Statistical Methods for NLP

Part I Markov Random Fields
Part II Equations to Implementation

Sameer Maskey

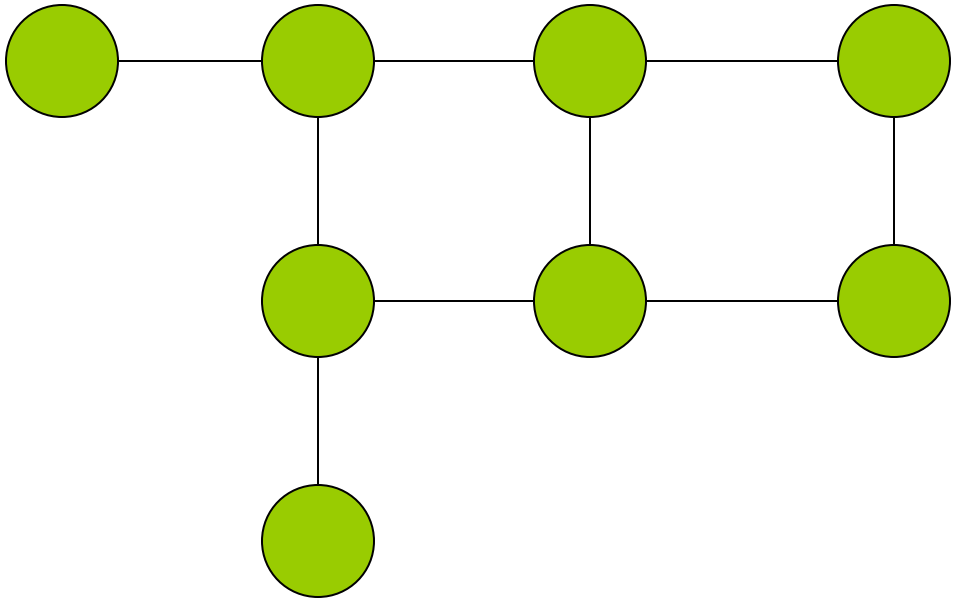
Week 14, April 20, 2010

Announcement

- Next class : Project presentation
- 10 mins total time, strictly timed
  - 2 mins for questions
- Please come to my office hours to download the presentation to my laptop
- Or please come 15 mins early to the class

Markov Random Fields

- Let ‘x’ represent the cost of a shirt in a flea market
- The shirt price in flea market shops may be affected by proximity of each other
- We can take account of such dependency by potential function $\theta_{ij}(x_i, x_j)$



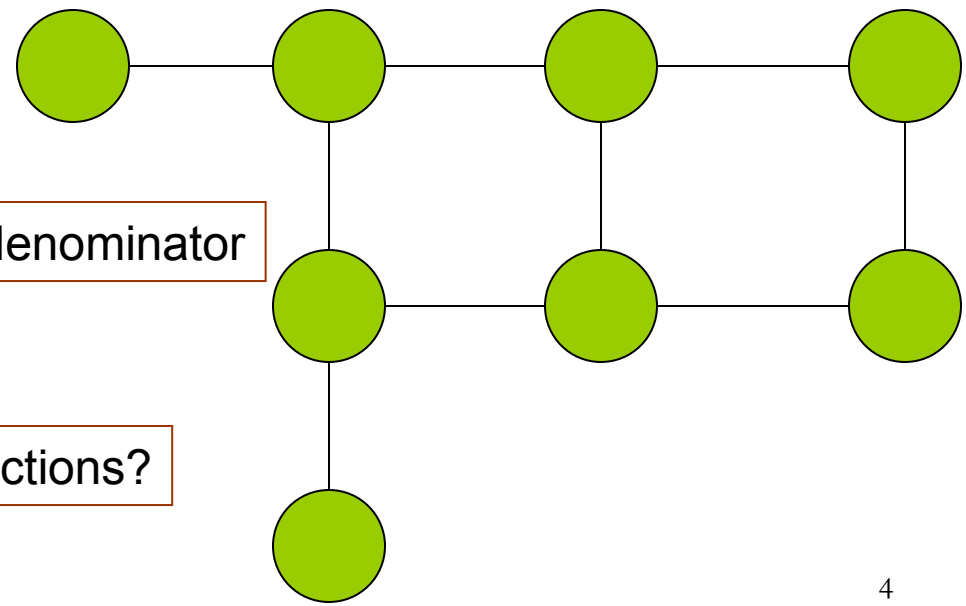
Markov Random Fields

- stronger dependency ~ larger values $\theta_{ij}(x_i, x_j)$
- Joint distribution over variables

Seems familiar?

$$\mathrm{P}(\mathrm{x}_1, ..., x_n) = \frac{exp(\sum_{(i,j)\in E} \theta_{ij}(x_i, x_j))}{\sum_{x_1,...,x_n} exp(\sum_{(i,j)\in E} \theta_{ij}(x_i, x_j))}$$

Global normalization in denominator



Log linear models, feature functions?

Markov Random Fields (cont'd)

- Rewriting in terms of factors

$$\mathrm{P}(\mathrm{x}_1, ..., x_n) = \frac{1}{Z(\theta)} exp(\sum_{(i,j)\in E} \theta_{ij}(x_i, x_j))$$

$$= \frac{1}{Z(\theta)} \prod_{(i,j)\in E} exp(\theta_{ij}(x_i, x_j))$$

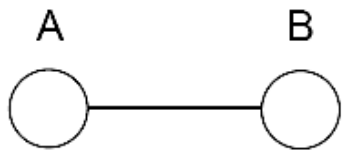
$$= \frac{1}{Z(\theta)} \prod_{(i,j)\in E} \psi_{ij}(x_i, x_j)$$

Potential Functions

Positive functions over groups of variables

Potential Functions

- Potential is just a table (all node combinations represented as entries)



A \ B	0	1
0	1.5	.4
1	.7	1.2

- Marginalizing the potential entails collapsing into one dimension

A	
0	1.9
1	1.9

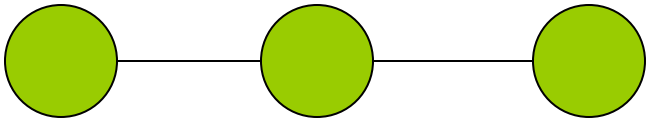
Marginalize over B

0	1
2.2	1.6

Marginalize over A

Factorization

Each of this term is a clique,
in fact maximal clique



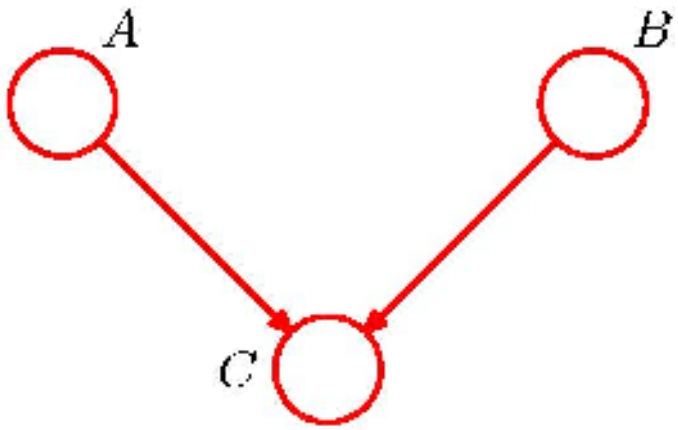
$$P(x_1, x_2, x_3) = \frac{1}{Z(\theta)} \psi(x_1, x_2)\psi(x_2, x_3)\psi(x_2, x_3)$$

- Represent global configuration as product of local potentials
- Hammersley-Clifford theorem tell us how to represent a graph that is consistent with given distribution
- Clique : set of nodes such that there is an edge between every node

Graph Separation

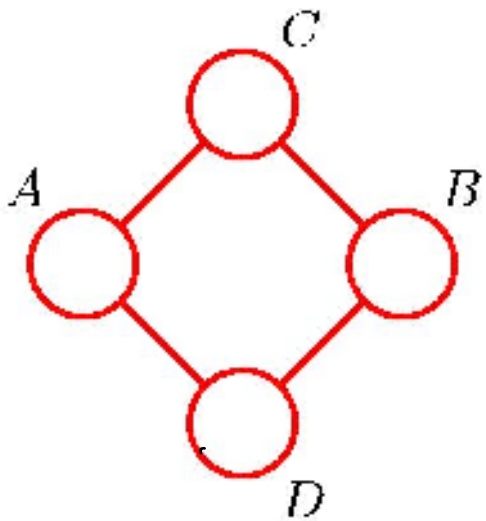
- We liked Bayesian Network because it allowed to represent conditional independence well
- Conditional independence governed by D-separation criterion in Bayes Net
- Simple representation of independence properties in Markov Random Field
  - Check graph reachability

