

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 6: Conditional Random Fields

Previous Lecture

- Sequence Prediction
 - Input: $x = (x^1, \dots, x^M)$
 - Predict: $y = (y^1, \dots, y^M)$
 - Naïve full multiclass: exponential explosion
 - Independent multiclass: strong independence assumption
- Hidden Markov Models
 - Generative model: $P(y^i | y^{i-1}), P(x^i | y^i)$
 - Prediction using Bayes's Rule + Viterbi
 - Train using Maximum Likelihood

Outline of Today

- Long Prelude:
 - Generative vs Discriminative Models
 - Naïve Bayes
- Conditional Random Fields
 - Discriminative version of HMMs

Generative vs Discriminative

- Generative Models: Hidden Markov Models
 - Joint Distribution: $P(x,y)$ ← Mismatch!
 - Uses Bayes's Rule to predict: $\operatorname{argmax}_y P(y|x)$ ↗
 - Can generate new samples (x,y)
- Discriminative Models: Conditional Random Fields
 - Conditional Distribution: $P(y|x)$ ← Same thing!
 - Can use model directly to predict: $\operatorname{argmax}_y P(y|x)$ ↗
- Both trained via Maximum Likelihood

Naïve Bayes

- Binary (or Multiclass) prediction $x \in R^D$
- Model joint distribution (Generative): $y \in \{-1, +1\}$

$$P(x, y) = P(x | y)P(y)$$

- “Naïve” independence assumption:

$$P(x | y) = \prod_{d=1}^D P(x^d | y)$$

- Prediction via:

$$\operatorname{argmax}_y P(y | x) = \operatorname{argmax}_y P(x | y)P(y) = \operatorname{argmax}_y P(y) \prod_{d=1}^D P(x^d | y)$$

Naïve Bayes

$P(x_d=1 y)$	$y=-1$	$y=+1$
$P(x^1=1 y)$	0.5	0.7
$P(x^2=1 y)$	0.9	0.4
$P(x^3=1 y)$	0.1	0.5

$P(y)$
$P(y=-1) = 0.4$
$P(y=+1) = 0.6$

$$x \in R^D$$

$$y \in \{-1, +1\}$$

- Prediction:

$$\operatorname{argmax}_y P(y | x) = \operatorname{argmax}_y P(x | y)P(y) = \operatorname{argmax}_y P(y) \prod_{d=1}^D P(x^d | y)$$

x	$P(y=-1 x)$	$P(y=+1 x)$	Predict
(1,0,0)	$0.4 * 0.5 * 0.1 * 0.9 = 0.018$	$0.6 * 0.7 * 0.6 * 0.5 = 0.126$	$y = +1$
(0,1,1)	$0.4 * 0.5 * 0.9 * 0.1 = 0.018$	$0.6 * 0.3 * 0.4 * 0.5 = 0.036$	$y = +1$
(0,1,0)	$0.4 * 0.5 * 0.9 * 0.9 = 0.162$	$0.6 * 0.3 * 0.4 * 0.5 = 0.036$	$y = -1$

Naïve Bayes

- Matrix Formulation:

$$P(x, y) = P(y) \prod_{d=1}^D P(x^d | y) = A_y \prod_{d=1}^D O_{x^d, y}^d$$

$$O_{a,b}^d = P(x^d = a | y = b) \quad A_b = P(y = b)$$

(Each $O_{*,b}^d$ Sums to 1) (Sums to 1)

$P(x_d=1 y)$	$y=-1$	$y=+1$
$P(x^1=1 y)$	0.5	0.7
$P(x^2=1 y)$	0.9	0.4
$P(x^3=1 y)$	0.1	0.5

$P(y)$
$P(y=-1) = 0.4$
$P(y=+1) = 0.6$

$$x \in R^D$$

$$y \in \{-1, +1\}$$

Naïve Bayes

- Train via Max Likelihood:

$$S = \{(x_i, y_i)\}_{i=1}^N$$

$$\operatorname{argmax}_{A,O} \prod_{i=1}^N P(x_i, y_i) = \prod_{i=1}^N P(y_i) \prod_{d=1}^D P(x_i^d | y_i) \quad \begin{array}{l} x \in R^D \\ y \in \{-1, +1\} \end{array}$$

- Estimate $P(y)$ and each $P(x^d | y)$ from data
 - Count frequencies

$$A_z = P(y = z) = \frac{\sum_{i=1}^N 1_{[y_i=z]}}{N} \quad O_{a,z}^d = P(x^d = a | y = z) = \frac{\sum_{i=1}^N 1_{[(y_i=z) \wedge (x_i^d=a)]}}{\sum_{i=1}^N 1_{[y_i=z]}}$$

Naïve Bayes vs HMMs

- Naïve Bayes:

$$P(x, y) = P(y) \prod_{d=1}^D P(x^d | y)$$

- Hidden Markov Models:

$$P(x, y) = P(\text{End} | y^M) \underbrace{\prod_{j=1}^M P(y^j | y^{j-1})}_{P(y)} \prod_{i=1}^M P(x^i | y^i)$$

“Naïve” Generative Independence Assumption

- **HMMs $\approx$ 1st order variant of Naïve Bayes!**
(just one interpretation...)

Naïve Bayes vs HMMs

- Naïve Bayes:

$$P(x, y) = A_y \prod_{d=1}^D O_{x^d, y}^d$$

Diagram annotations for Naïve Bayes:
- $P(y)$ points to A_y
- $P(x|y)$ points to $O_{x^d, y}^d$

- Hidden Markov Models:

$$P(x, y) = A_{End, y^M} \underbrace{\prod_{j=1}^M A_{y^j, y^{j-1}}}_{P(y)} \prod_{j=1}^M O_{x^j, y^j}$$

Diagram annotations for HMMs:
- $P(y)$ points to the underbraced transition term $\prod_{j=1}^M A_{y^j, y^{j-1}}$
- $P(x|y)$ points to O_{x^j, y^j}
- **"Naïve" Generative Independence Assumption** points to the observation term O_{x^j, y^j}

- HMMs $\approx$ 1st order variant of Naïve Bayes!
(just one interpretation...)

