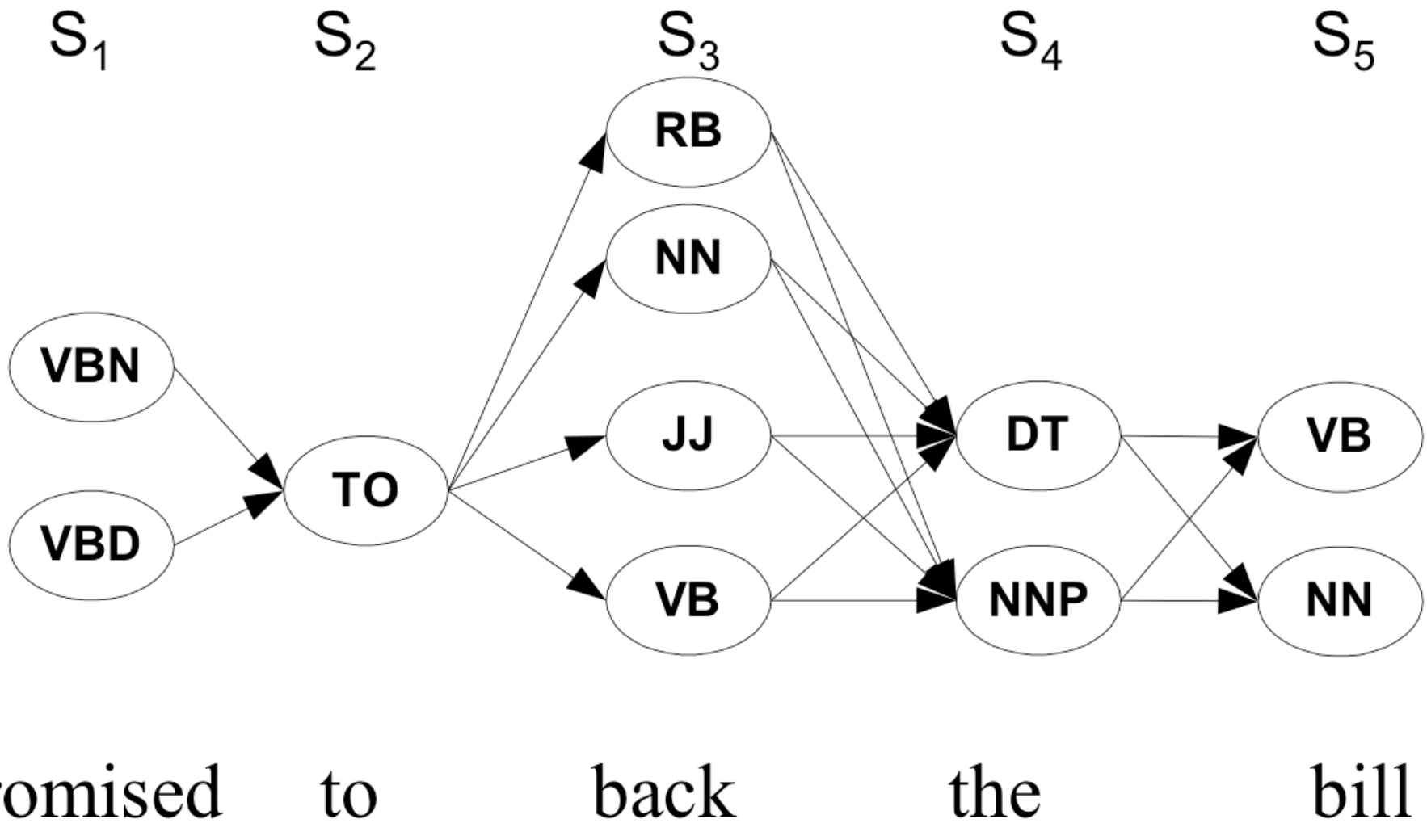
Viterbi Decoding for HMM, Parameter Learning