Directed vs. Undirected Graphs



$A \perp\!\!\!\perp B \mid \emptyset$

$A \not\perp\!\!\!\perp B \mid C$



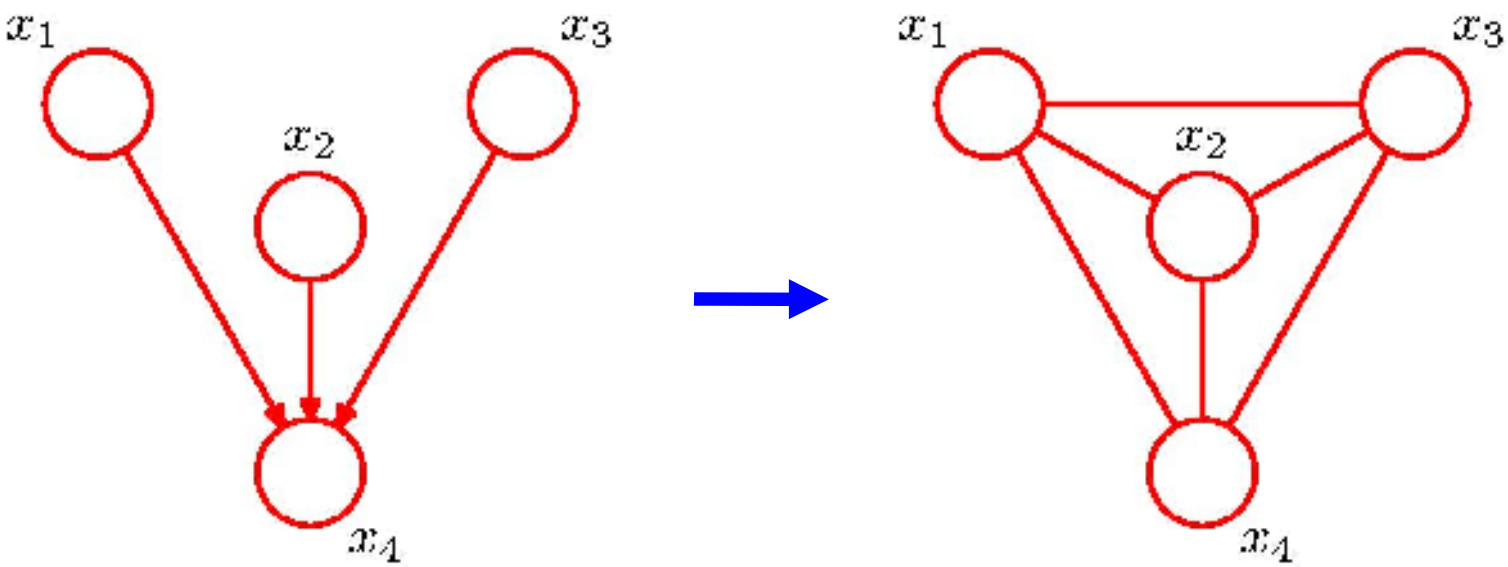
$A \not\perp\!\!\!\perp B \mid \emptyset$

$A \perp\!\!\!\perp B \mid C \cup D$

$C \perp\!\!\!\perp D \mid A \cup B$

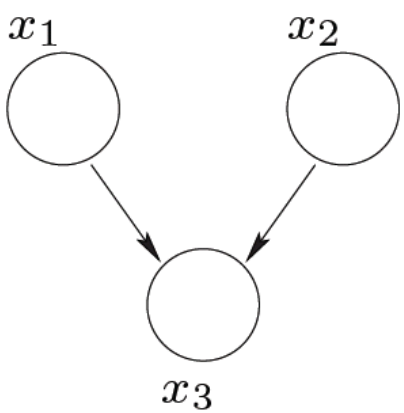
Converting Directed to Undirected Graphs

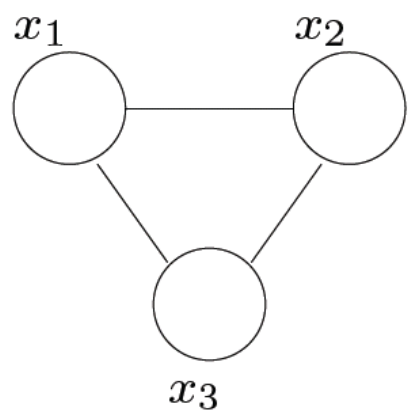
- Additional links



$$
\begin{aligned}
p(\mathbf{x}) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3) \\
&= \frac{1}{Z}\psi_A(x_1,x_2,x_3)\psi_B(x_2,x_3,x_4)\psi_C(x_1,x_2,x_4)
\end{aligned}
$$

Directed vs. Undirected

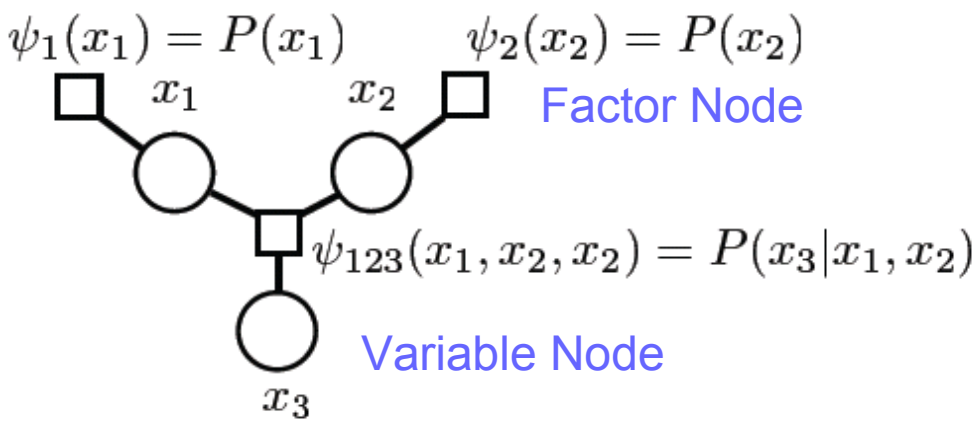




$$P(x_1, x_2, x_3) = P(x_1)P(x_2)P(x_3|x_1, x_2) \quad P(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$$

- Distribution has not changed but the graph representation has

Factor Graphs



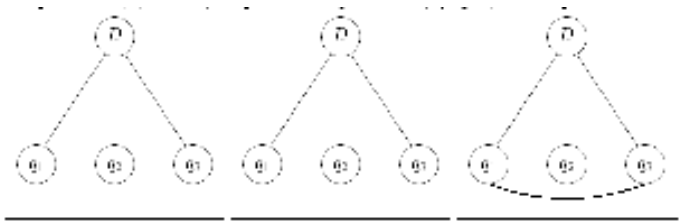
- We saw directed graphical model represent a given distribution
- We also found how to represent the same distribution using undirected models
- Factors graphs another way to represent same distribution
- Variables involved in a factor connected to the factor node
- Number of factors equal number of factor nodes

Graph Representation

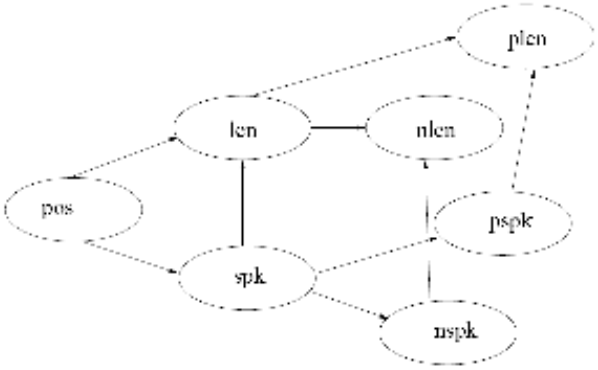
- We saw that we can represent distributions in three different types of graphs
  - Directed Acyclic Graphs
  - Undirected Graphs
  - Factor Graphs
- Depending on a problem one type of graph may be favored against another

Graphical Models in NLP

- Markov Random Field for term dependencies [Metzler W and Croft D, 05]

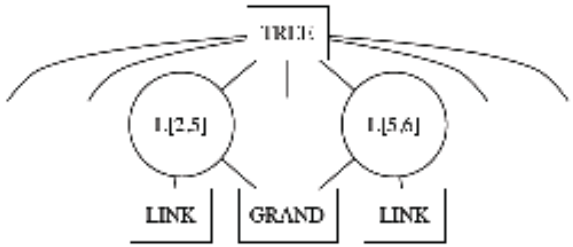


- Conditional Random fields for shallow parsing [Sha F. and Pereira F, 03]



- Bayesian Network for speech summarization [Maskey S. and Hirschberg J., 03]

- Dependency parsing with belief propagation [Smith D and Eisner J, 08]



Inference in Graphical Model

- Weights in Network make local assertions on how two nodes are related
- Inference algorithm takes these local assertions into global assertions between nodes[Jordan, M, 02]
- Many inference algorithms
  - Popular inference algorithm : Junction Tree algorithm

Inference

Inference in Naïve Bayes?