Summary: Naïve Bayes

- Joint model of (x, y) :

- “Naïve” independence assumption each x^d

$$P(x, y) = P(y) \prod_{d=1}^D P(x^d | y)$$

“Generative Model”
(can sample new data)

- Use Bayes’s Rule for prediction:

$$\operatorname{argmax}_y P(y | x) = \operatorname{argmax}_y P(x | y) P(y) = \operatorname{argmax}_y P(y) \prod_{d=1}^D P(x^d | y)$$

- Maximum Likelihood Training:
 - Count Frequencies

Learn Conditional Prob.?

- Weird to train to maximize:

$$\operatorname{argmax}_{A,O} \prod_{i=1}^N P(x_i, y_i) = \operatorname{argmax}_{A,O} \prod_{i=1}^N P(y_i) \prod_{d=1}^D P(x_i^d | y_i)$$

$$S = \{(x_i, y_i)\}_{i=1}^N$$

$$x \in R^D$$

$$y \in \{-1, +1\}$$

- When goal should be to maximize:

$$\operatorname{argmax}_{A,O} \prod_{i=1}^N P(y_i | x_i)$$

Breaks independence!

Can no longer use count statistics

~~$$P(x^d = a | y = z) = \frac{\sum_{i=1}^N 1_{[(y_i=z) \wedge (x_i^d=a)]}}{\sum_{i=1}^N 1_{[y_i=z]}}$$~~

$$p(x) = \sum_y P(x, y) = \sum_y P(y) \prod_{d=1}^D P(x^d | y)$$

*HMMs suffer same problem

Learn Conditional Prob.?

- Weird to train to maximize:

$$S = \{(x_i, y_i)\}_{i=1}^N$$

In general, you should maximize the likelihood of the model you define!

So if you define joint model $P(x, y)$, then maximize $P(x, y)$ on training data.

*HMMs suffer same problem

Summary: Generative Models

- Joint model of (x,y) :
 - Compact & easy to train...
 - ...with ind. assumptions
 - E.g., Naïve Bayes & HMMs

$$P(x, y)$$

Θ often used to denote
all parameters of model

- Maximize Likelihood Training:


$$\operatorname{argmax}_{\Theta} \prod_{i=1}^N P(x_i, y_i)$$

- Mismatch w/ prediction goal:
 - But hard to maximize $P(y|x)$

$$\operatorname{argmax}_y P(y|x)$$

$$S = \{(x_i, y_i)\}_{i=1}^N$$

Discriminative Models

- Conditional model: $P(y | x)$
 - Directly model prediction goal
- Maximum Likelihood: $\operatorname{argmax}_{\Theta} \prod_{i=1}^N P(y_i | x_i)$
- Matches prediction goal: $\operatorname{argmax}_y P(y | x)$
- **What does $P(y | x)$ look like?**

First Try

- Model $P(y|x)$ for every possible x

$P(y=1 x)$	x^1	x^2
0.5	0	0
0.7	0	1
0.2	1	0
0.4	1	1

$$x \in \{0,1\}^D$$

$$y \in \{-1,+1\}$$

- Train by counting frequencies
- **Exponential in # input variables D !**
 - Need to assume something... what?

Log Linear Models!

(Logistic Regression)

$$P(y | x) = \frac{\exp\{w_y^T x - b_y\}}{\sum_k \exp\{w_k^T x - b_k\}} \quad \begin{array}{l} x \in R^D \\ y \in \{1, 2, \dots, K\} \end{array}$$

- “Log-Linear” assumption
 - Model representation to linear in D
 - Most common discriminative probabilistic model

Prediction:

$$\operatorname{argmax}_y P(y | x)$$

Training:

$$\operatorname{argmax}_{\Theta} \prod_{i=1}^N P(y_i | x_i)$$

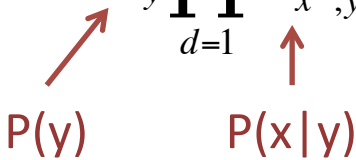
← Match! →

Naïve Bayes vs Logistic Regression

- Naïve Bayes:

- Strong ind. assumptions
- Super easy to train...
- ...but mismatch with prediction

$$P(x, y) = A_y \prod_{d=1}^D O_{x^d, y}^d$$


 $P(y)$ $P(x|y)$

- Logistic Regression:

- “Log Linear” assumption
 - Often more flexible than Naïve Bayes
- Harder to train (gradient desc.)...
- ...but matches prediction

$$P(y|x) = \frac{\exp\{w_y^T x - b_y\}}{\sum_k \exp\{w_k^T x - b_k\}}$$

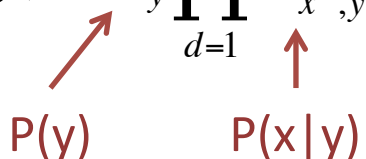
$$x \in R^D$$
$$y \in \{1, 2, \dots, K\}$$

Naïve Bayes vs Logistic Regression

- NB has K parameters for $P(y)$ (i.e., A)
- LR has K parameters for bias b
- NB has $K \cdot D$ parameters for $P(x|y)$ (i.e, O)
- LR has $K \cdot D$ parameters for w
- **Same number of parameters!**

Naïve Bayes

$$P(x, y) = A_y \prod_{d=1}^D O_{x^d, y}^d$$



$P(y)$ $P(x|y)$

Logistic Regression

$$P(y | x) = \frac{e^{w_y^T x - b_y}}{\sum_k e^{w_k^T x - b_k}}$$

$$x \in \{0, 1\}^D$$
$$y \in \{1, 2, \dots, K\}$$

Naïve Bayes vs Logistic Regression

Intuition:

Both models have same “capacity”
NB spends a lot of capacity on $P(x)$
LR spends all of capacity on $P(y|x)$

No Model Is Perfect!

