Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 6:

Conditional Random Fields

Previous Lecture

- Sequence Prediction
  - Input: $x = (x\ ,\ldots,x^M)$
  - Predict: $y = (y^1,\ldots,y^M)$
  - 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: $\text{argmax}_y P(y|x)$
  - Can generate new samples (x,y)
- Discriminative Models: **Conditional Random Fields**
  - Conditional Distribution: P(y|x) ← **Same thing!**
  - Can use model directly to predict: $\text{argmax}_y P(y|x)$
- Both trained via Maximum Likelihood

Naïve Bayes

- Binary (or Multiclass) prediction
- Model joint distribution (Generative):

$$x \in R^D$$

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

$$P(x,y) = P(x \mid y)P(y)$$

- "Naïve" independence assumption:

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

- Prediction via:

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

http://en.wikipedia.org/wiki/Naive_Bayes_classifier

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:

$$\underset{y}{\operatorname{argmax}}\, P(y \mid x) = \underset{y}{\operatorname{argmax}}\, P(x \mid y) P(y) = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{d=1}^{D} P(x^d \mid y)$$

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:

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

$$O^d_{a,b} = P(x^d = a \mid y = b) \qquad A_b = P(y = b)$$

(Each $O^d_{*,b}$ 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:

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

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

$$x \in R^D$$

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

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

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

Naïve Bayes vs HMMs

- Naïve Bayes:

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

"Naïve" Generative Independence Assumption

- Hidden Markov Models:

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

- **HMMs ≈ 1st order variant of Naïve Bayes!**

(just one interpretation…)

Naïve Bayes vs HMMs

- Naïve Bayes:

$$P(x,y) = \overbrace{A_y}^{\text{P(y)}} \overbrace{\prod_{d=1}^{D} O^d_{x^d,y}}^{\text{P(x|y)}}$$

"Naïve" Generative Independence Assumption

- Hidden Markov Models:

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

P(x|y)

- HMMs ≈ 1st order variant of Naïve Bayes!

(just one interpretation…)

Summary: Naïve Bayes

- Joint model of (x,y):
  - "Naïve" independence assumption each x^d

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

"Generative Model"
(can sample new data)

- Use Bayes's Rule for prediction:

$$\underset{y}{\operatorname{argmax}}\, P(y \mid x) = \underset{y}{\operatorname{argmax}}\, P(x \mid y)P(y) = \underset{y}{\operatorname{argmax}}\, P(y)\prod_{d=1}^{D} P(x^d \mid y)$$

- Maximum Likelihood Training:
  - Count Frequencies

Learn Conditional Prob.?

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

$$x \in R^D$$

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

- Weird to train to maximize:

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

- When goal should be to maximize:

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

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

Breaks independence!

Can no longer use count statistics

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

*HMMs suffer same problem

Learn Conditional Prob.?

- Weird to train to maximize:

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

arg

- W

arg

Br

Ca

$$\cancel{P(x \ldots = a \mid y = z) = \frac{\ldots}{\sum_{i=1}^{N} 1_{[y_i = z]}}}$$

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

Θ often used to denote all parameters of model

- Maximize Likelihood Training: $\underset{\Theta}{\operatorname{argmax}} \prod_{i=1}^{N} P(x_i, y_i)$

- Mismatch w/ prediction goal: $\underset{y}{\operatorname{argmax}}\, P(y \mid x)$
  - But hard to maximize P(y|x)

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

Discriminative Models

- Conditional model: $P(y \mid x)$
  - Directly model prediction goal

- Maximum Likelihood: $\operatorname*{argmax}_{\Theta} \prod_{i=1}^{N} P(y_i \mid x_i)$

- Matches prediction goal: $\operatorname*{argmax}_{y} P(y \mid x)$

- **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)

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

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

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

Prediction: **Training:**

$$\underset{y}{\operatorname{argmax}}\, P(y \mid x) \quad \leftarrow \textbf{Match!} \rightarrow \quad \underset{\Theta}{\operatorname{argmax}} \prod_{i=1}^{N} P(y_i \mid x_i)$$