- Given a graph and probability function we want to compute $\mathrm{P}(\mathrm{x}_1, x_2, \ldots, x_n)$

$$\mathrm{P}(\mathrm{X}_h | X_e) = \frac{p(X_h, X_e)}{p(X_e)}$$

$$\mathrm{P}(\mathrm{X}_e) = \sum_{X_h} P(X_h, X_e)$$

- Need to compute marginals
  - Computation of marginals needed in both directed and undirected graphs

$$p(x_j, x_k) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_M} \prod_{i=1}^{M} p(x_i | \pi_i)$$

$$p(x_j, x_k) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_M} \frac{1}{Z} \prod_{c \in C} \psi_c(X_c)$$

Inference : Exploit Graph Structure

- We can compute marginals by summing over all other variables
  - Brute force, computationally expensive, not efficient
- Better algorithm?
  - Pass messages in the graph
  - Enforce consistency among messages

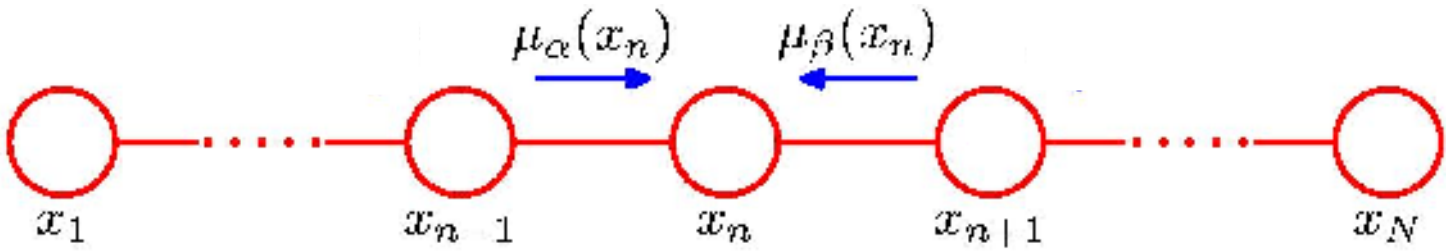
Inference on a Chain

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

$$p(x_n) = \sum_{x_1}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\cdots\sum_{x_N} p(\mathbf{x})$$

*Next 4 slides are from Bishop book resource [1]

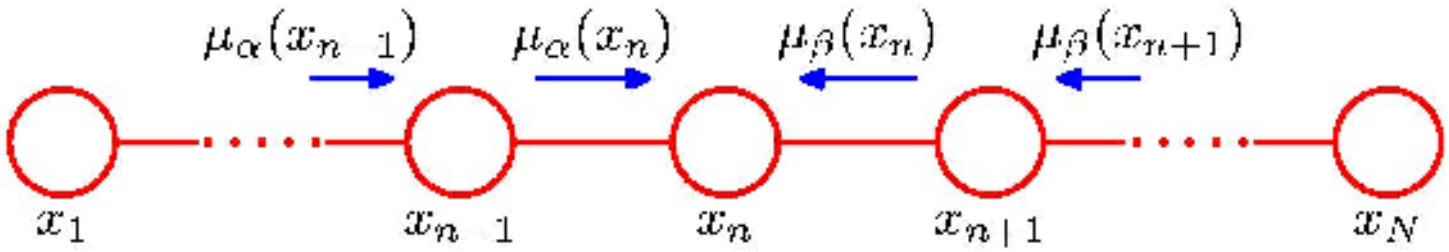
Inference on a Chain



$$p(x_n) = \frac{1}{Z}\underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right] \cdots\right]}_{\mu_\alpha(x_n)}$$

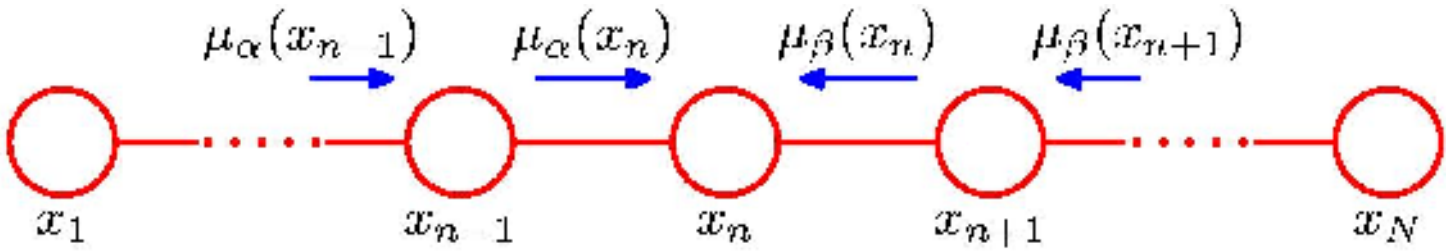
$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots\right]}_{\mu_\beta(x_n)}$$

Inference on a Chain



$$\begin{aligned}
\mu_\alpha(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right] \\
&= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}). \\
\mu_\beta(x_n) &= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots \right] \\
&= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \mu_\beta(x_{n+1}).
\end{aligned}$$

Inference on a Chain



$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \qquad \mu_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

$$Z = \sum_{x_n} \mu_\alpha(x_n)\mu_\beta(x_n)$$

Inference on a Chain

- To compute local marginals:
  - Compute and store all forward messages, $\mu_\alpha(x_n)$
  - Compute and store all backward messages, $\mu_\beta(x_n)$
  - Compute Z at any node x_m
  - Compute

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n)$$

for all variables required.

Graphical Model for Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

POS-tagged sentence

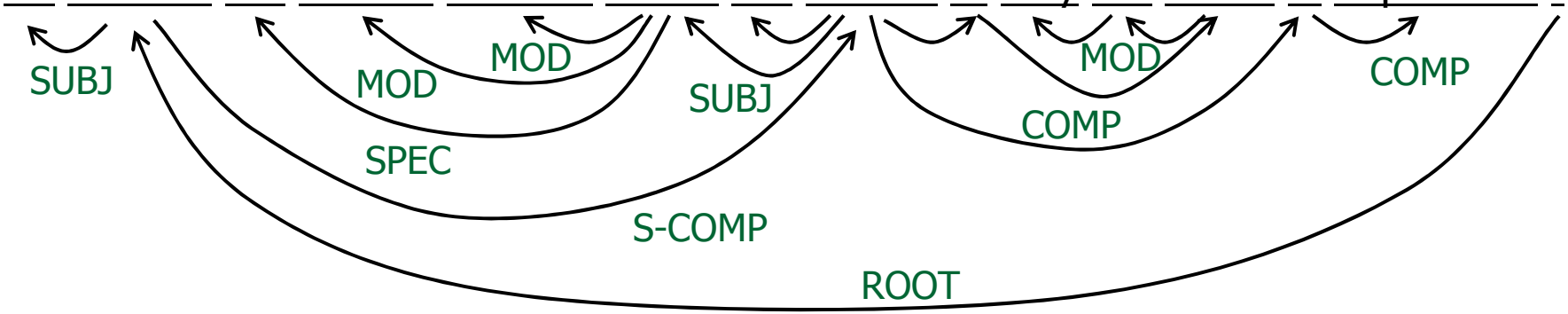
He reckons the current account deficit will narrow to only 1.8 billion in September.

PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .

Word dependency parsing

Word dependency parsed sentence

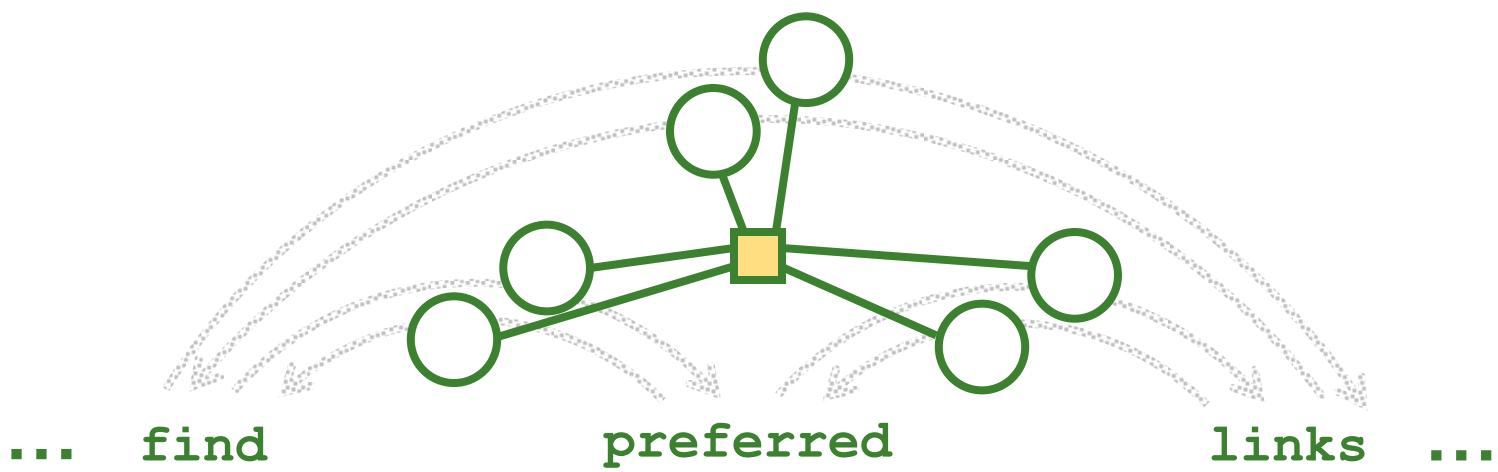
He reckons the current account deficit will narrow to only 1.8 billion in September .



slide from Smith D and Eisner J, 08, [2]

Dependency Parsing with Belief Propagation [Smith D and Eisner J, 08]

- We can have dependencies represented as nodes of a factor graph
- Add constraints to make it a legal tree, no loops



Graphical Models Summary

- Represent distributions with graphs (directed, undirected, factor graphs)
- Inference on the graph can be done by message passing algorithms
- Gaining more attention in NLP community

Statistical Methods for NLP

Part II Equations to Implementation

Topics We Covered

NLP -- ML

Text Mining	Linear Models of Regression
Text Categorization	Linear Methods of Classification Support Vector Machines
Information Extraction	Hidden Markov Model Maximum Entropy Models Conditional Random Fields
Topic and Document Clustering	K-means Expectation Maximization Viterbi Search
Machine Translation	
Language Modeling	
Speech-to-Speech Translation	Graphical Models

Scoring Unstructured Text

All Amazon reviewers may not rate the product, may just write reviews, we may have to infer the rating based on text review

Rate This Item to Improve Your Recommendations

I own it Rate this item

Customer Reviews

Average Customer Rating

(585 customer reviews)

5 star:	(460)
4 star:	(86)
3 star:	(13)
2 star:	(11)
1 star:	(15)

Ease of use	(11
Image quality	(112)
Construction quality	(110)
Battery life	(109)

› See and rate all 14 attributes.