(Especially on finite training set)
NB will trade off $P(y|x)$ with $P(x)$
LR will fit $P(y|x)$ as well as possible

Generative	Discriminative
$P(x,y)$ - Joint model over x and y - Cares about everything	$P(y x)$ (when probabilistic) - Conditional model - Only cares about predicting well
Naïve Bayes, HMMs	Logistic Regression, CRFs
Max Likelihood	Max (Conditional) Likelihood (=minimize log loss) - Can pick any loss based on y - Hinge Loss, Squared Loss, etc.
Always Probabilistic	Not Necessarily Probabilistic Certainly never joint over $P(x,y)$
Often strong assumptions Keeps training tractable	More flexible assumptions Focuses entire model on $P(y x)$
Mismatch between train & predict Requires Bayes's rule	Train to optimize predict goal
Can sample anything	Can only sample y given x
Can handle missing values in x	Cannot handle missing values in x

Conditional Random Fields

“Log-Linear” 1st Order Sequential Model

$$P(y | x) = \frac{1}{Z(x)} \exp \left\{ \sum_{j=1}^M \left(u_{y^j, y^{j-1}} + w_{y^j, x^j} \right) \right\}$$

$$Z(x) = \sum_{y'} \exp \{ F(y', x) \}$$

aka “Partition Function”

$$F(y, x) \equiv \sum_{j=1}^M \left(u_{y^j, y^{j-1}} + w_{y^j, x^j} \right)$$

Scoring Function

Scoring transitions

Scoring input features

$$P(y | x) = \frac{\exp \{ F(y, x) \}}{Z(x)}$$

$$\log P(y | x) = F(y, x) - \log(Z(x))$$

y^0 = special start state

- $x = \text{"Fish Sleep"}$
- $y = (N, V)$

$$P(y | x) = \frac{1}{Z(x)} \exp \left\{ \sum_{j=1}^M (u_{y^j, y^{j-1}} + w_{y^j, x^j}) \right\}$$

$u_{N,V}$ →

	$u_{N,*}$	$u_{V,*}$
$u_{*,N}$	-2	1
$u_{*,V}$	2	-2
$u_{*,Start}$	1	-1

← $w_{V,Fish}$

	$w_{N,*}$	$w_{V,*}$
$w_{*,Fish}$	2	1
$w_{*,Sleep}$	1	0

$$P(N, V | \text{"Fish Sleep"}) = \frac{1}{Z(x)} \exp \{ u_{N,Start} + w_{N,Fish} + u_{V,N} + w_{V,Sleep} \} = \frac{1}{Z(x)} \exp \{ 4 \}$$

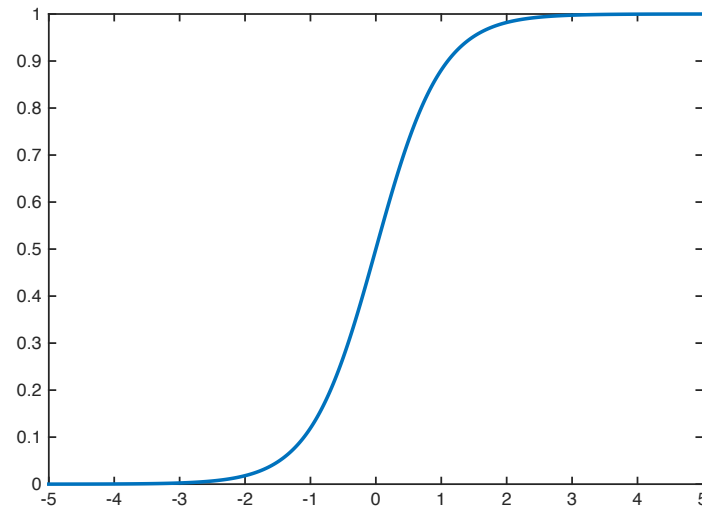
$$Z(x) = \text{Sum} \left(\begin{array}{|c|c|} \hline y & \exp(F(y,x)) \\ \hline (N,N) & \exp(1+2-2+1) = \exp(2) \\ \hline (N,V) & \exp(1+2+2+0) = \exp(4) \\ \hline (V,N) & \exp(-1+1+2+1) = \exp(3) \\ \hline (V,V) & \exp(-1+1-2+0) = \exp(-2) \\ \hline \end{array} \right)$$

- $x = \text{"Fish Sleep"}$
- $y = (N, V)$

$$P(N, V | \text{"Fish Sleep"}) = \frac{1}{Z(x)} \exp \{ u_{N, \text{Start}} + w_{N, \text{Fish}} + u_{V, N} + w_{V, \text{Sleep}} \}$$

$$P(N, V | \text{"Fish Sleep"})$$

*hold other parameters fixed



$$u_{N, \text{Start}} + v_{N, \text{Fish}} + u_{V, N} + v_{V, \text{Sleep}}$$

Basic Conditional Random Field

- Directly models $P(y|x)$
 - Discriminative
 - Log linear assumption
 - Same #parameters as HMM
 - 1st Order Sequential LR