Niloy Ganguly

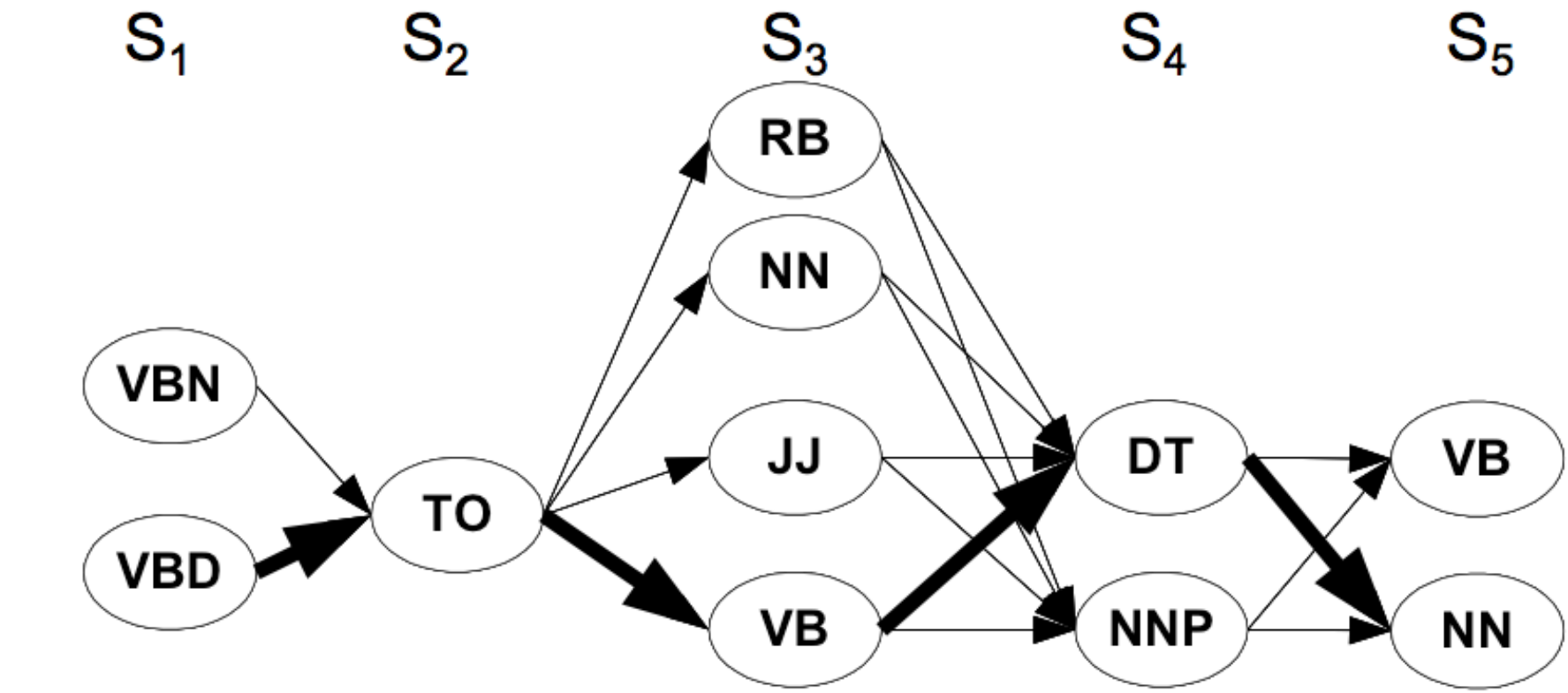
CSE, IIT Kharagpur

Week 4, Lecture 1

Walking through the states: best path



Walking through the states: best path



Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$
- $\psi_j(s+1) = argmax_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$
- $\psi_j(s+1) = argmax_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

Best final state is $argmax_{1 \leq i \leq N} \delta_i(|S|)$, we can backtrack from there

Practice Question

- Suppose you want to use a HMM tagger to tag the phrase, “the light book”, where we have the following probabilities:
- P(the|Det) = 0.3, P(the|Noun) = 0.1, P(light|Noun) = 0.003, P(light|Adj) = 0.002, P(light|Verb) = 0.06, P(book|Noun) = 0.003, P(book|Verb) = 0.01
- P(Verb|Det) = 0.00001, P(Noun|Det) = 0.5, P(Adj|Det) = 0.3, P(Noun|Noun) =0.2, P(Adj|Noun) = 0.002, P(Noun|Adj) = 0.2, P(Noun|Verb) = 0.3, P(Verb|Noun) = 0.3, P(Verb|Adj) = 0.001, P(Verb|Verb) = 0.1
- Work out in details the steps of the Viterbi algorithm. You can use a Table to show the steps. Assume all other conditional probabilities, not mentioned to be zero. Also, assume that all tags have the same probabilities to appear in the beginning of a sentence.

