Conditional Random Fields and beyond …

DANIEL KHASHABI
CS 546
UIUC, 2013

Outline

- Modeling
- Inference
- Training
- Applications

Outline

- Modeling
  - Problem definition
  - Discriminative vs. Generative
  - Chain CRF
  - General CRF
- Inference
- Training
- Applications

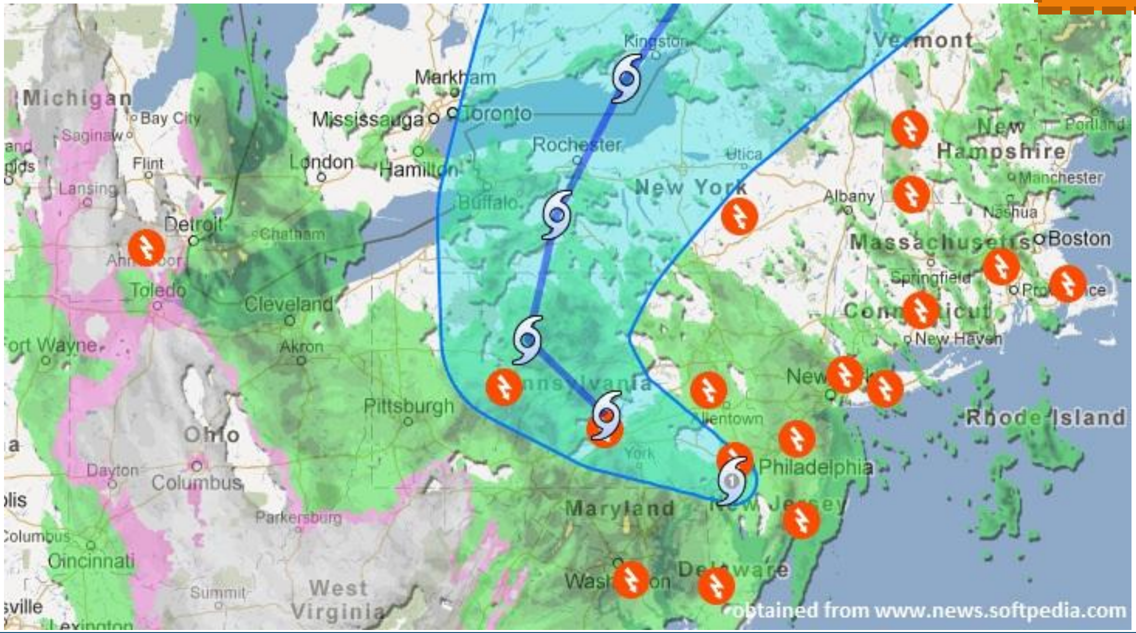
Problem Description

- Given X(observations), find Y(predictions)
- For example,

$$\begin{cases} X = \{temperature, moisture, pressure, \ldots\} \\ Y = \{Sunny, Rainy, Stormy, \ldots\} \end{cases}$$

Might depend on previous days and each other

Might depend on previous days and each other



Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ...

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ….
- Traditionally in graphical models,

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ….
- Traditionally in graphical models, ⇒ $p(\mathbf{x}, \mathbf{y})$

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ….
- Traditionally in graphical models, → $p(\mathbf{x},\mathbf{y}) = p(\mathbf{y}\,|\,\mathbf{x})\,p(\mathbf{x})$

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ...
- Traditionally in graphical models, ⇒ $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ….
- Traditionally in graphical models, ⇨ $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties
  - rich local features occur in relational data,

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ...
- Traditionally in graphical models, ⇨ $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties
  - rich local features occur in relational data, ⇨ $p(\mathbf{x})$

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ...
- Traditionally in graphical models, ⇨ $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties
  - rich local features occur in relational data, ⇨ $p(\mathbf{x})$

Intractable!

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ….
- Traditionally in graphical models, → $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties
  - rich local features occur in relational data, → $p(\mathbf{x})$
  - features may have complex dependencies,
    - constructing probability distribution over them is difficult

Intractable!

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ...
- Traditionally in graphical models, ⇨ $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties
  - rich local features occur in relational data, ⇨ $p(\mathbf{x})$
  - features may have complex dependencies,
    - constructing probability distribution over them is difficult

Intractable!

- **Solution:** directly model the conditional, $p(\mathbf{y} \mid \mathbf{x})$
  - is sufficient for classification!

Problem Description

- The relational connection occurs in many applications, NLP, Computer Vision, Signal Processing, ….
- Traditionally in graphical models, → $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x})$
  - Modeling the joint distribution can lead to difficulties
  - rich local features occur in relational data, → $p(\mathbf{x})$
  - features may have complex dependencies,
    - constructing probability distribution over them is difficult

Intractable!

- **Solution:** directly model the conditional, $p(\mathbf{y} \mid \mathbf{x})$
  - is sufficient for classification!
- **CRF** is simply a conditional distribution $p(\mathbf{y} \mid \mathbf{x})$ with an associated graphical structure

Discriminative Vs. Generative

$p(\mathbf{y}, \mathbf{x})$

$p(\mathbf{y} \mid \mathbf{x})$

Discriminative Vs. Generative

- **Generative Model:** A model that generate observed data randomly

$p(\mathbf{y}, \mathbf{x})$

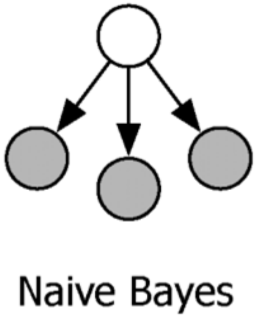
$p(\mathbf{y} \mid \mathbf{x})$

Discriminative Vs. Generative

$p(\mathbf{y}, \mathbf{x})$

- **Generative Model:** A model that generate observed data randomly
- **Naïve Bayes:** once the class label is known, all the features are independent

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k | y)$$



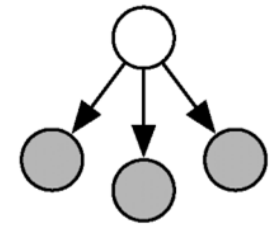
$p(\mathbf{y} | \mathbf{x})$

Discriminative Vs. Generative

$p(\mathbf{y},\mathbf{x})$

- **Generative Model:** A model that generate observed data randomly
- **Naïve Bayes:** once the class label is known, all the features are independent

$$p(y,\mathbf{x}) = p(y)\prod_{k=1}^{K} p(x_k|y)$$



Naive Bayes

- **Discriminative:** Directly estimate the posterior probability; Aim at modeling the "discrimination" between different outputs

$p(\mathbf{y}\,|\,\mathbf{x})$

Discriminative Vs. Generative

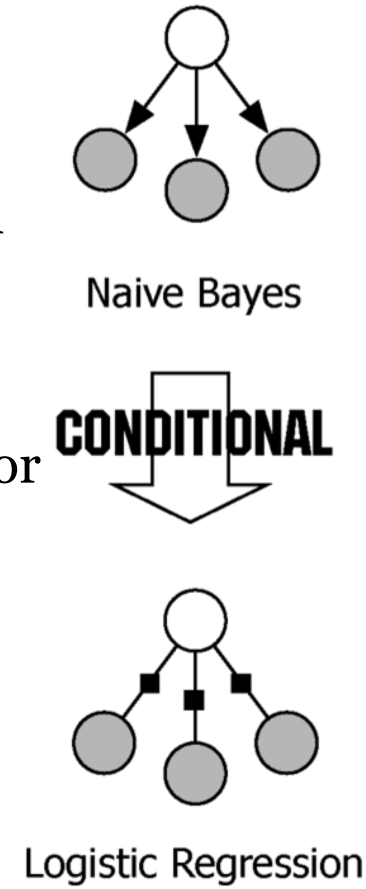
$p(\mathbf{y}, \mathbf{x})$

- **Generative Model:** A model that generate observed data randomly
- **Naïve Bayes:** once the class label is known, all the features are independent

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k | y)$$

$p(\mathbf{y} \mid \mathbf{x})$

- **Discriminative:** Directly estimate the posterior probability; Aim at modeling the "discrimination" between different outputs

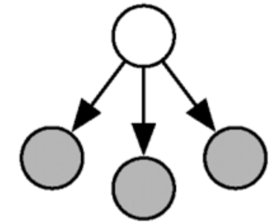


Discriminative Vs. Generative

$p(\mathbf{y}, \mathbf{x})$

- **Generative Model:** A model that generate observed data randomly
- **Naïve Bayes:** once the class label is known, all the features are independent

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k | y)$$

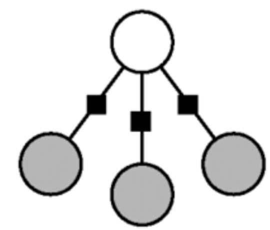


Naive Bayes

$p(\mathbf{y} \mid \mathbf{x})$

- **Discriminative:** Directly estimate the posterior probability; Aim at modeling the "discrimination" between different outputs
- **MaxEnt** classifier: linear combination of feature function in the exponent,

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y, \mathbf{x}) \right\}$$



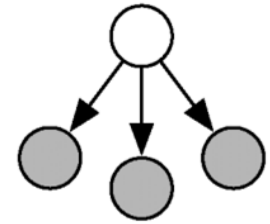
Logistic Regression

Discriminative Vs. Generative

$p(\mathbf{y}, \mathbf{x})$

- **Generative Model:** A model that generate observed data randomly
- **Naïve Bayes:** once the class label is known, all the features are independent

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k | y)$$

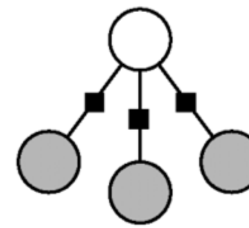


Naive Bayes

$p(\mathbf{y} | \mathbf{x})$

- **Discriminative:** Directly estimate the posterior probability; Aim at modeling the "discrimination" between different outputs
- **MaxEnt** classifier: linear combination of feature function in the exponent,

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{k=1}^{K} \theta_k f_k(y, \mathbf{x})\right\}$$



Logistic Regression

Both generative models and discriminative models describe distributions over (y , x), but they work in different directions.