CRF spends all model capacity on $P(y|x)$, rather than $P(x,y)$



$$F(y, x) \equiv \sum_{j=1}^M (u_{y^j, y^{j-1}} + w_{y^j, x^j})$$

$$P(y|x) = \frac{\exp\{F(y, x)\}}{\sum_{y'} \exp\{F(y', x)\}}$$

- **How to Predict?**
- **How to Train?**
- **Extensions?**

$$\log P(y|x) = F(y, x) - \log \left(\sum_{y'} \exp\{F(y', x)\} \right)$$

Predict via Viterbi

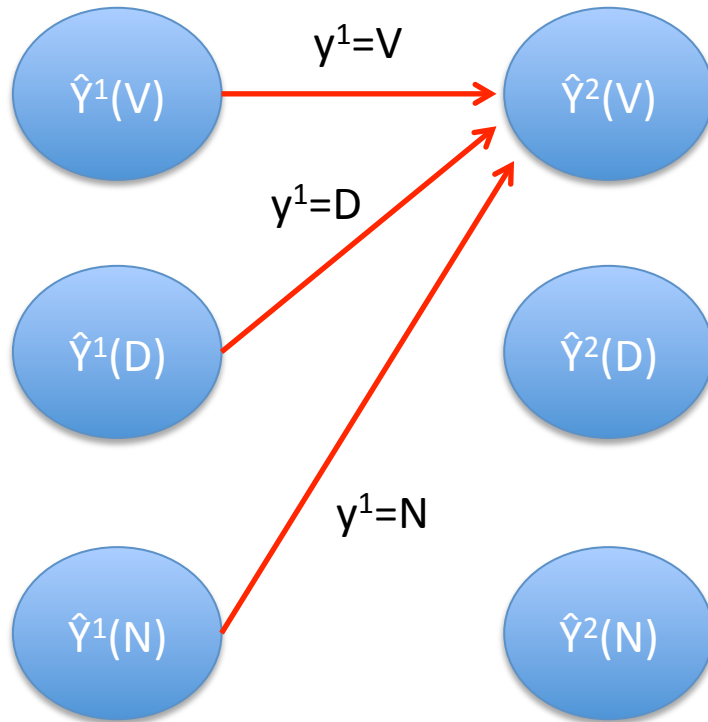
$$\begin{aligned}
 \operatorname{argmax}_y P(y | x) &= \operatorname{argmax}_y \log P(y | x) = \operatorname{argmax}_y F(y, x) \\
 &= \operatorname{argmax}_y \sum_{j=1}^M \left(u_{y^j, y^{j-1}} + w_{y^j, x^j} \right)
 \end{aligned}$$

Scoring transitions
Scoring input features

Maintain length-k prefix solutions	$\hat{Y}^k(T) = \left(\operatorname{argmax}_{y^{1:k-1}} F(y^{1:k-1} \oplus T, x^{1:k}) \right) \oplus T$
Recursively solve for length-(k+1) solutions	$ \begin{aligned} \hat{Y}^{k+1}(T) &= \left(\operatorname{argmax}_{y^{1:k} \in \{\hat{Y}^k(T)\}_T} F(y^{1:k} \oplus T, x) \right) \oplus T \\ &= \left(\operatorname{argmax}_{y^{1:k} \in \{\hat{Y}^k(T)\}_T} F(y^{1:k}, x) + u_{T, y^k} + w_{T, x^{k+1}} \right) \oplus T \end{aligned} $
Predict via best length-M solution	$\operatorname{argmax}_y F(y, x) = \operatorname{argmax}_{y \in \{\hat{Y}^M(T)\}_T} F(y, x)$

Solve: $\hat{Y}^2(V) = \left(\operatorname{argmax}_{y^1 \in \{\hat{Y}^1(T)\}_T} F(y^1, x) + u_{V, y^1} + w_{V, x^2} \right) \oplus V$

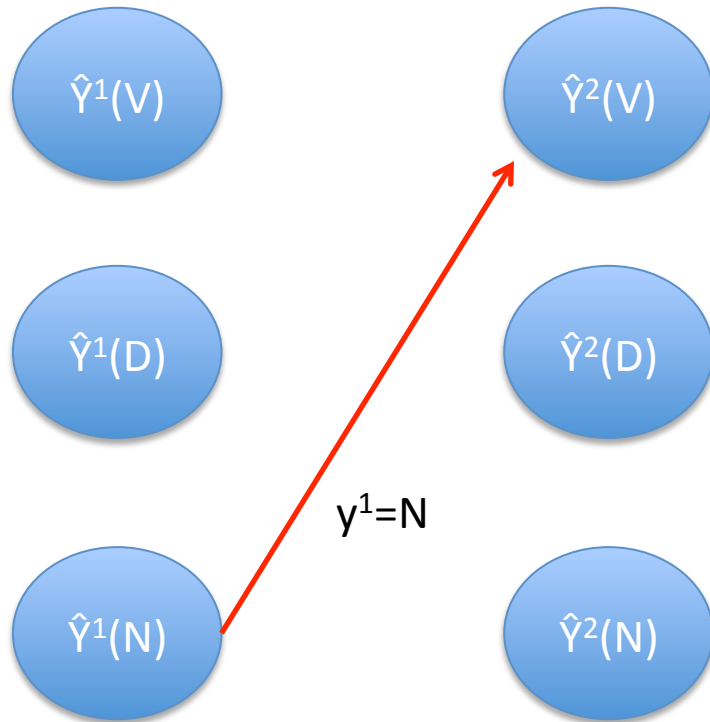
Store each
 $\hat{Y}^1(T)$ & $F(\hat{Y}^1(T), x)$



$\hat{Y}^1(T)$ is just T

Solve: $\hat{Y}^2(V) = \left(\operatorname{argmax}_{y^1 \in \{\hat{Y}^1(T)\}_T} F(y^1, x) + u_{V, y^1} + w_{V, x^2} \right) \oplus V$

