CS 6840: Natural Language Processing

Sequence Tagging with CRFs: Named Entity Recognition

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

Probabilistic Graphical Models

- PGMs use a graph for **compactly**:
  1. <u>Encoding a complex distribution</u> over a multi-dimensional space.
  2. <u>Representing a set of independencies</u> that hold in the distribution.
  - Properties 1 and 2 are, in a "deep sense", equivalent.

- Probabilistic Graphical Models:
  - **Directed**:
    - i.e. Bayesian Networks i.e. Belief Networks.
  - **Undirected**:
    - i.e. Markov Random Fields

Probabilistic Graphical Models

- **Directed PGMs**:
  - Bayesian Networks:
    - Dynamic Bayesian Networks:
      - State Observation Models:
        - Hidden Markov Models.
        - Linear Dynamical Systems (Kalman filters).

- **Undirected PGMs**:
  - Markov Random Fields (MRF).
    - Conditional Random Fields (CRF).
      - Sequential CRFs.

Markov Random Fields (MRF)

- V – a set of (discrete) random variables
- $G = (V, E)$ an undirected graph

Definition:

V is said to be a *Markov Random Field* with respect to G if:

$$P(V_i \mid V - V_i) = P(V_i \mid N(V_i)) \quad \text{, where } N(V_i) = \{V_j \mid (V_i, V_j) \in E\}$$

i.e. $N(V_i)$ is the *neighborhood* of V_i

Gibbs Random Fields (GRF)

- $G = (V, E)$ – an undirected graph
  - V is a set of (discrete) random variables
  - $C(G)$ is the set of all cliques of G
  - V_c is the set of vertices in a clique $c \in C(G)$

Definition:

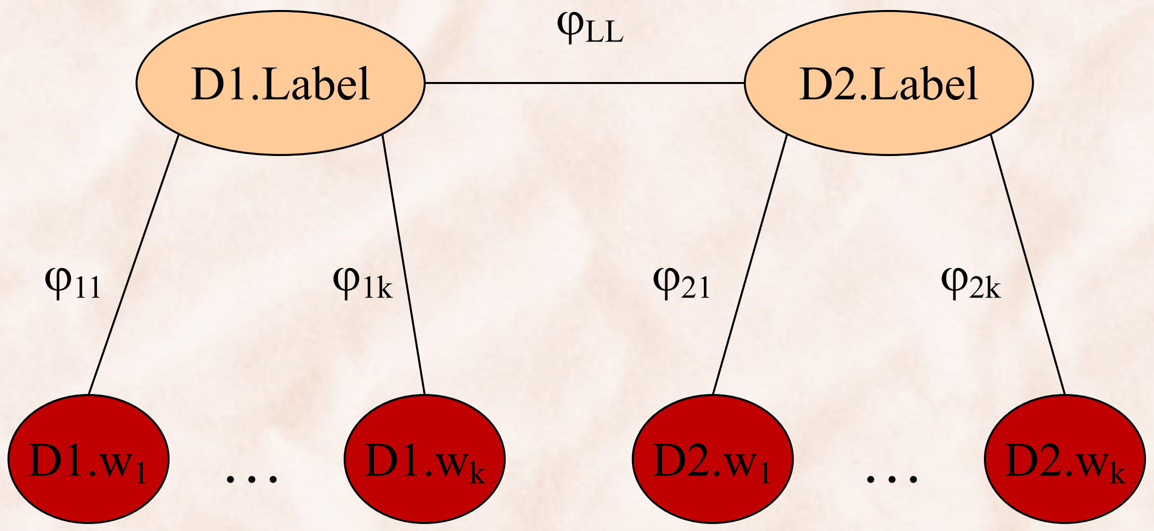
V is said to be a *Gibbs Random Field* with respect to G if:

$$P(V) = \frac{1}{Z} \exp \sum_{c \in C(G)} \varphi_c(V_c)$$

$\Phi = \{\varphi_c \mid \varphi_c : V_c \to R,\ c \in C(G)\}$ is the set of *clique potentials*

Z is the normalization constant

Gibbs Random Fields – Example



φ_{LL}	D1.Label	D2.Label
$\varphi_{LL}(0,0)$	0	0
$\varphi_{LL}(0,1)$	0	1
$\varphi_{LL}(1,0)$	1	0
$\varphi_{LL}(1,1)$	1	1

- D1, D2 are linked webpages
- D.Label $\in \{0,1\}$
- D.w is true if word w $\in$D, otherwise false
- k is the size of the vocabulary

φ_{1j}	D1.Label	D1.w_j
φ_{1j}(0,false)	0	false
φ_{1j}(0,true)	0	true
φ_{1j}(1,false)	1	false
φ_{1j}(1,true)	1	true

Markov-Gibbs Equivalence

- A GRF is characterized by its global property
  => *the Gibbs distribution*
- An MRF is characterized by its local property
  => *the Markov assumption*

Theorem [Hammersley & Clifford, 1971]

V is an *MRF* w.r.t. *G* ⇔ *V* is a *GRF* w.r.t. *G*

Discriminative MRF (CRF)

- $V = X \cup Y$ is a set of discrete random variables:
  - X are *observed* variables
  - Y are *hidden* variables (labels)
- $G = (V, E)$ is an undirected graph.

