

Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

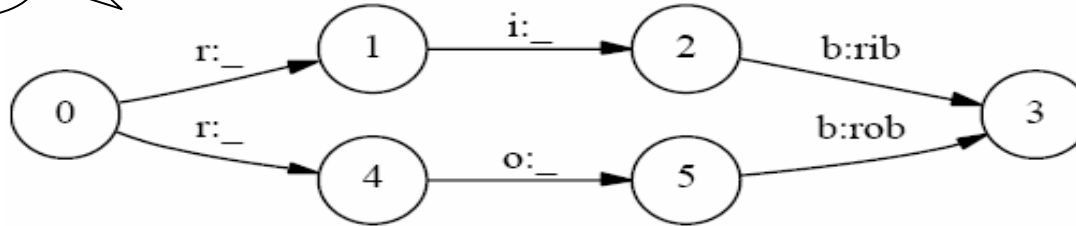
John Lafferty, Andrew McCallum, Fernando Pereira

Presenter: Yejin Choi

Label Bias Problem

arcs: observations (X)

nodes: outputs (Y)



S1	Herbivores (-)	like (r)	merino (i)	sheep (b)
	NNS (0)	IN (1)	JJ (2)	NN (3)
S2	Carnivores (-)	like (r)	eating (o)	sheep (b)
	NNS (0)	VB (4)	VBG (5)	NN (3)

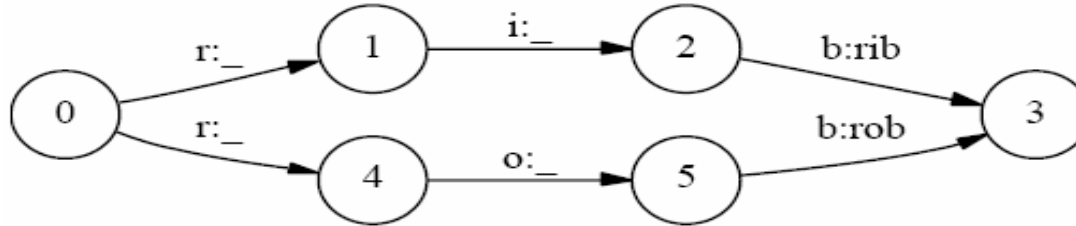
• Suppose [NNS => VB] transition more frequent than [NNS => IN]

• Suppose from [VB], only [VB => VBG] transition is possible

➔ now, What is $P(Y_i = \text{VBG} \mid Y_{i-1} = \text{VB}, X_i = \text{merino})$???

• Recall MEMM models $P(Y_i \mid Y_{i-1}, X_i)$

Label Bias Problem



Herbivores (h)	like	(r)	merino (i)	sheep (b)
NNS (0)	IN	(1)	JJ (2)	NN (3)
Carnivores (c)	like	(r)	eating (o)	sheep (b)
NNS (0)	VB	(4)	VBG (5)	NN (3)

So, how do we fix this nonsense ?

$$P(Y_i = \text{VBG} \mid Y_{i-1} = \text{VB}, X_i = ???) = 1 \text{ for any } X_i$$

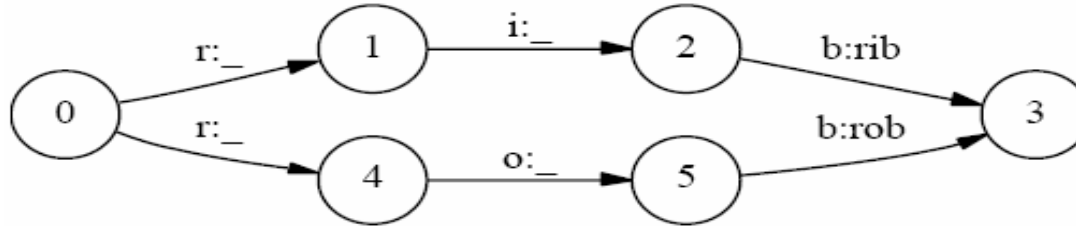
→ **Do not normalize on each node!**

Instead, normalize over the entire sequence.

This motivates the "global normalization" scheme of CRFs.

Other approaches: Cohen and Carvalho (2005), Sutton and McCallum (2005)

Label Bias Problem



Herbivores (h)	like	(r)	merino (i)	sheep (b)
NNS (0)	IN	(1)	JJ (2)	NN (3)
Carnivores (c)	like	(r)	eating (o)	sheep (b)
NNS (0)	VB	(4)	VBG (5)	NN (3)

Wait, what about HMMs?