Store each
 $\hat{Y}^1(T)$ & $F(\hat{Y}^1(T), x^1)$



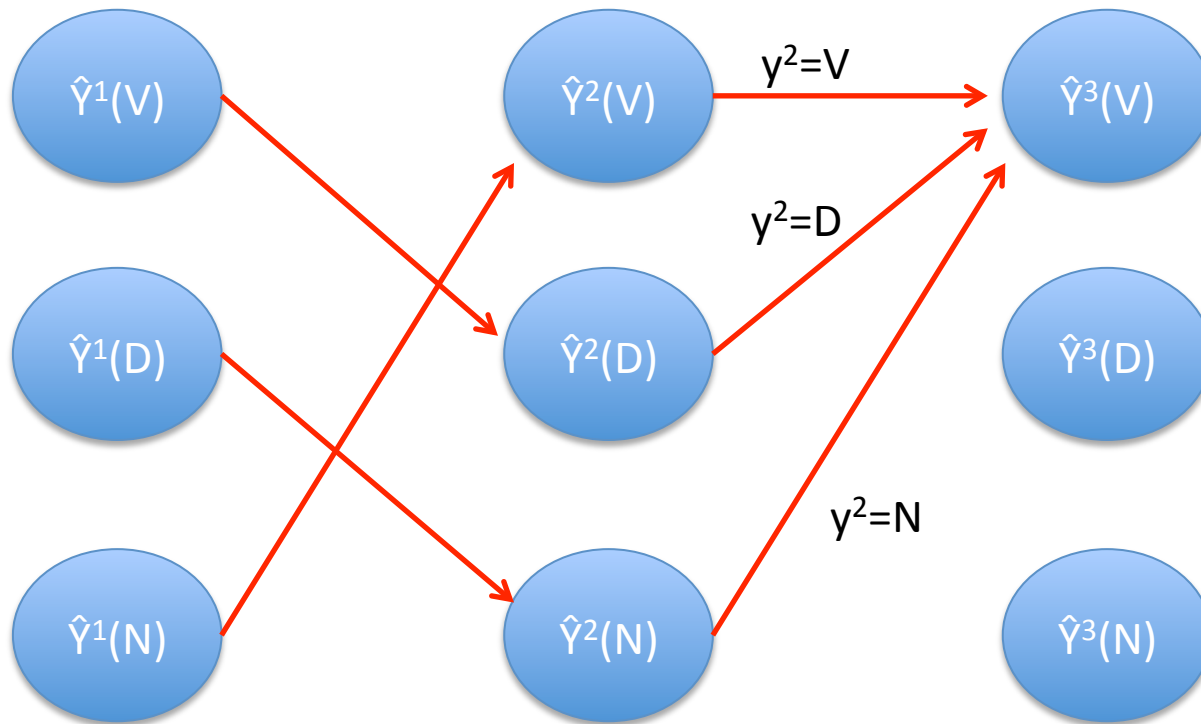
$\hat{Y}^1(T)$ is just T

Ex: $\hat{Y}^2(V) = (N, V)$

Solve:
$$\hat{Y}^3(V) = \left(\operatorname{argmax}_{y^{1:2} \in \{\hat{Y}^2(T)\}_T} F(y^{1:2}, x) + u_{V, y^2} + w_{V, x^3} \right) \oplus V$$

Store each
 $\hat{Y}^1(T)$ & $F(\hat{Y}^1(T), x^1)$

Store each
 $\hat{Y}^2(Z)$ & $F(\hat{Y}^2(Z), x)$



$\hat{Y}^1(Z)$ is just Z

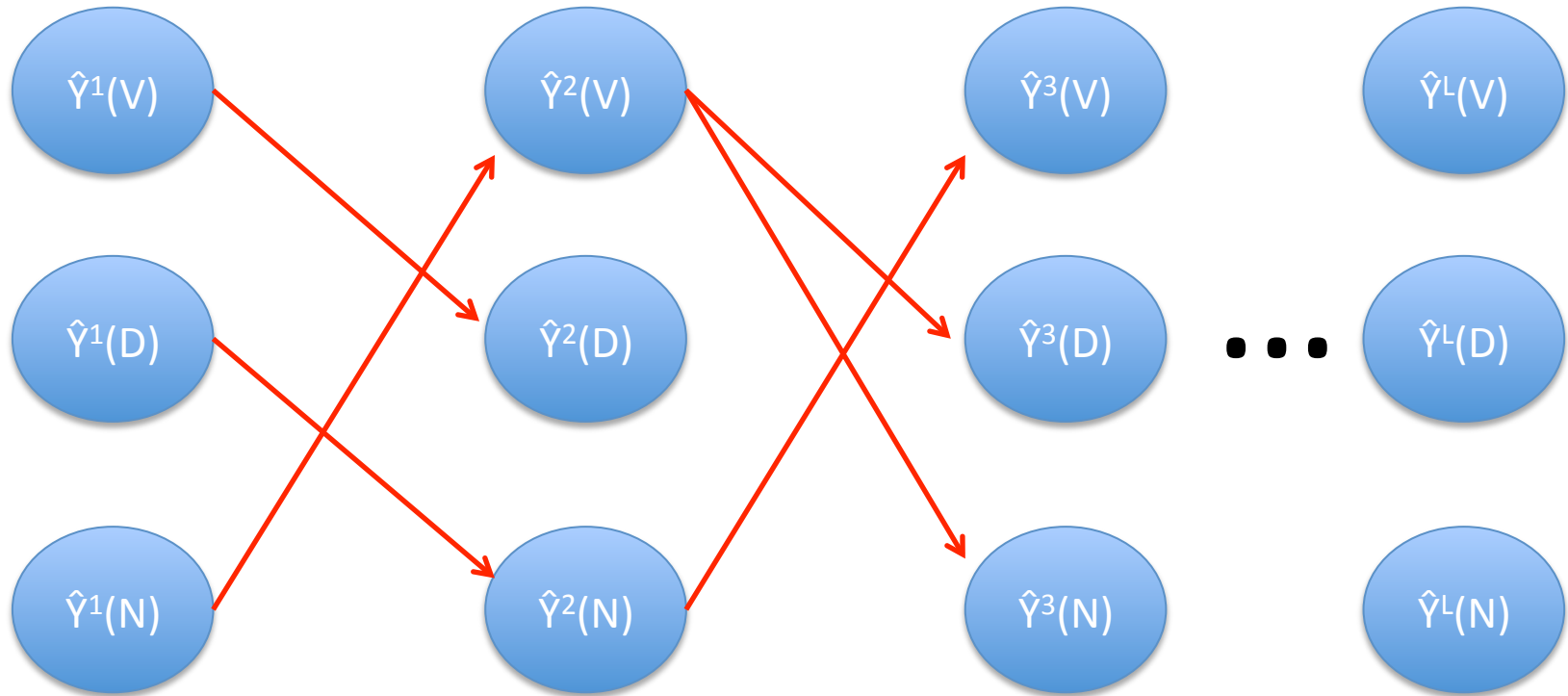
Ex: $\hat{Y}^2(V) = (N, V)$

Solve:
$$\hat{Y}^M(V) = \left(\operatorname{argmax}_{y^{M-1} \in \{\hat{Y}^M(T)\}_T} F(y^{M-1}, x) + u_{V, y^{M-1}} + w_{V, x^M} \right) \oplus V$$

