Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 9:

Conditional Random Fields

Announcements

- Homework 5 released
  - Skeleton code available on Moodle
  - Due in 2 weeks (2/16)
- Kaggle competition closes 2/9
  - SHORT report due 2/11 via Moodle
  - Submit as a group
- Nothing due week of 2/23

Today

- Recap of Sequence Prediction

- **Conditional Random Fields**
  - Sequential version of logistic regression
    - Analogous to how HMMs generalize Naïve Bayes
  - Discriminative sequence prediction
    - Learns to optimize P(y|x) for sequences

Recap: Sequence Prediction

- Input: $x = (x^1,\ldots,x^M)$
- Predict: $y = (y^1,\ldots,y^M)$
  - Each y^i one of L labels.

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

L=6

- x = "Fish Sleep"
- y = (N, V)

- x = "The Dog Ate My Homework"
- y = (D, N, V, D, N)

- x = "The Fox Jumped Over The Fence"
- y = (D, N, V, P, D, N)

Recap: General Multiclass

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

L=6

- x = "Fish sleep"
- y = (N, V)
- Multiclass prediction:
  - All possible length-M sequences as different class
  - (D, D), (D, N), (D, V), (D, Adj), (D, Adv), (D, Pr) (N, D), (N, N), (N, V), (N, Adj), (N, Adv), ...
- **L^M classes!**
  - **Length 2: $6^2 = 36$!**

Recap: General Multiclass

- x = “Fish sleep”
- y = (N, V)

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

L=6

Can Model Everything!
(In Theory)

Exponential Explosion in #Classes!
(Not Tractable)

Recap: Independent Multiclass

x="I fish often"

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

L=6

- Treat each word independently (assumption)
  - Independent multiclass prediction per word
  - Predict for x="I" independently
  - Predict for x="fish" independently
  - Predict for x="often" independently
  - Concatenate predictions.

Assume pronouns are nouns for simplicity.

Recap: Independent Multiclass

x="I fish often"

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

L=6

#Classes = #POS Tags
(6 in our example)

Solvable using standard multiclass prediction.
But ignores context!

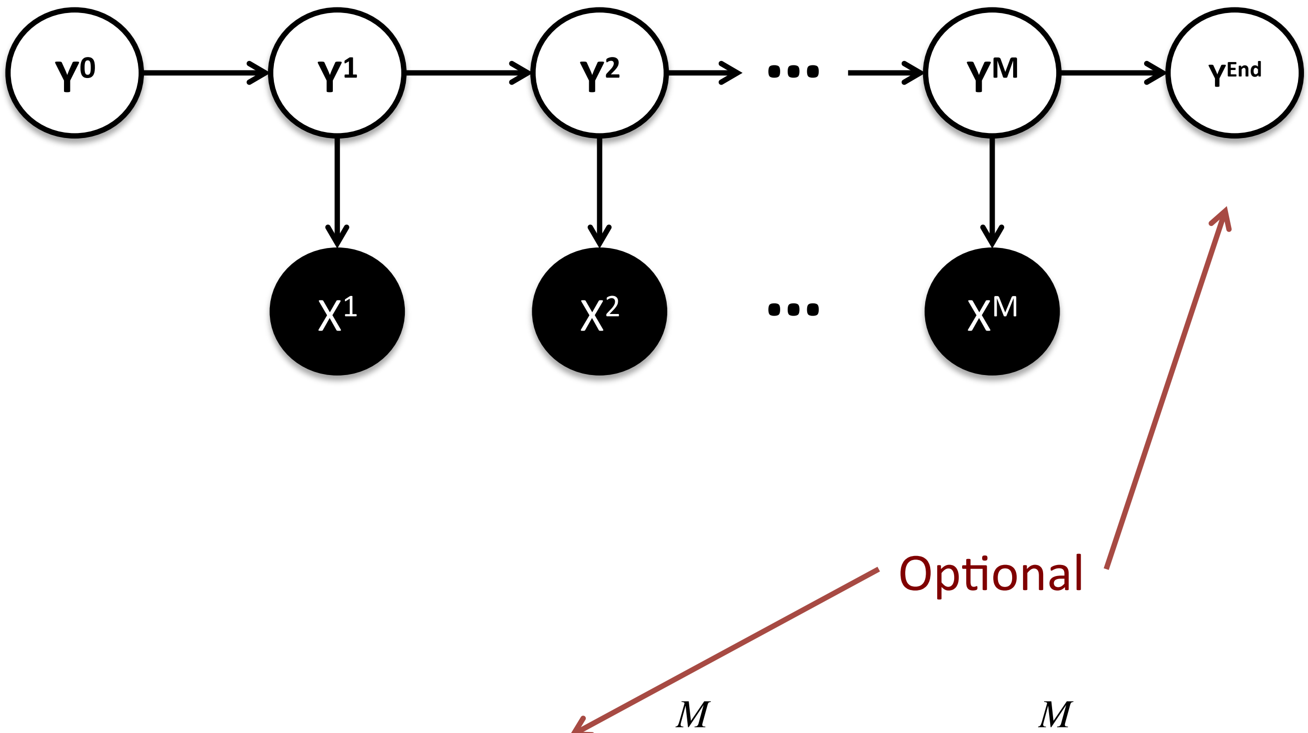
Assume pronouns are nouns for simplicity.