Definition:

V is said to be a *Conditional Random Field* (*CRF*) w.r.t. G if:

$$P(Y_i \mid X, Y - Y_i) = P(Y_i \mid X, N(V_i)) \quad \text{, where } N(Y_i) = \{Y_j | (Y_i, Y_j) \in E\}$$

i.e. $N(Y_i)$ is the *neighborhood* of Y_i

[Lafferty, McCallum & Pereira 2000]

Discriminative GRF (CMN)

- $V = X \cup Y$ is a set of discrete random variables
  - X are *observed* variables
  - Y are *hidden* variables (labels)
- $G = (V, E)$ is an undirected graph:
  - $C(G)$ are the cliques of G
  - $V_c = X_c \cup Y_c$ is the set of vertices in a clique $c \in C(G)$

Definition:

V is said to be a *Conditional Markov Network* w.r.t. G if:

$$P(Y \mid X) = \frac{1}{Z(X)} \exp \sum_{c \in C(G)} \varphi_c(X_c, Y_c)$$

$Z(X)$ is the normalization constant

Markov-Gibbs Equivalence

Theorem [Hammersley & Clifford, 1971] :

V is a *Conditional Random Field* w.r.t. *G*

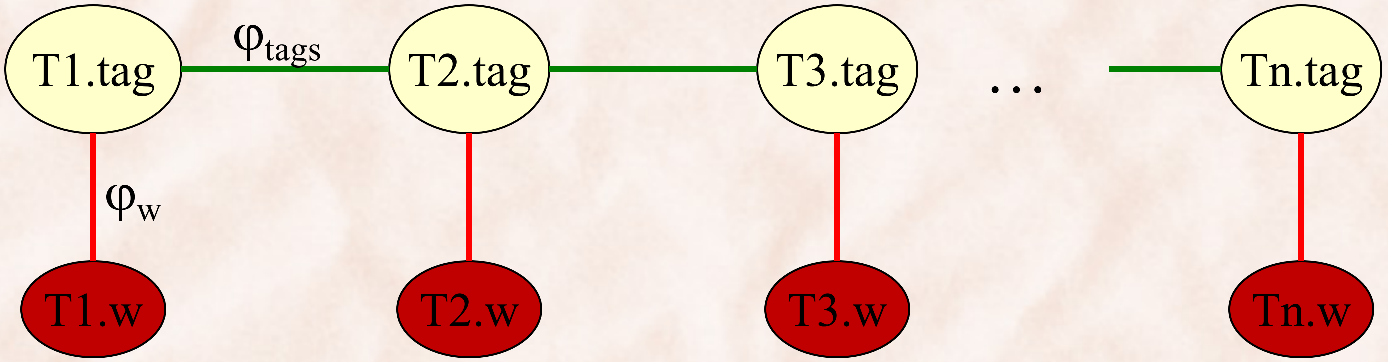
⇔ *V* is a *Conditional Markov Network* w.r.t. *G*

From HMMs to CRFs

$$
\begin{aligned}
p(\mathbf{y}, \mathbf{x}) &= \prod_{t=1}^{T} p(y_t | y_{t-1}) p(x_t | y_t) \\
&= \prod_{t=1}^{T} a_{y_{t-1} y_t} b_{x_t y_t} \\
&= \exp\left(\sum_{t=1}^{T} \log a_{y_{t-1} y_t} + \sum_{t=1}^{T} \log b_{x_t y_t} \right)
\end{aligned}
$$

[Lafferty, McCallum & Pereira 2000]

"Discriminative HMMs"



φ_{tags} and φ_w play a similar role to the (logarithms of the) usual HMM parameters *$P(T_{j+1}.tag|T_j.tag)$ and $P(T.w|T.tag)$.*