Store each
 $\hat{Y}^1(Z)$ & $F(\hat{Y}^1(Z), x^1)$

Store each
 $\hat{Y}^2(T)$ & $F(\hat{Y}^2(T), x)$

Store each
 $\hat{Y}^3(T)$ & $F(\hat{Y}^3(T), x)$



$\hat{Y}^1(T)$ is just T

Ex: $\hat{Y}^2(V) = (N, V)$

Ex: $\hat{Y}^3(V) = (D, N, V)$

Computing $P(y|x)$

- Viterbi doesn't compute $P(y|x)$
 - Just maximizes the numerator $F(y,x)$

$$P(y|x) = \frac{\exp\{F(y,x)\}}{\sum_{y'} \exp\{F(y',x)\}} \equiv \frac{1}{Z(x)} \exp\{F(y,x)\}$$

- Also need to compute $Z(x)$
 - aka the “Partition Function”

$$Z(x) = \sum_{y'} \exp\{F(y',x)\}$$

Computing Partition Function

- Naive approach is iterate over all y'
 - Exponential time, L^M possible y' !

$$Z(x) = \sum_{y'} \exp\{F(y', x)\} \quad F(y, x) \equiv \sum_{j=1}^M (u_{y^j, y^{j-1}} + w_{y^j, x^j})$$

- Notation: $G^j(a, b) = \exp\{u_{a,b} + w_{a, x^j}\}$

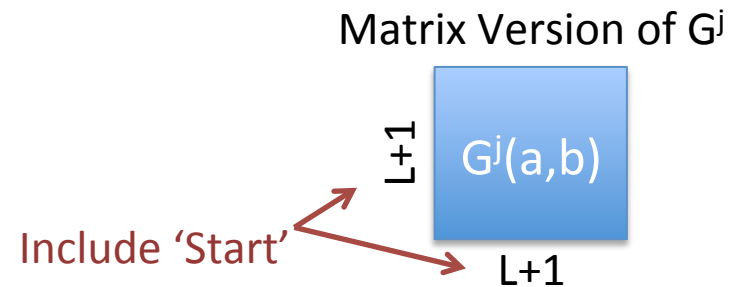
$$P(y | x) = \frac{1}{Z(x)} \prod_{j=1}^M G^j(y^j, y^{j-1})$$

$$Z(x) = \sum_{y'} \prod_{j=1}^M G^j(y'^j, y'^{j-1})$$

Matrix Semiring

$$Z(x) = \sum_{y'} \prod_{j=1}^M G^j(y'^j, y'^{j-1})$$

$$G^j(a, b) = \exp \left\{ u_{a,b} + w_{a,x^j} \right\}$$



$$G^{1:2}(a, b) \equiv \sum_c G^2(a, c) G^1(c, b)$$



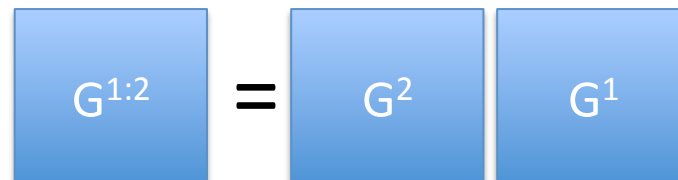
$$G^{i:j}(a, b) \equiv$$

Path Counting Interpretation

- Interpretation $G^1(a,b)$
 - L+1 start & end locations
 - Weight of path from 'b' to 'a' in step 1



- $G^{1:2}(a,b)$
 - Weight of all paths
 - Start in 'b' beginning of Step 1
 - End in 'a' after Step 2



Computing Partition Function

- Consider Length-1 ($M=1$)

$$Z(x) = \sum_a G^1(a, \text{Start})$$

Sum column 'Start' of G^1 !

- $M=2$

$$Z(x) = \sum_{a,b} G^2(b,a)G^1(a, \text{Start}) = \sum_b G^{1:2}(b, \text{Start})$$

Sum column 'Start' of $G^{1:2}$!

- General M

Sum column 'Start' of $G^{1:M}$!

- Do $M (L+1) \times (L+1)$ matrix computations to compute $G^{1:M}$
- $Z(x) = \text{sum column 'Start' of } G^{1:M}$

$$G^{1:M} = G^M G^{M-1} \dots G^2 G^1$$

Train via Gradient Descent

- Similar to Logistic Regression
 - Gradient Descent on negative log likelihood (log loss)

$$\operatorname{argmin}_{\Theta} \sum_{i=1}^N -\log P(y_i | x_i) = \operatorname{argmin}_{\Theta} \sum_{i=1}^N -F(y_i, x_i) + \log(Z(x))$$

Θ often used to denote all parameters of model

Harder to
differentiate!

- First term is easy:

- Recall:

$$F(y, x) \equiv \sum_{j=1}^M \left(u_{y^j, y^{j-1}} + w_{y^j, x^j} \right)$$

$$\partial_{u_{ab}} - F(y, x) = - \sum_{j=1}^M 1_{[(y^j, y^{j-1}) = (a, b)]}$$

$$\partial_{w_{az}} - F(y, x) = - \sum_{j=1}^M 1_{[(y^j, x^j) = (a, z)]}$$

Differentiating Log Partition

Lots of Chain Rule & Algebra!

$$\begin{aligned}
 \partial_{u_{ab}} \log(Z(x)) &= \frac{1}{Z(x)} \partial_{u_{ab}} Z(x) = \frac{1}{Z(x)} \partial_{u_{ab}} \sum_{y'} \exp\{F(y', x)\} \\
 &= \frac{1}{Z(x)} \sum_{y'} \partial_{u_{ab}} \exp\{F(y', x)\} \\
 &= \frac{1}{Z(x)} \sum_{y'} \exp\{F(y', x)\} \partial_{u_{ab}} F(y', x) = \sum_{y'} \frac{\exp\{F(y', x)\}}{Z(x)} \partial_{u_{ab}} F(y', x) \\
 &\stackrel{\text{Definition of } P(y' | x)}{=} \sum_{y'} P(y' | x) \partial_{u_{ab}} F(y', x) = \sum_{y'} \left[P(y' | x) \sum_{j=1}^M 1_{[(y'^j, y'^{j-1})=(a,b)]} \right] \\
 &= \sum_{j=1}^M \sum_{y'} P(y' | x) 1_{[(y'^j, y'^{j-1})=(a,b)]} = \sum_{j=1}^M P(y^j = a, y^{j-1} = b | x)
 \end{aligned}$$

Forward-Backward!