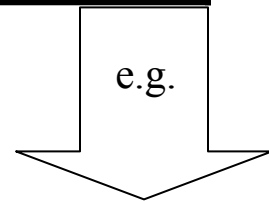
• Recall HMM models $P(X_i | Y_i)$ and $P(Y_i | Y_{i-1})$

(so that $P(X_i | Y_i, Y_{i-1}) \cdot P(Y_i | Y_{i-1}) = P(X_i, Y_i | Y_{i-1})$)

then we can get $P(\text{merino} | \text{VBG}) = 0 !$

Label Bias v.s. Observation Bias

Label Bias (Bottou, 1991; Lafferty et. al. 2001)	Observation Bias (Klein and Manning 2002)
Previous state explains current state so well (hence ignoring observation)	Observation explains current state so well (hence ignoring state transition)



	All the indexes above	HMM	MEMM
Correct	PDT DT NNS VBD	-0.0	-1.3
Incorrect	DT DT NNS VBD	-5.4	-0.3

→ Both are to do with **local conditional normalization**.

So, Let's normalize globally! (But, how?)

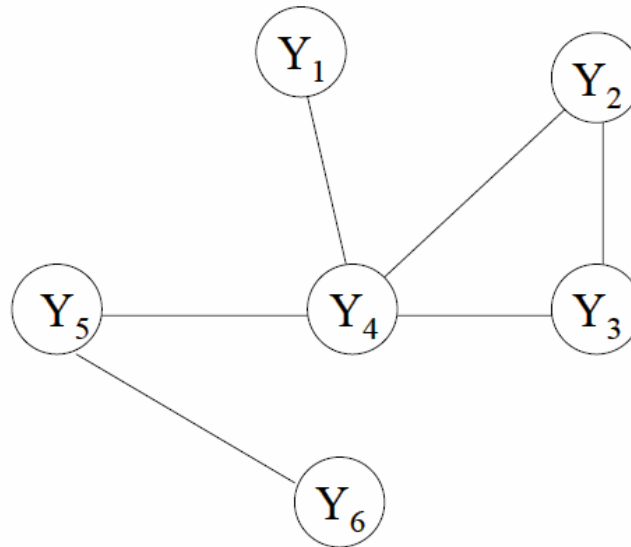
- What we need is the global joint distribution. i.e., $\mathbf{p}(\mathbf{y} \mid \mathbf{x})$
 - where $\mathbf{y} = (y_1, \dots, y_n)$ and $\mathbf{x} = (x_1, \dots, x_n)$
 - we do not want distributions on individual node. i.e., $\mathbf{p}(\mathbf{y}_i \mid \mathbf{x}_i)$
 - Instead, we want non-probabilistic *potential* function. i.e., $\phi(\mathbf{y}_i, \mathbf{x}_i)$
 - $p(\mathbf{y} \mid \mathbf{x}) \approx g(\phi(y_1, x_1), \dots, \phi(y_n, x_n))$????????
- Problem with directed graphs (like Bayesian Network)
 - a probability distribution should be given for each node
 - then, the joint probability $\mathbf{p}(\mathbf{y}) = \prod_i \mathbf{p}(\mathbf{y}_i \mid \mathbf{parents}(\mathbf{y}_i))$
 - ➔ btw, what is $\mathbf{parents}(\mathbf{y}_i)$ for MEMM ?
- Markov Random Field ! (= Markov Network, Random Field)

Markov Random Field

*Just for now,
forget about conditioning on X...*

Let $G = (Y, E)$ be a graph where each vertex Y_v is a random variable
Suppose $P(Y_v | \text{all other } Y) = P(Y_v | \text{neighbors}(Y_v))$ then Y is a
random field

Example:



- $P(Y_5 | \text{all other } Y) = P(Y_5 | Y_4, Y_6)$
- But, how do we compute the global joint distribution $P(Y)$ out of this ?
- Besides, We don't want to compute $P(Y_i | \text{neighbors}(Y_i))$!!!

Hammersley-Clifford theorem (1971)

$$P(\vec{Y}) = \frac{1}{Z} \prod_{\text{clique } C} \phi(\vec{Y}_C)$$

$$\text{where } Z = \sum_{\mathbf{Y}} \prod_C \phi(Y_C)$$

1. Given MRF $G=(Y,E)$ such that $P(Y_i | Y \setminus Y_i) = P(Y_i | \text{nbr}(Y_i))$
2. Given $\phi(Y_C)$ for \forall clique C in G , such that $\phi(Y_C) \geq 0$
 - cliques may overlap.
 - cliques may not be maximal.

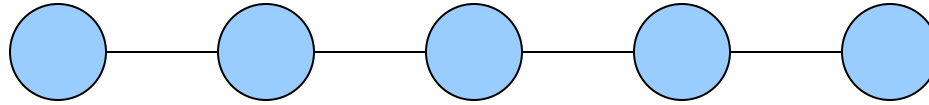
→ this implies we don't need to compute $P(Y_i | \text{nbr}(Y_i))$ to get $P(Y)$!

Definition of CRFs

Definition. Let $G = (V, E)$ be a graph such that $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, so that \mathbf{Y} is indexed by the vertices of G . Then (\mathbf{X}, \mathbf{Y}) is a conditional random field in case, when conditioned on \mathbf{X} , the random variables \mathbf{Y}_v obey the Markov property with respect to the graph: $p(\mathbf{Y}_v | \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v | \mathbf{X}, \mathbf{Y}_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G .