Recap: 1st Order HMM

- $x = (x^1, x^2, x^4, x^4, \ldots, x^M)$ (sequence of words)
- $y = (y^1, y^2, y^3, y^4, \ldots, y^M)$ (sequence of POS tags)

- $P(x^i | y^i)$ Probability of state y^i generating x^i
- $P(y^{i+1} | y^i)$ Probability of state y^i transitioning to y^{i+1}
- $P(y^1 | y^0)$ y^0 is defined to be the Start state
- $P(End | y^M)$ Prior probability of y^M being the final state
  - Not always used

Graphical Model Representation



$$P(x,y) = P(End \mid y^M)\prod_{i=1}^{M} P(y^i \mid y^{i-1})\prod_{i=1}^{M} P(x^i \mid y^i)$$

HMM Matrix Formulation

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

$$= A_{END,y^M}\prod_{j=1}^{M} A_{y^j y^{j-1}} O_{y^j,x^j}$$

Transition Probabilities

Emission Probabilities
(Observation Probabilities)

Recap: 1st-Order Sequence Models

- General multiclass:
  - Unique scoring function per entire seq.
  - Very intractable
- Independent multiclass
  - Scoring function per token, apply to each token in seq.
  - Ignores context, low accuracy
- **First-order models**
  - **Scoring function per pair of tokens.**
  - **"Sweet spot" between fully general & ind. multiclass**

Recap: Naïve Bayes & HMMs

- Naïve Bayes:

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

"Naïve" Generative Independence Assumption

- Hidden Markov Models:

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

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

(just one interpretation…)

Recap: 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)

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

Learn Conditional Prob.?

- Weird to train to maximize:

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

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

- When goal should be to maximize:

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

$$p(x) = \sum_{y} P(x, y) = \sum_{y} P(y) P(x \mid y)$$

Breaks independence!

Can no longer use count statistics

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

Both HMMs & Naïve Bayes suffer this 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.

-

B

Ca

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

Both HMMs & Naïve Bayes suffer this problem!

Generative vs Discriminative

- Generative Models:
  - Joint Distribution: P(x,y) ← **Mismatch!**
  - Uses Bayes's Rule to predict: argmax_y P(y|x)
  - Can generate new samples (x,y)

Hidden Markov Models
Naïve Bayes

- Discriminative Models:
  - Conditional Distribution: P(y|x) ← **Same thing!**
  - Can directly to predict: argmax_y P(y|x)

Conditional Random Fields
Logistic Regression

- Both trained via Maximum Likelihood

First Try

(for classifying a single y)

- 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

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

Log Linear Models!