↑
Marginalize over all y'

Optimality Condition

$$\operatorname{argmin}_{\Theta} \sum_{i=1}^N -\log P(y_i | x_i) = \operatorname{argmin}_{\Theta} \sum_{i=1}^N -F(y_i, x_i) + \log(Z(x_i))$$

- Consider one parameter:

$$\partial_{u_{ab}} \sum_{i=1}^N -F(y_i, x_i) = -\sum_{i=1}^N \sum_{j=1}^{M_i} 1_{[(y_i^j, y_i^{j-1})=(a,b)]} \quad \partial_{u_{ab}} \sum_{i=1}^N \log(Z(x_i)) = \sum_{i=1}^N \sum_{j=1}^{M_i} P(y_i^j = a, y_i^{j-1} = b | x_i)$$

- Optimality condition:

$$\sum_{i=1}^N \sum_{j=1}^{M_i} 1_{[(y_i^j, y_i^{j-1})=(a,b)]} = \sum_{i=1}^N \sum_{j=1}^{M_i} P(y_i^j = a, y_i^{j-1} = b | x_i)$$

- Frequency counts = Cond. expectation on training data!**
 - Holds for each component of the model
 - Each component is a “log-linear” model and requires gradient desc.

Forward-Backward for CRFs

$$\alpha^1(a) = G^1(a, \text{Start})$$

$$\alpha^j(a) = \sum_b \alpha^{j-1}(b) G^j(a, b)$$

$$\beta^M(b) = 1$$

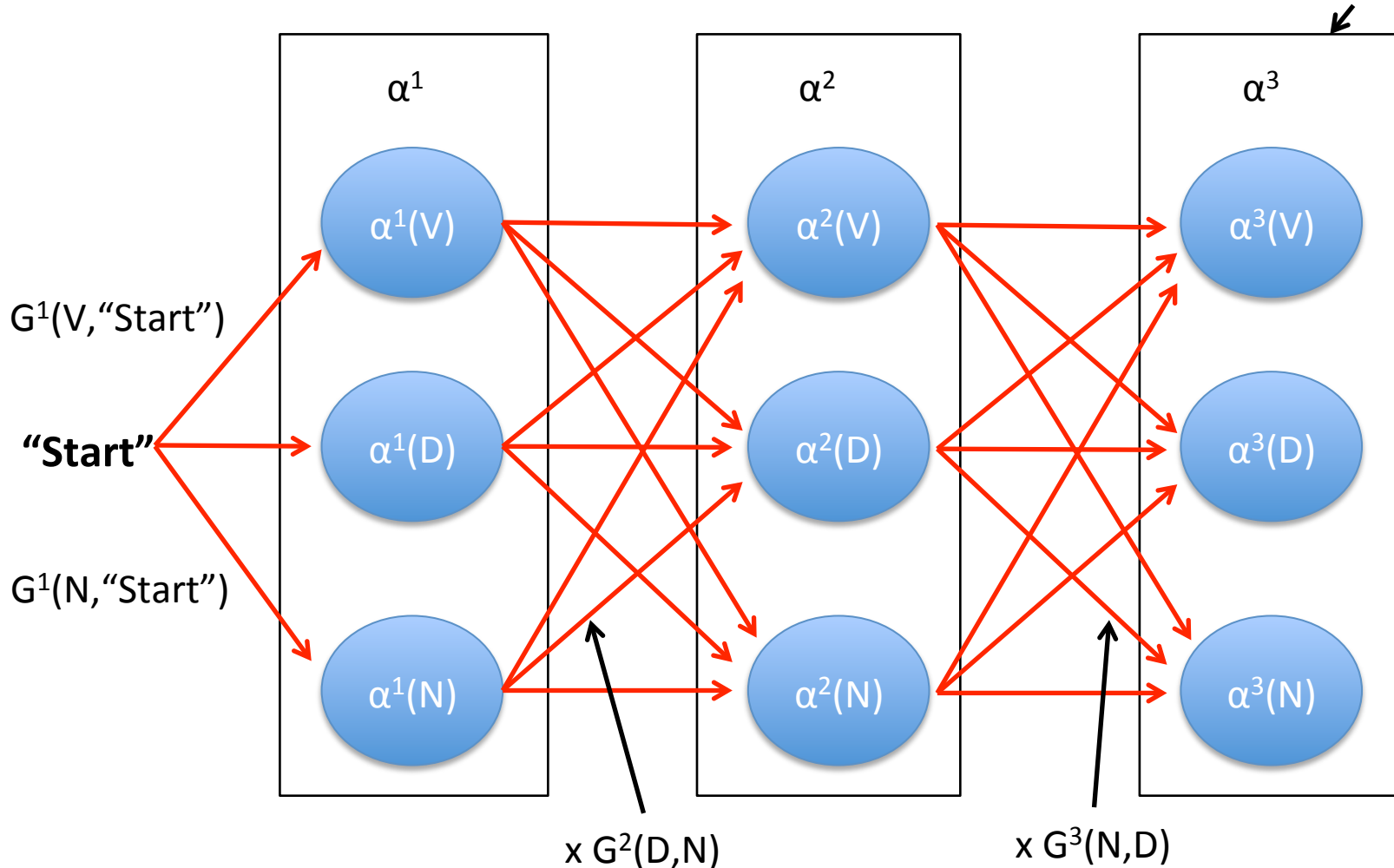
$$\beta^j(b) = \sum_a \beta^{j+1}(a) G^{j+1}(a, b)$$

$$P(y^j = b, y^{j-1} = a \mid x) = \frac{\alpha^{j-1}(a) G^j(b, a) \beta^j(b)}{Z(x)}$$

$$Z(x) = \sum_{y'} \exp\{F(y', x)\} \quad F(y, x) \equiv \sum_{j=1}^M (u_{y^j, y^{j-1}} + w_{y^j, x^j}) \quad G^j(a, b) = \exp\{u_{a,b} + w_{a, x^j}\}$$