From HMMs to CRFs

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

$$= \exp\left(\sum_{t=1}^{T} \log a_{y_{t-1}y_t} + \sum_{t=1}^{T} \log b_{x_t y_t}\right)$$

$$= \exp\left(\sum_{t=1}^{T} \sum_{i,j\in S} w_{ij}\mathbf{1}[y_{t-1} = i]\mathbf{1}[y_t = j] + \sum_{t=1}^{T}\sum_{j\in S}\sum_{o\in O} u_{jo}\mathbf{1}[y_t = j]\mathbf{1}[x_t = o]\right)$$

$$\text{where} \begin{cases} w_{ij} = \log a_{ij} = \log p(y_t = j|y_{t-1} = i) \\ u_{jo} = \log b_{jo} = \log p(x_t = o|y_t = j) \end{cases}$$

From HMMs to CRFs

Define feature functions:

$$f^s_{ij}(y_{t-1}, y_t) = \mathbf{1}[y_{t-1} = i]\mathbf{1}[y_t = j]$$

$$f^o_{jo}(y_t, t, \mathbf{x}) = \mathbf{1}[y_t = j]\mathbf{1}[x_t = o]$$

$$p(\mathbf{y}, \mathbf{x}) = \exp\left(\sum_{t=1}^{T}\sum_{i,j\in S} w_{ij}\mathbf{1}[y_{t-1} = i]\mathbf{1}[y_t = j] + \sum_{t=1}^{T}\sum_{j\in S}\sum_{o\in O} u_{jo}\mathbf{1}[y_t = j]\mathbf{1}[x_t = o]\right)$$

$$= \exp\left(\sum_{t=1}^{T}\sum_{i,j\in S} w_{ij} f^s_{ij}(y_{t-1}, y_t) + \sum_{t=1}^{T}\sum_{j\in S}\sum_{o\in O} u_{jo} f^o_{jo}(y_t, t, \mathbf{x})\right)$$

$$= \exp\left(\sum_{t=1}^{T} \mathbf{w}^T \mathbf{f}_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \mathbf{u}^T \mathbf{f}_o(y_t, t, \mathbf{x})\right)$$

From HMMs to CRFs

$$p(\mathbf{y}, \mathbf{x}) = \exp\left(\sum_{t=1}^{T} \mathbf{w}^T \mathbf{f}_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \mathbf{u}^T \mathbf{f}_o(y_t, \mathbf{x})\right)$$

But $p(\mathbf{y}|\mathbf{x}) = \dfrac{p(\mathbf{y}, \mathbf{x})}{p(\mathbf{x})} = \dfrac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}', \mathbf{x})}$

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} \mathbf{w}^T \mathbf{f}_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \mathbf{u}^T \mathbf{f}_o(y_t, \mathbf{x})\right)$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp\left(\sum_{t=1}^{T} \mathbf{w}^T \mathbf{f}_s(y'_{t-1}, y'_t) + \sum_{t=1}^{T} \mathbf{u}^T \mathbf{f}_o(y'_t, \mathbf{x})\right)$

Linear-Chain Conditional Random Fields

- A linear-chain CRF is a distribution $p(\mathbf{y}|\mathbf{x})$ over sequences of labels $\mathbf{y}$ and conditioned on observations $\mathbf{x}$ that takes the form:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} \mathbf{w}^T \mathbf{f}_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \mathbf{u}^T \mathbf{f}_o(y_t, t, \mathbf{x})\right)$$

$$\text{where } Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp\left(\sum_{t=1}^{T} \mathbf{w}^T \mathbf{f}_s(y'_{t-1}, y'_t) + \sum_{t=1}^{T} \mathbf{u}^T \mathbf{f}_o(y'_t, t, \mathbf{x})\right)$$

- The state transition features $\mathbf{f}_s$ and observation emission features $\mathbf{f}_o$ can be any real-valued functions.

Inference (Decoding) with CRFs: Viterbi

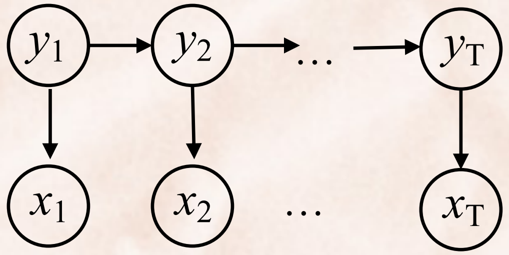
- Define *transition scores*: $\phi_s(y_{t-1}, y_t) = \mathbf{w}^T \mathbf{f}_s(y_{t-1}, y_t)$
- Define *emission scores*: $\phi_o(y_t, \mathbf{x}) = \mathbf{u}^T \mathbf{f}_o(y_t, t, \mathbf{x})$
- Then the CRF probability distribution can be written as:

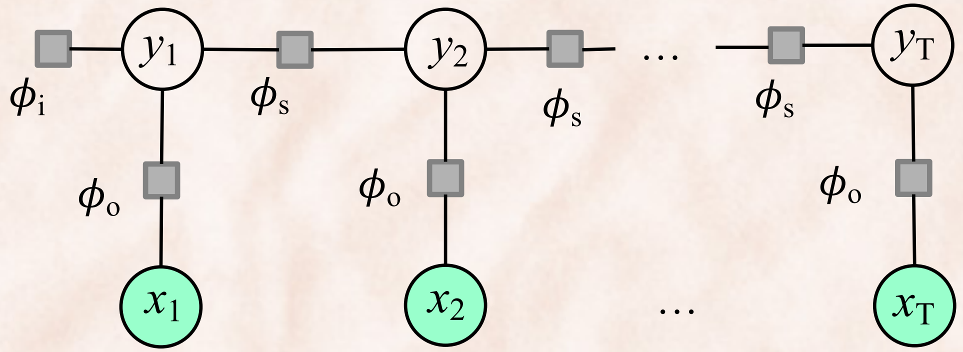
$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} \phi_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \phi_o(y_t, t, \mathbf{x}) \right)$$

- Now the emission features / scores at a position *t* can use the entire observation vector **x**.
- Features can be *overlapping* i.e. no need to model conditional (in)dependencies:
  - Conditioning on **x** => discriminative model.

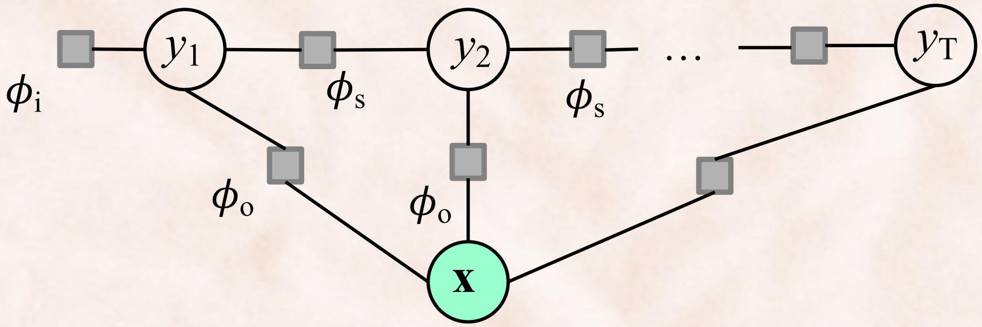
From HMMs to CRFs

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



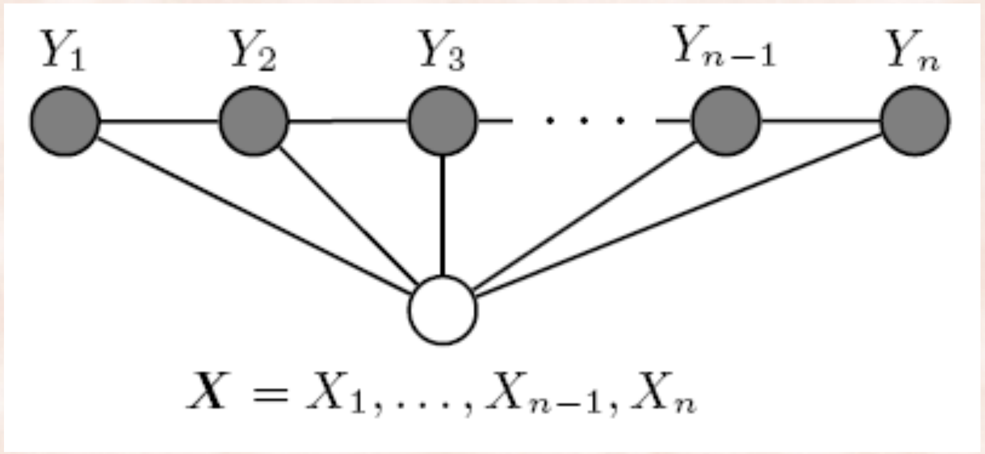


- CRFs: $p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} \phi_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \phi_o(y_t, t, \mathbf{x})\right)$



[Lafferty, McCallum & Pereira 2001]

Linear-Chain CRFs



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

$$\boxed{P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \exp \sum_j \underbrace{\lambda_j F_j(\mathbf{y},\mathbf{x})}_{\varphi_j(\mathbf{y},\mathbf{x})}}$$

Inference (Decoding) with CRFs: Viterbi

- Define *transition scores*: $\phi_s(y_{t-1}, y_t) = \mathbf{w}^T \mathbf{f}_s(y_{t-1}, y_t)$
- Define *emission scores*: $\phi_o(y_t, \mathbf{x}) = \mathbf{u}^T \mathbf{f}_o(y_t, t, \mathbf{x})$
- Then the CRF probability distribution can be written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} \phi_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \phi_o(y_t, t, \mathbf{x})\right)$$