(Logistic Regression)

$$P(y \mid x) = \frac{\exp\left\{w_y^T x - b_y\right\}}{\sum_k \exp\left\{w_k^T x - b_k\right\}} \qquad \begin{array}{l} x \in R^D \\ y \in \{1, 2, \ldots, L\} \end{array}$$

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

Prediction: **Training:**

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

Naïve Bayes vs Logistic Regression

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

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

P(y) → A_y; P(x|y) → $O^{d}_{x^d,y}$

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

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

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

Naïve Bayes vs Logistic Regression

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

Naïve Bayes

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

P(y) → A_y; P(x|y) → $O^d_{x^d,y}$

Logistic Regression

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

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

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

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

Conditional Random Fields

Sequential Version of Logistic Regression

"Log-Linear" 1st Order Sequential Model

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

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

aka "Partition Function"

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

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, excluding end state

- x = “Fish Sleep”
- y = (N,V)

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

$A_{N,V}$

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

$w_{V,Fish}$

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

$$P(N,V \mid \text{"Fish Sleep"}) = \frac{1}{Z(x)} \exp\left\{ A_{N,Start} + O_{N,Fish} + A_{V,N} + O_{V,Sleep} \right\} = \frac{1}{Z(x)} \exp\{4\} \approx 0.66$$

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(-2)

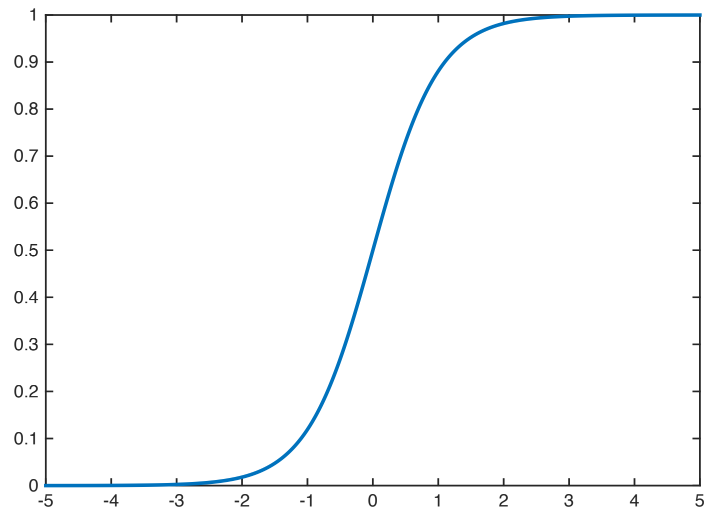
)

- x = “Fish Sleep”
- y = (N,V)

$$P(N,V\,|\,"Fish\ Sleep") = \frac{1}{Z(x)}\exp\left\{F(x,y)\right\}$$

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

*hold other parameters fixed



$$F(y,x)$$

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)

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

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

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

Predict via Viterbi

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

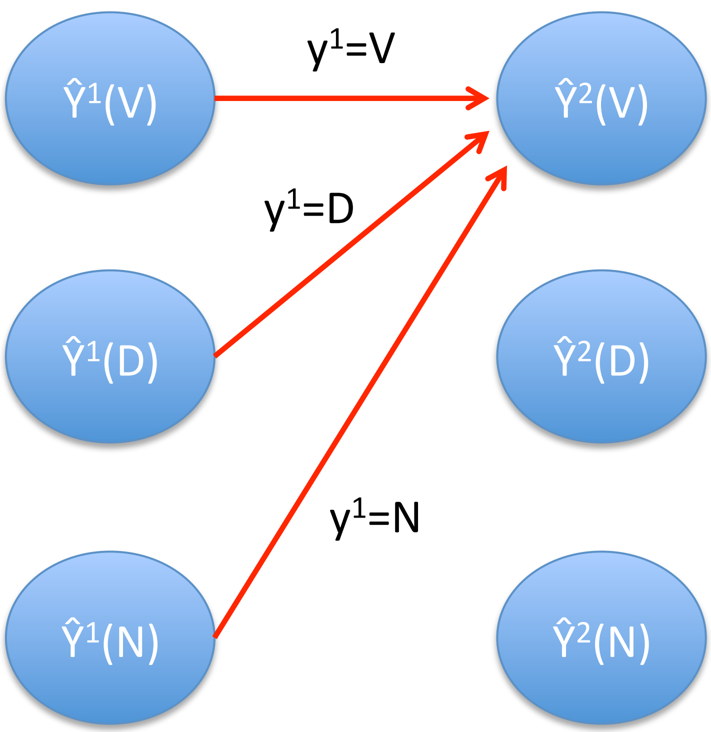
$$= \underset{y}{\operatorname{argmax}} \sum_{j=1}^{M} \left(A_{y^j, y^{j-1}} + O_{y^j, x^j} \right)$$