Naïve Bayes vs Logistic Regression

- Naïve Bayes:
  - Strong ind. assumptions
  - Super easy to train…
  - …but mismatch with prediction

$$P(x,y) = \underbrace{A_y}_{P(y)} \prod_{d=1}^{D} \underbrace{O^d_{x^d,y}}_{P(x|y)}$$

- Logistic Regression:
  - "Log Linear" assumption
    - Often more flexible than Naïve Bayes
  - Harder to train (gradient desc.)…
  - …but matches prediction

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

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

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*D parameters for P(x|y) (i.e, O)
- LR has K*D parameters for w
- **Same number of parameters!**

Naïve Bayes

$$P(x,y) = \underbrace{A_y}_{P(y)} \prod_{d=1}^{D} \underbrace{O^d_{x^d,y}}_{P(x|y)}$$

Logistic Regression

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

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

$$y \in \{1,2,\ldots,K\}$$

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

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

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

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

$$P(y \mid x) = \frac{\exp\left\{ F(y, x) \right\}}{Z(x)}$$

$$\log P(y \mid x) = F(y, x) - \log\left(Z(x) \right)$$

y^0 = special start state

- x = “Fish Sleep”
- y = (N,V)

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

$u_{N,V}$

	$u_{N,*}$	$u_{V,*}$
$u_{*,N}$	-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

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

Z(x) = Sum (

y	exp(F(y,x))
(N,N)	exp(1+2-2+1) = exp(2)
(N,V)	exp(1+2+2+0) = exp(4)
(V,N)	exp(-1+1+2+1) = exp(3)
(V,V)	exp(-1+1-2+0) = exp(-1)

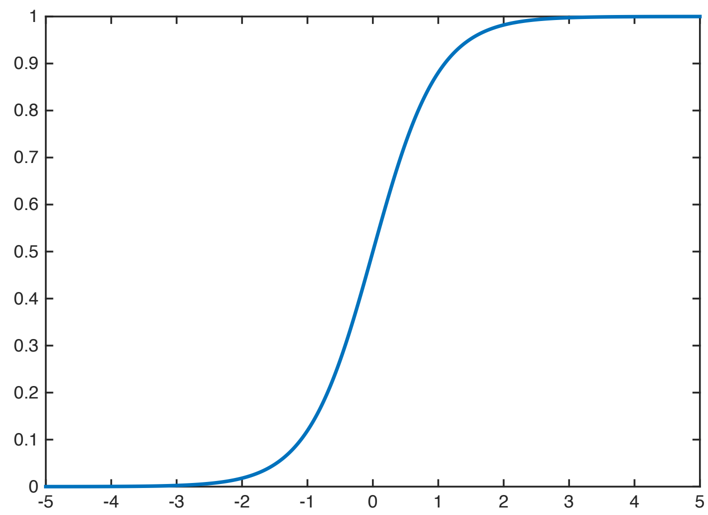
)

- x = “Fish Sleep”
- y = (N,V)

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

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

*hold other parameters fixed



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

Basic Conditional Random Field

- Directly models P(y|x)
  - Discriminative
  - Log linear assumption
  - Same #parameters as HMM
  - 1st Order Sequential LR
- **How to Predict?**
- **How to Train?**
- **Extensions?**

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

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

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

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

Predict via Viterbi

$$\operatorname*{argmax}_y P(y \mid x) = \operatorname*{argmax}_y \log P(y \mid x) = \operatorname*{argmax}_y F(y,x)$$