- Almost identical to the definition of MRFs, except CRFs condition on \mathbf{X} . and notice that \mathbf{X} is not part of \mathbf{V} .
- Hammersley-Clifford theorem can be extended to the conditional cases (on \mathbf{X}) as well!

$P(y | x)$ for linear-chain CRFs



Let

$$\phi(y_i, \mathbf{x}) = \exp \sum_k \lambda_k f_k(y_i, \mathbf{x})$$
$$\phi(y_i, y_{i+1}, \mathbf{x}) = \exp \sum_k \lambda_k f_k(y_i, y_{i+1}, \mathbf{x})$$

Can use any other non negative function, but exp function will give us a nice continuous convex objective function later.

Then,

$$p(y | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \left(\phi(y_i, \mathbf{x}) \phi(y_i, y_{i+1}, \mathbf{x}) \right)$$
$$= \frac{1}{Z(\mathbf{x})} \exp \left(\sum_{i,k} \lambda_k f_k(y_i, \mathbf{x}) + \sum_{i,k} \lambda_k f_k(y_i, y_{i+1}, \mathbf{x}) \right)$$
$$= \frac{1}{Z(\mathbf{x})} \exp \left(\boldsymbol{\lambda} \bullet \mathbf{F}(y, \mathbf{x}) \right)$$

where $Z(\mathbf{x}) = \sum_y \exp \left(\boldsymbol{\lambda} \bullet \mathbf{F}(y, \mathbf{x}) \right)$

$$p_{\theta}(y | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left(\sum_{e \in E, k} \lambda_k f_k(e, y|_e, \mathbf{x}) + \sum_{v \in V, k} \mu_k g_k(v, y|_v, \mathbf{x}) \right)$$

Objective function for training

Given the training data $D = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}_{j=1}^N$

and $p(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}, \mathbf{x})$

Objective function :

conditional likelihood $L(\boldsymbol{\lambda}) = L(\boldsymbol{\lambda} | D) = P(D | \boldsymbol{\lambda}) = \prod_j p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)})$

equiv. to optimize $l(\boldsymbol{\lambda}) = \log L(\boldsymbol{\lambda}) = \sum_j \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)})$

$$l(\boldsymbol{\lambda}) = \sum_j \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)}) = \sum_j \log \frac{1}{Z(\mathbf{x}^{(j)})} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}^{(j)}, \mathbf{x}^{(j)})$$

$$= \sum_j \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}^{(j)}, \mathbf{x}^{(j)}) - \log Z(\mathbf{x}^{(j)})$$

$$= \sum_j \left(\boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}^{(j)}, \mathbf{x}^{(j)}) - \log \sum_{\mathbf{y}} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}, \mathbf{x}^{(j)}) \right)$$

Objective function for training

Given the training data $D = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}_{j=1}^N$

and $p(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}, \mathbf{x})$

Objective function :

conditional likelihood $L(\boldsymbol{\lambda}) = L(\boldsymbol{\lambda} | D) = P(D | \boldsymbol{\lambda}) = \prod_j p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)})$

equiv. to optimize $l(\boldsymbol{\lambda}) = \log L(\boldsymbol{\lambda}) = \sum_j \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)})$

$$l(\boldsymbol{\lambda}) = \sum_j \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)}) = \sum_j \log \frac{1}{Z(\mathbf{x}^{(j)})} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}^{(j)}, \mathbf{x}^{(j)})$$

$$= \sum_j \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}^{(j)}, \mathbf{x}^{(j)}) - \log Z(\mathbf{x}^{(j)})$$

$$= \sum_j \left(\boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}^{(j)}, \mathbf{x}^{(j)}) - \log \sum_{\mathbf{y}} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(\mathbf{y}, \mathbf{x}^{(j)}) \right)$$