Some of these patterns could be exploited to discover knowledge

Most Helpful Customer Reviews

1,568 of 1,594 people found the following review helpful:

Great camera, one of the best low(er)-end DSLRs on the market, April 23, 2008

By **Hyun Yu** - See all my reviews

TOP 1000 REVIEWER REAL NAME VINE™ VOICE

My journey with DSLRs began back in 2003 with the original Digital Rebel. DSLRs changed my photography for the better like nothing else. Five years and some 25,000 shots later, it's still going strong. Along the way I upgraded to the Canon 30D, which is a fantastic camera as well. When the 40D was announced, I decided to wait until the 50D sometime in 2009, but wanted a newer backup/second body for my photography needs. So when the XSi/450D was announced, it sounded like a perfect fit for my needs.

I got it from Amazon.com three days ago, and have given it a pretty good workout since then, having shot about 650 shots under a variety of shooting conditions and with a number of different Canon and third-party lenses. The following are my impressions.

The build feels very good. The camera feels wonderfully light yet well built. I'm 6ft tall with average size hands, and the camera feels good in my hand. The battery grip, to me, defeats the purpose of having a small, light DSLR, so I opted for a Hakuba/Opteka grip (it's a plate that screws into the tripod socket that enables you to use the excellent Canon E1 hand strap with it) and I couldn't be happier. I'm not a fan of neck straps, so this works well for me (see the uploaded photo for the configuration).

Patterns may exist in unstructured text

Review of a camera in Amazon

Linear Regression

- Empirical Loss (Predicted vs. Original)

$$J(\theta) = \frac{1}{2N} \left\| \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix} - \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1K} \\ 1 & x_{21} & x_{22} & \dots & x_{2K} \\ & & \dots & & \\ & & \dots & & \\ & & \dots & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \cdot \\ \cdot \\ \cdot \\ \theta_K \end{bmatrix} \right\|^2$$

Y **X** **θ**

$$\theta^* = (X^T X)^{-1} X^T Y$$

- Given out N training data points we can build X and Y matrix and perform the matrix operations
- For any new test data plug in the x values (features) in our regression function with the best theta values we have

Implementation of Multiple Linear Regression

$$\theta^* = (X^T X)^{-1} X^T Y$$

- Given out N training data points we can build X and Y matrix and perform the matrix operations
- Can use MATLAB or write your own, Matrix multiplication implementation to get theta matrix
- For any new test data plug in the x values (features) in our regression function with the best theta values we have

Regression Pseudocode

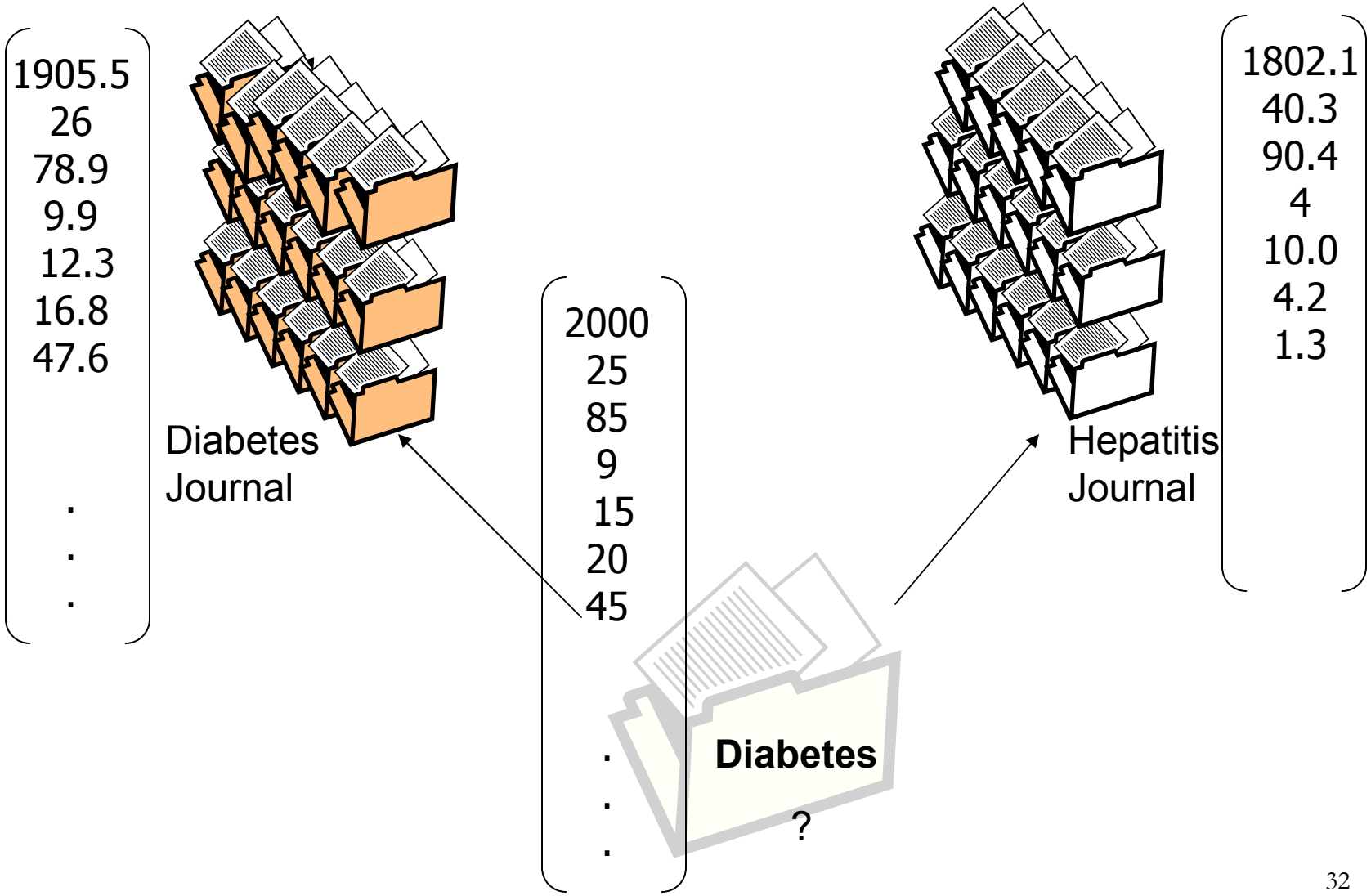
Load X1, … XN
Load Y1,…. YN

Build X Matrix (NxK)

$$\theta^* = (X^T X)^{-1} X^T Y$$

Test Y = $\theta^* X$

Text Classification



Perceptron

- We want to find a function that would produce least training error

$$R_n(w) = \frac{1}{n} \sum_{i=1}^{n} Loss(y_i, f(x_i; w))$$

Perceptron Pseudocode

```
Load X1, …, XN
Load Y1,…., YN

for (i = 1 to N)
   if (y(i) * x(i) * w <= 0)
     w = w + y(i) * x(i)
   end
end
```

Naïve Bayes Classifier for Text

$$P(Y_k, X_1, X_2, ..., X_N) = P(Y_k)\Pi_i P(X_i|Y_k)$$

Prior Probability of the Class

Conditional Probability of feature given the Class

Here N is the number of words, not to confuse with the total vocabulary size

Naïve Bayes Classifier for Text

- Given the training data what are the parameters to be estimated?

$$P(Y) \qquad P(X|Y_1) \qquad P(X|Y_2)$$

Diabetes : 0.8
Hepatitis : 0.2

the: 0.001
diabetic : 0.02
blood : 0.0015
sugar : 0.02
weight : 0.018
…

the: 0.001
diabetic : 0.0001
water : 0.0118
fever : 0.01
weight : 0.008
…

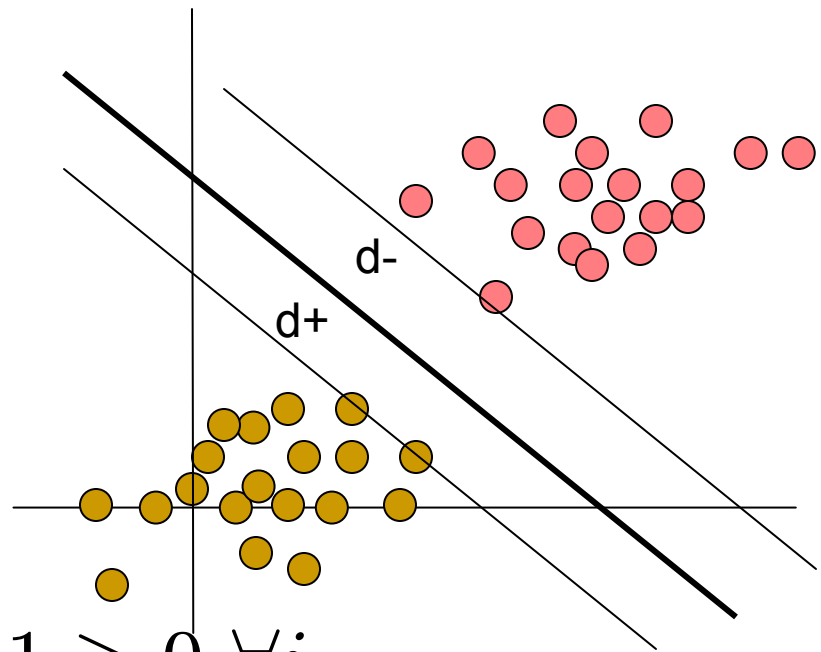
Naïve Bayes Pseudocode

```
Foreach Class C
   totalWCount=0
     for J = 1 to |V|
       count(WJ)
     end
     totalCount = totalCount + totalCount(WJ)
    for J= 1 to |V|
      C.WJProb = count(WJ) + delta /totalCount(WJ) + delta * |V|
    end
end
```