Discriminative Vs. Generative

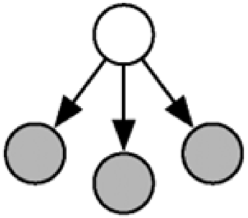
$p(\mathbf{y}, \mathbf{x})$

$p(\mathbf{y} \mid \mathbf{x})$

○=observable ○=unobservable

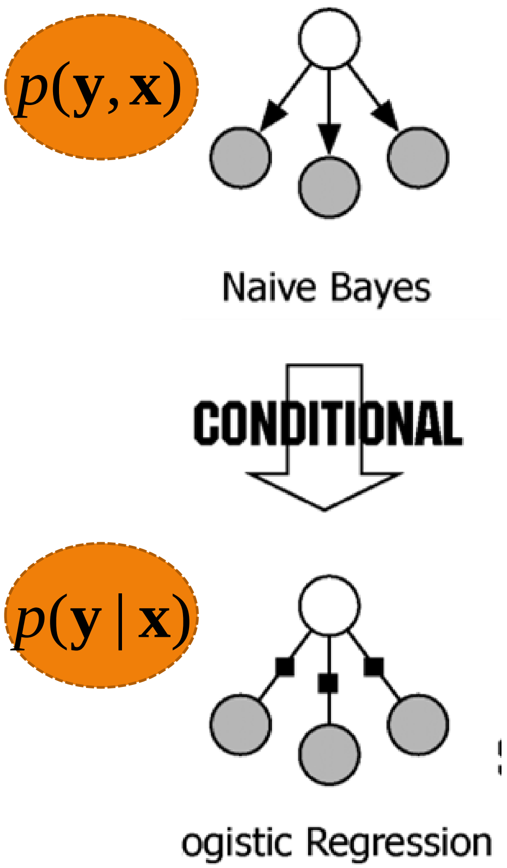
Discriminative Vs. Generative

$p(\mathbf{y}, \mathbf{x})$



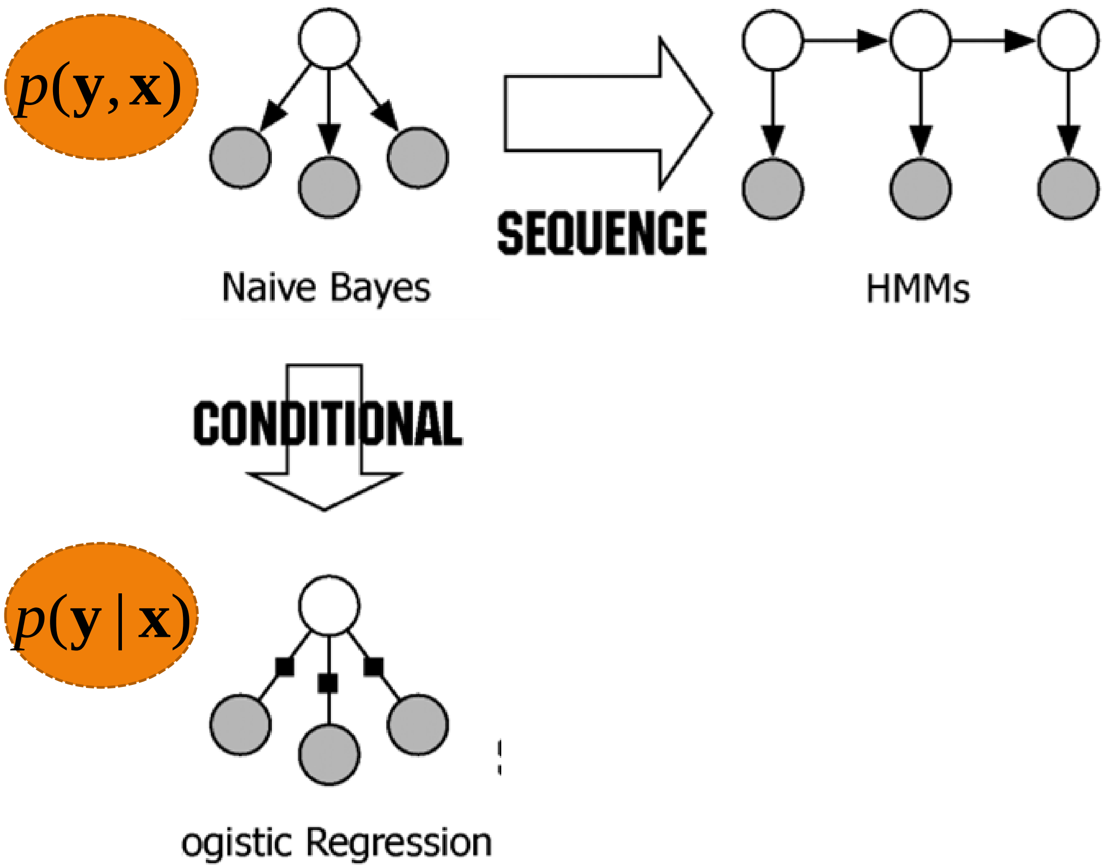
$p(\mathbf{y} | \mathbf{x})$

Discriminative Vs. Generative



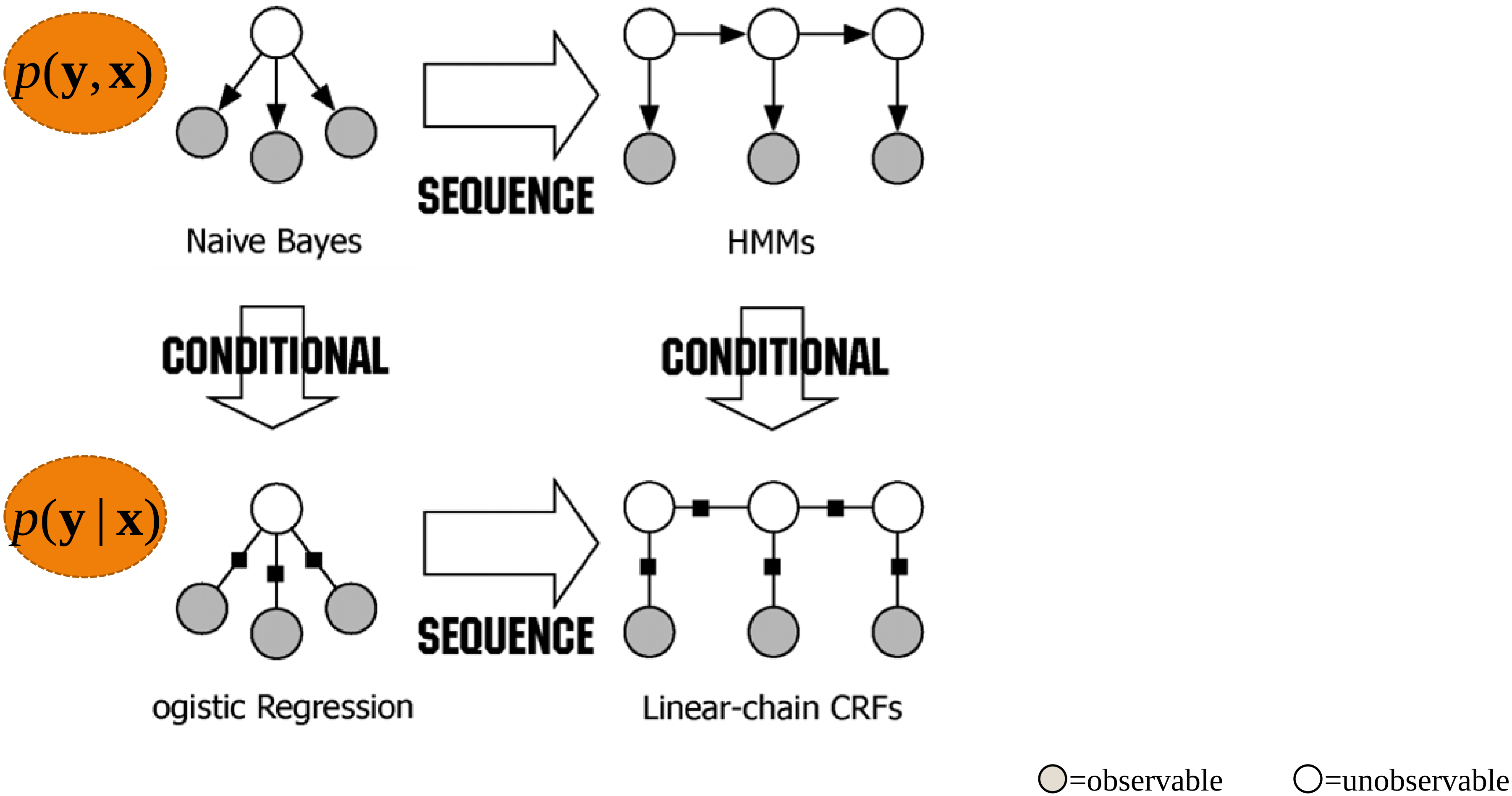
=observable =unobservable

Discriminative Vs. Generative

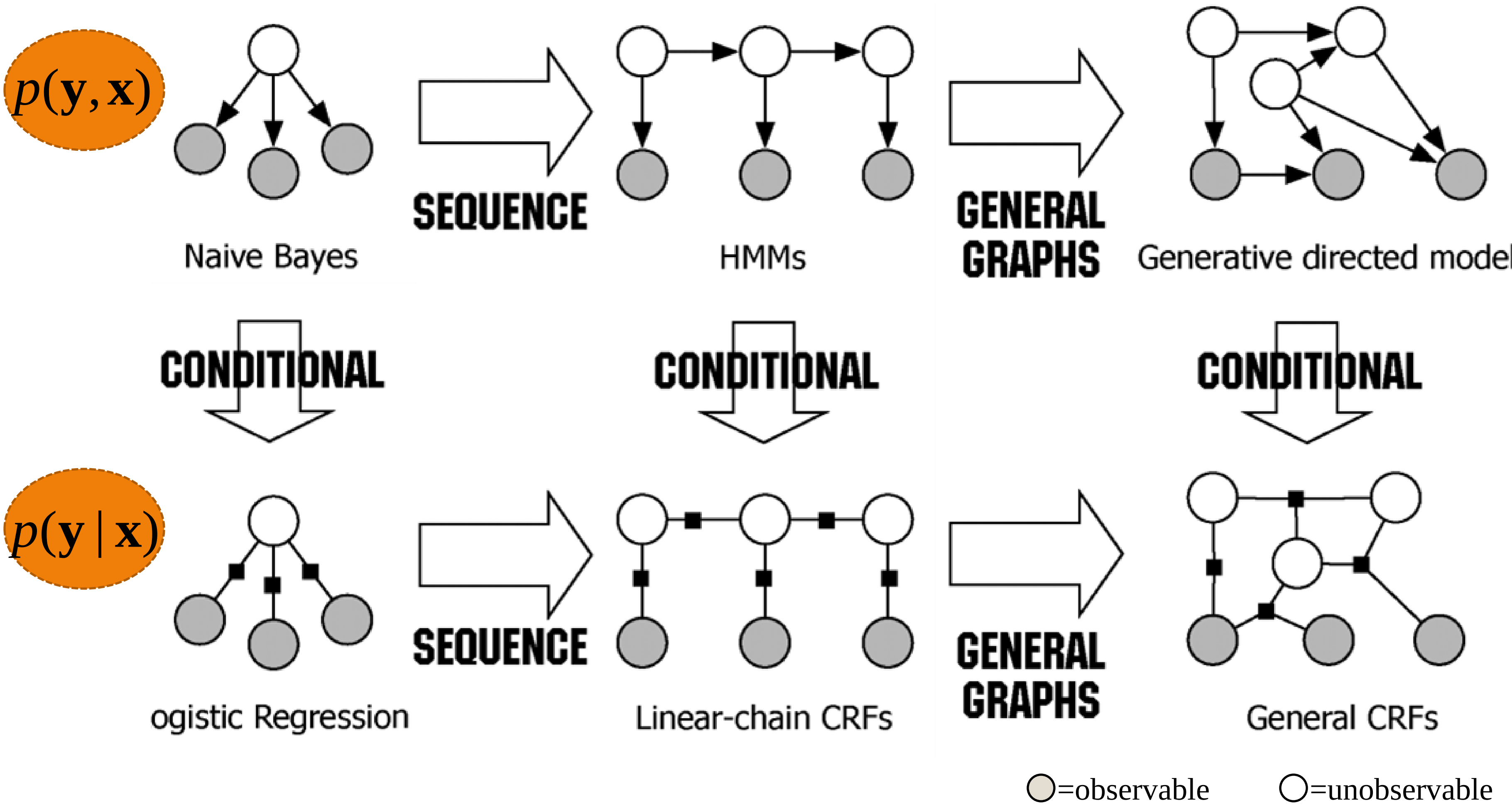


=observable =unobservable

Discriminative Vs. Generative



Discriminative Vs. Generative

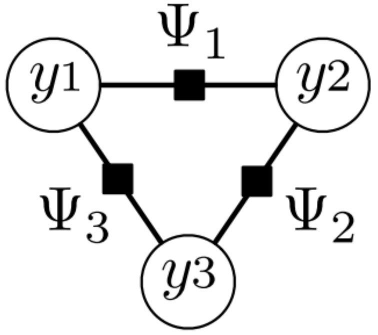


Markov Random Field(MRF) and Factor Graphs

- On an undirected graph, the joint distribution of variables $\mathbf{y}$

$$p(\mathbf{y}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{y}_C), \;\; Z = \sum_{\mathbf{y}}\prod_C \psi_C(\mathbf{y}_C)$$

  - $\psi_C(\mathbf{y}_C) \geq 0$:Potential function
  - Typically : $\psi_C(\mathbf{y}_C) = \exp\{-E(\mathbf{y}_C)\}$
  - Z :Partition function

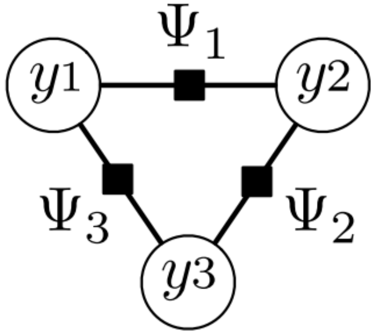


Markov Random Field(MRF) and Factor Graphs

- On an undirected graph, the joint distribution of variables $\mathbf{y}$

$$p(\mathbf{y}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{y}_C),\ \ Z = \sum_{\mathbf{y}}\prod_C \psi_C(\mathbf{y}_C)$$

  - $\psi_C(\mathbf{y}_C) \geq 0$:Potential function
  - Typically : $\psi_C(\mathbf{y}_C) = \exp\{-E(\mathbf{y}_C)\}$
  - Z :Partition function



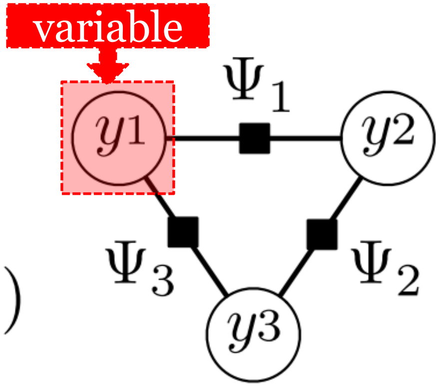
$$p(y_1, y_2, y_3) \propto \Psi_1(y_1, y_2)\Psi_2(y_2, y_3)\Psi_3(y_1, y_3)$$

Markov Random Field(MRF) and Factor Graphs

- On an undirected graph, the joint distribution of variables $\mathbf{y}$

$$p(\mathbf{y}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{y}_C), \quad Z = \sum_{\mathbf{y}}\prod_C \psi_C(\mathbf{y}_C)$$

  - $\psi_C(\mathbf{y}_C) \geq 0$:Potential function
  - Typically : $\psi_C(\mathbf{y}_C) = \exp\{-E(\mathbf{y}_C)\}$
  - Z :Partition function



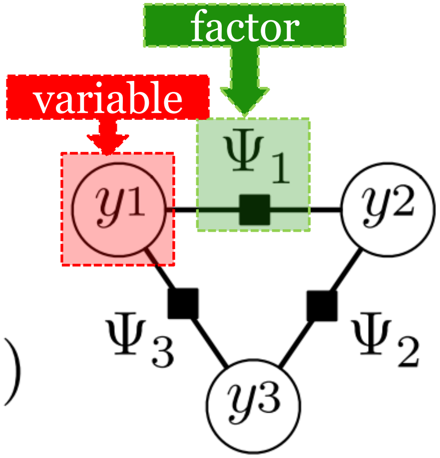
$$p(y_1, y_2, y_3) \propto \Psi_1(y_1, y_2)\Psi_2(y_2, y_3)\Psi_3(y_1, y_3)$$

Markov Random Field(MRF) and Factor Graphs

- On an undirected graph, the joint distribution of variables $\mathbf{y}$

$$p(\mathbf{y}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{y}_C), \quad Z = \sum_{\mathbf{y}}\prod_C \psi_C(\mathbf{y}_C)$$

  - $\psi_C(\mathbf{y}_C) \geq 0$:Potential function
  - Typically : $\psi_C(\mathbf{y}_C) = \exp\{-E(\mathbf{y}_C)\}$
  - Z :Partition function



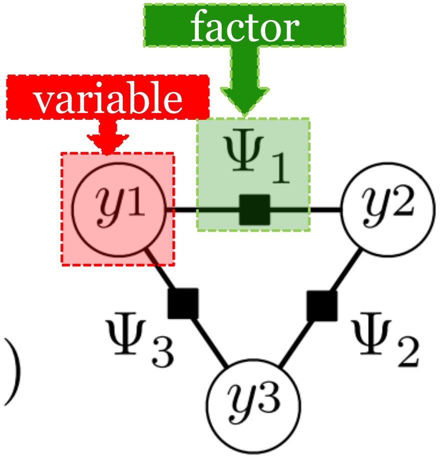
$$p(y_1, y_2, y_3) \propto \Psi_1(y_1, y_2)\Psi_2(y_2, y_3)\Psi_3(y_1, y_3)$$