Learning the Parameters

Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

Learning the Parameters

Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

Methods for these scenarios

- For the first scenario, parameters can be directly estimated using maximum likelihood estimate from the labeled dataset
- For the second scenario, *Baum-Welch Algorithm* is used to estimate the parameters of the hidden markov model.

Baum Welch Algorithm

Niloy Ganguly

CSE, IIT Kharagpur

Week 4, Lecture 2

Baum Welch Algorithm

Uses the well-known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden markov model

Baum Welch Algorithm

Uses the well-known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden markov model

Parameters of HMM

Let X_t be the random variable denoting hidden state at time t, and Y_t be the observation variable at time T. HMM parameters are given by $\theta = (A, B, \pi)$ where

- $A = \{a_{ij}\} = P(X_t = j | X_{t-1} = i)$ is the state transition matrix
- $\pi = \{\pi_i\} = P(X_1 = i)$ is the initial state distribution
- $B = \{b_j(y_t)\} = P(Y_t = y_t | X_t = j)$ is the emission matrix

Baum Welch Algorithm

Uses the well-known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden markov model

Parameters of HMM

Let X_t be the random variable denoting hidden state at time t, and Y_t be the observation variable at time T. HMM parameters are given by $\theta = (A, B, \pi)$ where

- $A = \{a_{ij}\} = P(X_t = j | X_{t-1} = i)$ is the state transition matrix
- $\pi = \{\pi_i\} = P(X_1 = i)$ is the initial state distribution
- $B = \{b_j(y_t)\} = P(Y_t = y_t | X_t = j)$ is the emission matrix

Given observation sequences $Y = (Y_1 = y_1, Y_2 = y_2, \ldots, Y_T = y_T)$, the algorithm tries to find the parameters θ that maximise the probability of the observation.

The Algorithm

The basic idea is to start with some random initial conditions on the parameters θ, estimate best values of state paths X_t using these, then re-estimate the parameters θ using the just-computed values of X_t, iteratively.

The Algorithm

The basic idea is to start with some random initial conditions on the parameters θ, estimate best values of state paths X_t using these, then re-estimate the parameters θ using the just-computed values of X_t, iteratively.

Intuition

- Choose some initial values for $\theta = (A, B, \pi)$.
- *Repeat the following step until convergence:*
- Determine probable (state) paths $\ldots X_{t-1} = i, X_t = j \ldots$
- Count the expected number of transitions a_{ij} as well as the expected number of times, various emissions $b_j(y_t)$ are made
- Re-estimate $\theta = (A, B, \pi)$ using a_{ij} and $b_j(y_t)$s.

A forward-backward algorithm is used for finding probable paths.

Forward-Backward Algorithm

Forward Procedure

$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$ be the probability of seeing $y_1, \ldots, y_t$ and being in state i at time t. Found recursively using:

- $\alpha_i(1) = \pi_i b_i(y_1)$

Forward-Backward Algorithm

Forward Procedure

$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$ be the probability of seeing $y_1, \ldots, y_t$ and being in state i at time t. Found recursively using:

- $\alpha_i(1) = \pi_i b_i(y_1)$
- $\alpha_j(t+1) = b_j(y_{t+1}) \sum_{i=1}^{N} \alpha_i(t) a_{ij}$

Forward-Backward Algorithm

Forward Procedure

$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$ be the probability of seeing $y_1, \ldots, y_t$ and being in state i at time t. Found recursively using:

- $\alpha_i(1) = \pi_i b_i(y_1)$
- $\alpha_j(t+1) = b_j(y_{t+1}) \sum_{i=1}^{N} \alpha_i(t) a_{ij}$

Backward Procedure

$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T|X_t = i, \theta)$ be the probability of ending partial sequence $y_{t+1}, \ldots, y_T$ given starting state i at time t. $\beta_i(t)$ is computed recursively as:

- $\beta_i(T) = 1$

Forward-Backward Algorithm

Forward Procedure

$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$ be the probability of seeing $y_1, \ldots, y_t$ and being in state i at time t. Found recursively using:

- $\alpha_i(1) = \pi_i b_i(y_1)$
- $\alpha_j(t+1) = b_j(y_{t+1}) \sum_{i=1}^{N} \alpha_i(t) a_{ij}$

Backward Procedure

$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T|X_t = i, \theta)$ be the probability of ending partial sequence $y_{t+1}, \ldots, y_T$ given starting state i at time t. $\beta_i(t)$ is computed recursively as:

- $\beta_i(T) = 1$
- $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(y_{t+1})$

Finding probabilities of paths

We compute the following variables:

- Probability of being in state i at time t given the observation Y and parameters θ

$$\gamma_i(t) = P(X_t = i|Y,\theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)}$$

- Probability of being in state i and j at time t and $t+1$ respectively given the observation Y and parameters θ

$$\zeta_{ij}(t) = P(X_t = i, X_{t+1} = j|Y,\theta) = \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}$$

Updating the parameters

- $\pi_i = \gamma_i(1)$, expected number of times state i was seen at time 1
- $a_{ij} = \frac{\sum_{t=1}^{T} \zeta_{ij}(t)}{\sum_{t=1}^{T} \gamma_i(t)}$, expected number of transitions from state i to state j, compared to the total number of transitions away from state i
- $b_i(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$ with $1_{y_t=v_k}$ being an indicator function, is the expected number of times the output observations are v_k while being in state i compared to the expected total number of times in state i.

Maximum Entropy Models

Niloy Ganguly

CSE, IIT Kharagpur

Week 4, Lecture 3

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

- Use morphological cues (capitalization, suffix) to assign a more calculated guess

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

- Use morphological cues (capitalization, suffix) to assign a more calculated guess

Limited Context

- "is clearly **marked**" → verb, past participle
- "he clearly **marked**" → verb, past tense

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

- Use morphological cues (capitalization, suffix) to assign a more calculated guess

Limited Context

- "is clearly **marked**" → verb, past participle
- "he clearly **marked**" → verb, past tense

Possible solution:

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

- Use morphological cues (capitalization, suffix) to assign a more calculated guess

Limited Context

- "is clearly **marked**" → verb, past participle
- "he clearly **marked**" → verb, past tense

Possible solution: Use higher order model, combine various n-gram models to avoid sparseness problem

Maximum Entropy Modeling: Discriminative Model

Maximum Entropy Modeling: Discriminative Model

- We may identify a heterogeneous set of features which contribute in some way to the choice of POS tag of the current word.

Maximum Entropy Modeling: Discriminative Model

- We may identify a heterogeneous set of features which contribute in some way to the choice of POS tag of the current word.
  - Whether it is the first word in the article

Maximum Entropy Modeling: Discriminative Model

- We may identify a heterogeneous set of features which contribute in some way to the choice of POS tag of the current word.
  - Whether it is the first word in the article
  - Whether the next word is *to*

Maximum Entropy Modeling: Discriminative Model

- We may identify a heterogeneous set of features which contribute in some way to the choice of POS tag of the current word.
  - Whether it is the first word in the article
  - Whether the next word is *to*
  - Whether one of the last 5 words is a preposition, etc.
- MaxEnt combines these features in a probabilistic model

Maximum Entropy: The Model

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

Maximum Entropy: The Model

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

where

- $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

Maximum Entropy: The Model

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

where

- $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

- λ_i is a weight given to a feature f_i

Maximum Entropy: The Model

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

where

- $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

- λ_i is a weight given to a feature f_i
- x denotes an observed datum and y denotes a class

What is the form of the features?