SVM: Maximizing Margin with Constraints

- We can combine the two inequalities to get

$$y_i(\mathbf{w}^T x_i + b) - 1 \geq 0 \ \forall i$$



- Problem formulation
  - Minimize $\frac{\|w\|^2}{2}$
  - Subject to

$$y_i(\mathbf{w}^T x_i + b) - 1 \geq 0 \ \forall i$$

Dual Problem

- Solve dual problem instead
- Maximize

$$J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i . x_j)$$

- subject to constraints of

$$\alpha_i \geq 0 \ \forall i$$

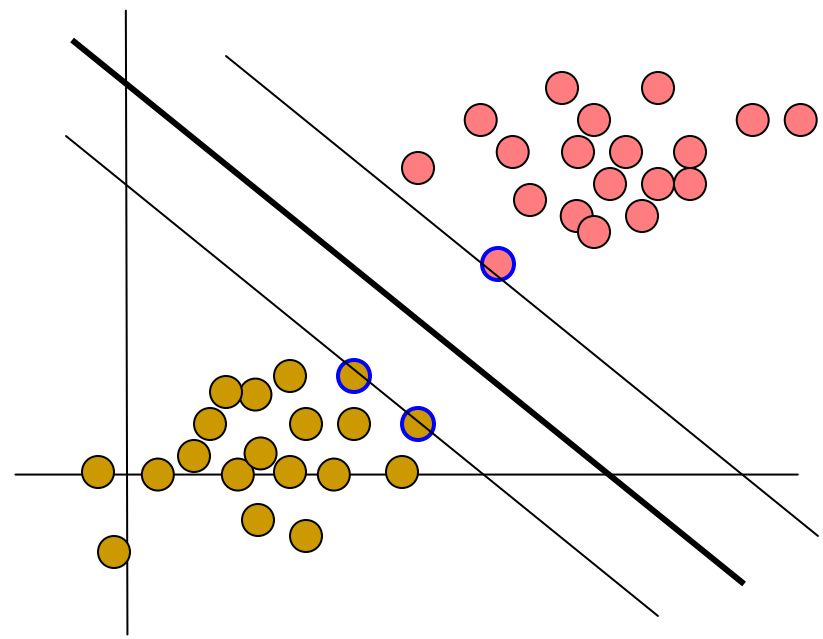
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

SVM Solution

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i$$

- Linear combination of weighted training example
- Sparse Solution, why?
  - Weights zero for non-support vectors

$$\sum_{i \in SV} \widehat{\alpha}_i y_i (x_i . x) + \widehat{b}$$



Sequential Minimal Optimization (SMO) Algorithm

- The weights are just linear combinations of training vectors weighted with alphas
- We still have not answered how do we get alphas
  - Coordinate ascent

Do until converged

select pair of alpha(i) and alpha(j)

reoptimize W(alpha) with respect to alpha(i) and alpha(j)

holding all other alphas constant

done

Information Extraction Tasks

- Named Entity Identification
- Relation Extraction
- Coreference resolution
- Term Extraction
- Lexical Disambiguation
- Event Detection and Classification

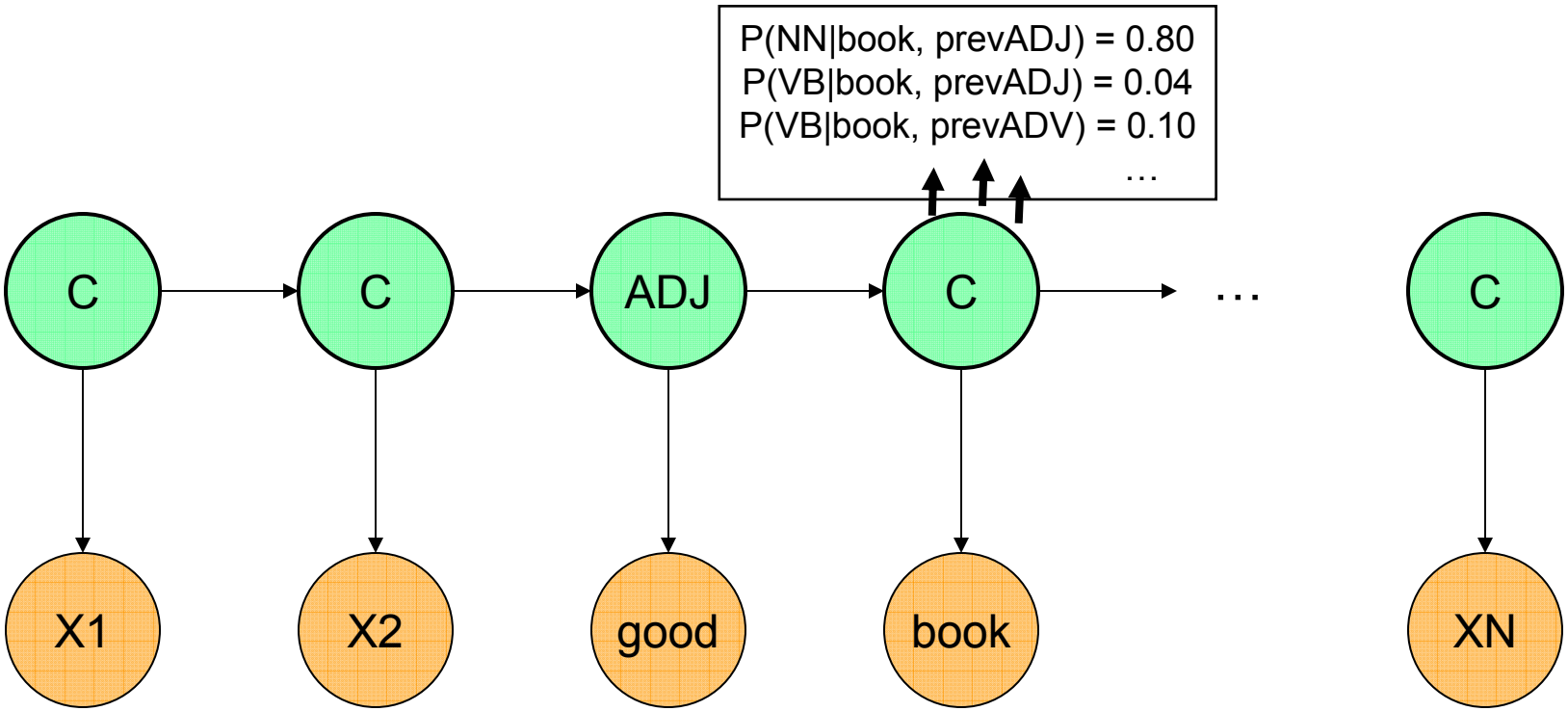
Classifiers for Information Extraction

- Sometimes extracted information can be a sequence
- Extract Parts of Speech for the given sentence