Scoring transitions   Scoring observations

Maintain length-k prefix solutions	$\hat{Y}^k(T) = \left(\underset{y^{1:k-1}}{\operatorname{argmax}}\, F(y^{1:k-1} \oplus T, x^{1:k}) \right) \oplus T$
Recursively solve for length-(k+1) solutions	$\hat{Y}^{k+1}(T) = \left(\underset{y^{1:k} \in \left\{ \hat{Y}^k(T) \right\}_T}{\operatorname{argmax}}\ F(y^{1:k} \oplus T, x^{1:k+1}) \right) \oplus T$ $= \left(\underset{y^{1:k} \in \left\{ \hat{Y}^k(T) \right\}_T}{\operatorname{argmax}}\ F(y^{1:k}, x^{1:k}) + A_{T,y^k} + O_{T,x^{k+1}} \right) \oplus T$
Predict via best length-M solution	$\underset{y}{\operatorname{argmax}}\, F(y,x) = \underset{y \in \left\{ \hat{Y}^M(T) \right\}_T}{\operatorname{argmax}}\ F(y,x)$

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

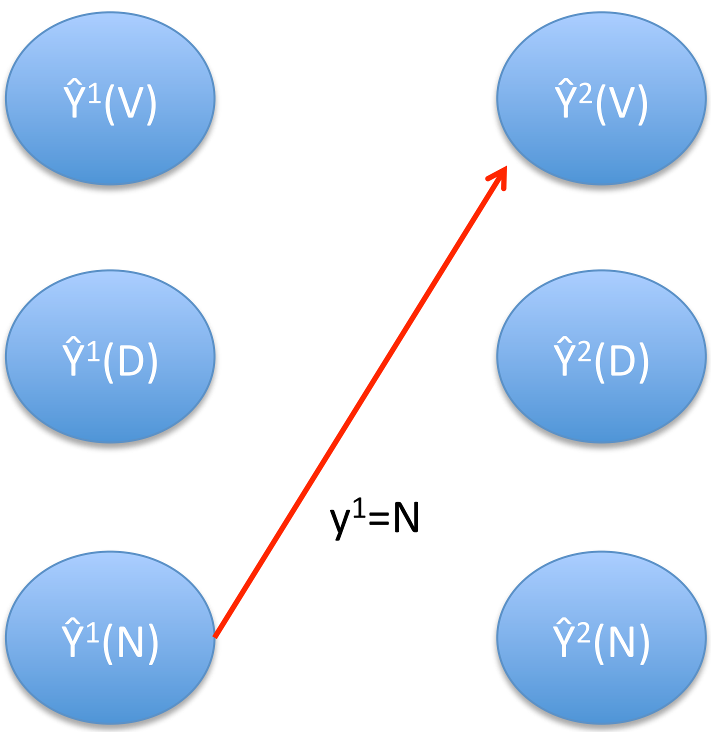
Store each
$\hat{Y}^1(T)$ & $F(\hat{Y}^1(T),x)$