Features in Maximum Entropy Models

- Features encode elements of the context x for predicting tag y
- Context x is taken around the word w, for which a tag y is to be predicted

Features in Maximum Entropy Models

- Features encode elements of the context x for predicting tag y
- Context x is taken around the word w, for which a tag y is to be predicted
- Features are binary values functions, e.g.,

$$f(x,y) = \left\{ \begin{array}{ll} 1 & \text{if } isCapitalized(w) \& y = NNP \\ 0 & otherwise \end{array} \right\}$$

Example Features

Example: Named Entities

- LOCATION (in Arcadia)
- LOCATION (in Québec)
- DRUG (taking Zantac)
- PERSON (saw Sue)

Example Features

Example: Named Entities

- LOCATION (in Arcadia)
- LOCATION (in Québec)
- DRUG (taking Zantac)
- PERSON (saw Sue)

Example Features

- $f_1(x,y) = [y = LOCATION \wedge w_{-1} = \text{“}in\text{”} \wedge isCapitalized(w)]$
- $f_2(x,y) = [y = LOCATION \wedge hasAccentedLatinChar(w)]$
- $f_3(x,y) = [y = DRUG \wedge ends(w, \text{“}c\text{”})]$

Tagging with Maximum Entropy Model

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Tagging with Maximum Entropy Model

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Tag sequence candidate $\{t_1, \ldots, t_n\}$ has conditional probability:

$$P(t_1, \ldots, t_n | w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i | x_i)$$

Tagging with Maximum Entropy Model

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Tag sequence candidate $\{t_1, \ldots, t_n\}$ has conditional probability:

$$P(t_1, \ldots, t_n | w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i | x_i)$$

- The context x_i also includes previously assigned tags for a fixed history.
- Beam search is used to find the most probable sequence

Practice Question

Consider the maximum entropy model for POS tagging, where you want to estimate $P(tag|word)$. In a hypothetical setting, assume that *tag* can take the values *D*, *N* and *V* (short forms for Determiner, Noun and Verb). The variable *word* could be any member of a set V of possible words, where V contains the words *a*, *man*, *sleeps*, as well as additional words. The distribution should give the following probabilities

- $P(D|a) = 0.9$
- $P(N|man) = 0.9$
- $P(V|sleeps) = 0.9$
- $P(D|word) = 0.6$ for any word other than *a*, *man* or *sleeps*
- $P(N|word) = 0.3$ for any word other than *a*, *man* or *sleeps*
- $P(V|word) = 0.1$ for any word other than *a*, *man* or *sleeps*

It is assumed that all other probabilities, not defined above could take any values such that $\sum_{tag} P(tag|word) = 1$ is satisfied for any word in V.

- Define the features of your maximum entropy model that can model this distribution. Mark your features as f_1, f_2 and so on. Each feature should have the same format as explained in the class. [**Hint:** 6 Features should make the analysis easier]
- For each feature f_i, assume a weight λ_i. Now, write expression for the following probabilities in terms of your model parameters
  - $P(D|cat)$
  - $P(N|laughs)$
  - $P(D|man)$
- What value do the parameters in your model take to give the distribution as described above. (i.e. $P(D|a) = 0.9$ and so on. *You may leave the final answer in terms of equations*)

Solution to Practice Question

Let $\mathcal{V}$ be the vocabulary. Let $V' = \mathcal{V} - \{a, man, sleeps\}$. Consider features as:

- f_1: word = 'a' and tag = 'D'
- f_2: word = 'man' and tag = 'N'
- f_3: word = 'sleeps' and tag = 'V'
- f_4: word in V' and tag = 'D'
- f_5: word in V' and tag = 'N'
- f_6: word in V' and tag = 'V'

The MaxEnt model gives us $p_\lambda(y|x) = \frac{e^{\sum_i \lambda_i f_i(x,y)}}{Z}$. Thus we have

- $P(D|cat) = \frac{e^{\lambda_4}}{Z}$
- $P(N|laughs) = \frac{e^{\lambda_5}}{Z}$
- $P(D|man) = \frac{e^0}{Z} = \frac{1}{Z}$

Since $\sum_{tag} P(tag|word) = 1$ is satisfied for any word in $\mathcal{V}$,
$P(D|a) + P(N|a) + P(V|a) = 1 \implies \frac{e^{\lambda_1}+2}{Z} = 1 \implies Z = e^{\lambda_1} + 2$. Therefore
$P(D|a) = 0.9 \implies \frac{e^{\lambda_1}}{e^{\lambda_1}+2} = 0.9 \implies e^{\lambda_1} = 0.9(e^{\lambda_1}+2) \implies 0.1(e^{\lambda_1}) = 1.8 \implies \lambda_1 = \log 18$
Similarly you can find out $\lambda_2, \lambda_3 \ldots \lambda_6$