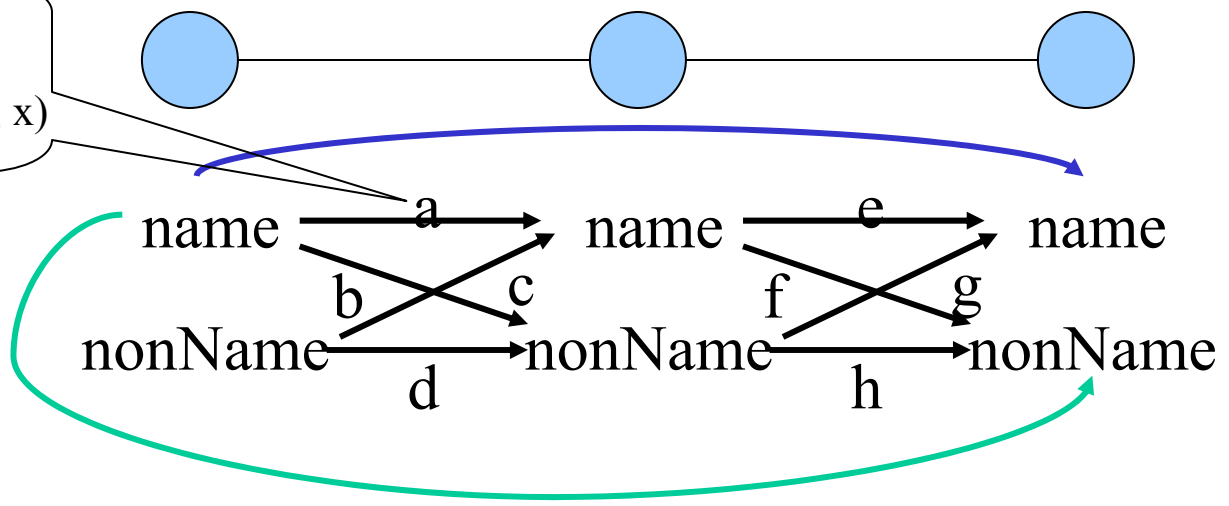


nasty!

Computing $Z(x)$! ... for linear-chain CRFs

$$l(\lambda) = \sum_j \log p(y^{(j)} | x^{(j)}) = \sum_j \left(\lambda \bullet F(y^{(j)}, x^{(j)}) - \log \sum_y \exp \lambda \bullet F(y, x^{(j)}) \right)$$

$\phi(y_i, y_{i+1}, x)$
 $= \exp \sum_k \lambda_k f_k(y_i, y_{i+1}, x)$



$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} \underline{ae + bg} & \underline{af + bh} \\ \dots & \dots \end{bmatrix}$$

Parameter Estimation for CRFs

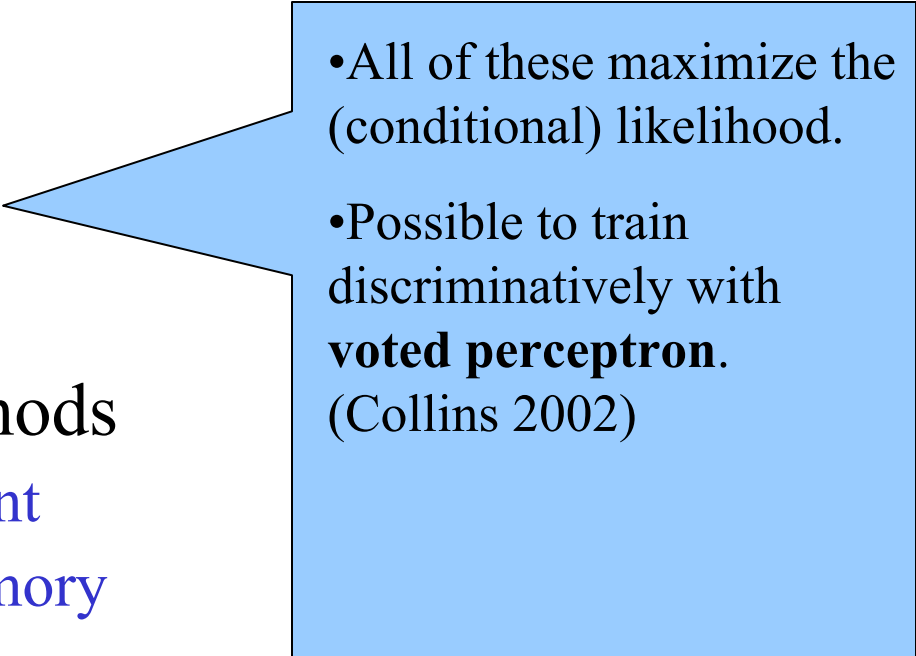
→ how to compute $\operatorname{argmax}_{\lambda} l(\lambda)$?

- Iterative scaling algorithms
 - Generalized iterative scaling (GIS)
 - Improved iterative scaling (IIS)
 - easy to implement
 - both really slow to converge
- Gradient descent methods
 - CG : conjugate gradient
 - L-BFGS : limited memory Newton's method
 - much harder to implement, but lots of code available
 - you only need to provide $l(\lambda)$ and $l'(\lambda)$
 - much faster to converge

Parameter Estimation for CRFs

- Iterative scaling algorithms
 - Generalized iterative scaling (GIS)
 - Improved iterative scaling (IIS)

- Gradient descent methods
 - CG : conjugate gradient
 - L-BFGS : limited memory
Newton's method



- All of these maximize the (conditional) likelihood.