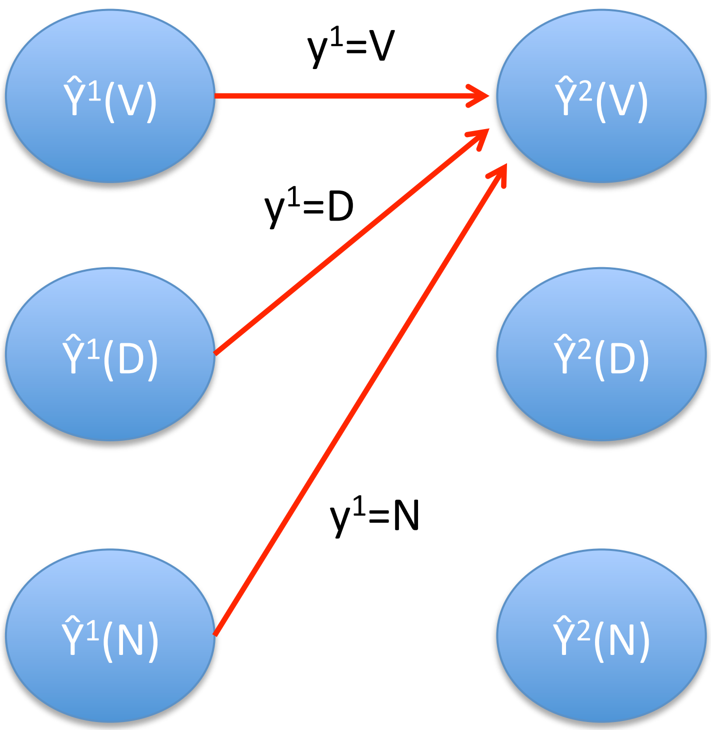
$$= \operatorname*{argmax}_y \sum_{j=1}^{M} \left(u_{y^j, y^{j-1}} + w_{y^j, x^j} \right)$$

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

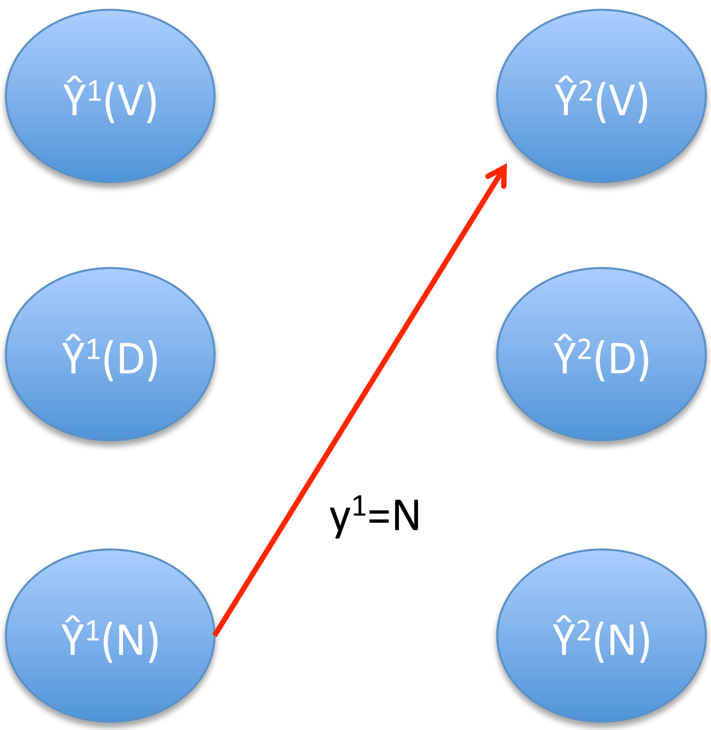
Store each
$\hat{Y}^1(T)$ & $F(\hat{Y}^1(T),x^1)$



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

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

Store each
Ŷ¹(T) & F(Ŷ¹(T),x¹)