Maximum Entropy Models

Niloy Ganguly

CSE, IIT Kharagpur

Week 4, Lecture 4

Features for POS Tagging (Ratnaparakhi, 1996)

The specific word and tag context available to a feature is

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

Features for POS Tagging (Ratnaparakhi, 1996)

The specific word and tag context available to a feature is

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

Example: $f_j(h_i, t_i) = 1$ if $suffix(w_i) = \text{“}ing''\&t_i = VBG$

Example Features

Condition	Features	
w_i is not rare	$w_i = X$	& $t_i = T$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$	& $t_i = T$
	X is suffix of w_i, $\|X\| \leq 4$	& $t_i = T$
	w_i contains number	& $t_i = T$
	w_i contains uppercase character	& $t_i = T$
	w_i contains hyphen	& $t_i = T$
$\forall\ w_i$	$t_{i-1} = X$	& $t_i = T$
	$t_{i-2}t_{i-1} = XY$	& $t_i = T$
	$w_{i-1} = X$	& $t_i = T$
	$w_{i-2} = X$	& $t_i = T$
	$w_{i+1} = X$	& $t_i = T$
	$w_{i+2} = X$	& $t_i = T$

Example Features

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

Example Features

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

$w_i = \texttt{about}$ & $t_i = \texttt{IN}$
$w_{i-1} = \texttt{stories}$ & $t_i = \texttt{IN}$
$w_{i-2} = \texttt{the}$ & $t_i = \texttt{IN}$
$w_{i+1} = \texttt{well-heeled}$ & $t_i = \texttt{IN}$
$w_{i+2} = \texttt{communities}$ & $t_i = \texttt{IN}$
$t_{i-1} = \texttt{NNS}$ & $t_i = \texttt{IN}$
$t_{i-2}t_{i-1} = \texttt{DT NNS}$ & $t_i = \texttt{IN}$

$w_{i-1} = \texttt{about}$ & $t_i = \texttt{JJ}$
$w_{i-2} = \texttt{stories}$ & $t_i = \texttt{JJ}$
$w_{i+1} = \texttt{communities}$ & $t_i = \texttt{JJ}$
$w_{i+2} = \texttt{and}$ & $t_i = \texttt{JJ}$
$t_{i-1} = \texttt{IN}$ & $t_i = \texttt{JJ}$
$t_{i-2}t_{i-1} = \texttt{NNS IN}$ & $t_i = \texttt{JJ}$
$\text{prefix}(w_i)$=w & $t_i = \texttt{JJ}$
$\text{prefix}(w_i)$=we & $t_i = \texttt{JJ}$
$\text{prefix}(w_i)$=wel & $t_i = \texttt{JJ}$
$\text{prefix}(w_i)$=well & $t_i = \texttt{JJ}$
$\text{suffix}(w_i)$=d & $t_i = \texttt{JJ}$
$\text{suffix}(w_i)$=ed & $t_i = \texttt{JJ}$
$\text{suffix}(w_i)$=led & $t_i = \texttt{JJ}$
$\text{suffix}(w_i)$=eled & $t_i = \texttt{JJ}$
w_i contains hyphen & $t_i = \texttt{JJ}$

Conditional Probability

Given a sentence $\{w_1, \ldots, w_n\}$, a tag sequence candidate $\{t_1, \ldots, t_n\}$ has conditional probability:

$$P(t_1, \ldots, t_n | w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i | x_i)$$

Search Algorithm

Conditional Probability

Given a sentence $\{w_1, \ldots, w_n\}$, a tag sequence candidate $\{t_1, \ldots, t_n\}$ has conditional probability:

$$P(t_1, \ldots, t_n | w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i | x_i)$$

A *Tag Dictionary* is used, which, for each known word, lists the tags that it has appeared with in the training set.

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability tag sequence up to and including word w_i.

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability tag sequence up to and including word w_i.