Markov Random Field(MRF) and Factor Graphs

- On an undirected graph, the joint distribution of variables $\mathbf{y}$

$$p(\mathbf{y}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{y}_C),\ \ Z = \sum_{\mathbf{y}}\prod_C \psi_C(\mathbf{y}_C)$$

  - $\psi_C(\mathbf{y}_C) \geq 0$:Potential function
  - Typically : $\psi_C(\mathbf{y}_C) = \exp\{-E(\mathbf{y}_C)\}$
  - Z :Partition function



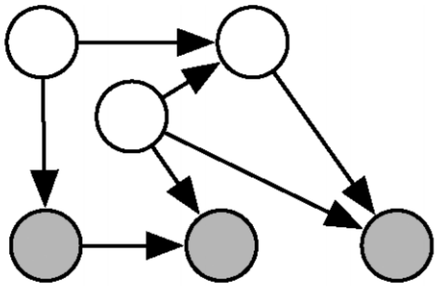
$$p(y_1, y_2, y_3) \propto \Psi_1(y_1, y_2)\Psi_2(y_2, y_3)\Psi_3(y_1, y_3)$$

- Not all distributions satisfy Markovian properties
  - Hammersley-Clifford Theorem
    - The ones which do can be factorized

Directed Graphical Models(Bayesian Network)

- Local conditional distributions
  - If $\pi(s)$ indices of the parents of y_s

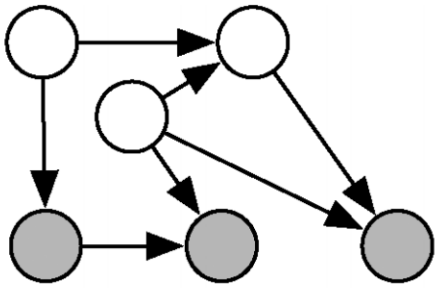
$$p(\mathbf{y}) = \prod_{s=1}^{S} p(y_s | \mathbf{y}_{\pi(s)})$$



Directed Graphical Models(Bayesian Network)

- Local conditional distributions
  - If $\pi(s)$ indices of the parents of y_s

$$p(\mathbf{y}) = \prod_{s=1}^{S} p(y_s|\mathbf{y}_{\pi(s)}).$$

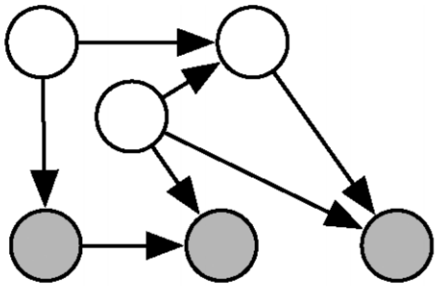


- Generally used as generative models
  - E.g. Naïve Bayes: once the class label is known, all the features are independent

Directed Graphical Models(Bayesian Network)

- Local conditional distributions
  - If $\pi(s)$ indices of the parents of y_s

$$p(\mathbf{y}) = \prod_{s=1}^{S} p(y_s | \mathbf{y}_{\pi(s)})$$



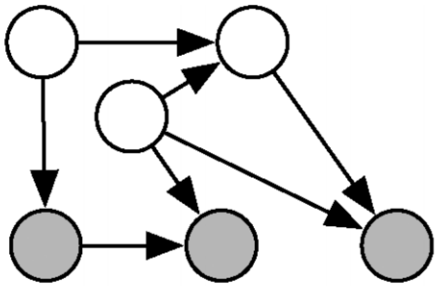
- Generally used as generative models
  - E.g. Naïve Bayes: once the class label is known, all the features are independent

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k | y)$$

Directed Graphical Models(Bayesian Network)

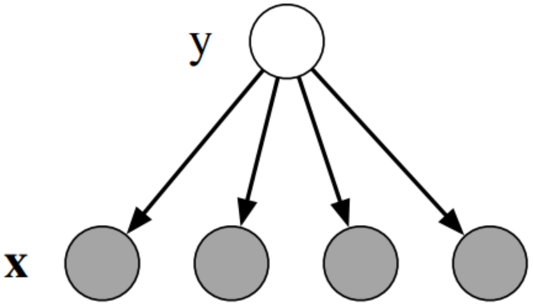
- Local conditional distributions
  - If $\pi(s)$ indices of the parents of y_s

$$p(\mathbf{y}) = \prod_{s=1}^{S} p(y_s|\mathbf{y}_{\pi(s)})$$



- Generally used as generative models
  - E.g. Naïve Bayes: once the class label is known, all the features are independent

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k|y)$$



Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation,

Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, → $X = \{x_t\}_{t=1}^{\mathrm{T}}$

Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, → $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states,

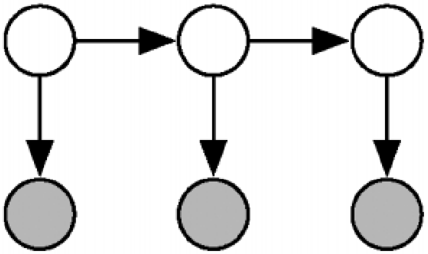
Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, → $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states, → $Y = \{y_t\}_{t=1}^{T}$

Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states, $Y = \{y_t\}_{t=1}^{T}$
- HMM is generative:

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} p(y_t | y_{t-1}) p(x_t | y_t)$$

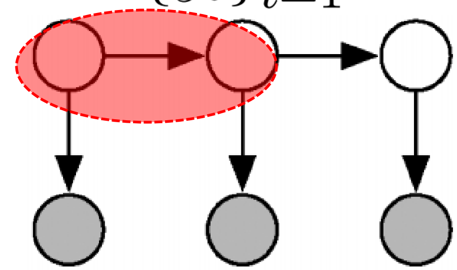


Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states, $Y = \{y_t\}_{t=1}^{T}$
- HMM is generative:

Transition probability

$$p(\mathbf{y},\mathbf{x}) = \prod_{t=1} p(y_t|y_{t-1})p(x_t|y_t)$$



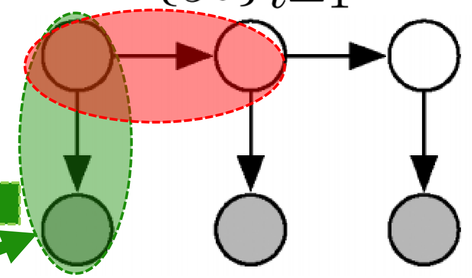
Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states. $Y = \{y_t\}_{t=1}^{T}$
- HMM is generative:

Transition probability

$$p(\mathbf{y},\mathbf{x}) = \prod_{t=1} p(y_t|y_{t-1})p(x_t|y_t)$$

Observation probability



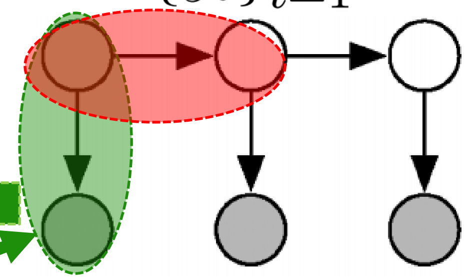
Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states. $Y = \{y_t\}_{t=1}^{T}$
- HMM is generative:

Transition probability

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1} p(y_t | y_{t-1}) p(x_t | y_t)$$

Observation probability



- Doesn't model long-range dependencies
- Not practical to represent multiple interacting features (hard to model p(x))

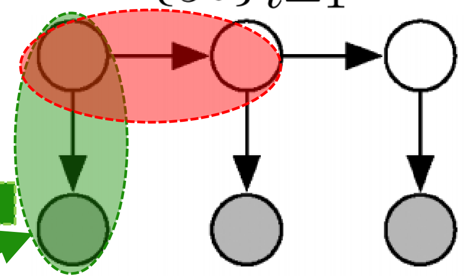
Sequence prediction

- Like NER: identifying and classifying proper names in text, e.g. China as location; George Bush as people; United Nations as organizations
  - Set of observation, $X = \{x_t\}_{t=1}^{T}$
  - Set of underlying sequence of states, $Y = \{y_t\}_{t=1}^{T}$
- HMM is generative:

Transition probability

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1} p(y_t | y_{t-1}) p(x_t | y_t)$$

Observation probability

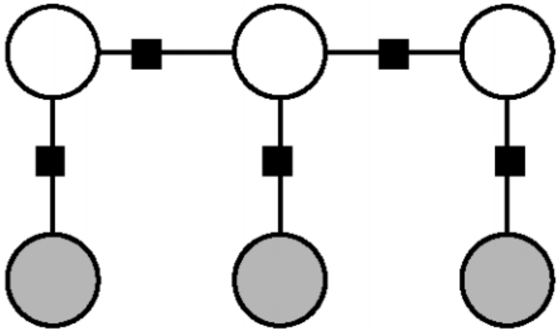


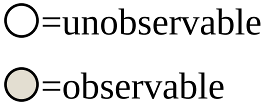
- Doesn't model long-range dependencies
- Not practical to represent multiple interacting features (hard to model p(x))
- The primary advantage of CRFs over hidden Markov models is their conditional nature, resulting in the relaxation of the independence assumption
- And it can handle overlapping features

Chain CRFs

- Each potential function will operate on pairs of adjacent label variables

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_j \lambda_j F_j(\boldsymbol{y},\boldsymbol{x})\right)$$



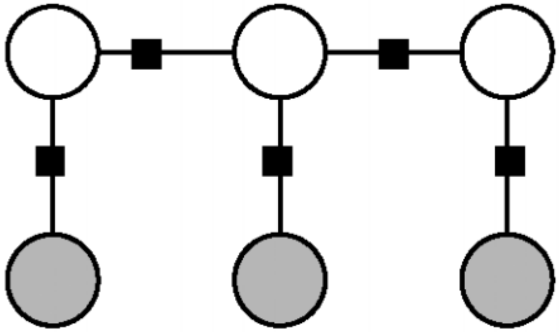


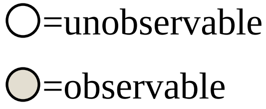
Chain CRFs

- Each potential function will operate on pairs of adjacent label variables

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})}\exp\left(\sum_j \lambda_j F_j(\boldsymbol{y},\boldsymbol{x})\right)$$

$$F_j(\boldsymbol{y},\boldsymbol{x}) = \sum_{i=1} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$





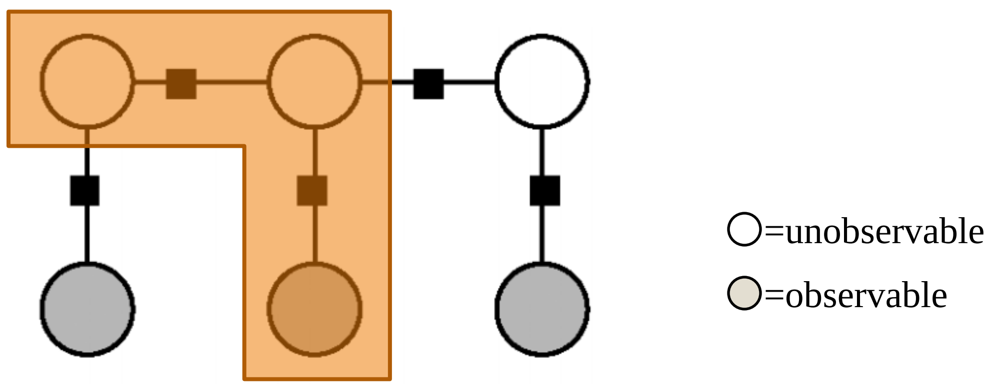
Chain CRFs

- Each potential function will operate on pairs of adjacent label variables

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_j \lambda_j F_j(\boldsymbol{y}, \boldsymbol{x})\right)$$

$$F_j(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$

Feature functions



Chain CRFs

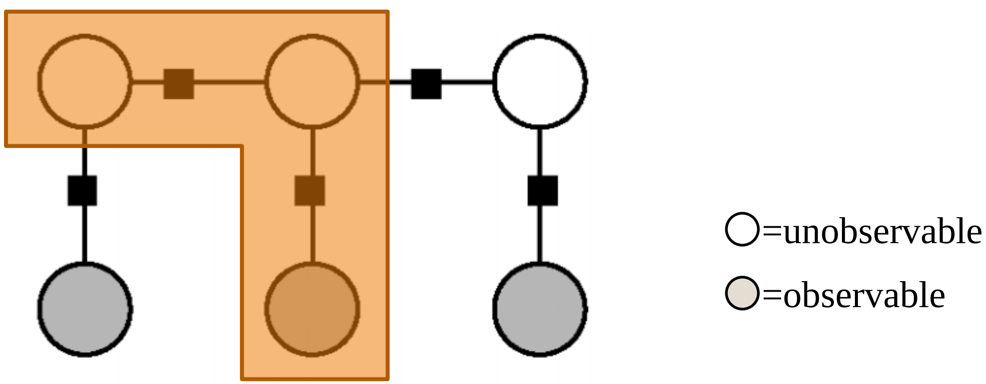
- Each potential function will operate on pairs of adjacent label variables

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})}\exp\left(\sum_j \lambda_j F_j(\boldsymbol{y},\boldsymbol{x})\right)$$

$$F_j(\boldsymbol{y},\boldsymbol{x}) = \sum_{i=1} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$