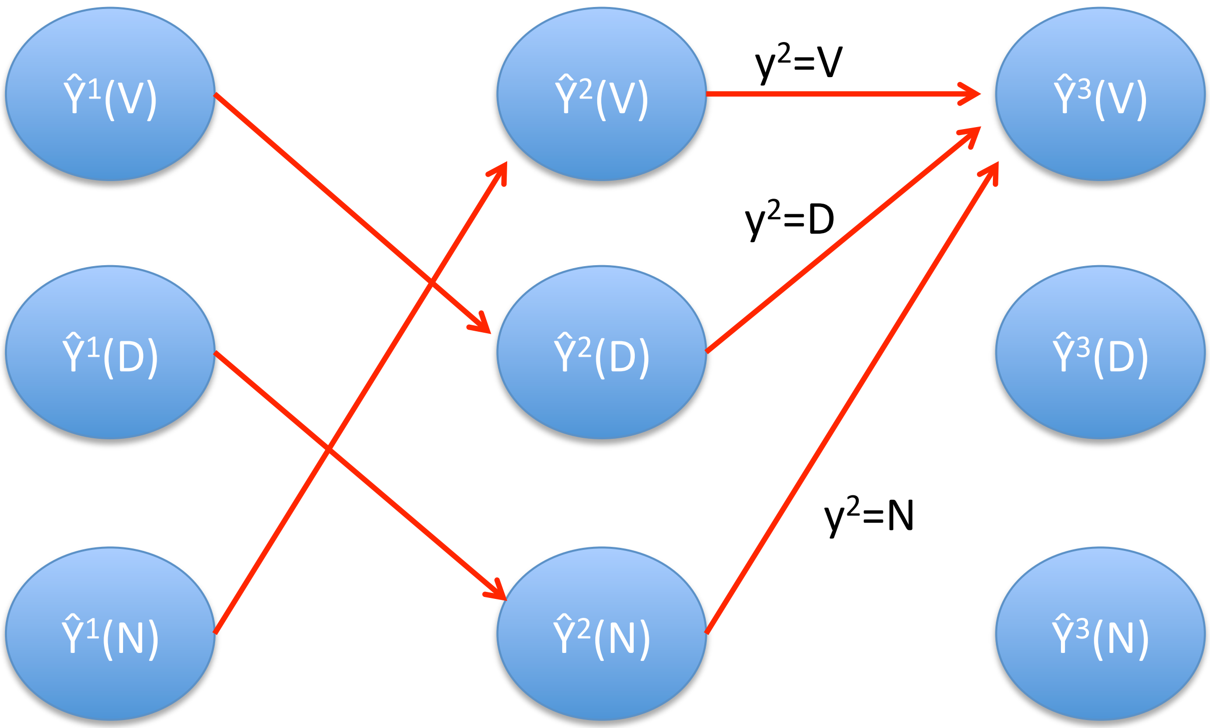
Ŷ¹(T) is just T

Ex: Ŷ²(V) = (N, V)