- **Inference** = finding most likely sequence of states:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathrm{argmax}_{\mathbf{y}}\ p(\mathbf{y}|\mathbf{x}) \\ &= \mathrm{argmax}_{\mathbf{y}} \sum_{t=1}^{T} \phi_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \phi_o(y_t, t, \mathbf{x})\end{aligned}$$

The Viterbi Algorithm

1. Initialization

$$\delta_j(1) = \phi_s(\triangleright, j) + \phi_o(j, 1, \mathbf{x})$$

$$\psi_j(1) = 0$$

2. Recursion

$$\delta_j(t) = \max_{1 \le i \le N} \delta_i(t-1) + \phi_s(i,j) + \phi_o(j,t,\mathbf{x})$$

$$\psi_j(t) = \arg \max_{1 \le i \le N} \delta_i(t-1) + \phi_s(i,j) + \phi_o(j,t,\mathbf{x})$$

3. Termination

$$p(\hat{\mathbf{y}}) = \max_{1 \le j \le N} \delta_j(T)$$

$$\hat{\mathbf{y}}_T = \arg \max_{1 \le j \le N} \delta_j(T)$$

$\triangleright$ represents state before first position. Could also add a state $\triangleleft$ after the last position => run recursion until T + 1

4. State sequence backtracking

$$\hat{\mathbf{y}}_{t-1} = \psi_{\hat{\mathbf{y}}_t}(t)$$

Time complexity?

Training CRFs: Gradient

- If $\mathbf{y}^*$ is the true tag sequence for $\mathbf{x}$, then minimize the negative log-likelihood $J(\mathbf{w}, \mathbf{u}) = -\log p(\mathbf{y}^* \mid \mathbf{x})$.
- Gradient formula resembles logistic regression:

$$\frac{\delta J}{\delta \mathbf{w}} = \underbrace{\sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \sum_{t=1}^{T} \mathbf{f}_s(y_{t-1}, y_t)}_{\textit{expected counts}} - \underbrace{\sum_{t=1}^{T} \mathbf{f}_s(y_{t-1}^*, y_t^*)}_{\textit{true counts}}$$

$$\frac{\delta J}{\delta \mathbf{u}} = \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \sum_{t=1}^{T} \mathbf{f}_o(y_t, t, \mathbf{x}) - \sum_{t=1}^{T} \mathbf{f}_o(y_t^*, t, \mathbf{x})$$

Training CRFs: Forward-Backward

- For computing the emission parameters **u**, the expected counts are:

$$\sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \sum_{t=1}^{T} \mathbf{f}_o(y_t, t, \mathbf{x}) = \sum_{t=1}^{T} \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \mathbf{f}_o(y_t, t, \mathbf{x})$$

$$= \sum_{t=1}^{T} \sum_{j\in S} p(y_t = j|\mathbf{x}) \mathbf{f}_o(j, t, \mathbf{x})$$

(because $\mathbf{f}_o$ depends only on the state at position *t*, we marginalized over all possible states at all other positions)

- To compute $p(y_t = j|\mathbf{x})$ we use the same forward-backward procedure as for HMMs ($\gamma_j(t)$ on slide 97).

$$p(y_t = j|\mathbf{x}) = \frac{\alpha_j(t)\beta_j(t)}{\sum_{i\in S} \alpha_i(t)\beta_i(t)}$$

The Forward Procedure for CRFs

Define *forward score* $\alpha_j(t)$ = the sum of the (unnormalized) scores of all paths leading to state *j* at position *t*.

1. Initialization

$$\alpha_j(1) = \exp\{\phi_s(\triangleright, j) + \phi_o(j, 1, \mathbf{x})\}$$

2. Recursion:

$$\alpha_j(t) = \sum_{i \in S} \alpha_i(t-1) \exp\{\phi_s(i, j) + \phi_o(j, t, \mathbf{x})\}$$

3. Termination:

$$Z(\mathbf{x}) = \sum_{j \in S} \alpha_j(T)$$

The Backward Procedure for CRFs

Define *backward score* $\beta_i(t)$ = the sum of the (unnormalized) scores of all paths leading backwards to state *i* at position *t*.