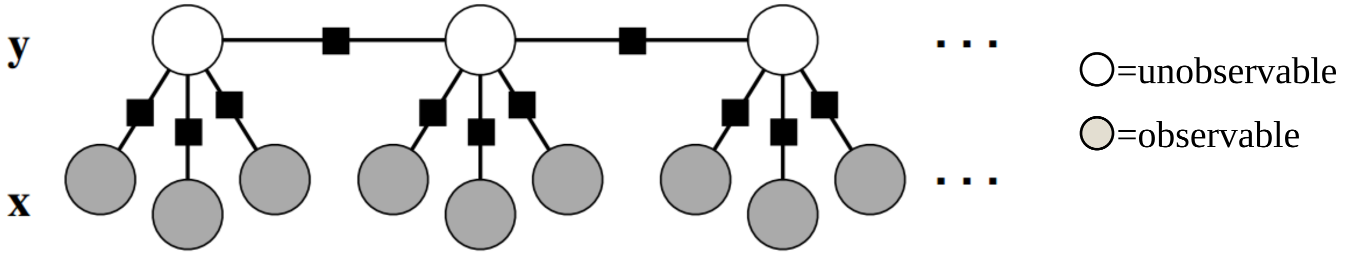
Feature functions

- Parameters to be estimated, λ_j



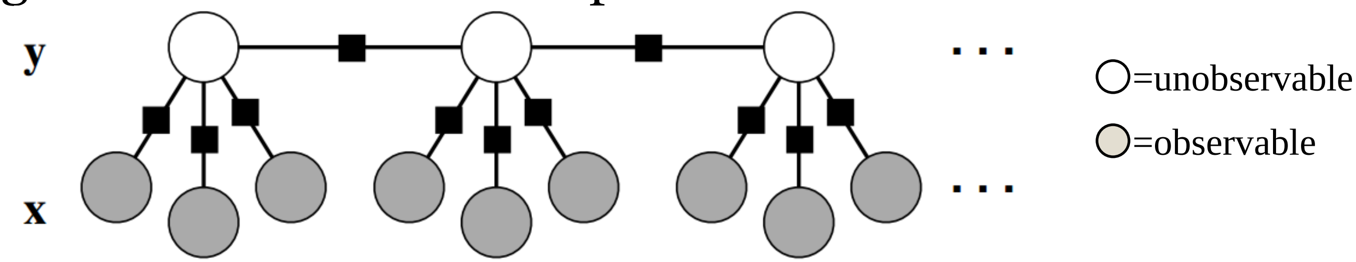
Chain CRF

- We can change it so that each state depends on more observations

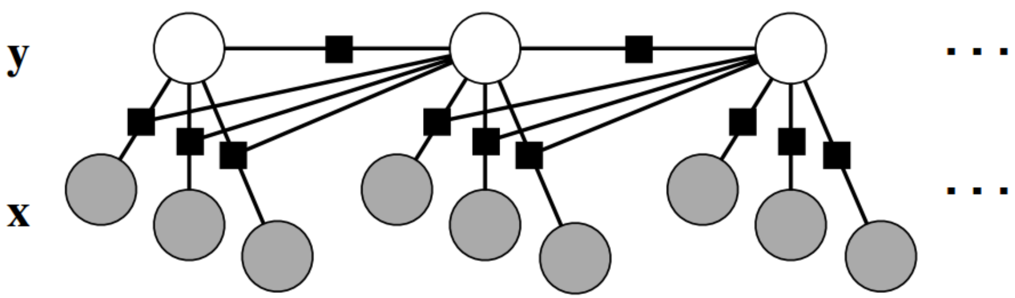


Chain CRF

- We can change it so that each state depends on more observations

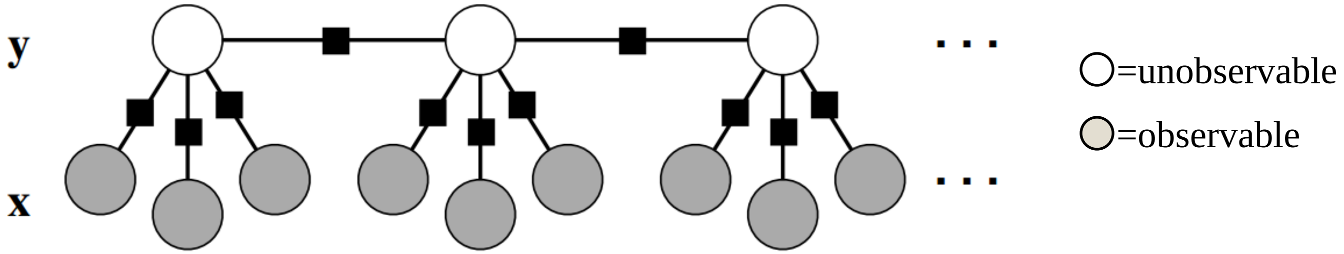


- Or inputs at previous steps

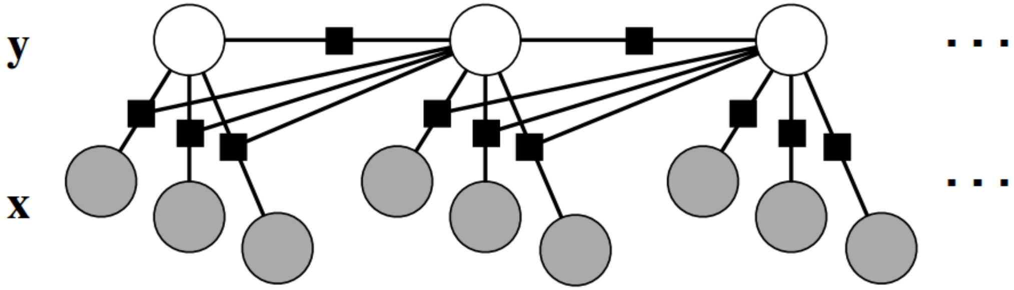


Chain CRF

- We can change it so that each state depends on more observations



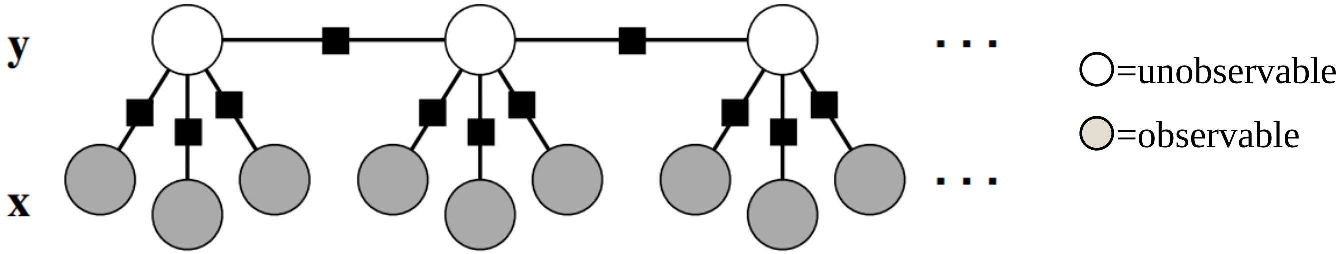
- Or inputs at previous steps



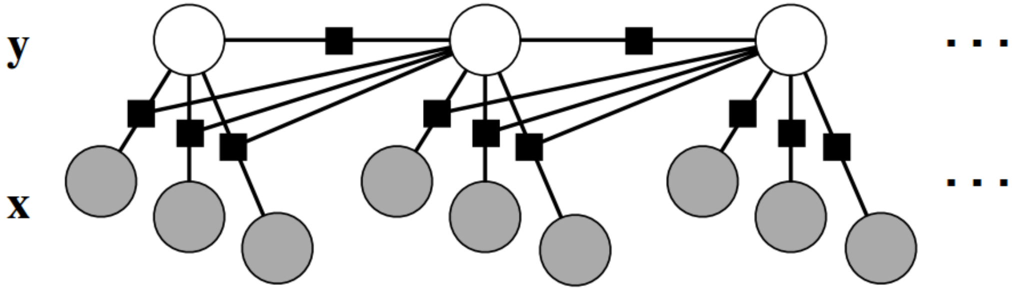
- Or all inputs

Chain CRF

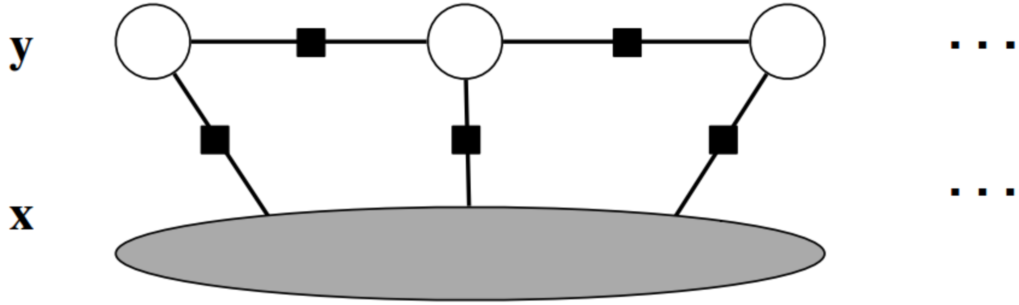
- We can change it so that each state depends on more observations



- Or inputs at previous steps

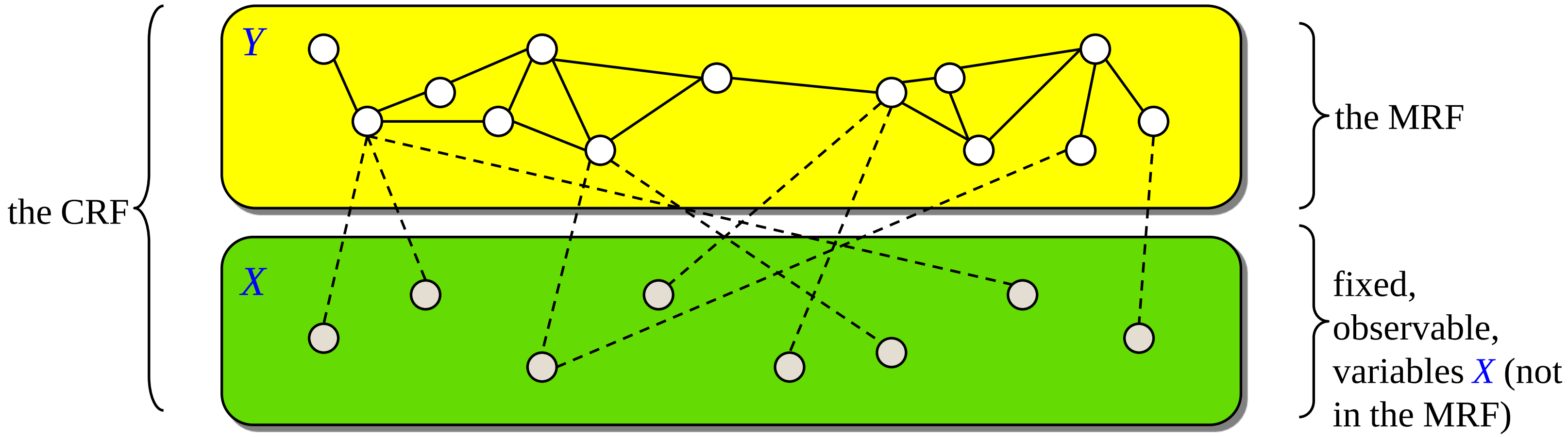


- Or all inputs



General CRF: visualization

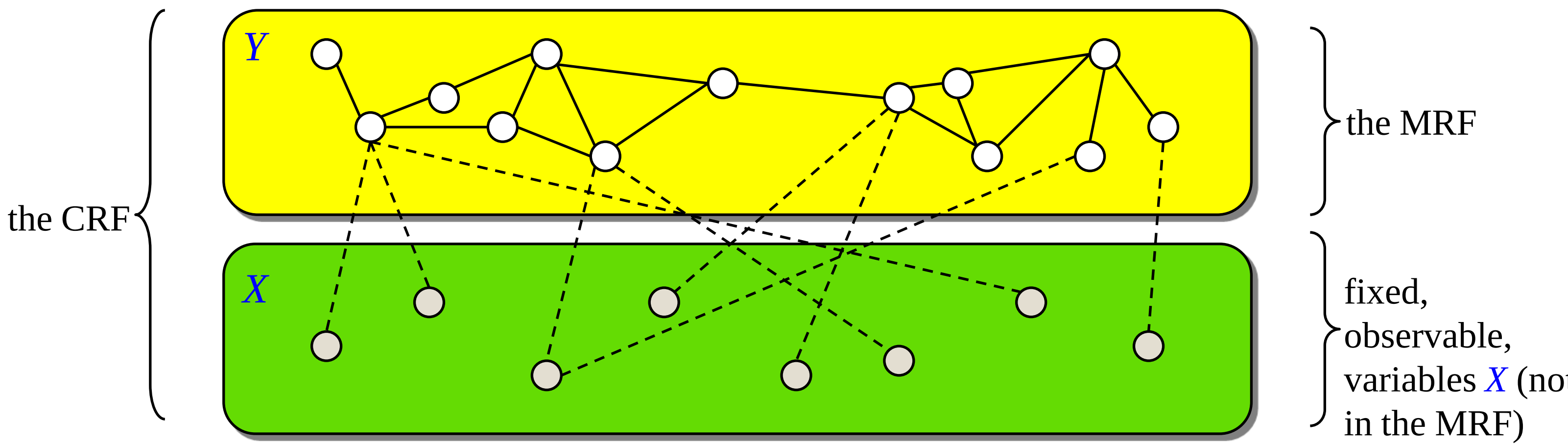
- If $G = (V, E)$, and $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$;
  $w \sim v \Leftrightarrow w$ and v are neighbors
  $(\mathbf{X}, \mathbf{Y})$ is a CRF, if $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \sim v)$;



General CRF: visualization

- If $G = (V, E)$, and $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$;
  $w \sim v \Leftrightarrow w$ and v are neighbors

  $(\mathbf{X}, \mathbf{Y})$ is a CRF, if $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \sim v)$



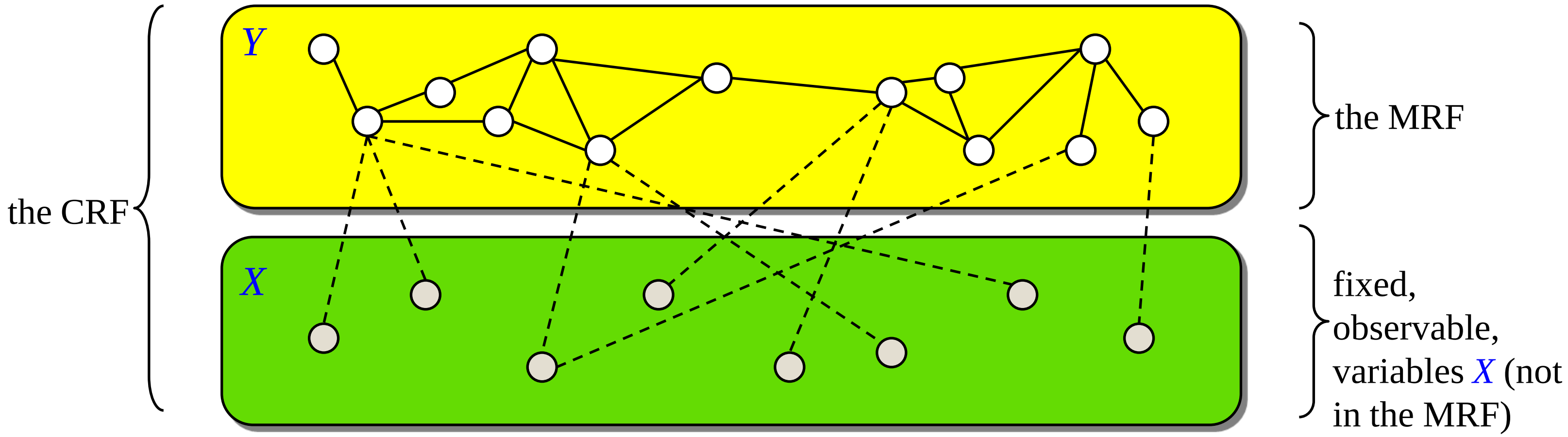
Note that in a CRF *we do not explicitly model any direct relationships between the observables (i.e., among the X)* (Lafferty *et al.*, 2001).

