Linear term}} - \underbrace{\log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}_{\text{Normalization term}}$$

Features in Log-linear Models

$$S = S_i, S_2, \ldots, S_n$$

$$X = X_i, X_2, \ldots, X_n$$

NNP	VBZ	IN	NNP
USC	is	in	California

Theoretically, we can use any features in X and S: $f(X, S)$

- The current word: <is>
- The surrounding words: <USC>, <in>, …
- The current POS tag: <VBZ>
- The surrounding tags: <NNP>, <IN>, …

How to design these features into numerical vectors?

Binary Feature Vectors

$$f_1 = \begin{cases} 1, & \text{if } x_i = \text{is},\ s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

$$f_2 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC},\ x_i = \text{is},\ s_{i-1} = \text{NNP, and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

$$f_3 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC},\ x_i = \text{is},\ x_{i+1} = \text{in},\ s_{i-1} = \text{NNP, and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

Task-Specific Features

- Spelling features for prefixes/suffixes

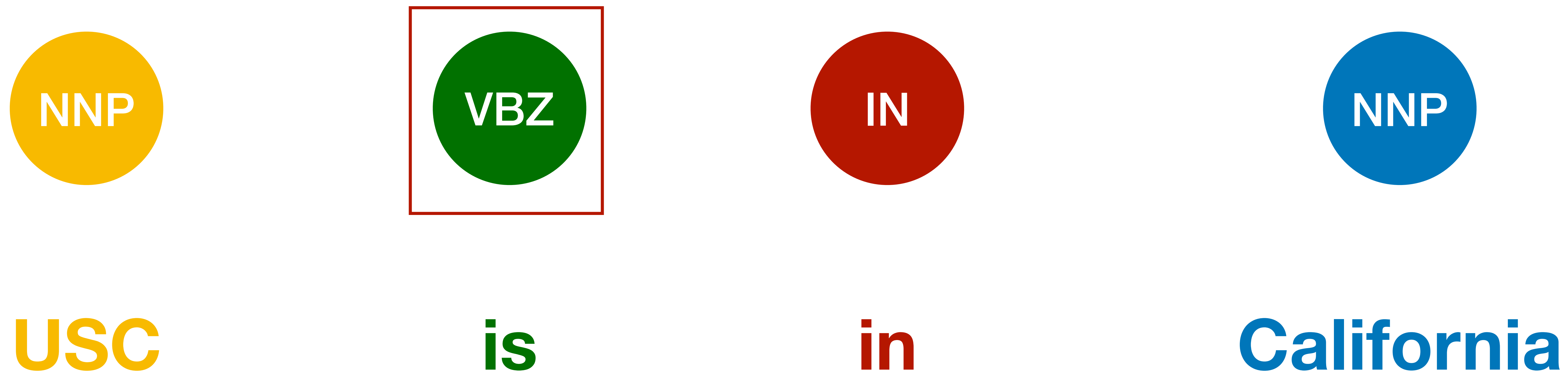
$$f_4 = \begin{cases} 1, & \text{if } x_i \text{ ends in } \textit{ing},\ s_{i-1} = \text{NNP, and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

$$f_5 = \begin{cases} 1, & \text{if } x_i \text{ starts with } \textit{pre},\ s_{i-1} = \text{NNP, and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

Features in Log-linear Models

$$S = S_i, S_2, \ldots, S_n$$

$$X = X_i, X_2, \ldots, X_n$$



Theoretically, we can use any features in X and S: $f(X, S)$

- The current word: <is>
- The surrounding words: <USC>, <in>, …
- The current POS tag: <VBZ>
- The surrounding tags: <NNP>, <IN>, …

How to design these features into numerical vectors?

Can we design any features in practice?