- Possible to train discriminatively with **voted perceptron**.
(Collins 2002)

Sha and Pereira (2003)

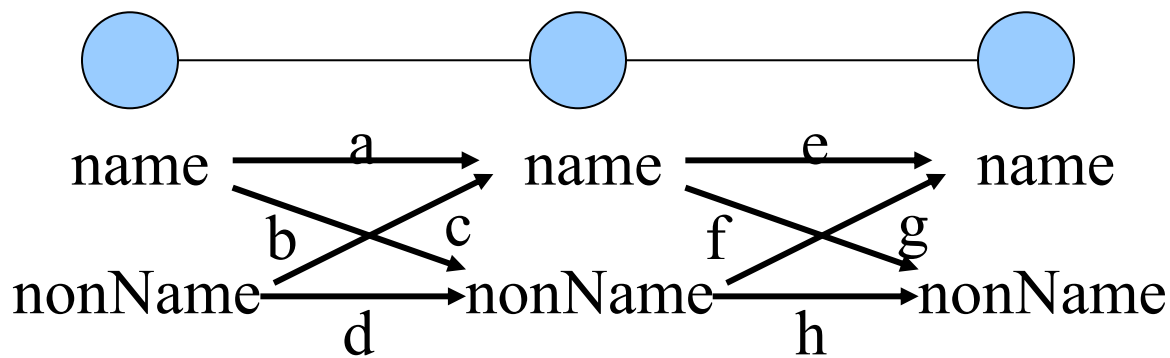
training method	time	F score	\mathcal{L}'_{λ}
Precond. CG	130	94.19%	-2968
Mixed CG	540	94.20%	-2990
Plain CG	648	94.04%	-2967
L-BFGS	84	94.19%	-2948
GIS	3700	93.55%	-5668

Table 3: Runtime for various training methods in minutes, 375k examples

$\operatorname{argmax}_y P(y \mid \mathbf{x}) \dots$ for linear-chain CRFs

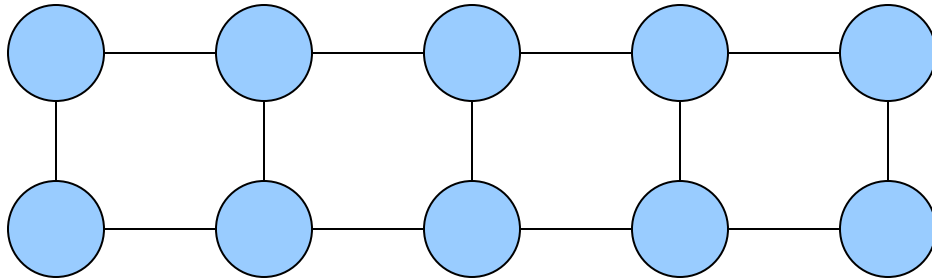
A universal
component for
SVM, HMM,
MEMM, etc

$$\begin{aligned}\operatorname{argmax}_y p(y \mid \mathbf{x}) &= \operatorname{argmax}_y \log p(y \mid \mathbf{x}) \\ &= \operatorname{argmax}_y \left(\lambda \bullet \mathbf{F}(y, \mathbf{x}) - \log \mathbf{Z}(\mathbf{x}) \right) \\ &= \operatorname{argmax}_y \lambda \bullet \mathbf{F}(y, \mathbf{x})\end{aligned}$$

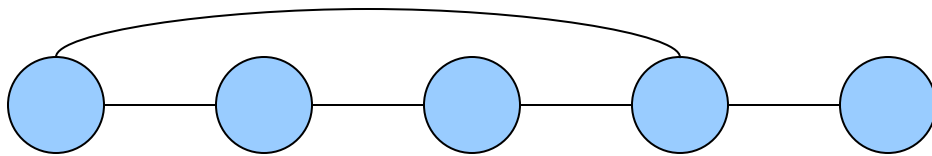


Other CRFs

Factorial CRFs (Sutton et.al., 2004)

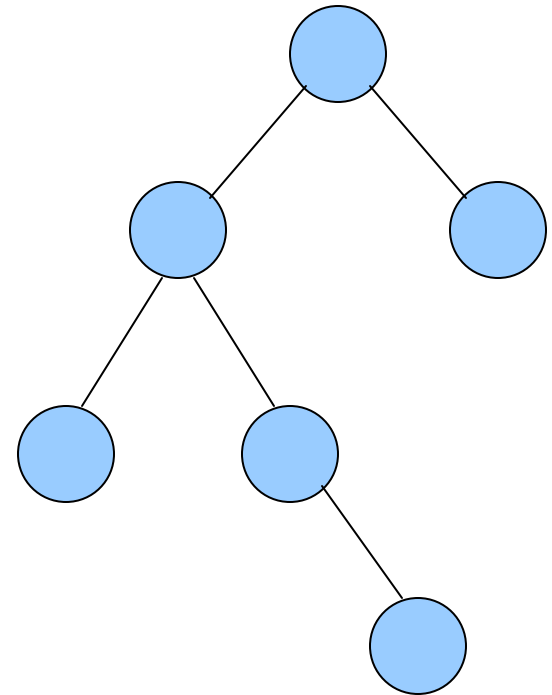


Skip-chain CRFs (Sutton and McCallum, 2004)