Search description

- Generate tags for w_1, find top N, set $s_{1j}, 1 \leq j \leq N$, accordingly.

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability tag sequence up to and including word w_i.

Search description

- Generate tags for w_1, find top N, set $s_{1j}, 1 \leq j \leq N$, accordingly.
- Initialize $i = 2$
  - Initialize $j = 1$
  - Generate tags for w_i, given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
  - $j = j + 1$, repeat if $j \leq N$

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability tag sequence up to and including word w_i.

Search description

- Generate tags for w_1, find top N, set $s_{1j}, 1 \leq j \leq N$, accordingly.
- Initialize $i = 2$
  - Initialize $j = 1$
  - Generate tags for w_i, given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
  - $j = j + 1$, repeat if $j \leq N$
- Find N highest probability sequences generated by above loop, set s_{ij} accordingly

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability tag sequence up to and including word w_i.

Search description

- Generate tags for w_1, find top N, set $s_{1j}, 1 \leq j \leq N$, accordingly.
- Initialize $i = 2$
  - Initialize $j = 1$
  - Generate tags for w_i, given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
  - $j = j + 1$, repeat if $j \leq N$
- Find N highest probability sequences generated by above loop, set s_{ij} accordingly
- $i = i + 1$, repeat if $i \leq n$

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability tag sequence up to and including word w_i.

Search description

- Generate tags for w_1, find top N, set $s_{1j}, 1 \leq j \leq N$, accordingly.
- Initialize $i = 2$
  - Initialize $j = 1$
  - Generate tags for w_i, given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
  - $j = j + 1$, repeat if $j \leq N$
- Find N highest probability sequences generated by above loop, set s_{ij} accordingly
- $i = i + 1$, repeat if $i \leq n$
- Return highest probability sequence s_{n1}

Practice Question

Suppose you want to use a MaxEnt tagger to tag the sentence, "the light book". We know that the top 2 POS tags for the words *the*, *light* and *book* are $\{Det, Noun\}$, $\{Verb, Adj\}$ and $\{Verb, Noun\}$, respectively. Assume that the MaxEnt model uses the following history h_i (context) for a word w_i:

$$h_i = \{w_i, w_{i-1}, w_{i+1}, t_{i-1}\}$$

where w_{i-1} and w_{i+1} correspond to the previous and next words and t_{i-1} corresponds to the tag of the previous word. Accordingly, the following features are being used by the MaxEnt model:

- f_1: $t_{i-1} = Det$ and $t_i = Adj$
- f_2: $t_{i-1} = Noun$ and $t_i = Verb$
- f_3: $t_{i-1} = Adj$ and $t_i = Noun$
- f_4: $w_{i-1} = the$ and $t_i = Adj$
- f_5: $w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- f_6: $w_{i-1} = light$ and $t_i = Noun$
- f_7: $w_{i+1} = light$ and $t_i = Det$
- f_8: $w_{i-1} = NULL$ and $t_i = Noun$

Assume that each feature has a uniform weight of 1.0.
Use Beam search algorithm with a beam-size of 2 to identify the highest probability tag sequence for the sentence.

Solution to Practice Question

The solution using beam search with beam size 2 is as follows:

$i = 1$	$i = 2$	$i = 3$
the	light	book
Det - $\frac{e^1}{2e^1} = 0.5$ ✓	Det, Verb - $0.5 \times \frac{e^0}{e^0+e^3} = 0.0237$	Det, Adj, Verb - $0.4763 \times \frac{e^0}{e^0+e^2} = 0.0568$
Noun - $\frac{e^1}{2e^1} = 0.5$ ✓	Det, Adj - $0.5 \times \frac{e^3}{e^0+e^3} = 0.4763$ ✓	Det, Adj, Noun - $0.4763 \times \frac{e^2}{e^0+e^2} = 0.4195$ ✓
	Noun, Verb - $0.5 \times \frac{e^1}{e^1+e^2} = 0.1345$	Noun, Adj, Verb - $0.3655 \times \frac{e^0}{e^0+e^2} = 0.0436$
	Noun, Adj - $0.5 \times \frac{e^2}{e^1+e^2} = 0.3655$ ✓	Noun, Adj, Noun - $0.3655 \times \frac{e^2}{e^0+e^2} = 0.3219$

At each step, the possible tag sequences with corresponding probabilities are shown. The sequences retained in the beam for $i = 1$ and $i = 2$ are indicated by ✓. Since the weights are all 1, the probability of a tag sequence at a step is (prob. of tag sequence in the last step)*softmax(no. of features matched) where the softmax is taken over all the possible tags for the word (in this case, 2).
The highest probability tag sequence is Det, Adj, Noun.