Feature Sparsity

Number of Features

$$f_1 = \begin{cases} 1, & \text{if } x_i = \text{is},\ s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

VK

$$f_2 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC},\ x_i = \text{is},\ s_{i-1} = \text{NNP, and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

V^2K^2

$$f_3 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC},\ x_i = \text{is},\ x_{i+1} = \text{in},\ s_{i-1} = \text{NNP, and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

V^3K^2

Features vs. Independence

- We need independence assumptions to compute the nominator
- Stronger assumptions lead to less flexible features

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\boxed{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}}$$

Overview

- The Sequence Labeling Problem
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
  - Maximum Likelihood Estimation (MLE)
- Hidden Markov Model (HMM)
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Generative Models vs. Discriminative Models
  - General Form of Log-Linear Models
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Maximum-Entropy Markov Models (MEMMs)

- Goal: modeling the distribution

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m)$$

Independence Assumptions in MEMMs

- **Markov Assumption on S**

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \prod_{j=1}^{m} p(s_j \,|\, s_1, \ldots, s_{j-1}, x_1, \ldots, x_m)$$

chain rule (no assumptions)

$$= \prod_{j=1}^{m} p(s_j \,|\, s_{j-1}, x_1, \ldots, x_m)$$

Markov assumption

Using Log-Linear Models

- **We then model each term using a log-linear model:**

$$p(s_j \mid s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_j \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s'_j))}$$

- **Here $f(x_1, \ldots, x_m, j, s, s')$ is the feature vector:**
  - $x_1, \ldots, x_m$ is the sequence of words to be tagged
  - j is the position to be tagged (any value from $1, \ldots, m$)
  - s is the previous state
  - s' is the new state

Using Log-Linear Models

- **We then model each term using a log-linear model:**

$$p(s_j \mid s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, \cancel{i}, s_{j-1}, s_j))}{\boxed{\sum_{s'_j \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s'_j))}}$$

Trackable

i ==j

- **Here $\boxed{f(x_1, \ldots, x_m, j, s, s')}$ is the feature vector:**
  - $x_1, \ldots, x_m$ is the sequence of words to be tagged
  - j is the position to be tagged (any value from $1, \ldots, m$)
  - s is the previous state
  - s' is the new state

The whole sequence of X

Only two successive tags

Features in MEMMs

$$S = S_1, S_2, \ldots, S_n$$

$$X = X_1, X_2, \ldots, X_n$$

NNP	VBZ	IN	NNP
USC	is	in	California

What are the most important features $f(x_1, \ldots, x_m, j, s, s')$?

- The current word: <is>
- The current tag: <VBZ>
- The surrounding words: <USC>, <in>
- The previous POS tag: <NNP>
- Prefix or suffix features
- …

Decoding with MEMMs: Viterbi Algorithm

- Goal: for a given input sequence $x_1, \ldots, x_m$, find

$$\arg \max_{s_1, \ldots, s_m} p(s_1 \ldots s_m | x_1 \ldots x_m)$$

- We can use the *Viterbi* algorithm again (see last lecture on HMMs). Basic data structure:

$$\pi[j, s]$$

will be a table entry that stores the maximum probability for any state sequence ending in state s at position j. More formally:

$$\pi[j, s] = \max_{s_1 \ldots s_{j-1}} \left(p(s | s_{j-1}, x_1 \ldots x_m) \prod_{k=1}^{j-1} p(s_k | s_{k-1}, x_1 \ldots x_m) \right)$$

Slide credit: Michael Collins COMS W4705

Decoding with MEMMs: Viterbi Algorithm

- Initialization: for $s \in \mathcal{S}$

$$\pi[1, s] = p(s | s_0, x_1 \ldots x_m)$$

where s_0 is a special "initial" state.