DET	VB	AA	ADJ	NN
This	is	a	good	book

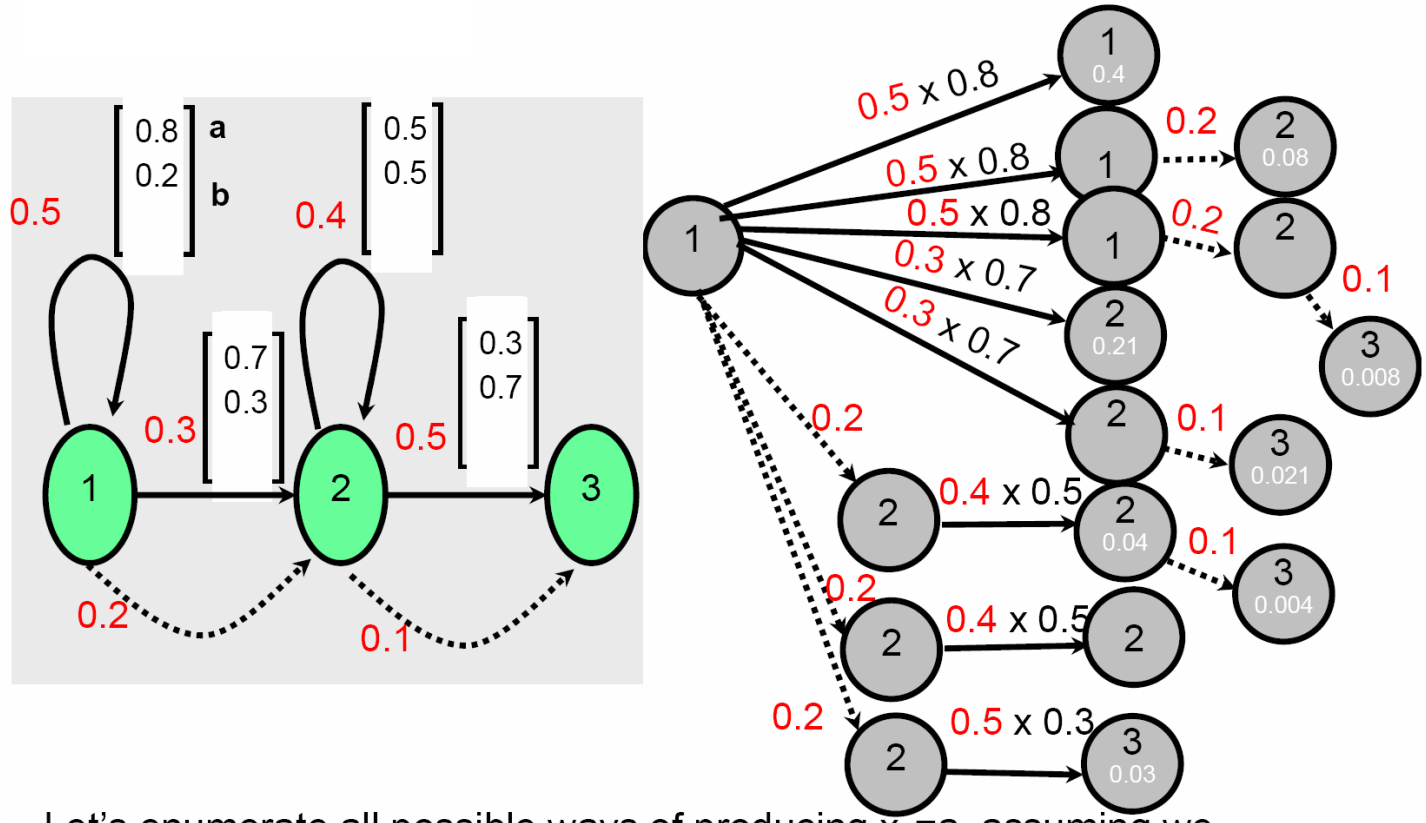
- What kind of classifier may work well for this kind of sequence classification?

Classification with Memory



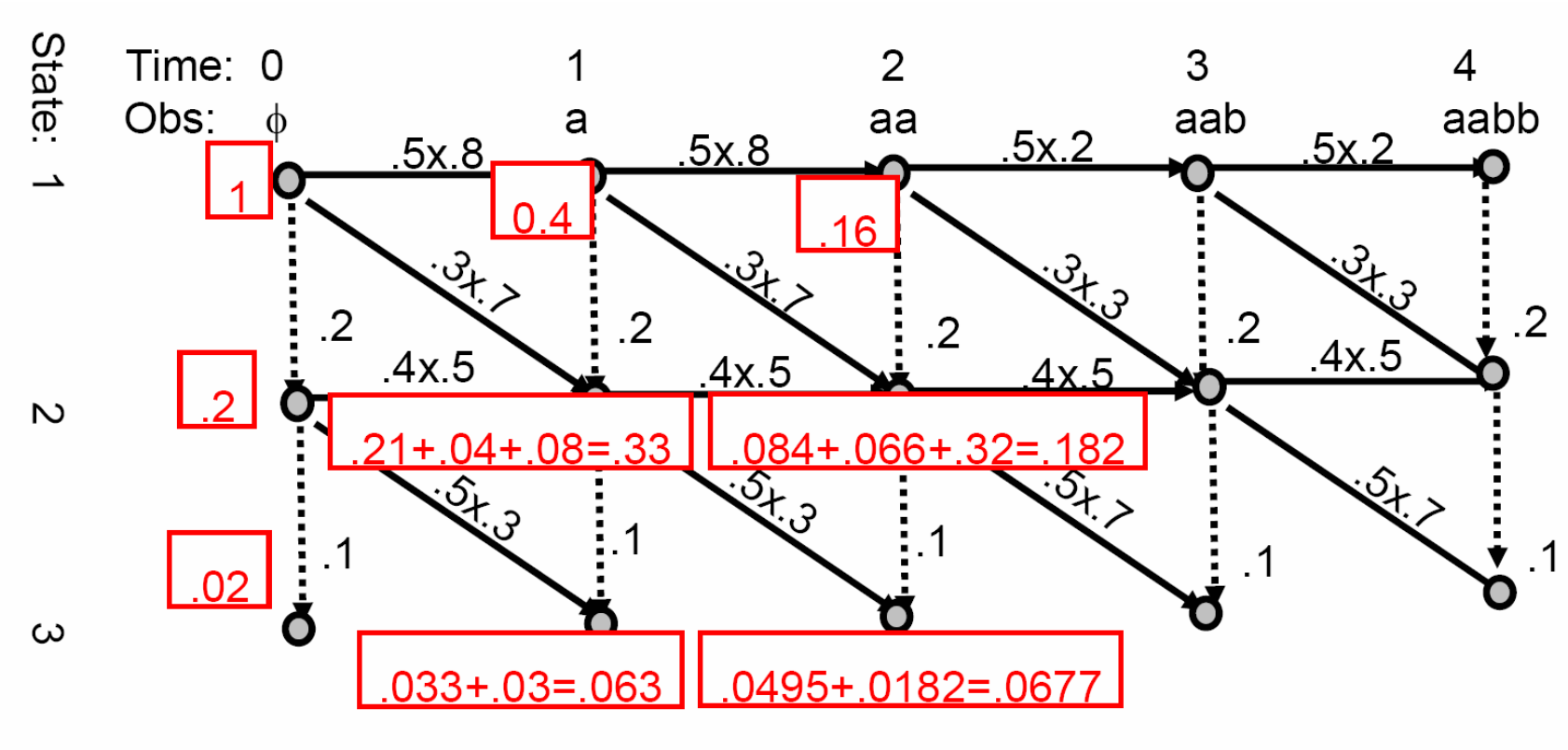
- C(t) (class) is dependent on current observations X(t) and previous state of classification (C(t-1))
- C(t) can be POS tags, document class, word class, X(t) can be text based features

HMM Example



Let's enumerate all possible ways of producing x_1=a, assuming we start in state 1.