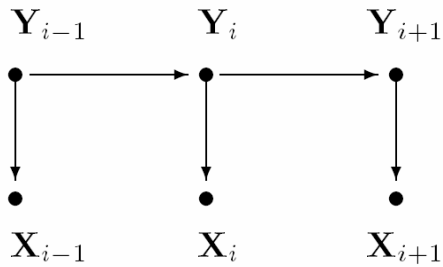
Tree CRFs

(Cohn and Blunsom, 2005)

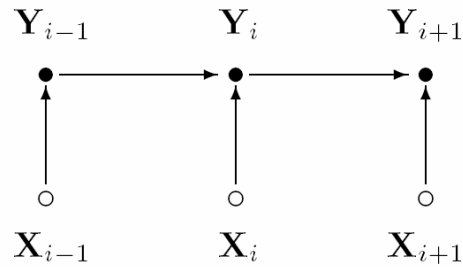


Graphical representation

HMM



MEMM



CRF

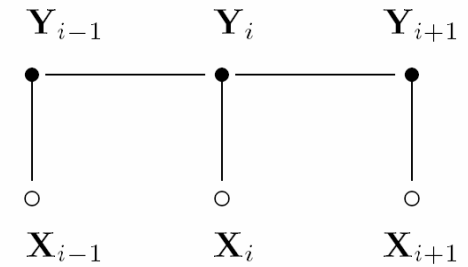
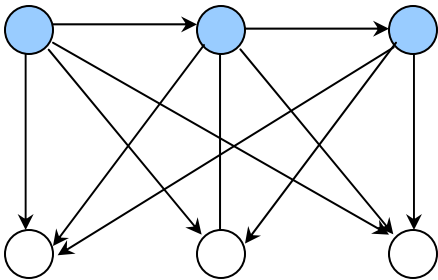
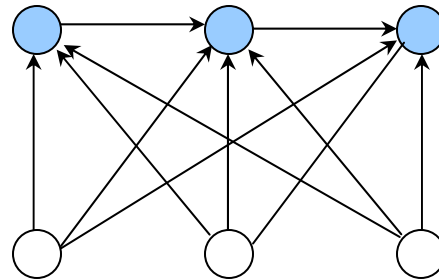


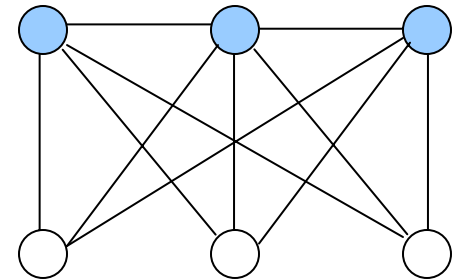
Figure 2. Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.



?



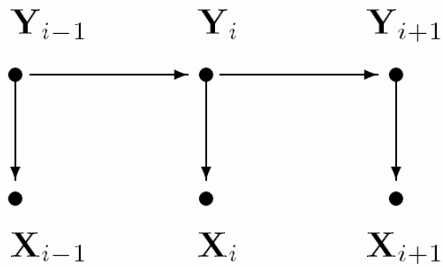
?



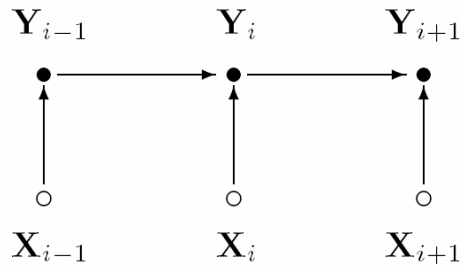
?

Graphical representation

HMM



MEMM



CRF

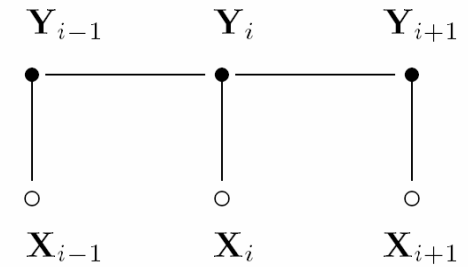
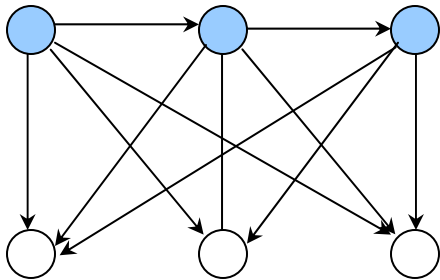
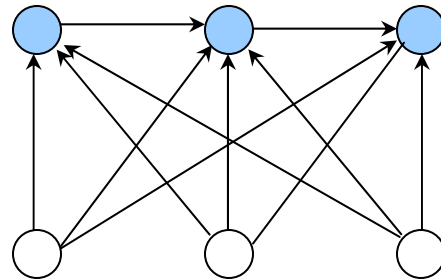


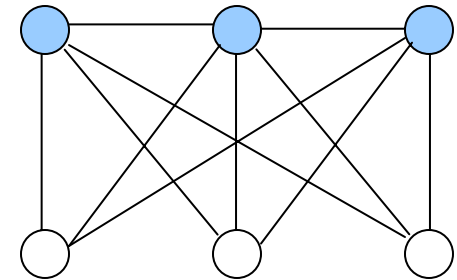
Figure 2. Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.