- For $j = 2 \ldots m$, $s = 1 \ldots k$:

$$\pi[j, s] = \max_{s' \in \mathcal{S}} \left[\pi[j-1, s'] \times p(s | s', x_1 \ldots x_m) \right]$$

- We then have

$$\max_{s_1 \ldots s_m} p(s_1 \ldots s_m | x_1 \ldots x_m) = \max_{s} \pi[m, s]$$

Slide credit: Michael Collins COMS W4705

Model Performance

	POS Tagging	NER
HMM	96.4%	75.3
MEMM	96.9%	85.9

HMMs vs. MEMMs

- In MEMMs, each state transition has probability

$$p(s_j \mid s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s' \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s'_j))}$$

- In HMMs, each state transition has probability

$$p(s_j \mid s_{j-1}) p(x_j \mid s_j)$$

- Feature vectors f allows much richer representations in MEMMs:
  - Sensitivity to *any* word in the input sequence $x_1, \ldots, x_m$, not just x_j
  - Sensitivity to spelling features (prefixes, suffixes etc.) of current or surrounding words
- Parameter estimation in MEMMs is more expensive than in HMMs (but is still not prohibitive for most tasks)

Can we relax the Markov assumption in MEMMs but keep the same features?

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
  - Maximum Likelihood Estimation (MLE)
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Conditional Random Fields (CRFs)

- **Goal: modeling the distribution**

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m)$$

- **In MEMMs we had**

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \prod_{j=1}^{m} p(s_j \,|\, s_1, \ldots, s_{j-1}, x_1, \ldots, x_m)$$

chain rule (no assumptions)

$$= \prod_{j=1}^{m} p(s_j \,|\, s_{j-1}, x_1, \ldots, x_m)$$

Markov assumption

- **Using log-linear model**

$$p(s_j \,|\, s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s' \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s'_j))}$$

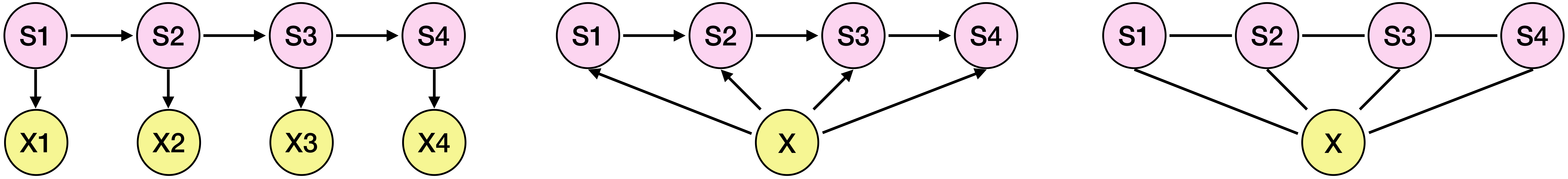
Can we build a *giant* log-linear model?

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m)$$

Conditional Random Fields (CRFs)