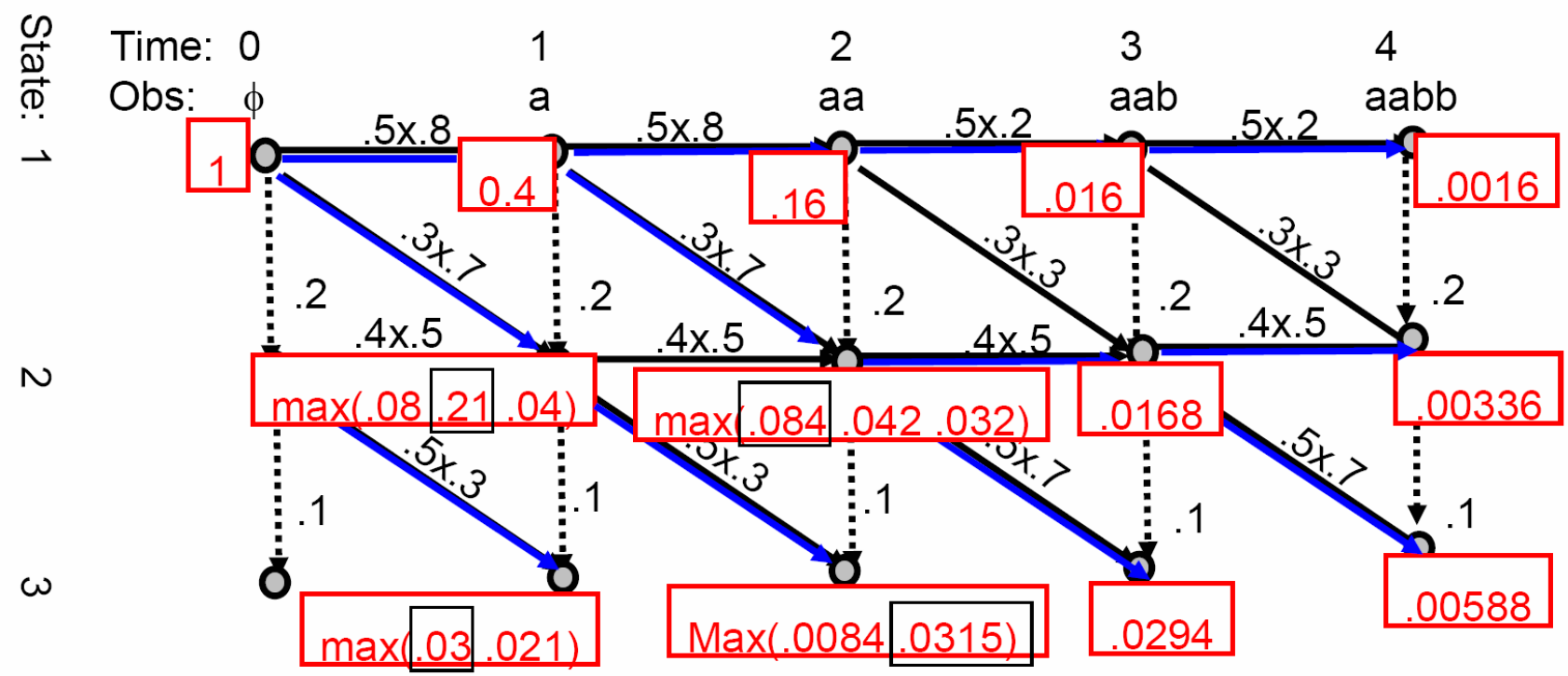
Forward Algorithm : Computing alphas



Forward Algorithm Perl Code

```
Sub Forward() {
   for (my $t = 0; $t < @obs ; $t++){
            #sum across i,j transitions for all starting i and ending at j
            for (my $i = 0; $i < $num_states; $i++){

               my $tot_forward_prob_for_cur_dest = 0;
               for (my $j = 0; $j < $num_states; $j++){

                        #multiply transition prob * obs prob
                        my $trans_prob = $trans_matrix[$i][$j];

                        #get obs_id
                        my $cur_obs = $obs[$t];
                        my $obs_id = $obs_vocab_id{$cur_obs};

                        #state i producing the given observation
                        my $obs_prob = $obs_matrix[$i][$obs_id];

                        #compute the forward prob
                        if ($t == 0){
                           my $start_trans_prob = $start_state_matrix[$j];
                           $tot_forward_prob_for_cur_dest =  $start_trans_prob * $obs_prob;
                        }
                        else{
                           $tot_forward_prob_for_cur_dest = $tot_forward_prob_for_cur_dest
                                    + $forward_prob_matrix[$i][$t] * $trans_prob * $obs_prob;
                        }
               }
                #this is for passing on forward pass to next iteration
               $forward_prob_matrix[$i][$t+1] = $tot_forward_prob_for_cur_dest;
            }
   }
}
```

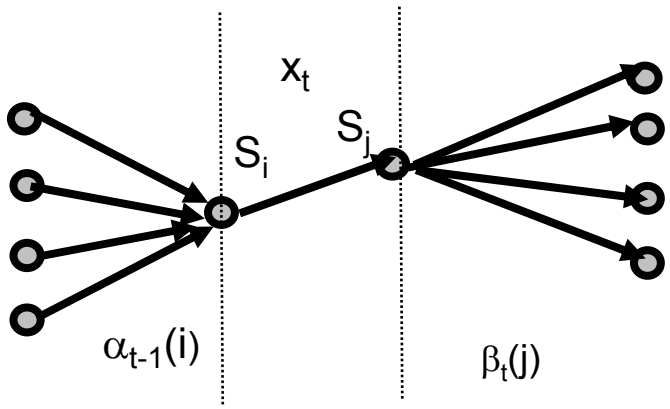
Viterbi Algorithm



Problem 3: Forward-Backward Algorithm

Consider transition from state i to j, tr_{ij}

Let $p_t(tr_{ij},X)$ be the probability that tr_{ij} is taken at time t, and the complete output is X.



$$p_t(tr_{ij},X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)$$

Problem 3: F-B algorithm cont'd

$p_t(tr_{ij},X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)$

where:

$\alpha_{t-1}(i) = Pr(state=i, x_1 \ldots x_{t-1})$ = probability of being in state i and having produced $x_1 \ldots x_{t-1}$

a_{ij} = transition probability from state i to j

$b_{ij}(x_t)$ = probability of output symbol x_t along transition ij

$\beta_t(j) = Pr(x_{t+1} \ldots x_T | state= j)$ = probability of producing $x_{t+1} \ldots x_T$ given you are in state j

Estimating Transition and Emission Probabilities

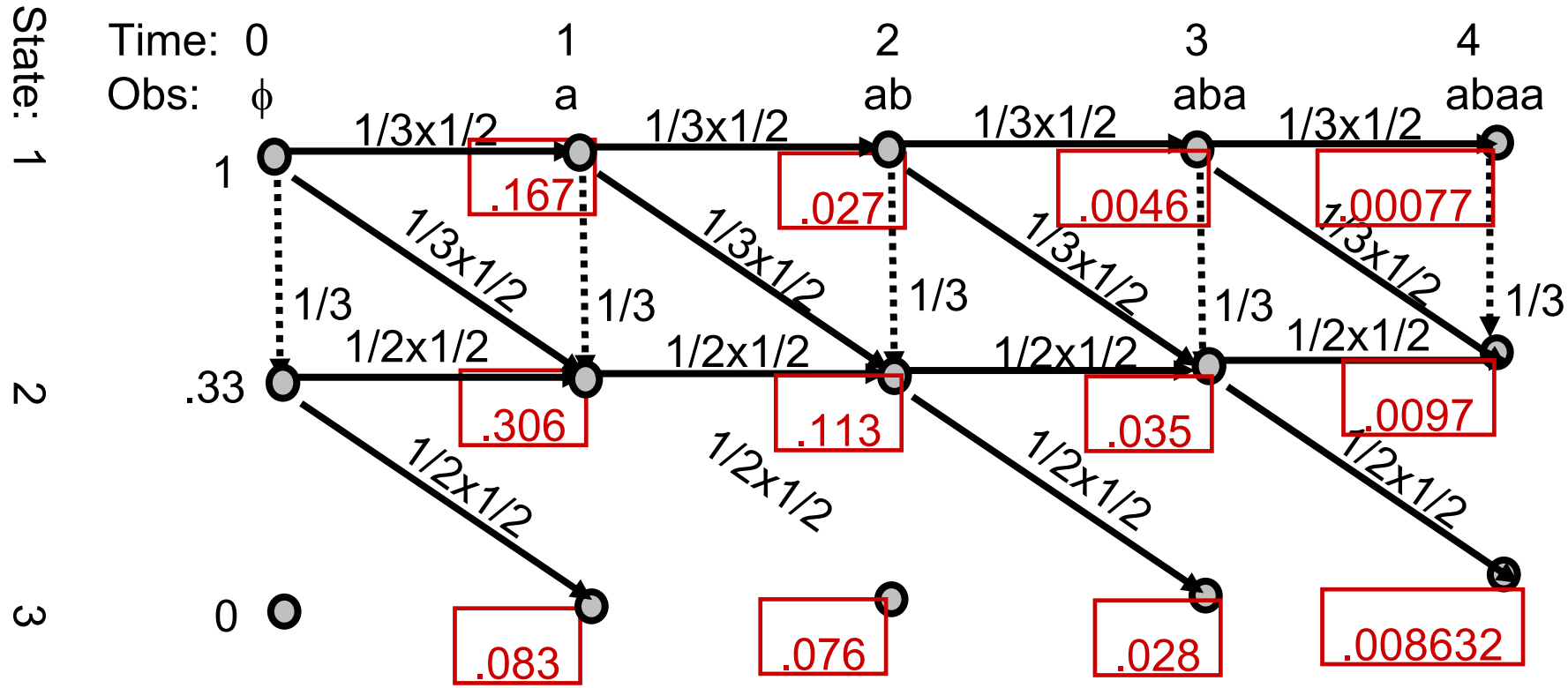
$$a_{ij} = \frac{count(i \rightarrow j)}{\sum_{q \in Q} count(i \rightarrow q)}$$

$$\hat{a}_{ij} = \frac{\text{Expected number of transitions from state i to j}}{\text{Expected number of transitions from state i}}$$

$$\hat{b}_j(x_t) = \frac{\text{Expected number of times in state j and observing symbol xt}}{\text{Expected number of time in state j}}$$