A Good Reference

Berger et al., *A Maximum Entropy Approach to Natural Language Processing*, Computational Linguistics, Vol. 22, No. 1.

Conditional Random Fields

Niloy Ganguly

CSE, IIT Kharagpur

Week 4, Lecture 5

Problem with Maximum Entropy Models

Per-state normalization

All the mass that arrives at a state must be distributed among the possible successor states

Problem with Maximum Entropy Models

Per-state normalization

All the mass that arrives at a state must be distributed among the possible successor states

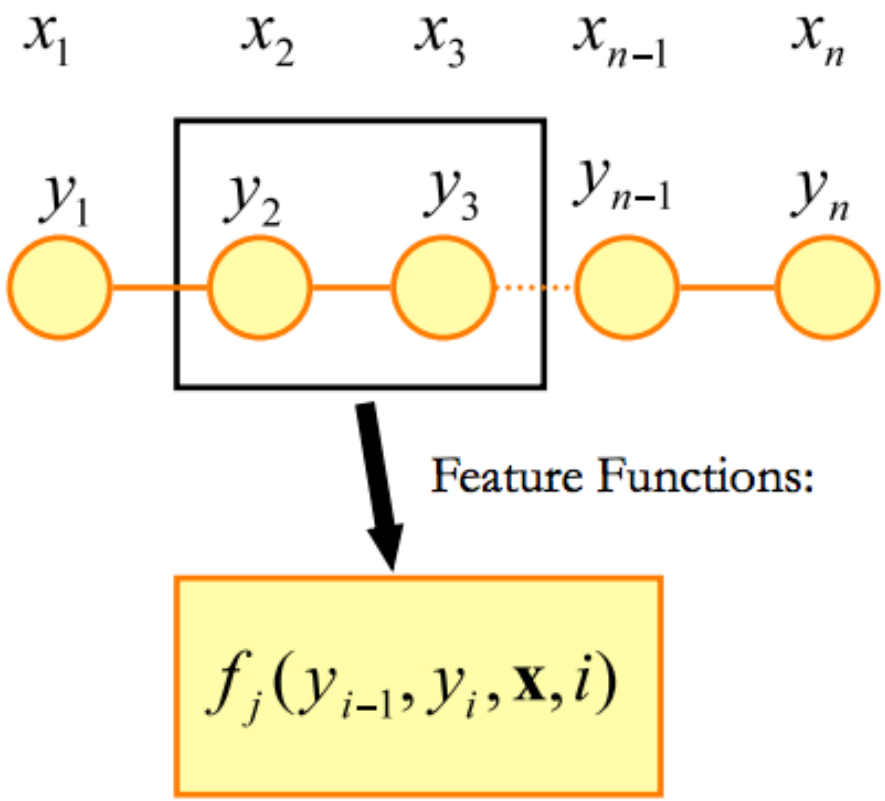
This gives a 'label bias' problem

Let's see the intuition

Conditional Random Fields

- CRFs are conditionally trained, undirected graphical models.
- Let's look at the linear chain structure

Conditional Random Fields: Feature Functions



Feature Functions

Express some characteristic of the empirical distribution that we wish to hold in the model distribution

$f_j(y_{i-1}, y_i, \mathbf{x}, i)$

1 *if* $y_{i-1} = IN$ *and*
$y_i = NNP$ *and*
$x_i = September$

0 *otherwise*

Conditional Random Fields: Distribution

Label sequence modelled as a normalized product of feature functions:

$$P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \exp \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in Y} \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

CRFs

- Have the advantages of MEMM but avoid the label bias problem
- CRFs are globally normalized, whereas MEMMs are locally normalized.
- Widely used and applied. CRFs have been (close to) state-of-the-art in many sequence labeling tasks.