- **Globally Normalized Model**

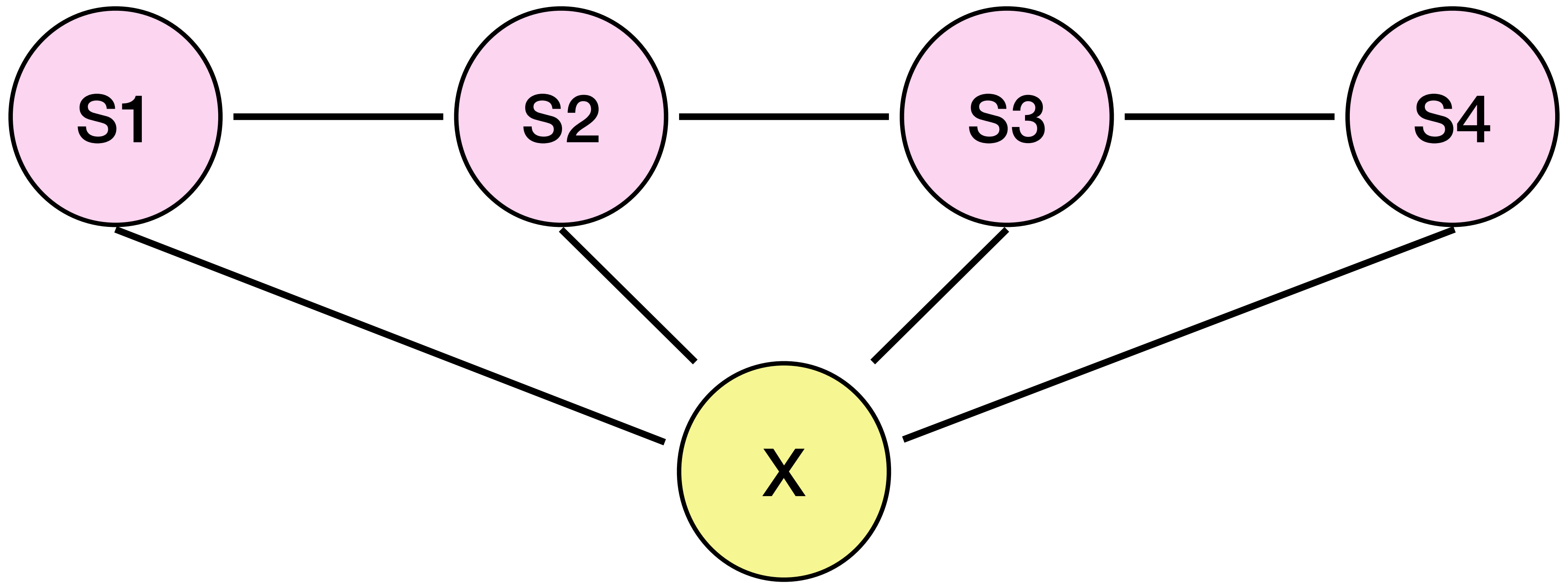
$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}$$



HMM **MEMM** **CRF**

Independence Assumptions in CRFs

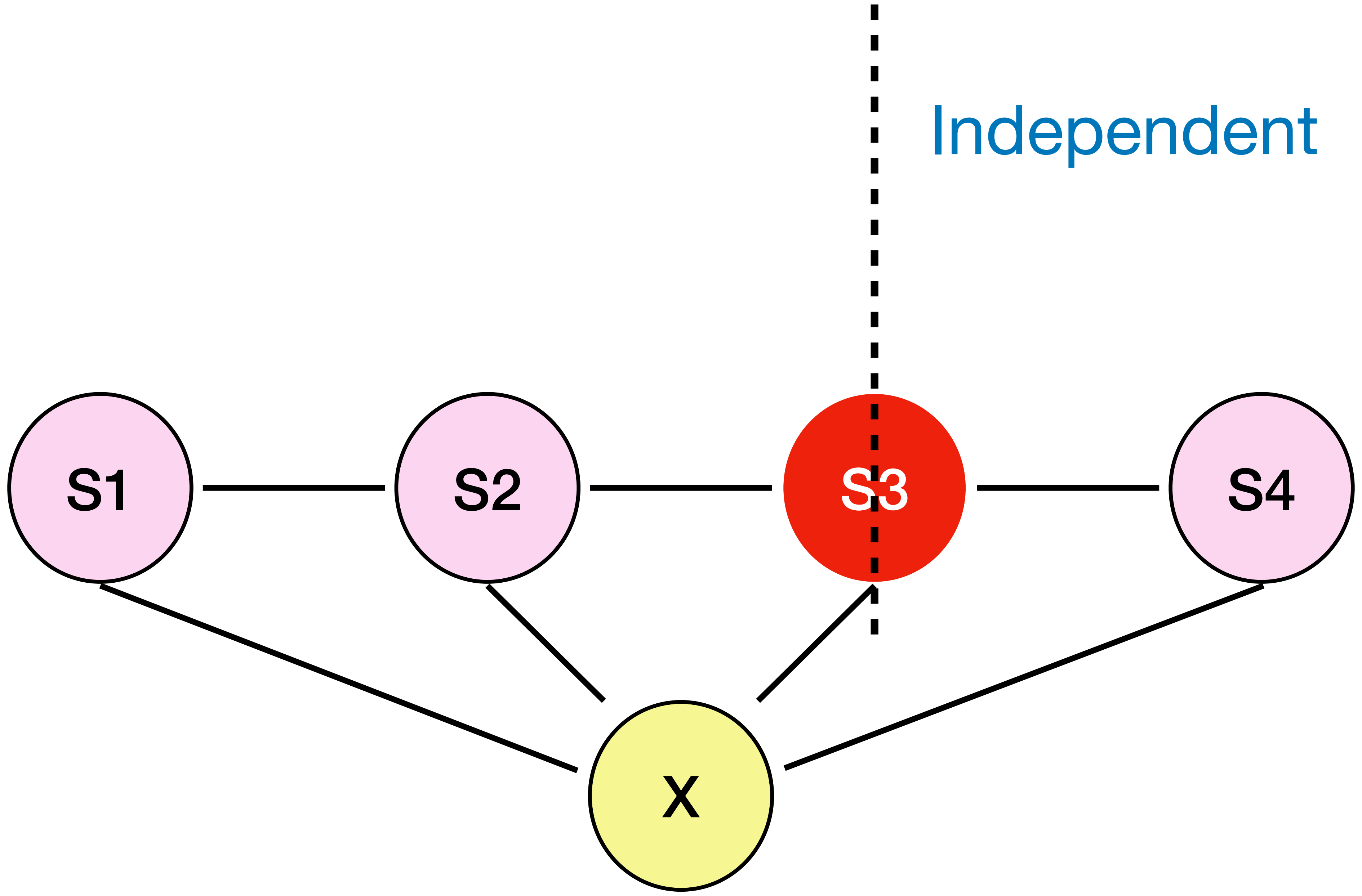
$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}$$