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

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

Store each
$\hat{Y}^2(Z)$ & $F(\hat{Y}^2(Z),x^{1:2})$



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

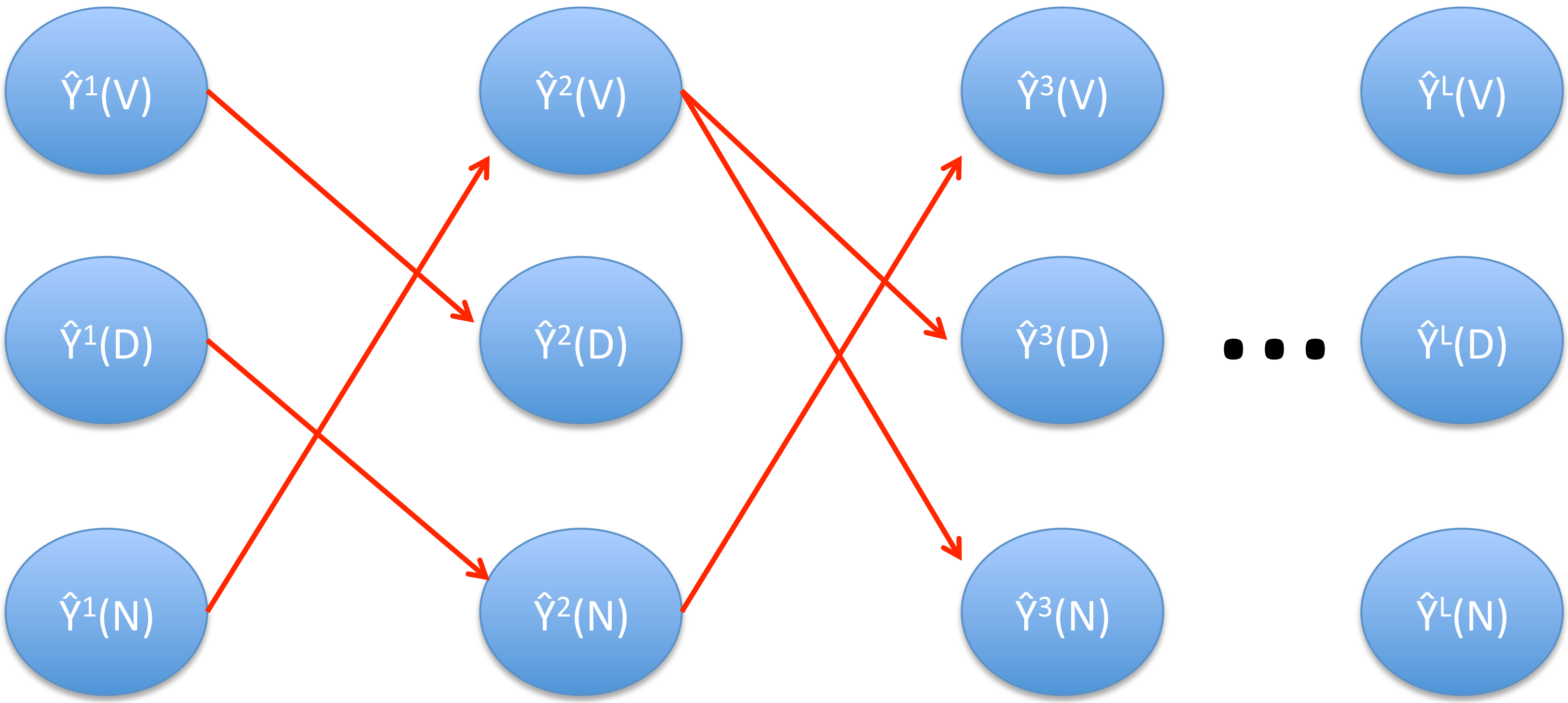
Ex: $\hat{Y}^2(V) = (N, V)$

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

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

Store each
$\hat{Y}^2(T)$ & $F(\hat{Y}^2(T), x^{1:2})$

Store each
$\hat{Y}^3(T)$ & $F(\hat{Y}^3(T), x^{1:3})$



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

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

- 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’!

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

- Notation:

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

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

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

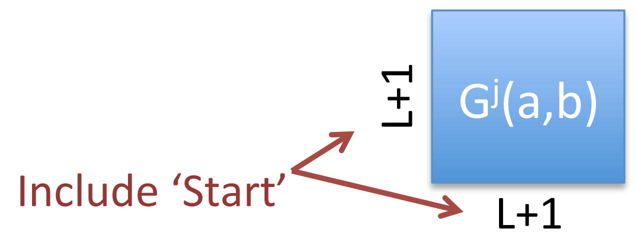
http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Matrix Semiring

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

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

Matrix Version of G^j



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

$$G^{1:2} = G^2 \; G^1$$

$$G^{i:j}(a,b) \equiv \quad G^{i:j} = G^j \; G^{j-1} \cdots G^{i+1} \; G^i$$

http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Path Counting Interpretation

G^1

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

$G^{1:2} = G^2 \; G^1$

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

http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Computing Partition Function

- Consider Length-1 (M=1)

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

Sum column 'Start' of G^1!

- M=2

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

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

- General M

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

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

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

http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Train via Gradient Descent

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

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

Θ often used to denote all parameters of model

Harder to differentiate!