NO!



YES



YES



Maximum Entropy II

- Also: regularization (smoothing)

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) - k \|\mathbf{w}\|^2$$

- Maximize likelihood = Minimize “log-loss”

$$\min_{\mathbf{w}} k \|\mathbf{w}\|^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- Motivation

- Connection to maximum entropy principle
- Might want to do a good job of being uncertain on noisy cases...
- ... in practice, though, posteriors are pretty peaked



Max vs “Soft-Max” Margin

- SVMs:

$$\min_{\mathbf{w}} k \|\mathbf{w}\|^2 - \sum_i \underbrace{\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}) \right) \right)}_{\text{Hard (Penalized) Margin}}$$

- Maxent:

$$\min_{\mathbf{w}} k \|\mathbf{w}\|^2 - \sum_i \underbrace{\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right) \right)}_{\text{Soft Margin}}$$

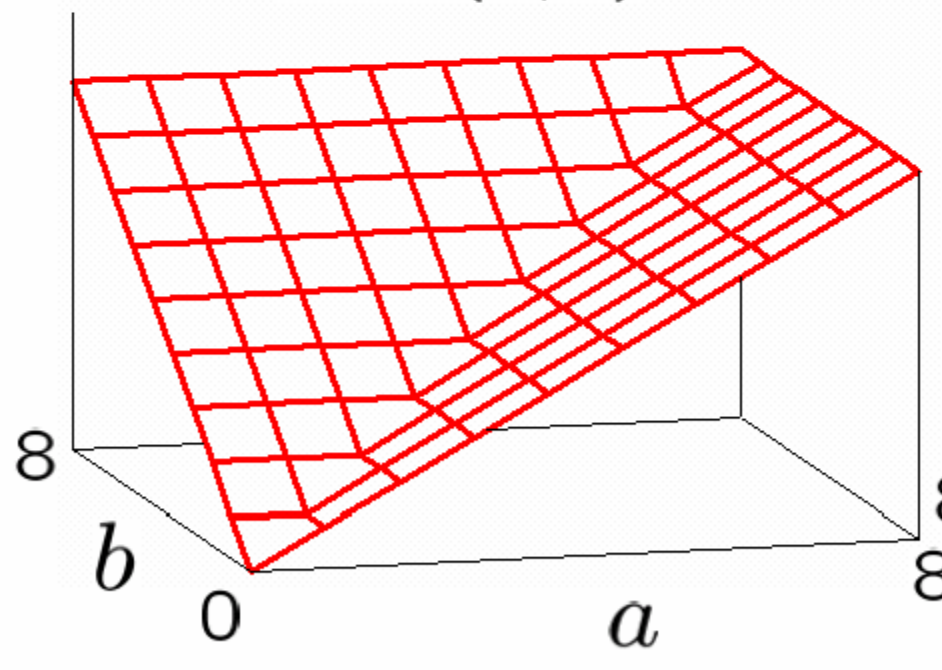
- Very similar! Both try to make the true score better than a function of the other scores.
 - The SVM tries to beat the augmented runner-up
 - The maxent classifier tries to beat the “soft-max”



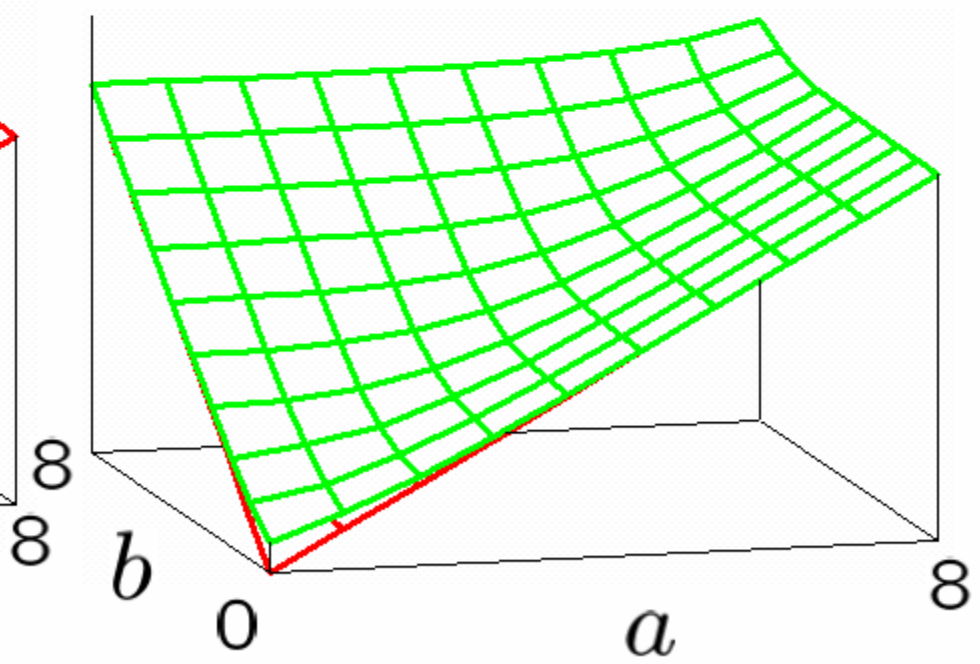
“Soft-Max”