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

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

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



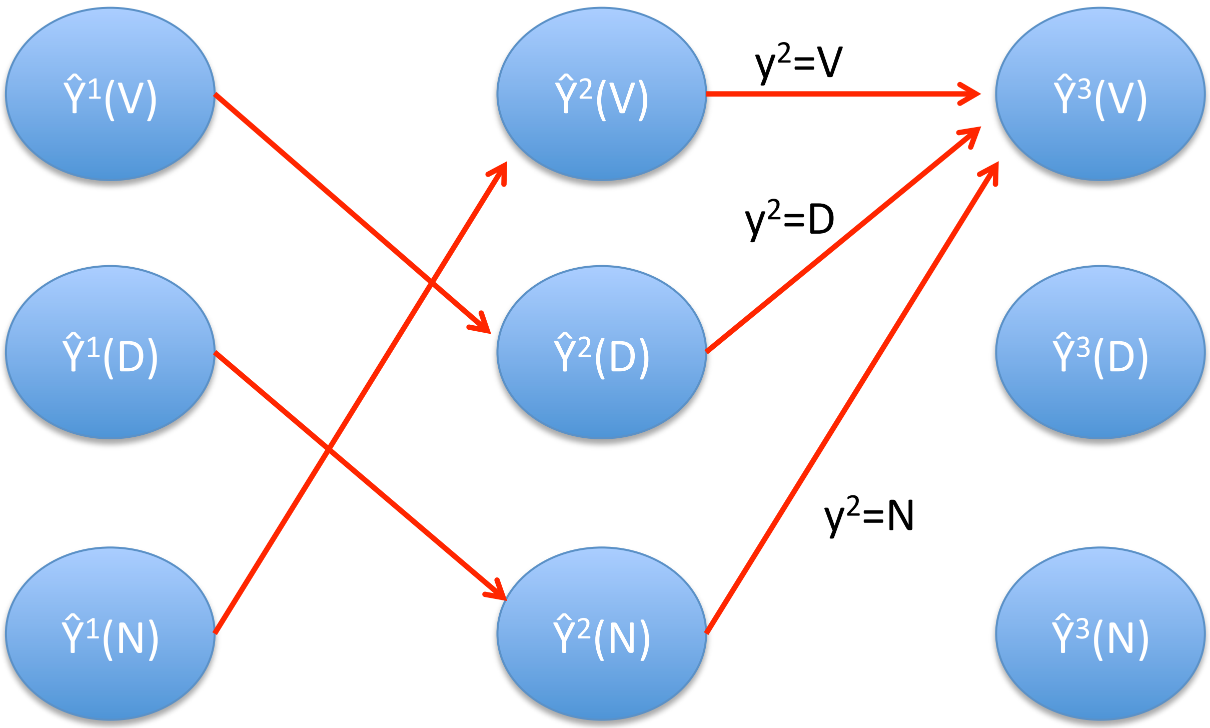
$\hat{Y}^1(T)$ is just T

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

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

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

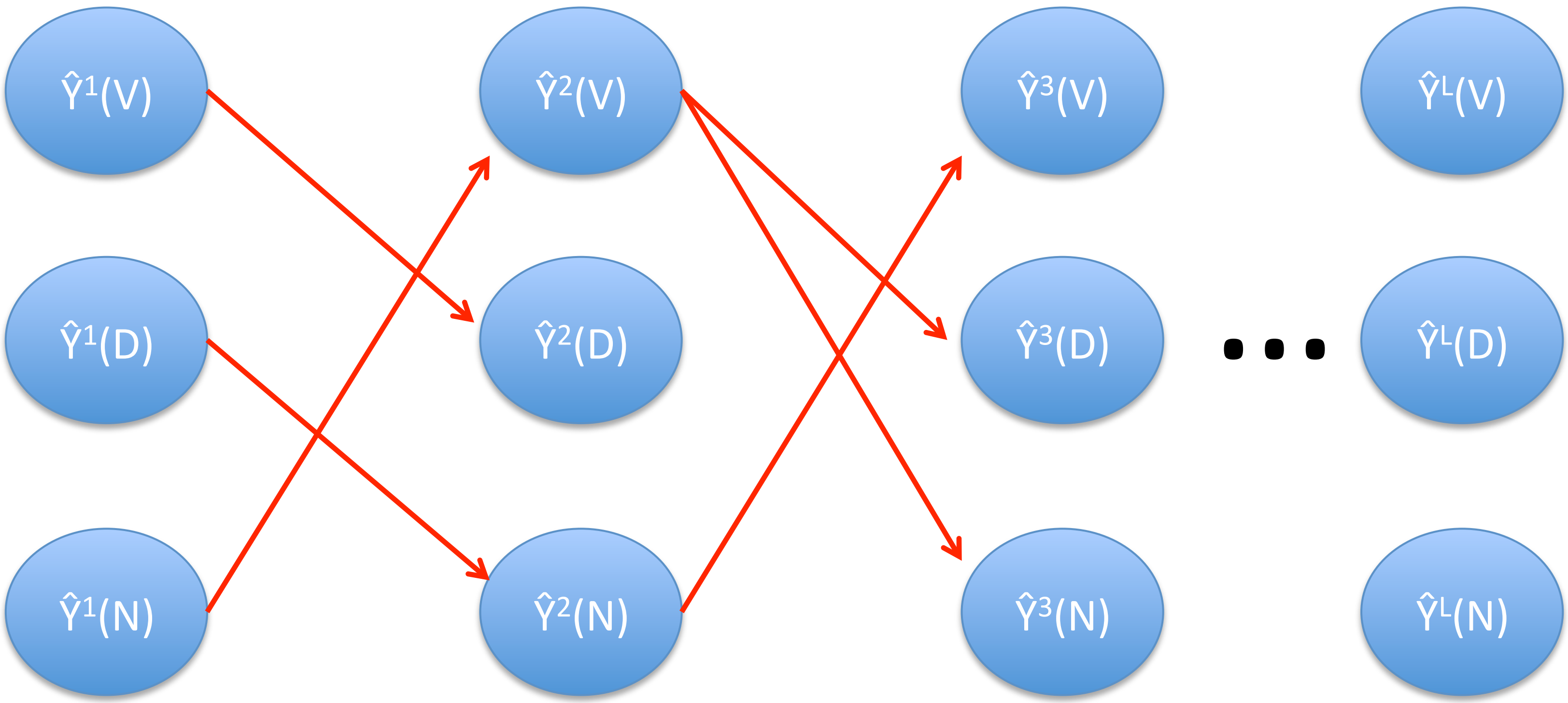
Ex: $\hat{Y}^2(V) = (N, V)$

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

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'!

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

- Notation:

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

Suppressing dependency on x for simpler notation

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

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

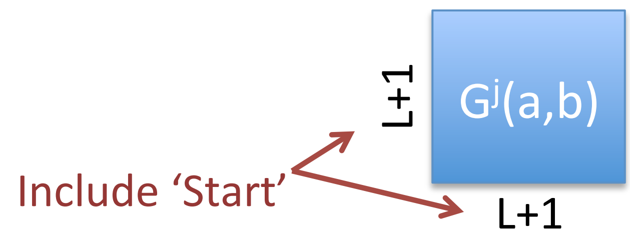
http://www.inference.phy.cam.ac.uk/hmw26/papers/crf_intro.pdf

Matrix Semiring

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

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

Matrix Version of G^j



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

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

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

Path Counting Interpretation

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

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

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

Computing Partition Function

- Consider Length-1 (M=1)

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

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 $G^{1:M}$

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

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

Computing Partition Function

- Consider Length-1 (M=1) $Z(x) = \sum_b G^1(b, Start)$

  tart' of G^1!

- M=2

  $(b, Start)$

  art' of $G^{1:2}$!

Numerical Instability Issues!

(See Course Notes)

- General M

  art' of $G^{1:M}$!

  – Do M (L+1)x(L+1) matrix computations to compute $G^{1:M}$

  – Z(x) = sum column 'Start' of $G^{1:M}$

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

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)

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

Θ often used to denote all parameters of model

Harder to differentiate!

- First term is easy:
  - Recall:

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

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

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

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

Differentiating Log Partition

Lots of Chain Rule & Algebra!

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

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

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

Definition of P(y'|x) ↘

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

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

Forward-Backward!

↑ Marginalize over all y'

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

Optimality Condition

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

- Consider one parameter:

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

- Optimality condition:

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

- **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

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

$$\alpha^j(a) = \sum_{a'} \alpha^{j-1}(a') G^j(a, a')$$

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

$$\beta^j(b) = \sum_{b'} \beta^{j+1}(b') G^{j+1}(b', b)$$

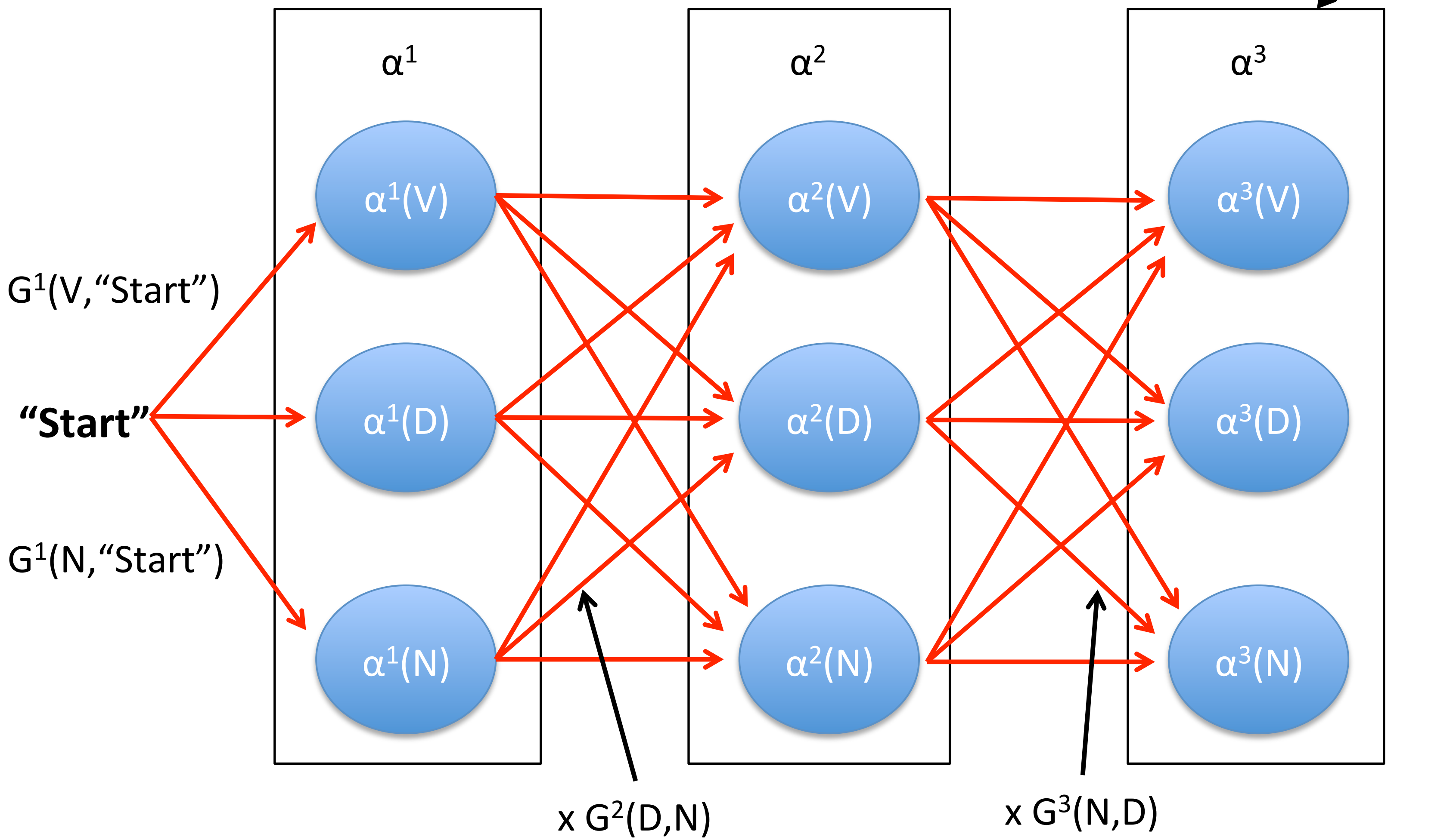
$$P(y^j = b, y^{j-1} = a \mid x) = \frac{\alpha^{j-1}(a) G^j(b, a) \beta^j(b)}{Z(x)}$$