- First term is easy:
  - Recall:

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

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

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

http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Differentiating Log Partition

Lots of Chain Rule & Algebra!

$$\partial_{u_{ab}} \log(Z(x)) = \frac{1}{Z(x)} \partial_{u_{ab}} Z(x) = \frac{1}{Z(x)} \partial_{u_{ab}} \sum_{y'} \exp\{F(y',x)\}$$

$$= \frac{1}{Z(x)} \sum_{y'} \partial_{u_{ab}} \exp\{F(y',x)\}$$

$$= \frac{1}{Z(x)} \sum_{y'} \exp\{F(y',x)\} \partial_{u_{ab}} F(y',x) = \sum_{y'} \frac{\exp\{F(y',x)\}}{Z(x)} \partial_{u_{ab}} F(y',x)$$

Definition of P(y'|x)

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

$$= \sum_{j=1}^{M} \sum_{y'} P(y' | x) 1_{\left[(y'^j, y'^{j-1})=(a,b)\right]} = \sum_{j=1}^{M} P(y^j = a, y^{j-1} = b | x)$$

Forward-Backward!

Marginalize over all y'

http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Optimality Condition

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

- Consider one parameter:

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

- Optimality condition:

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

- **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

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

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

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

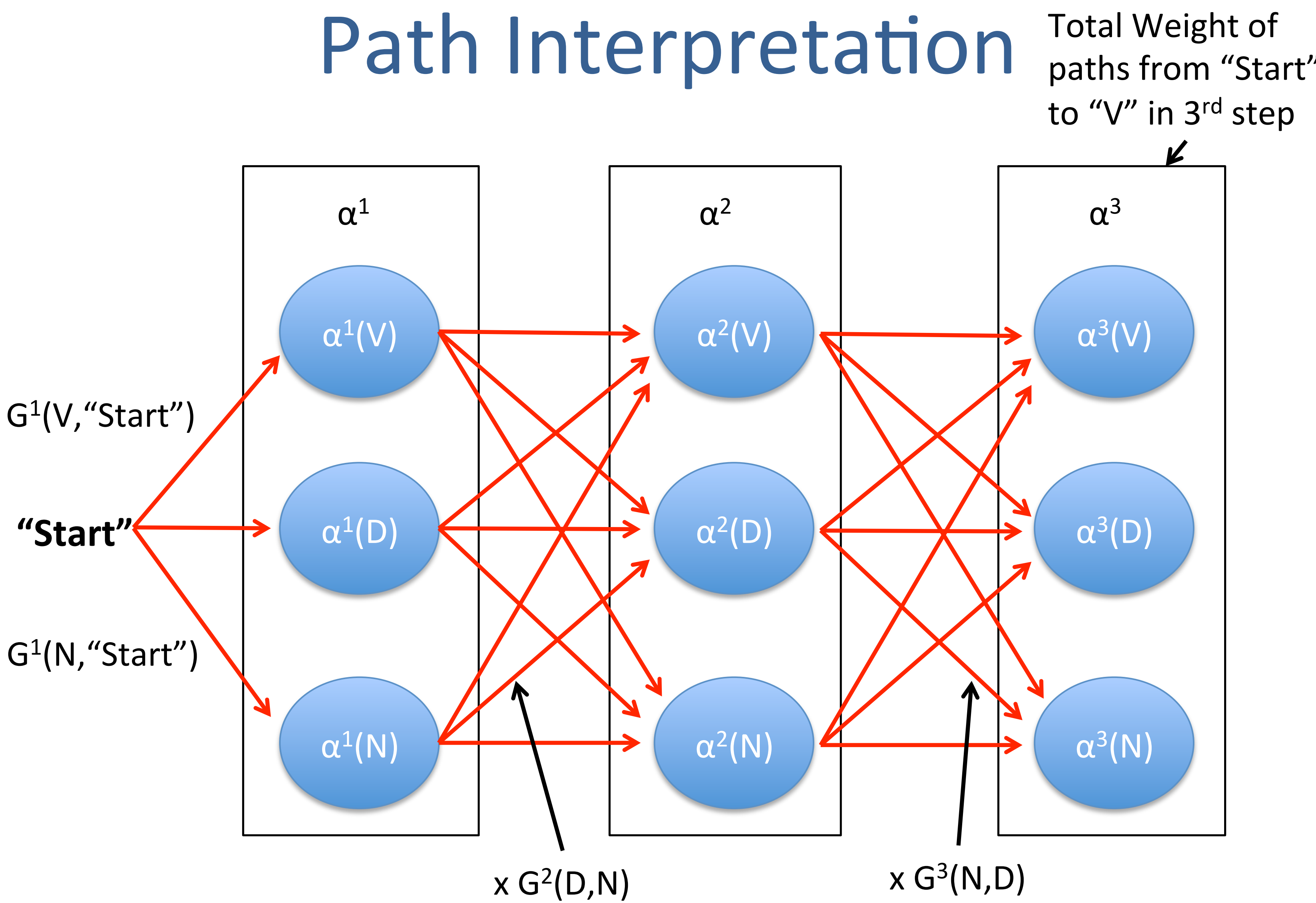
$$\beta^j(b) = \sum_a \beta^{j+1}(a) G^j(a,b)$$