1. Initialization

$$\beta_i(T) = 1$$

2. Recursion:

$$\beta_i(t) = \sum_{j \in S} \beta_j(t+1) \exp\{\phi_s(i,j) + \phi_o(j, t+1, \mathbf{x})\}$$

3. Termination:

$$Z(\mathbf{x}) = \sum_{j \in S} \beta_j(1) \exp\{\phi_s(\triangleright, j) + \phi_o(j, 1, \mathbf{x})\}$$

Homework Exercise

- Rewrite Viterbi, forward-backward for CRFs:
  - For when we also use a special end of sequence symbol $\triangleleft$.
  - For when we do not use either start or end of sequence symbols.
    - This is how the homework skeleton code is implemented.

Conditional Random Fields (CRFs)

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} \phi_s(y_{t-1}, y_t) + \sum_{t=1}^{T} \phi_o(y_t, t, \mathbf{x})\right)$$

- Inference with Viterbi:

$$\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$$

- Learning:
  - use **forward-backward** to compute marginals $p(y_t = j|\mathbf{x})$
  - then compute gradient with respect to observation features:

$$\frac{\delta J}{\delta \mathbf{u}} = \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \sum_{t=1}^{T} \mathbf{f}_o(y_t, t, \mathbf{x}) - \sum_{t=1}^{T} \mathbf{f}_o(y_t^*, t, \mathbf{x})$$

Conditional Random Fields (CRFs)

- For training the transition parameters **w**, the expected counts are:

$$\sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \sum_{t=1}^{T} \mathbf{f}_s(y_{t-1}, y_t) = \sum_{t=1}^{T} \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x})\mathbf{f}_s(y_{t-1}, y_t)$$

$$= \sum_{t=1}^{T} \sum_{i,j\in S} p(y_{t-1} = i, y_t = j|\mathbf{x})\mathbf{f}_s(i, j)$$

- To compute $p(y_{t-1} = i, y_t = j|\mathbf{x})$ we can use a similar forward-backward procedure as for HMMs ($\xi_{ij}(t)$ on slide 97).
  - We will not use them for the homework exercise, observation features are often sufficient for good performance.

CRF Training

```
for each epoch
    for each example
        extract features on each emission and transition (use cache)
        compute marginal probabilities with *forward-backward*
        compute potentials $\phi_s$ and $\phi_o$ based on features + weights
        accumulate gradient over all emissions and transitions
        do gradient update
```

Implementation & Debugging

- Cache feature vectors.
- Compute both the gradient and the objective value at the same time.
  - Same is done in neural networks.
- Exploit sparsity in feature vectors where possible, e.g. in feature vectors and gradients.
- Do all dynamic program computation in *log space* to avoid underflow / overflow.
- **Inference**:
  - Forward-Backward should be the same at all positions.
  - Check probabilities look reasonable for features correlated with the tag.
- **Learning**:
  - Train on a small subset, check that the loss is going down.,

Notes on CRFs

- Observation features can depend on tag bigrams $\phi_o(j, j-1, t, \mathbf{x})$
  - Still linear time complexity for inference.
- The **max-product** algorithm on factor graphs is a generalization of Viterbi to tree-structured CRFs:
  - **Sum-product** is generalization of forward-backward.
- Inference in general CRFs (arbitray graphs) is NP-hard:
  - Exact inference (but potential exponential) with **junction-tree** alg.
  - Approximate inference with **loopy belief propagation**.
  - Still an active area of research. **Beware the Deep Learning atractor**!

Named Entity Recognition (NER)

$\overbrace{\text{Judson Brewer}}^{\textit{Person}}$, a researcher at $\overbrace{\text{Yale}}^{\textit{Organization}}$, noticed that his scans and $\overbrace{\text{Robin}}^{\textit{Person}}$ ’s looked remarkably alike .

B-Per	I-Per	O	O	O	O	B-Org	O	O	O	O	O	O	B-Per	O	…
Judson	Brewer	,	a	researcher	at	Yale	,	noticed	that	his	scans	and	Robin	’s	…

Word features:
- Word identity.
- Prefix/suffix.
- Capitalization.
- Word ‘shape’.
- Word clusters.

Context features:
- Words before / after.
- Tags before / after.
- …

Gazeteeeers:
- An entry.
- First token in an entry.

Named Entity Recognition (NER)

Feature engineering for $\phi_o(j, t, \mathbf{x})$

”The key figure in the marriage of Al Hubbard and Silicon Valley was Myron Stolaroff. Born in Roswell, New Mexico, in 1920, Stolaroff studied engineering at Stanford and was one of Ampex’s very first employees.”

Pollan, Michael. How to Change Your Mind (p. 176). Penguin Publishing Group. Kindle Edition.

Collective NER

- If a name appears multiple times in a document, it is likely to have the same entity type:
  - The **Charlotte** area includes a diverse range of businesses
  - In December 2002, **Charlotte** was hit by an ice storm.