General CRF: visualization

- If $G = (V, E)$, and $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$;

  $w \sim v \Leftrightarrow w$ and v are neighbors

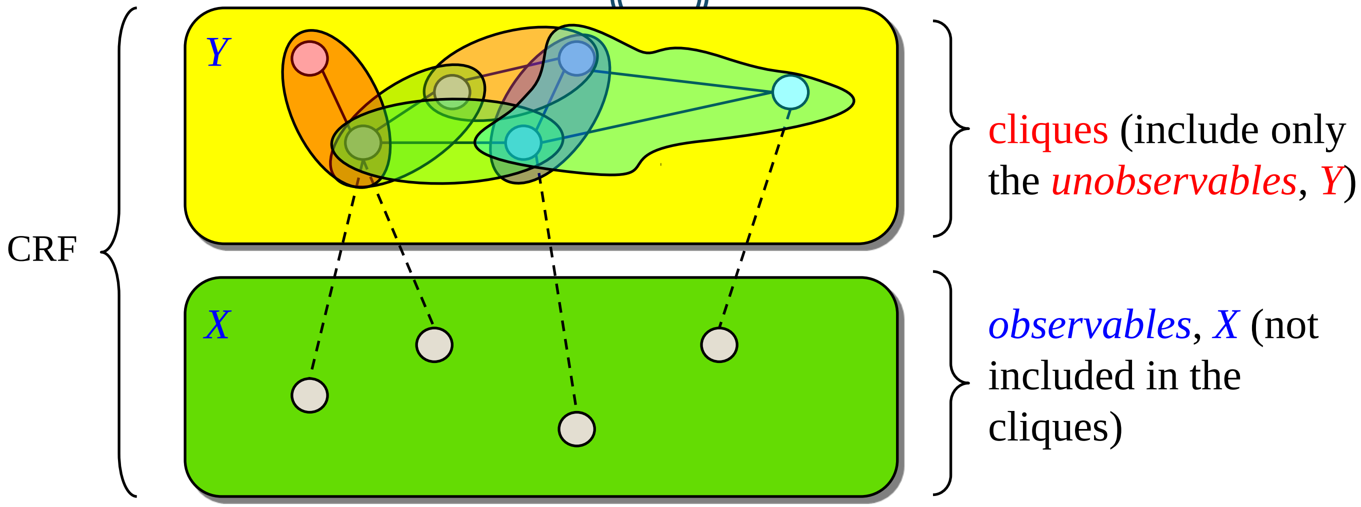
  $(\mathbf{X}, \mathbf{Y})$ is a CRF, if $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \sim v)$;



Note that in a CRF *we do not explicitly model any direct relationships between the observables (i.e., among the X)* (Lafferty *et al.*, 2001).

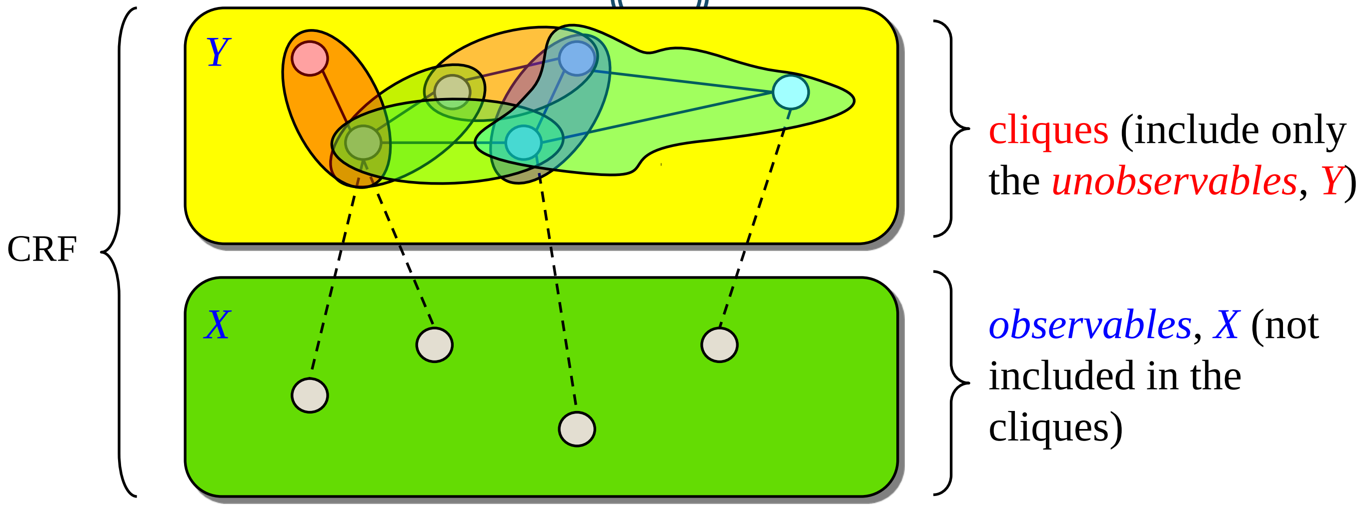
Hammersley-Clifford does not apply to X!

General CRF: visualization



- Divide **y** MRF into cliques. The parameters inside each template are tied $\Phi_c(\mathbf{y}_c, \mathbf{x})$--*potential functions*; functions for the template

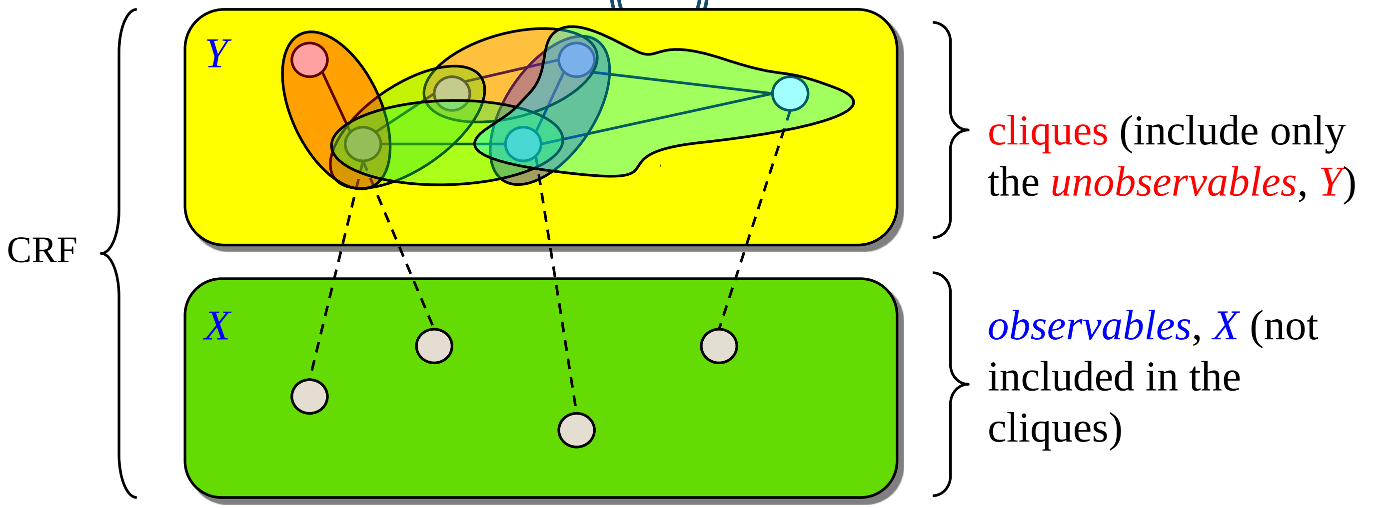
General CRF: visualization



- Divide $\mathbf{y}$ MRF into cliques. The parameters inside each template are tied $\Phi_c(\mathbf{y}_c, \mathbf{x})$--*potential functions*; functions for the template

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{x})} = \frac{1}{\sum_{\mathbf{y}'} e^{Q(\mathbf{y}',\mathbf{x})}} e^{Q(\mathbf{y},\mathbf{x})}, \quad Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

General CRF: visualization

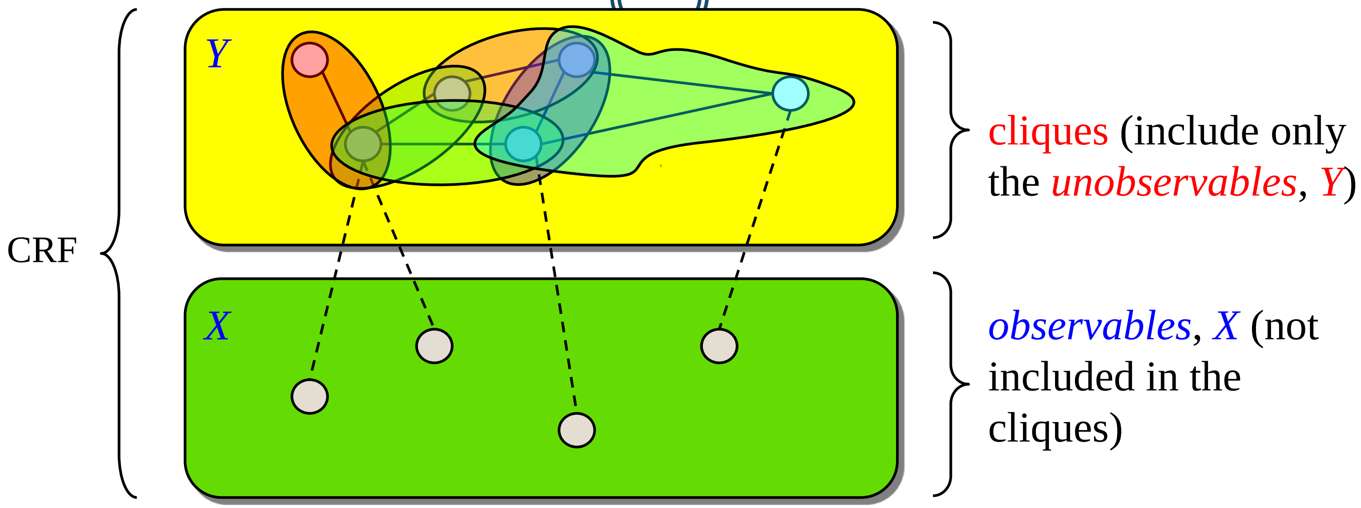


- Divide $\mathbf{y}$ MRF into cliques. The parameters inside each template are tied $\Phi_c(\mathbf{y}_c, \mathbf{x})$--*potential functions*; functions for the template

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{x})} = \frac{1}{\sum_{\mathbf{y}'} e^{Q(\mathbf{y}',\mathbf{x})}} e^{Q(\mathbf{y},\mathbf{x})}, \quad Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

Note that we are not summing over **x** in the denominator

General CRF: visualization



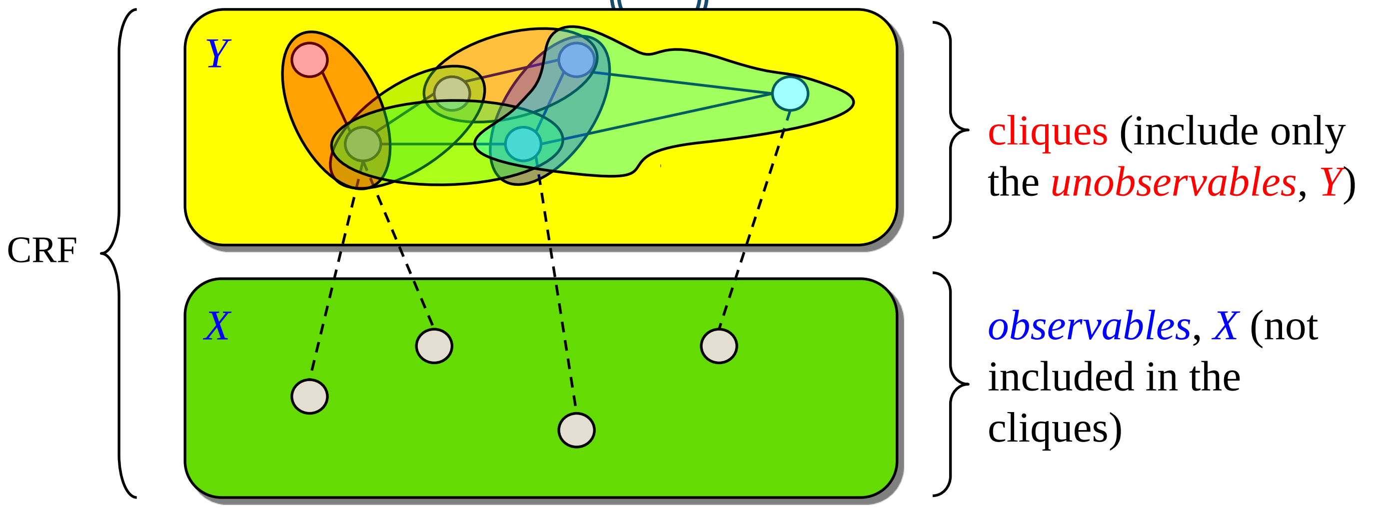
- Divide **y** MRF into cliques. The parameters inside each template are tied $\Phi_c(\mathbf{y}_c, \mathbf{x})$--*potential functions*; functions for the template