Path Interpretation

Total Weight of paths from "Start" to "V" in 3rd step



β just does it backwards

Matrix Formulation

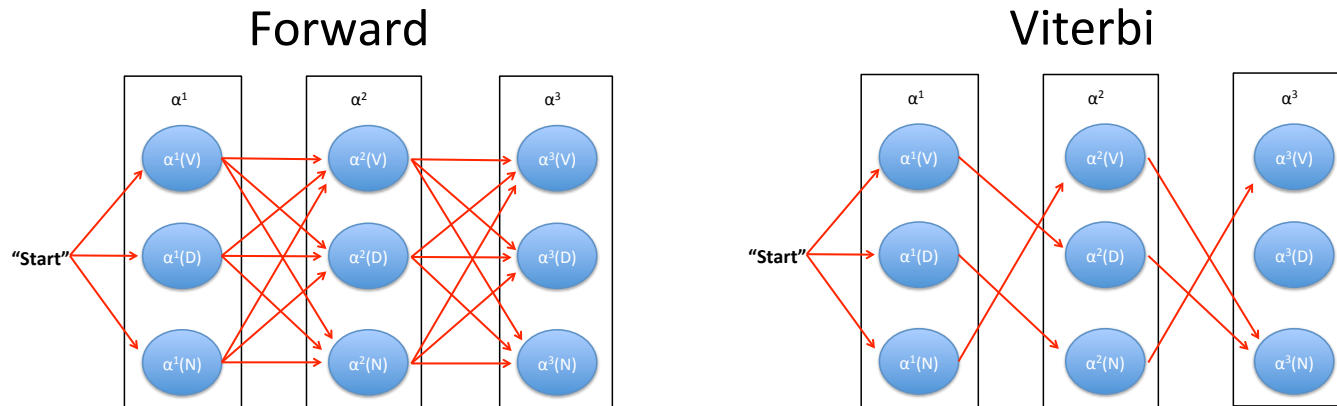
- Use Matrices!
- Fast to compute!
- Easy to implement!

$$\alpha^2 = G^2 \alpha^1$$

$$\beta^6 = (G^2)^T \beta^5$$

Path Interpretation:

Forward-Backward vs Viterbi



- Forward (and Backward) sums over all paths
 - Computes expectation of reaching each state
 - E.g., total (un-normalized) probability of $y^3=\text{Verb}$ over all possible $y^{1:2}$
- Viterbi only keeps the best path
 - Computes best possible path to reaching each state
 - E.g., single highest probability setting of $y^{1:3}$ such that $y^3=\text{Verb}$

Summary: Training CRFs

- Similar optimality condition as HMMs:
 - Match frequency counts of model components!
- $$\sum_{i=1}^N \sum_{j=1}^{M_i} 1_{[(y_i^j, y_i^{j-1})=(a,b)]} = \sum_{i=1}^N \sum_{j=1}^{M_i} P(y_i^j = a, y_i^{j-1} = b | x_i)$$
- Except HMMs can just set the model using counts
 - CRFs need to do gradient descent to match counts
- Run Forward-Backward for expectation
 - Just like HMMs as well

More General CRFs

$$P(y|x) = \frac{\exp\{F(y,x)\}}{\sum_{y'} \exp\{F(y',x)\}}$$

Reduction:

$$\phi_j(a,b|x) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$\leftarrow u_{a,b}$
 $\leftarrow w_{a,x^j}$



$$\theta^T \phi_j(y^j, y^{j-1} | x) = u_{y^j, y^{j-1}} + w_{y^j, x^j}$$

θ is “flattened” weight vector
Can extend $\phi_j(a,b|x)$

New:

$$F(y,x) \equiv \sum_{j=1}^M \theta^T \phi_j(y^j, y^{j-1} | x)$$

Old:

$$F(y,x) \equiv \sum_{j=1}^M (u_{y^j, y^{j-1}} + w_{y^j, x^j})$$

More General CRFs

$$P(y \mid x) = \frac{\exp\{F(y, x)\}}{\sum_{y'} \exp\{F(y', x)\}} \quad F(y, x) \equiv \sum_{j=1}^M \theta^T \phi_j(y^j, y^{j-1} \mid x)$$

1st order Sequence CRFs:

$$F(y, x) \equiv \sum_{j=1}^M \left[\theta_2^T \psi_j(y^j, y^{j-1}) + \theta_1^T \varphi_j(y^j \mid x) \right]$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad \phi_j(a, b \mid x) = \begin{bmatrix} \psi_j(a, b) \\ \varphi_j(b \mid x) \end{bmatrix}$$

Example

$$F(y, x) \equiv \sum_{j=1}^M \left[\theta_2^T \psi_j(y^j, y^{j-1}) + \theta_1^T \varphi_j(y^j | x) \right]$$

Basic formulation
only had first part



$$\varphi_{j,b}(x) = \begin{bmatrix} e_{x^j} \\ 1_{[x^j \in \text{animal}]} \\ e_{x^{j-1}} \end{bmatrix}$$

Various attributes of x

$$\varphi_j(b | x) = \begin{bmatrix} 1_{[b==1]} \varphi_{j,1}(x) \\ 1_{[b==1]} \varphi_{j,2}(x) \\ \vdots \\ 1_{[b==100]} \varphi_{j,100}(x) \\ \vdots \end{bmatrix}$$

All 0's except
1 sub-vector

Stack for each label $y^j=b$

Summary: CRFs

- “Log-Linear” 1st order sequence model
 - Multiclass LR + 1st order components
 - Discriminative Version of HMMs

$$P(y|x) = \frac{\exp\{F(y,x)\}}{\sum_{y'} \exp\{F(y',x)\}}$$

$$F(y,x) \equiv \sum_{j=1}^M [\theta_2^T \psi_j(y^j, y^{j-1}) + \theta_1^T \varphi_j(y^j | x)]$$

- Predict using Viterbi, Train using Gradient Descent
- Need forward-backward to differentiate partition function

Next Week

- Structural SVMs
 - Hinge loss for sequence prediction
- More General Structured Prediction
- Next Recitation:
 - Optimizing non-differentiable functions (Lasso)
 - Accelerated gradient descent
- Homework 2 due in 12 days
 - Tuesday, Feb 3rd at 2pm via Moodle