CRF

Independence Assumptions in CRFs

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}$$



CRF

weaker than MEMMs!

Decoding with CRFs

- **Viterbi Algorithm still applicable!**

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\boxed{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}}$$

Makes no effects on decoding!

Computation of the Global Nominator

- **How to compute the global nominator?**
  - Dynamic programming similar to the Viterbi algorithm
  - Replacing the maximum operation in decoding with sum operation

$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\boxed{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}}$$

Partition Function

$$\pi[j, s] = \boxed{\sum_{s_1, \ldots, s_{j-1}}} \left[\prod_{k=1}^{j-1} \exp(v \cdot f(x_1, \ldots, x_m, k, s_{k-1}, s_k)) \right] \exp(v \cdot f(x_1, \ldots, x_m, k, s_{j-1}, s)))$$

Overview

- The Sequence Labeling Problem
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
  - Maximum Likelihood Estimation (MLE)
- Hidden Markov Model (HMM)
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Maximum Likelihood Estimation

- **Need to maximize:**

$$\max_v L(v) = \sum_{i=1}^{N} \log P(S_i \mid X_i; v)$$

$$= \sum_{i=1}^{N} v \cdot f(X_i, S_i) - \sum_{i=1}^{N} \log \sum_{s' \in \mathbb{S}} e^{v \cdot f(X_i, S')}$$

- **Calculating gradients:**

$$\frac{\partial L(v)}{\partial v_k} = \underbrace{\sum_{i=1}^{N} f_k(X_i, S_i)}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{N} \sum_{S' \in \mathbb{S}} f_k(X_i, S') p(S' \mid X_i; v)}_{\text{Expected counts}}$$

See Collin's notes for derivations

Model Performance

	POS Tagging	NER
HMM	96.4%	75.3
MEMM	96.9%	85.9
CRF	97.3%	88.7

Reading Materials

- **Notes from Michael Collins:**
  - Log-linear Models
  - MEMMs and CRFs