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{x})} = \frac{1}{\sum_{\mathbf{y}'} e^{Q(\mathbf{y}',\mathbf{x})}} e^{Q(\mathbf{y},\mathbf{x})}, \quad Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

Note that we are not summing over **x** in the denominator

- The cliques contain only unobservables (**y**); though, **x** is an argument to Φ_c

General CRF: visualization



- Divide $\mathbf{y}$ MRF into cliques. The parameters inside each template are tied $\Phi_c(\mathbf{y}_c, \mathbf{x})$--*potential functions*; functions for the template

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{x})} = \frac{1}{\sum_{\mathbf{y}'} e^{Q(\mathbf{y}',\mathbf{x})}} e^{Q(\mathbf{y},\mathbf{x})}, \quad Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

Note that we are not summing over $\mathbf{x}$ in the denominator

- The cliques contain only unobservables ($\mathbf{y}$); though, $\mathbf{x}$ is an argument to Φ_c
- The probability $P_M(\mathbf{y}|\mathbf{x})$ is a *joint distribution* over the unobservables Y

General CRF: visualization

- A number of *ad hoc* modeling decisions are typically made with regard to the form of the potential functions.
- Φ_c is typically decomposed into a weighted sum of feature sensors f_i, producing:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{Q(\mathbf{y}, \mathbf{x})}$$

$$Q(\mathbf{y}, \mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

General CRF: visualization

- A number of *ad hoc* modeling decisions are typically made with regard to the form of the potential functions.
- Φ_c is typically decomposed into a weighted sum of feature sensors f_i, producing:

$$\left.\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \frac{1}{Z} e^{Q(\mathbf{y},\mathbf{x})} \\ Q(\mathbf{y},\mathbf{x}) &= \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x}) \\ \Phi_c(\mathbf{y}_c, \mathbf{x}) &= \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x}) \end{aligned}\right\}$$

General CRF: visualization

- A number of *ad hoc* modeling decisions are typically made with regard to the form of the potential functions.
- Φ_c is typically decomposed into a weighted sum of feature sensors f_i, producing:

$$\left.\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \frac{1}{Z} e^{Q(\mathbf{y},\mathbf{x})} \\ Q(\mathbf{y},\mathbf{x}) &= \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x}) \\ \Phi_c(\mathbf{y}_c, \mathbf{x}) &= \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x}) \end{aligned}\right\} \quad P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x})}$$

General CRF: visualization

- A number of *ad hoc* modeling decisions are typically made with regard to the form of the potential functions.
- Φ_c is typically decomposed into a weighted sum of feature sensors f_i, producing:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{Q(\mathbf{y},\mathbf{x})}$$

$$Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

$$\Phi_c(\mathbf{y}_c, \mathbf{x}) = \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x})$$

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x})}$$

- Back to the chain-CRF!

Cliques can be identified as *pairs* of adjacent Ys:

General CRF: visualization

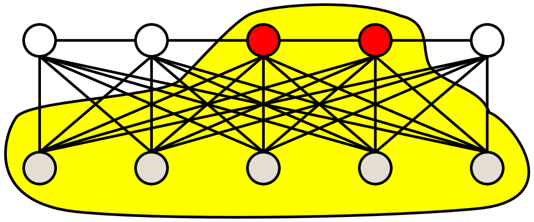
- A number of *ad hoc* modeling decisions are typically made with regard to the form of the potential functions.
- Φ_c is typically decomposed into a weighted sum of feature sensors f_i, producing:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{Q(\mathbf{y},\mathbf{x})}$$

$$Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

$$\Phi_c(\mathbf{y}_c, \mathbf{x}) = \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x})$$

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x})}$$



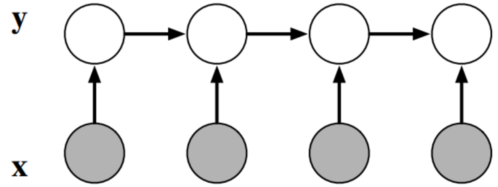
- Back to the chain-CRF!

Cliques can be identified as *pairs* of adjacent Ys:

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_j \lambda_j F_j(\boldsymbol{y},\boldsymbol{x})\right) \qquad F_j(\boldsymbol{y},\boldsymbol{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$

Chain CRFs vs. MEMM

- Linear-chain CRFs were originally introduced as an improvement to MEMM
- Maximum Entropy Markov Models (MEMM)
  - Transition probabilities are given by logistic regression



$$p_{\text{MEMM}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p(y_t|y_{t-1},\mathbf{x})$$

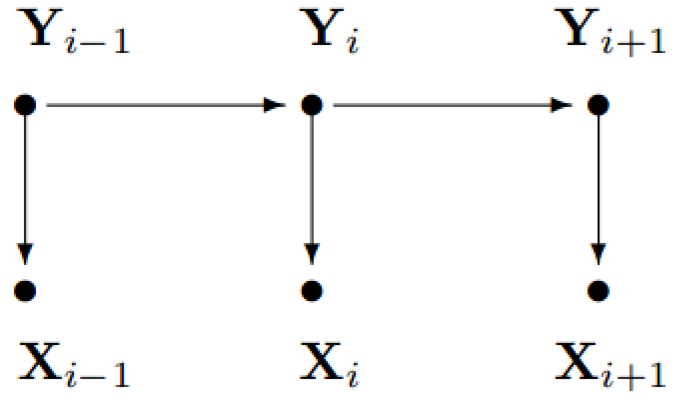
$$p(y_t|y_{t-1},\mathbf{x}) = \frac{1}{Z_t(y_{t-1},\mathbf{x})} \exp\left\{\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}$$

$$Z_t(y_{t-1},\mathbf{x}) = \sum_{y'} \exp\left\{\sum_{k=1}^{K} \theta_k f_k(y', y_{t-1}, \mathbf{x}_t)\right\}$$

- Notice the per-state normalization
- Only dependent on the previous inputs; no dependence on the future states.
  - Label-bias problem

CRFs vs. MEMM vs. HMM

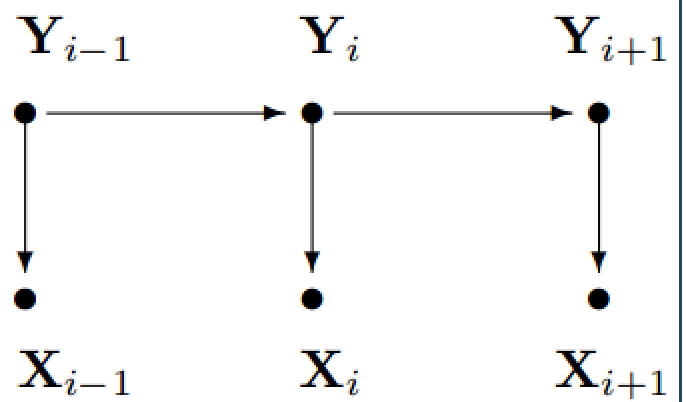
- HMM



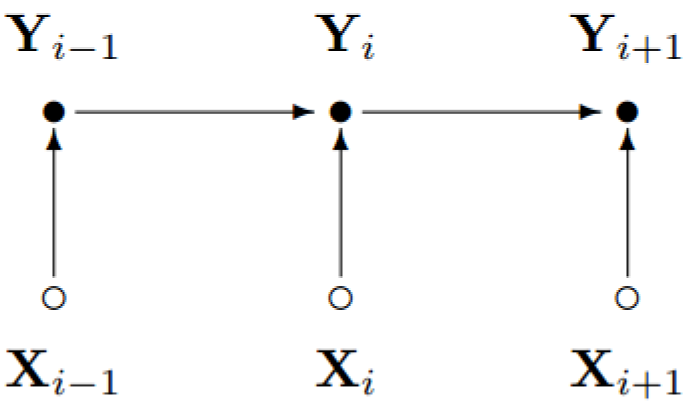
- MEMM
- CRF

CRFs vs. MEMM vs. HMM

- HMM



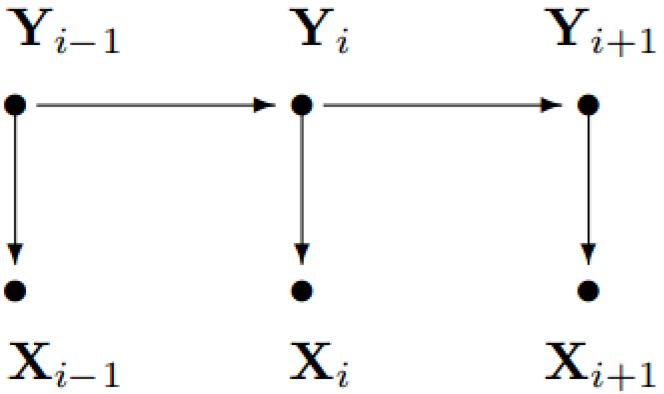
- MEMM



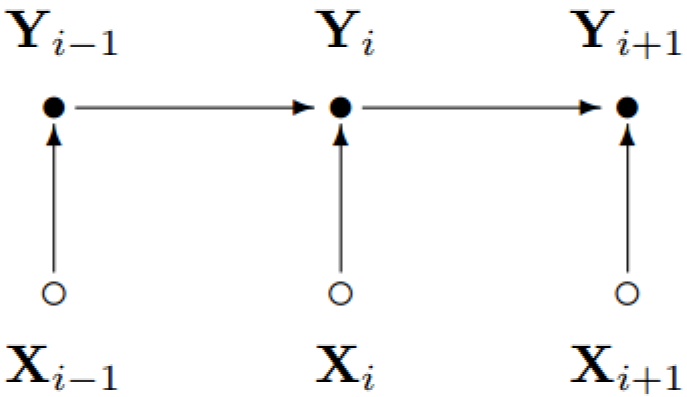
- CRF

CRFs vs. MEMM vs. HMM

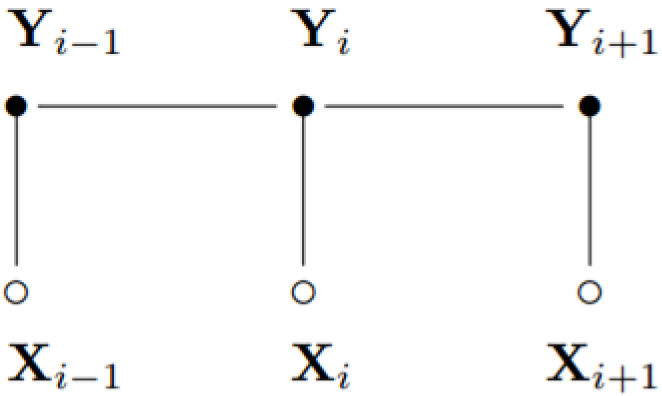
- HMM



- MEMM



- CRF



Outline

- Modeling
- Inference
  - General CRF
  - Chain CRF
- Training
- Applications

Inference

Inference

- Given the observations,{xi})and parameters, we target to find the best state sequence

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}).$$

Inference

- Given the observations,{xi})and parameters, we target to find the best state sequence $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$.
- For the general CRF:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y}} \frac{1}{Z} e^{\sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})} = \arg\max_{\mathbf{y}} \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

Inference

- Given the observations,{xi})and parameters, we target to find the best state sequence

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$$

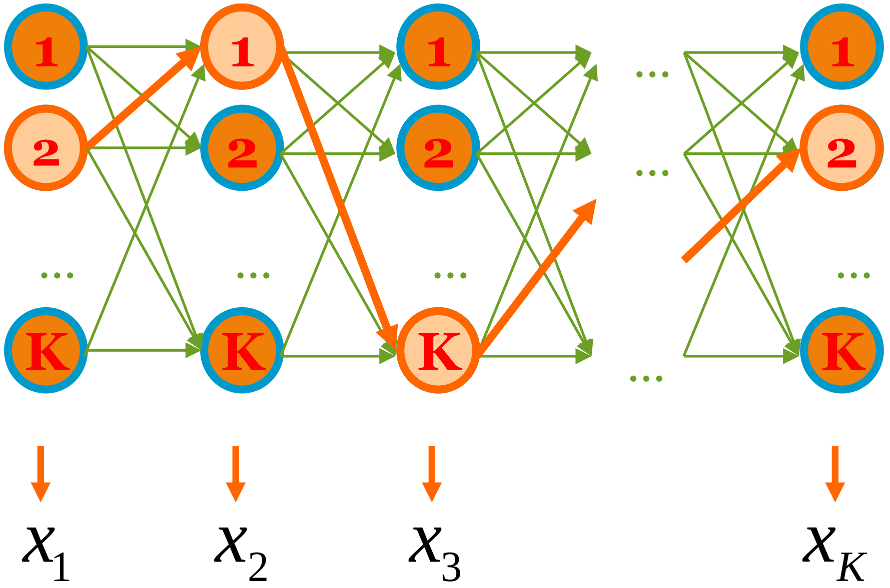
- For the general CRF:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y}} \frac{1}{Z} e^{\sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})} = \arg\max_{\mathbf{y}} \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

- For general graphs, the problem of exact inference in CRFs is intractable
  - Approximate methods ! A large literature …