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

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

Path Interpretation

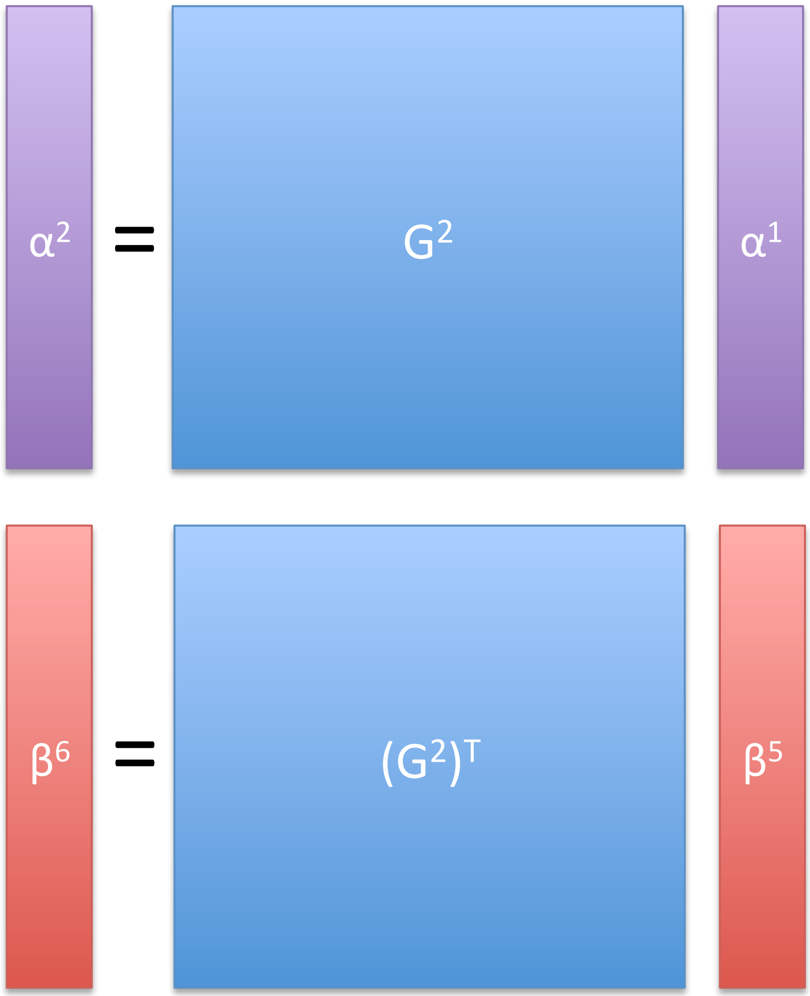
Total Weight of paths from "Start" to "V" in 3rd step



β just does it backwards

Matrix Formulation

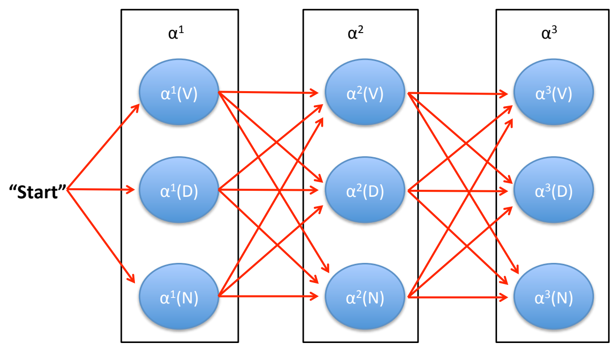
- Use Matrices!
- Fast to compute!
- Easy to implement!



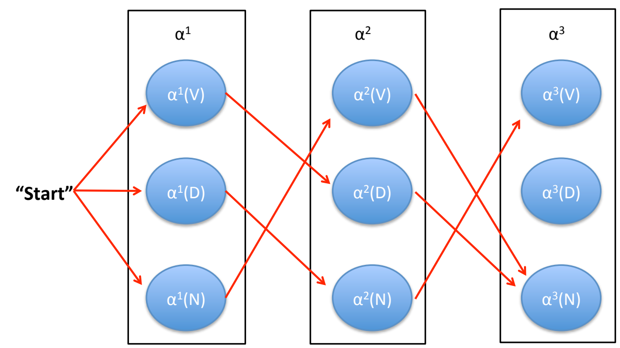
Path Interpretation:

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!

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

  - 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

Summary: CRFs

- Log-Linear Sequential Model:

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

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

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

- Same #parameters as HMMs
  - Focused on learning P(y|x)
  - Prediction via Viterbi
  - Gradient Descent via Forward-Backward

Next Lecture

- More General Formulation of CRFs
  - More concise notation
    - Matches logistic regression notation
    - Matches course notes (later this week)
  - Easier to reason about for implementation

- General Structured Prediction