- Use more complex PGMs with approximate inference to model these dependences:
  - Relational Markov Networks [Bunescu & Mooney, ACL 2004].
  - Skip CRFs [Sutton & McCallum, SRL workshop 2004].
  - Gibbs sampling with annealing [Finkel et al., ACL 2005].
  - Integer linear programming [Roth & Yih, ICML 2005].

Segmental vs. Linear-Chain CRFs

- Linear-chain CRFs map tokens to labels:

O	O	O	Per	Per	O	Loc	Loc	O	Per	Per
the	marriage	of	Al	Hubbard	and	Silicon	Valley	was	Myron	Stolaroff

- Segmental CRFs jointly segment the sequence and map segments to lables:

O	O	O	Per	O	Loc	O	Per
[the]	[marriage]	[of]	[Al Hubbard]	[and]	[Silicon Valley]	was	[Myron Stolaroff]

Semi-Markov CRFs

[Sarawagi & Cohen, ICML 2004]

Segmentation $\mathbf{s} = [s_1, s_2, \ldots, s_K]$, where each $s_k = [b_k, e_k)$, $b_{k+1} = e_k$

Labeling $\mathbf{y} = [y_1, y_2, \ldots, y_K]$

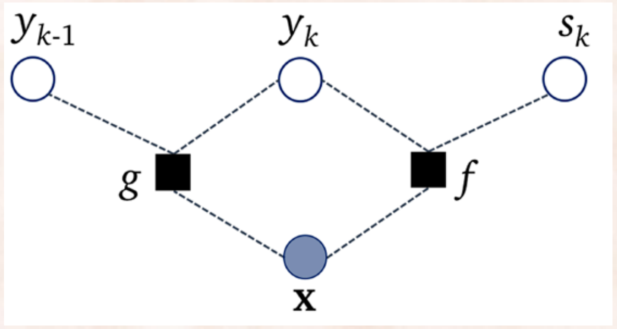
Segmentation and labeling distribution:

$$P(\mathbf{s},\mathbf{y}|\mathbf{x},\mathbf{w},\mathbf{u}) = \frac{e^{\mathbf{w}^T\mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x})+\mathbf{u}^T\mathbf{G}(\mathbf{s},\mathbf{y},\mathbf{x})}}{Z(\mathbf{x})}$$

$$\mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x}) = \sum_{k=1}^{K}\mathbf{f}(s_k, y_k, \mathbf{x})$$

$$\mathbf{G}(\mathbf{s},\mathbf{y},\mathbf{x}) = \sum_{k=1}^{K}\mathbf{g}(y_k, y_{k-1}, \mathbf{x})$$

Factor graph representation:



Semi-Markov CRFs

[Sarawagi & Cohen, ICML 2004]

$$P(\mathbf{s},\mathbf{y}|\mathbf{x},\mathbf{w},\mathbf{u}) = \frac{e^{\mathbf{w}^T\mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x})+\mathbf{u}^T\mathbf{G}(\mathbf{s},\mathbf{y},\mathbf{x})}}{Z(\mathbf{x})}$$

$$\mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x}) = \sum_{k=1}^{K}\mathbf{f}(s_k, y_k, \mathbf{x})$$

$$\mathbf{G}(\mathbf{s},\mathbf{y},\mathbf{x}) = \sum_{k=1}^{K}\mathbf{g}(y_k, y$$

Viterbi algorithm used here too …

$$\hat{\mathbf{s}},\hat{\mathbf{y}} = \arg\max_{\mathbf{s},\mathbf{y}} P(\mathbf{s},\mathbf{y}|\mathbf{x},\mathbf{w},\mathbf{u})$$

$$= \arg\max_{\mathbf{s},\mathbf{y}} \mathbf{w}^T\mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x}) + \mathbf{u}^T\mathbf{G}(\mathbf{s},\mathbf{y},\mathbf{x})$$

$$= \arg\max_{\mathbf{s},\mathbf{y}} \mathbf{w}^T\sum_{k=1}^{K}\mathbf{f}(s_k, y_k, \mathbf{x}) + \mathbf{u}^T\sum_{k=1}^{K}\mathbf{g}(y_k, y_{k-1}, \mathbf{x})$$

Semi-Markov CRFs

[Sarawagi & Cohen, ICML 2004]

$$\hat{\mathbf{s}}, \hat{\mathbf{y}} = \arg\max_{\mathbf{s},\mathbf{y}} P(\mathbf{s}, \mathbf{y} | \mathbf{x}, \mathbf{w}, \mathbf{u})$$

$$= \arg\max_{\mathbf{s},\mathbf{y}} \mathbf{w}^T \mathbf{F}(\mathbf{s}, \mathbf{y}, \mathbf{x}) + \mathbf{u}^T \mathbf{G}(\mathbf{s}, \mathbf{y}, \mathbf{x})$$

$$= \arg\max_{\mathbf{s},\mathbf{y}} \mathbf{w}^T \underbrace{\sum_{k=1}^{K} \mathbf{f}(s_k, y_k, \mathbf{x}) + \mathbf{u}^T \sum_{k=1}^{K} \mathbf{g}(y_k, y_{k-1}, \mathbf{x})}_{\text{score } V(|\mathbf{x}|, y_K)}$$

$V(i, y)$ = the largest score of a partial segmentation whose last segment ends at position i and has label y, where $i = 1 \ldots |\mathbf{x}|$.