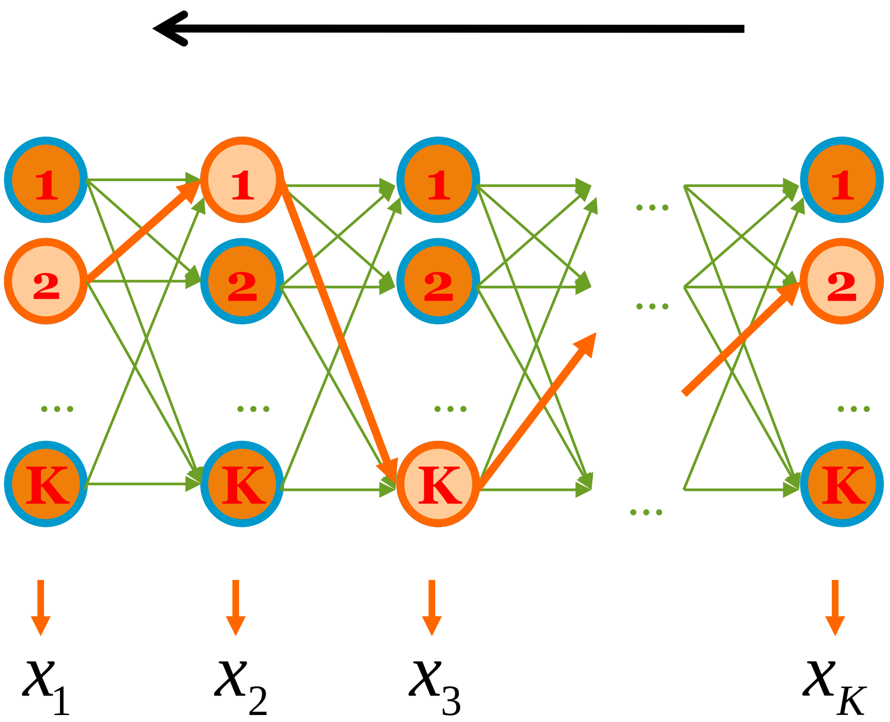
Inference in HMM

- Dynamic Programming:



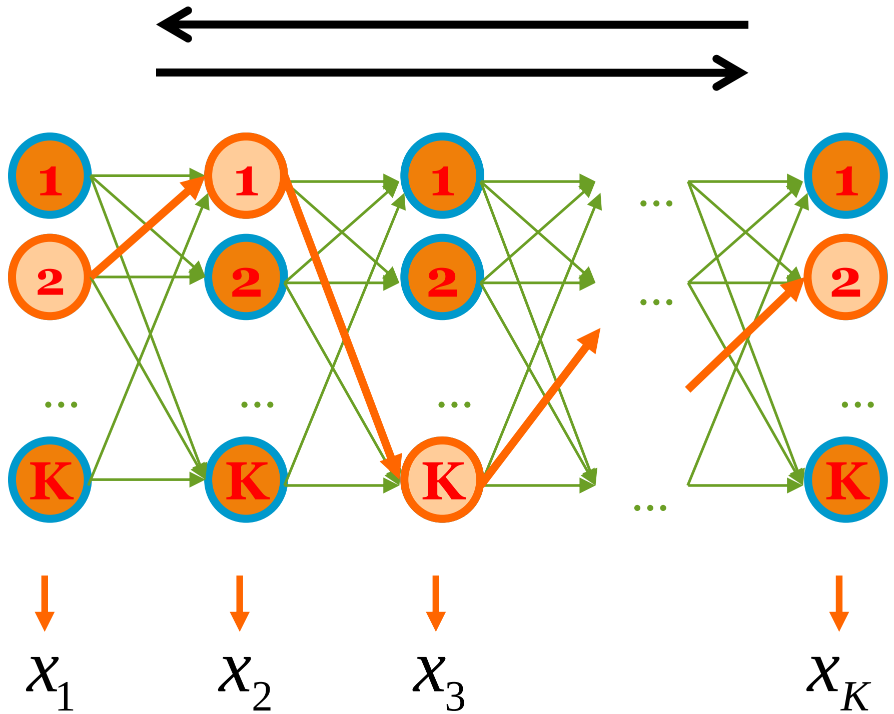
Inference in HMM

- Dynamic Programming:
  - Forward



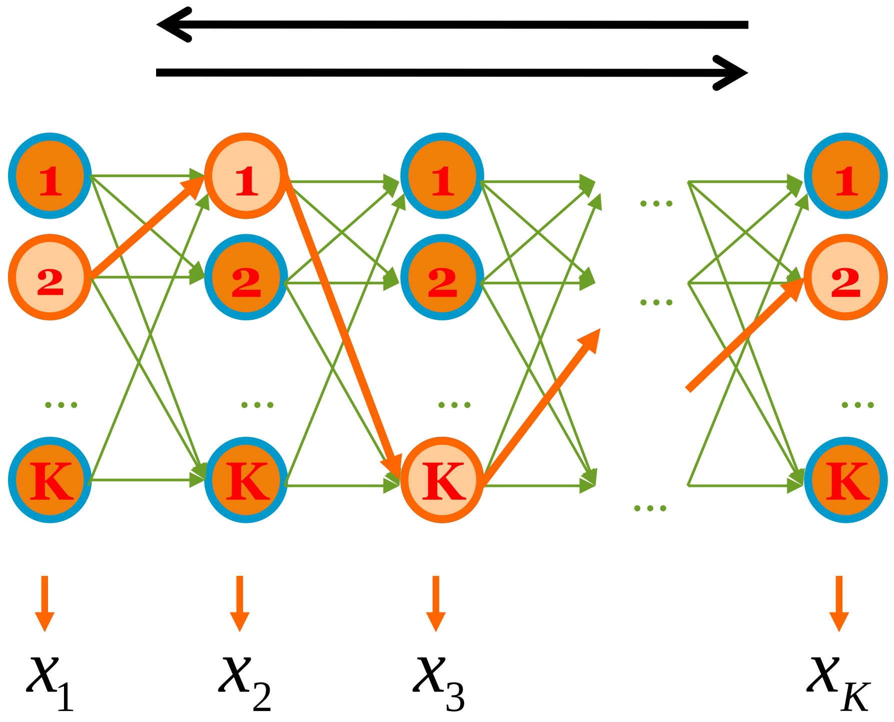
Inference in HMM

- Dynamic Programming:
  - Forward
  - Backward



Inference in HMM

- Dynamic Programming:
  - Forward
  - Backward
  - Viterbi



Parameter Learning: Chain CRF

- Chain CRF could be done using dynamic programming

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})}\exp\big(\sum_j \lambda_j F_j(\boldsymbol{y},\boldsymbol{x})\big) \qquad F_j(\boldsymbol{y},\boldsymbol{x}) = \sum_{i=1}^{n} f_j(y_{i-1},y_i,\boldsymbol{x},i),$$

Parameter Learning: Chain CRF

- Chain CRF could be done using dynamic programming

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_j \lambda_j F_j(\boldsymbol{y}, \boldsymbol{x})\right) \quad F_j(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$

- Assume $y \in \mathcal{Y}$
- Naively doing could be intractable: $n^{|\mathcal{Y}|}$

Parameter Learning: Chain CRF

- Chain CRF could be done using dynamic programming

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_j \lambda_j F_j(\boldsymbol{y}, \boldsymbol{x})\right) \qquad F_j(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$

- Assume $y \in \mathcal{Y}$
- Naively doing could be intractable: $n^{|\mathcal{Y}|}$
- Define a matrix $\{M_i(\boldsymbol{x}) | i = 1, \ldots, n+1\}$ with size $|\mathcal{Y} \times \mathcal{Y}|$

$$M_i(y', y|\boldsymbol{x}) = \exp\left(\sum_j \lambda_j f_j(y', y, \boldsymbol{x}, i)\right)$$

Parameter Learning: Chain CRF

- Chain CRF could be done using dynamic programming

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_j \lambda_j F_j(\boldsymbol{y},\boldsymbol{x})\right) \quad F_j(\boldsymbol{y},\boldsymbol{x}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \boldsymbol{x}, i),$$

- Assume $y \in \mathcal{Y}$
- Naively doing could be intractable: $n^{|\mathcal{Y}|}$
- Define a matrix $\{M_i(\boldsymbol{x}) | i = 1, \ldots, n+1\}$ with size $|\mathcal{Y} \times \mathcal{Y}|$

$$M_i(y', y|\boldsymbol{x}) = \exp\left(\sum_j \lambda_j f_j(y', y, \boldsymbol{x}, i)\right)$$

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{x})} \prod_{i=1}^{n+1} M_i(y_{i-1}, y_i|\boldsymbol{x})$$

$$Z(\boldsymbol{x}) = \left[\prod_{i=1}^{n+1} M_i(\boldsymbol{x})\right]_{\texttt{start},\texttt{end}}$$

Parameter Learning: Chain CRF

- By defining the following forward and backward parameters,

Parameter Learning: Chain CRF

- By defining the following forward and backward parameters,

$$\alpha_i(\boldsymbol{x})^T = \alpha_{i-1}(\boldsymbol{x})^T M_i(\boldsymbol{x})$$

$$\alpha_0(y|\boldsymbol{x}) = \begin{cases} 1 & \text{if } y = \texttt{start} \\ 0 & \text{otherwise} \end{cases}$$

Parameter Learning: Chain CRF

- By defining the following forward and backward parameters,

$$\beta_i(\boldsymbol{x}) = M_{i+1}(\boldsymbol{x})\beta_{i+1}(\boldsymbol{x}) \qquad \alpha_i(\boldsymbol{x})^T = \alpha_{i-1}(\boldsymbol{x})^T M_i(\boldsymbol{x})$$

$$\beta_{n+1}(y|\boldsymbol{x}) = \begin{cases} 1 & \text{if } y = \texttt{stop} \\ 0 & \text{otherwise} \end{cases} \qquad \alpha_0(y|\boldsymbol{x}) = \begin{cases} 1 & \text{if } y = \texttt{start} \\ 0 & \text{otherwise} \end{cases}$$

Parameter Learning: Chain CRF

- By defining the following forward and backward parameters,

$$\beta_i(\boldsymbol{x}) = M_{i+1}(\boldsymbol{x})\beta_{i+1}(\boldsymbol{x}) \qquad \alpha_i(\boldsymbol{x})^T = \alpha_{i-1}(\boldsymbol{x})^T M_i(\boldsymbol{x})$$

$$\beta_{n+1}(y|\boldsymbol{x}) = \begin{cases} 1 & \text{if } y = \texttt{stop} \\ 0 & \text{otherwise} \end{cases} \qquad \alpha_0(y|\boldsymbol{x}) = \begin{cases} 1 & \text{if } y = \texttt{start} \\ 0 & \text{otherwise} \end{cases}$$

$$Z(\mathbf{x}) = \sum_{i \in S} \alpha_{\mathrm{T}}(i). \qquad Z(\mathbf{x}) = \beta_0(y_0)$$

Inference: Chain-CRF

- The inference of linear-chain CRF is very similar to that of HMM
- We can write the marginal distribution:

$$p(Y_{i-1} = y', Y_i = y | \boldsymbol{x}^{(k)}, \boldsymbol{\lambda}) = \frac{\alpha_{i-1}(y'|\boldsymbol{x}) M_i(y', y|\boldsymbol{x}) \beta_i(y|\boldsymbol{x})}{Z(\boldsymbol{x})}$$

- Solve Chain-CRF using Dynamic Programming (Similar to Viterbi)!

Inference: Chain-CRF

- The inference of linear-chain CRF is very similar to that of HMM
- We can write the marginal distribution:

$$p(Y_{i-1} = y', Y_i = y | \boldsymbol{x}^{(k)}, \boldsymbol{\lambda}) = \frac{\alpha_{i-1}(y'|\boldsymbol{x}) M_i(y', y|\boldsymbol{x}) \beta_i(y|\boldsymbol{x})}{Z(\boldsymbol{x})}$$

- Solve Chain-CRF using Dynamic Programming (Similar to Viterbi)!
- 1. First computing α for all t (forward), then compute β for all t (backward).
- 2. Return the marginal distributions computed.
- 3. Run viterbi to find the optimal sequence $n.|\mathcal{Y}|^2$

Outline

- Modeling
- Inference
- Training
  - General CRF
  - Some notes on approximate learning
- Applications

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the model.

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, we wish to learn parameters of the model.
- For chain or tree structured CRFs, they can be trained by maximum likelihood

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, we wish to learn parameters of the model.
- For chain or tree structured CRFs, they can be trained by maximum likelihood
  - The objective function for chain-CRF is convex(see Lafferty et al(2001)).

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, we wish to learn parameters of the model.
- For chain or tree structured CRFs, they can be trained by maximum likelihood
  - The objective function for chain-CRF is convex(see Lafferty et al(2001)).
- General CRFs are intractable hence approximation solutions are necessary

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the mode.

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the mode.
- Conditional log-likelihood for a general CRF:

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the mode.
- Conditional log-likelihood for a general CRF:

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_k \left[\log \frac{1}{Z(\boldsymbol{x}^{(k)})} + \sum_j \lambda_j F_j(\boldsymbol{y}^{(k)}, \boldsymbol{x}^{(k)}) \right]$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_j} = E_{\tilde{p}(\boldsymbol{Y},\boldsymbol{X})} \left[F_j(\boldsymbol{Y}, \boldsymbol{X}) \right] - \sum_k E_{p(\boldsymbol{Y}|\boldsymbol{x}^{(k)},\boldsymbol{\lambda})} \left[F_j(\boldsymbol{Y}, \boldsymbol{x}^{(k)}) \right]$$

Empirical Distribution

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the mode.
- Conditional log-likelihood for a general CRF:

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_{k} \left[\log \frac{1}{Z(\boldsymbol{x}^{(k)})} + \sum_{j} \lambda_j F_j(\boldsymbol{y}^{(k)}, \boldsymbol{x}^{(k)}) \right]$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_j} = E_{\tilde{p}(\boldsymbol{Y}, \boldsymbol{X})} \left[F_j(\boldsymbol{Y}, \boldsymbol{X}) \right] - \sum_{k} E_{p(\boldsymbol{Y} | \boldsymbol{x}^{(k)}, \boldsymbol{\lambda})} \left[F_j(\boldsymbol{Y}, \boldsymbol{x}^{(k)}) \right]$$

Empirical Distribution

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the mode.
- Conditional log-likelihood for a general CRF:

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_k \left[\log \frac{1}{Z(\boldsymbol{x}^{(k)})} + \sum_j \lambda_j F_j(\boldsymbol{y}^{(k)}, \boldsymbol{x}^{(k)}) \right]$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_j} = E_{\tilde{p}(\boldsymbol{Y},\boldsymbol{X})}\left[F_j(\boldsymbol{Y},\boldsymbol{X})\right] - \sum_k E_{p(\boldsymbol{Y}|\boldsymbol{x}^{(k)},\boldsymbol{\lambda})}\left[F_j(\boldsymbol{Y},\boldsymbol{x}^{(k)})\right]$$

Empirical Distribution

Hard to calculate!