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

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

http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Path Interpretation



β just does it backwards

Matrix Formulation

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

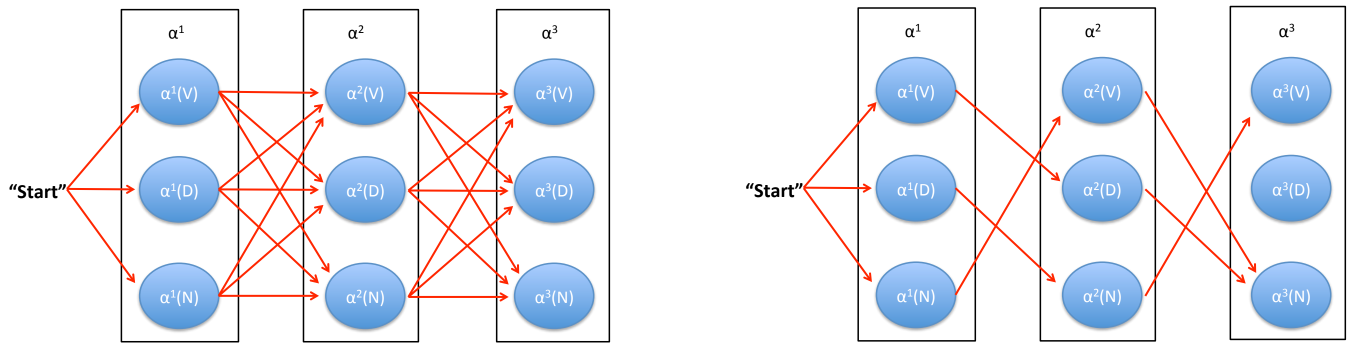
$$\alpha^2 = G^2 \alpha^1$$

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

Path Interpretation:
Forward-Backward vs Viterbi

Forward

Viterbi



- Forward (and Backward) sums over all paths
  - Computes expectation of reaching each state
  - E.g., total (un-normalized) probability of y^3=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=Verb

Summary: Training CRFs

- Similar optimality condition as HMMs:
  - Match frequency counts of model components!

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

  - 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

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

New:

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

Old:

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

Reduction:

$$\phi_j(a,b \mid x) = \begin{array}{cl} \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{array}\right] & \begin{array}{l} \\ \\ \\ \leftarrow u_{a,b} \\ \\ \\ \\ \leftarrow w_{a,x^j} \\ \\ \\ \\ \end{array} \end{array}$$

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

θ is "flattened" weight vector

Can extend $\phi_j(a,b|x)$

More General CRFs

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

1st order Sequence CRFs:

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

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

Example

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

Basic formulation only had first part

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

Various attributes of x

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

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

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

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

  - 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