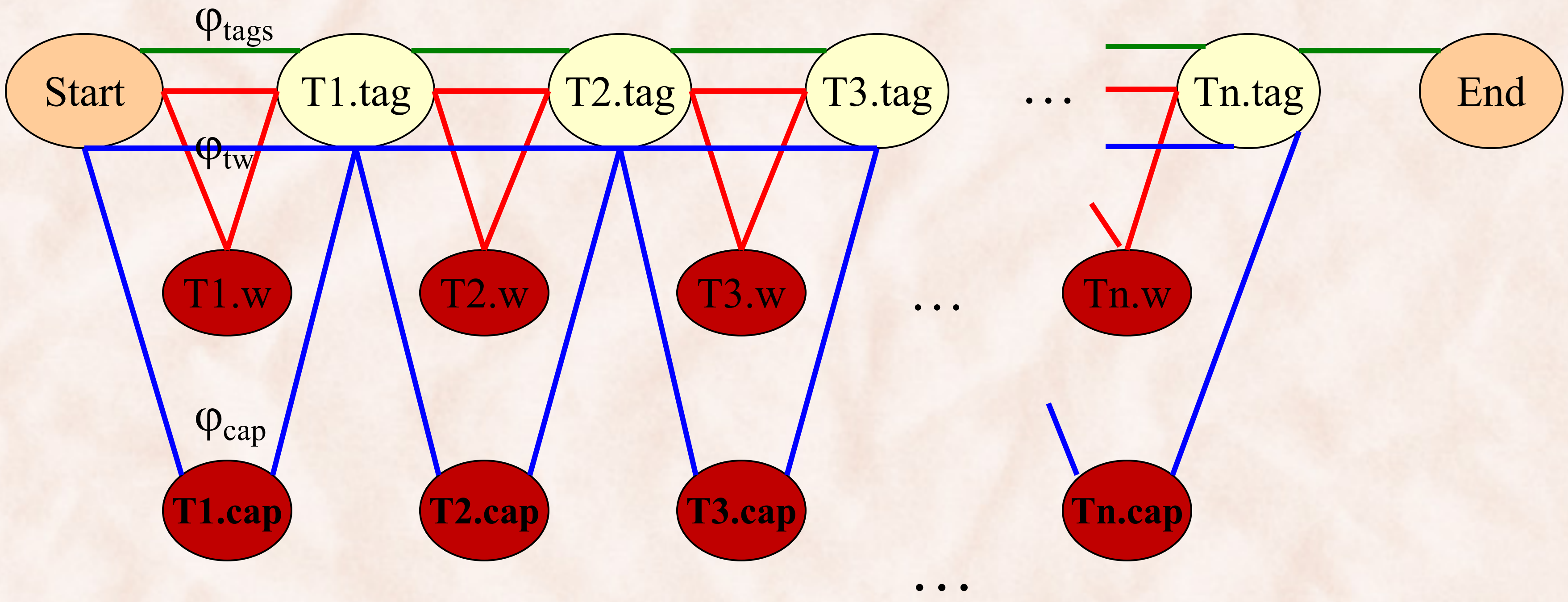
$V(i, y)$ can be computed using a Viterbi-like procedure:

$$V(i,y) = \max_{y', 1 \le l \le L} V(i-l, y') + \mathbf{w}^T \mathbf{f}(\langle i-l+1, i \rangle, y, \mathbf{x}) + \mathbf{u}^T \mathbf{g}(y, y', \mathbf{x})$$

[Lafferty, McCallum & Pereira 2001]

Part-of-speech Tagging

Sentence *S* = a sequence of tokens *T1, …, Tn (tokens as entities)*



- *Tj.tag* – the POS tag at position *j*
- *Tj.w* – *true* if word *w* occurs at position *j*
- *Tj.cap* – *true* if word at position *j* begins with capital letter
- …

Supplemental Reading

- Section 7.5 from Eisenstein (after reading previous ones).
  - Section 7.5.3 on CRFs.