Parameter Learning

- Given the training data, $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ we wish to learn parameters of the mode.
- Conditional log-likelihood for a general CRF:

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_k \left[\log \frac{1}{Z(\boldsymbol{x}^{(k)})} + \sum_j \lambda_j F_j(\boldsymbol{y}^{(k)}, \boldsymbol{x}^{(k)}) \right]$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_j} = E_{\tilde{p}(\boldsymbol{Y}, \boldsymbol{X})} \left[F_j(\boldsymbol{Y}, \boldsymbol{X}) \right] - \sum_k E_{p(\boldsymbol{Y} \mid \boldsymbol{x}^{(k)}, \boldsymbol{\lambda})} \left[F_j(\boldsymbol{Y}, \boldsymbol{x}^{(k)}) \right]$$

Empirical Distribution

Hard to calculate!

- It is not possible to analytically determine the parameter values that maximize the log-likelihood – setting the gradient to zero and solving for λ does not always yield a closed form solution. (Almost always)

Parameter Learning

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

- Until we reach convergence

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

- Until we reach convergence

$$\mid \mathcal{L}(\lambda_{i+1}; y \mid x) - \mathcal{L}(\lambda_i; y \mid x) \mid < \epsilon$$

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

- Until we reach convergence

$$\mid \mathcal{L}(\lambda_{i+1}; y \mid x) - \mathcal{L}(\lambda_i; y \mid x) \mid < \epsilon$$

- Or any other optimization:
  - Quasi-Newton methods: BFGS [Bertsekas,1999] or L-BFGS [Byrd, 1994]

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

- Until we reach convergence

$$| \mathcal{L}(\lambda_{i+1}; y \mid x) - \mathcal{L}(\lambda_i; y \mid x) | < \epsilon$$

- Or any other optimization:
  - Quasi-Newton methods: BFGS [Bertsekas,1999] or L-BFGS [Byrd, 1994]
- General CRFs are intractable hence approximation solutions are necessary

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

- Until we reach convergence

$$| \mathcal{L}(\lambda_{i+1}; y \mid x) - \mathcal{L}(\lambda_i; y \mid x) | < \epsilon$$

- Or any other optimization:
  - Quasi-Newton methods: BFGS [Bertsekas,1999] or L-BFGS [Byrd, 1994]
- General CRFs are intractable hence approximation solutions are necessary

Compared with Markov chains, CRF's should be more discriminative, much slower to train and possibly more susceptible to over-training.

Parameter Learning

- This could be done using gradient descent

$$\lambda \propto \max_{\lambda} \mathcal{L}(\lambda; y \mid x) \propto \max_{\lambda} \log \sum_{i=1}^{N} p(\mathbf{y} \mid \mathbf{x}; \lambda)$$

$$\lambda_{i+1} \leftarrow \lambda_i + \alpha . \nabla_{\lambda} \mathcal{L}(\lambda; y \mid x)$$

- Until we reach convergence

$$| \mathcal{L}(\lambda_{i+1}; y \mid x) - \mathcal{L}(\lambda_i; y \mid x) | < \epsilon$$

- Or any other optimization:
  - Quasi-Newton methods: BFGS [Bertsekas,1999] or L-BFGS [Byrd, 1994]
- General CRFs are intractable hence approximation solutions are necessary

Compared with Markov chains, CRF's should be more discriminative, much slower to train and possibly more susceptible to over-training.

- Regularization:
  - σ is a regularization parameter

$$f_{objective}(\theta) = P_{\theta}(\mathbf{y} \mid \mathbf{x}) - \frac{\| \theta \|^2}{2\sigma^2}$$

Training (and Inference): General Case

- Approximate solution, to get faster inference.

Training (and Inference): General Case

- Approximate solution, to get faster inference.
- Treat inference as shortest path problem in the network consisting of paths(with costs)
  - Max Flow-Min Cut (Ford-Fulkerson, 1956)

Training (and Inference): General Case

- Approximate solution, to get faster inference.
- Treat inference as shortest path problem in the network consisting of paths(with costs)
  - Max Flow-Min Cut (Ford-Fulkerson, 1956)
- Pseudo-likelihood approximation:
  - Convert a CRF into separate patches; each consists of a hidden node and true values of neighbors; Run ML on separate patches
  - Efficient but may over-estimate inter-dependencies

Training (and Inference): General Case

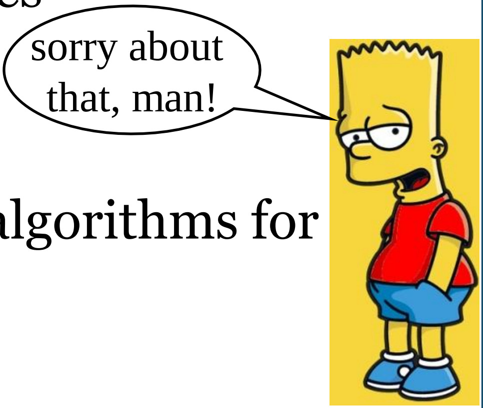
- Approximate solution, to get faster inference.
- Treat inference as shortest path problem in the network consisting of paths(with costs)
  - Max Flow-Min Cut (Ford-Fulkerson, 1956)
- Pseudo-likelihood approximation:
  - Convert a CRF into separate patches; each consists of a hidden node and true values of neighbors; Run ML on separate patches
  - Efficient but may over-estimate inter-dependencies
- Belief propagation?!
  - variational inference algorithm
  - it is a direct generalization of the exact inference algorithms for linear-chain CRFs

Training (and Inference): General Case

- Approximate solution, to get faster inference.
- Treat inference as shortest path problem in the network consisting of paths(with costs)
  - Max Flow-Min Cut (Ford-Fulkerson, 1956)
- Pseudo-likelihood approximation:
  - Convert a CRF into separate patches; each consists of a hidden node and true values of neighbors; Run ML on separate patches
  - Efficient but may over-estimate inter-dependencies
- Belief propagation?!
  - variational inference algorithm
  - it is a direct generalization of the exact inference algorithms for linear-chain CRFs
- Sampling based method(MCMC)

Training (and Inference): General Case

- Approximate solution, to get faster inference.
- Treat inference as shortest path problem in the network consisting of paths(with costs)
  - Max Flow-Min Cut (Ford-Fulkerson, 1956)
- Pseudo-likelihood approximation:
  - Convert a CRF into separate patches; each consists of a hidden node and true values of neighbors; Run ML on separate patches
  - Efficient but may over-estimate inter-dependencies
- Belief propagation?!
  - variational inference algorithm
  - it is a direct generalization of the exact inference algorithms for linear-chain CRFs
- Sampling based method(MCMC)



CRF frontiers

- Bayesian CRF:
  - Because of the large number of parameters in typical applications of CRFs
    -

CRF frontiers

- Bayesian CRF:
  - Because of the large number of parameters in typical applications of CRFs
    - prone to overfitting.
      -

CRF frontiers

- Bayesian CRF:
  - Because of the large number of parameters in typical applications of CRFs
    - prone to overfitting.
    - Regularization?
      -

CRF frontiers

- Bayesian CRF:
  - Because of the large number of parameters in typical applications of CRFs
    - prone to overfitting.
    - Regularization?
    - Instead of $\mathbf{y}^* = \max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x};\hat{\theta})$
      - $\mathbf{y}^* = \max_{\mathbf{y}} \int p(\mathbf{y}|\mathbf{x};\theta) p(\theta|\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{x}^{(N)}, \mathbf{y}^{(N)}) d\theta$
      - Too complicated! How can we approximate this?

CRF frontiers

- Bayesian CRF:
  - Because of the large number of parameters in typical applications of CRFs
    - prone to overfitting.
    - Regularization?
    - Instead of $\mathbf{y}^* = \max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x};\hat{\theta})$
      - $\mathbf{y}^* = \max_{\mathbf{y}} \int p(\mathbf{y}|\mathbf{x};\theta)p(\theta|\mathbf{x}^{(1)},\mathbf{y}^{(1)},\ldots,\mathbf{x}^{(N)},\mathbf{y}^{(N)})d\theta$
      - Too complicated! How can we approximate this?
- Semi-supervised CRF:
  - The need to have big labeled data!
  - Unlike in generative models, it is less obvious how to incorporate unlabelled data into a conditional criterion, because the unlabelled data is a sample from the distribution $p(\mathbf{x})$

Outline

- Modeling
- Inference
- Training
- Some Applications

Some applications: Part-of-Speech-Tagging

- POS(part of speech) tagging; the identification of words as nouns, verbs, adjectives, adverbs, etc.

Students	need	another	break
noun	verb	article	noun

- CRF features:

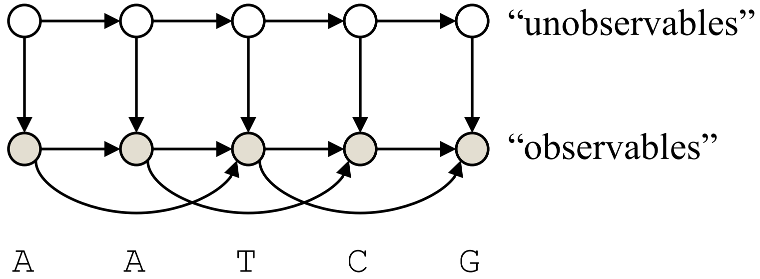
Feature Type	Description
Transition	$\forall k,k'$ $y_i = k$ and $y_{i+1}=k'$
Word	$\forall k,w$ $y_i = k$ and $x_i=w$ $\forall k,w$ $y_i = k$ and $x_{i-1}=w$ $\forall k,w$ $y_i = k$ and $x_{i+1}=w$ $\forall k,w,w'$ $y_i = k$ and $x_i=w$ and $x_{i-1}=w'$ $\forall k,w,w'$ $y_i = k$ and $x_i=w$ and $x_{i+1}=w'$
Orthography: Suffix	$\forall s$ in {“ing”,”ed”,”ogy”,”s”,”ly”,”ion”,”tion”, “ity”, …} and $\forall k$ $y_i=k$ and x_i ends with s
Orthography: Punctuation	$\forall k$ $y_i = k$ and x_i is capitalized $\forall k$ $y_i = k$ and x_i is hyphenated …

Is HMM(Gen.) better or CRF(Disc.)

- If your application gives you good structural information such that could be easily modeled by dependent distributions, and could be learnt tractably, go the generative way!

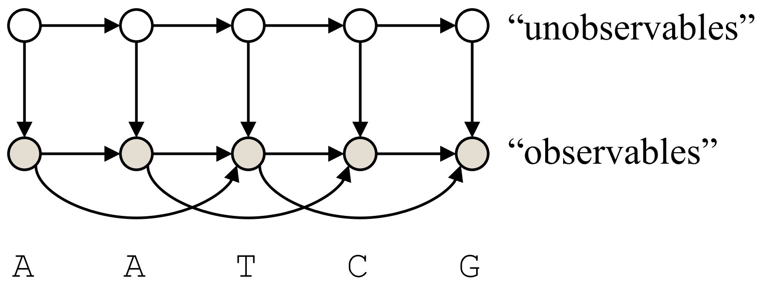
Is HMM(Gen.) better or CRF(Disc.)

- If your application gives you good structural information such that could be easily modeled by dependent distributions, and could be learnt tractably, go the generative way!
- Ex. Higher-order emissions from individual states

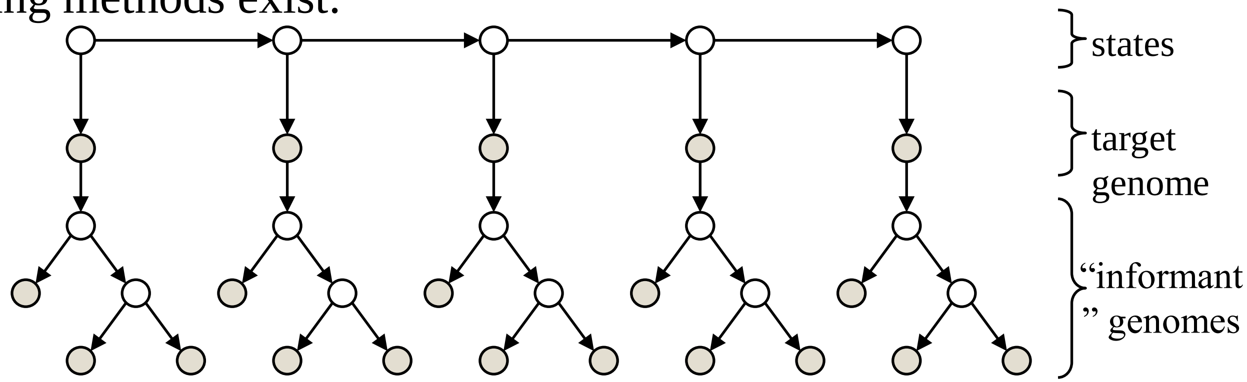


Is HMM(Gen.) better or CRF(Disc.)

- If your application gives you good structural information such that could be easily modeled by dependent distributions, and could be learnt tractably, go the generative way!
- Ex. Higher-order emissions from individual states



- Incorporating *evolutionary conservation* from an alignment: *PhyloHMM*, for which efficient decoding methods exist:



References

- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. ICML01, 2001.
- Charles Elkan, "Log-linear Models and Conditional Random Field," Notes for a tutorial at CIKM, 2008.
- Charles Sutton and Andrew McCallum, "An Introduction to Conditional Random Fields for Relational Learning," MIT Press, 2006
- Slides: An Introduction to Conditional Random Field, Ching-Chun Hsiao
- Hanna M. Wallach , Conditional Random Fields: An Introduction, 2004
- Sutton, Charles, and Andrew McCallum. *An introduction to conditional random fields for relational learning*. Introduction to statistical relational learning. MIT Press, 2006.
- Sutton, Charles, and Andrew McCallum. "An introduction to conditional random fields." *arXiv preprint arXiv:1011.4088* (2010).
- *B. Majoros,* Conditional Random Fields, for eukaryotic gene prediction