$$\max(a, b) \approx \log(\exp(a) + \exp(b))$$

$\max(a, b)$



$\log(\exp(a) + \exp(b))$





Log-Loss

- If we view maxent as a minimization problem:

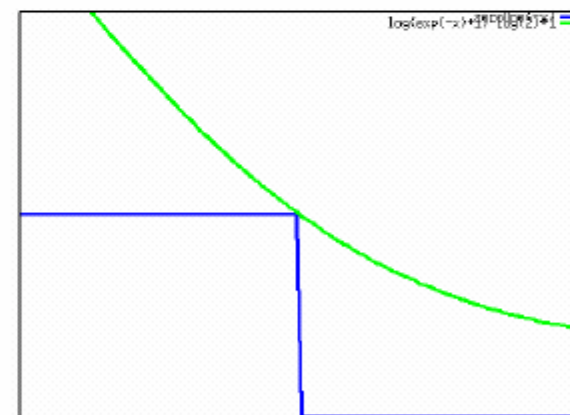
$$\min_{\mathbf{w}} k \|\mathbf{w}\|^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- This minimizes the “log-loss” on each example

$$- \left[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right]$$

$$- \log \left(\frac{\exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i))}{\sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))} \right) = - \log P(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w})$$

- Log-loss bounds zero-one loss



$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$



Loss Functions: I

- Zero-One Loss

$$\sum_i \text{step} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$

- Hinge

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] \right)$$

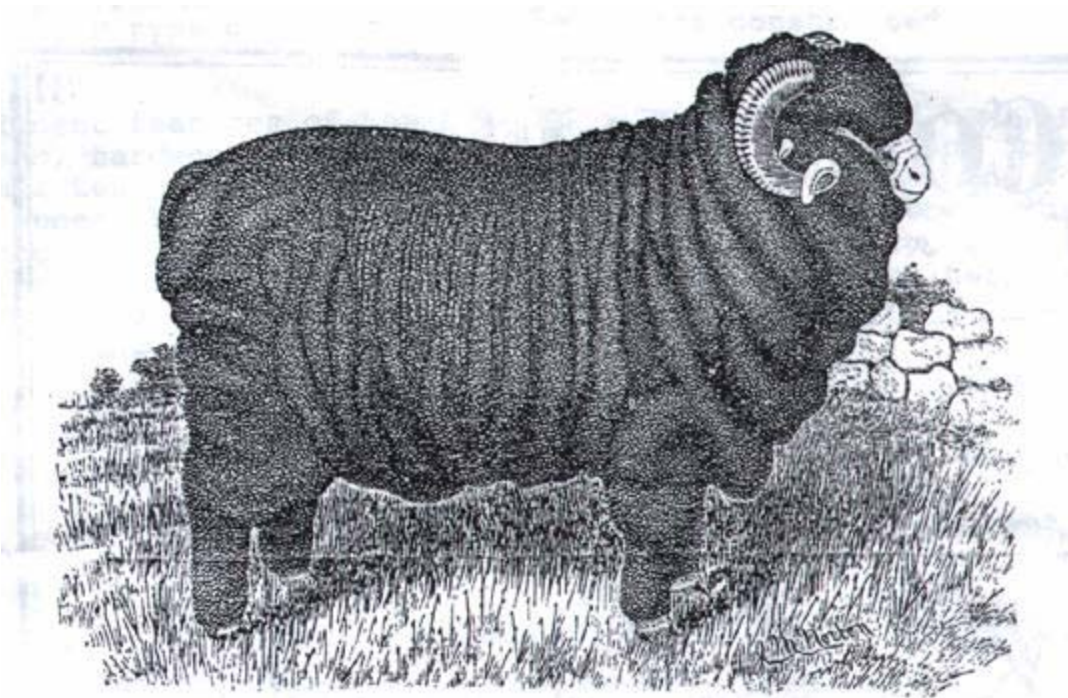
- Log

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$



$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

Merino Sheep



CHERBINO & WILLIAMSON 589.

Bred by Cherbino & Williamson, Middlebury, Vt.; owned by I. L. Eells, Middlebury, Vt.