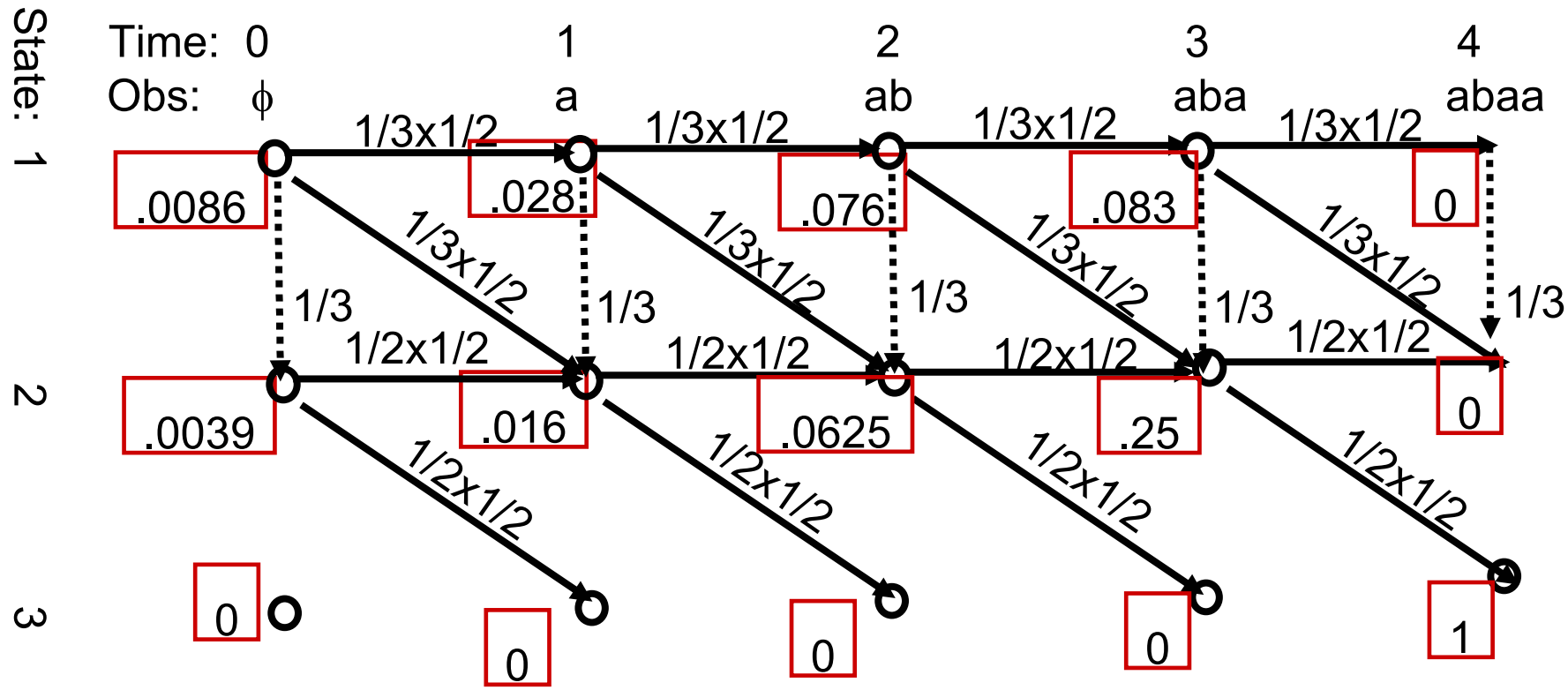
Problem 3: F-B algorithm

Compute α's. since forced to end at state 3, α_T=.008632=Pr(X)



Problem 3: F-B algorithm, cont'd

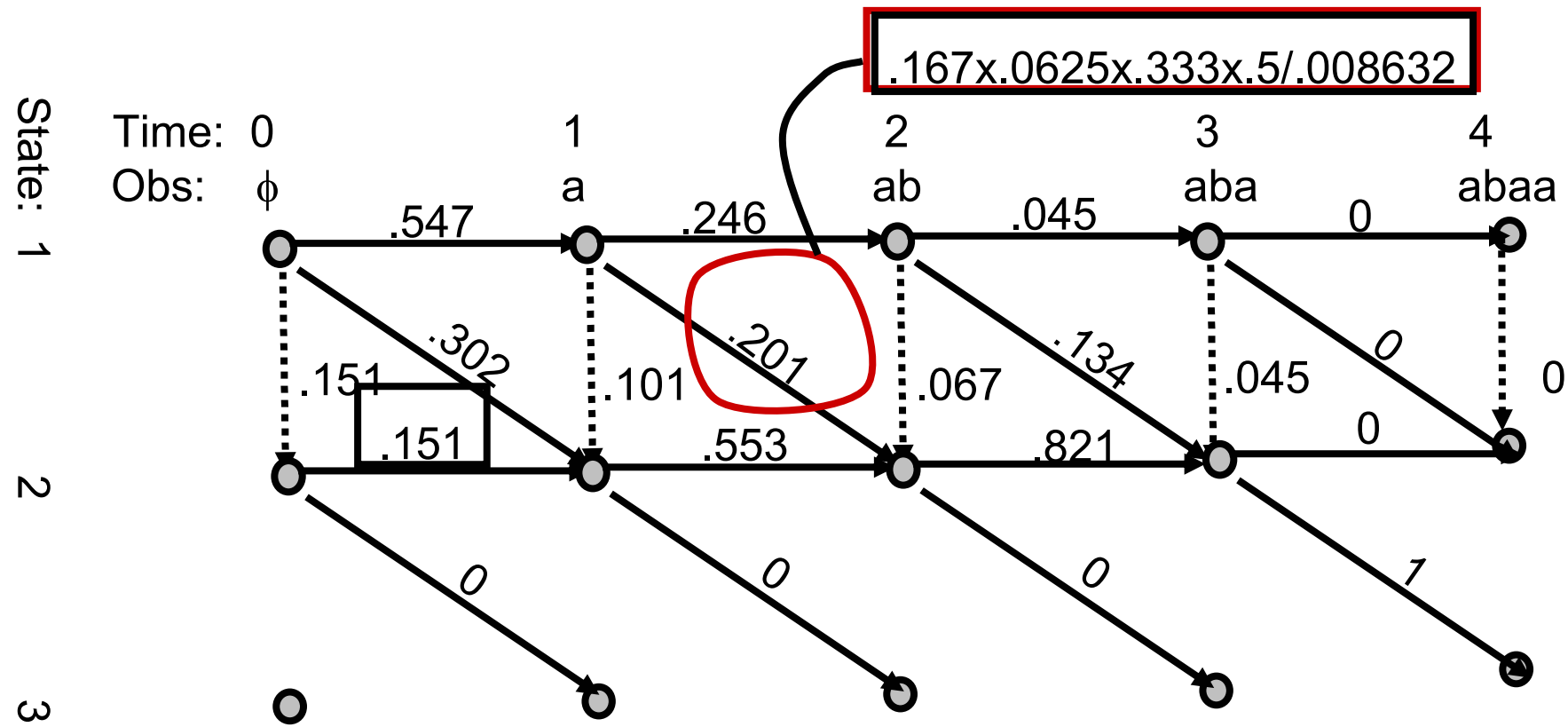
Compute β's.



Problem 3: F-B algorithm, cont'd

Compute counts. (a posteriori probability of each transition)

$$c_t(tr_{ij}|X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)/\ Pr(X)$$



Backward Algorithm Perl Code

```
sub Backward() {
   my (@obs) = @_;
   # we need to start at the end
   my $end_t = scalar(@obs);
   #go upto zero so start 1 less
   $end_t--;
   for (my $t = $end_t; $t >=0; $t--){
                #sum across i,j transitions for all starting i and ending at j
                for (my $i = 0; $i < $num_states; $i++){

                   my $tot_back_prob_for_cur_dest = 0;
                   for (my $j = 0; $j < $num_states; $j++){

                                #multiply transition prob * obs prob
                                my $trans_prob = $trans_matrix[$i][$j];

                                #get obs_id
                                my $cur_obs = $obs[$t];
                                my $obs_id = $obs_vocab_id{$cur_obs};

                                #state i producing the given observation
                                my $obs_prob = $obs_matrix[$i][$obs_id];

                                #compute the back prob
                                if ($t == $end_t){
                                   my $end_trans_prob = $end_state_matrix[$j];
                                   $tot_back_prob_for_cur_dest =  $end_trans_prob ;
                                }
                                else{
                                   $tot_back_prob_for_cur_dest = $tot_back_prob_for_cur_dest +
                                              $back_prob_matrix[$i][$t] * $trans_prob * $obs_prob;
                                }
                   }
                   $back_prob_matrix[$i][$t-1] = $tot_back_prob_for_cur_dest;
                }
    }
}
```

Finding Maximum Likelihood of our Conditional Models (Multinomial Logistic Regression)

$$(C|D,\lambda) = \prod_{(c,d)\in(C,D)} p(c|d,\lambda)$$

$$P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} logp(c|d,\lambda)$$

$$P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} log\frac{exp\sum_i \lambda_i f_i(c,d)}{\sum_{c'} exp\sum_i \lambda_i f_i(c',d)}$$

Maximizing Conditional Log Likelihood

$$P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} logexp \sum_i \lambda_i f_i(c,d)$$

$$-\sum_{(c,d)\in(C,D)} log \sum_{c'} exp \sum_i \lambda_i f_i(c',d)$$

Taking derivative and setting it to zero

$$\frac{\partial log(P|C,\lambda)}{\partial \lambda_i} = \sum_{(c,d)\in(C,D)} f_i(c,d) - \sum_{(c,d)\in(C,D)} \sum_{c'} P(c'|d,\lambda) f_i(c',d)$$

Empirical count (f_i, c)

Predicted count (f_i, λ)

Optimal parameters are obtained when empirical expectation equal predicted expectation

Finding Model Parameters

- We saw that optimum parameters are obtained when empirical expectation of a feature equals predicted expectation
- We are finding a model having maximum entropy and satisfying constraints for all features fj

$$E_p(f_j) = E_{\tilde{(p)}}(f_j)$$

- Hence finding the parameters of maximum entropy model entails to maximizing conditional log-likelihood and solving it
  - Conjugate Gradient Descent
  - Quasi Newton's Method
  - A simple iterative scaling
    - Features are non-negative (indicator functions are non-negative)
    - Add a slack feature $f_{m+1}(d,c) = M - \sum_{j=1}^{m} f_j(d,c)$
    - where $M = max_{i,c} \sum_{j=1}^{m} f_j(d_i, c)$

Finding Model Parameters

- We saw that optimum parameters are obtained when empirical expectation of a feature equals predicted expectation
- We are finding a model having maximum entropy and satisfying constraints for all features fj

$$E_p(f_j) = E_{\tilde{(p)}}(f_j)$$

- Hence finding the parameters of maximum entropy model entails to maximizing conditional log-likelihood and solving it
  - Conjugate Gradient Descent
  - Quasi Newton's Method
  - A simple iterative scaling
    - Features are non-negative (indicator functions are non-negative)
    - Add a slack feature
    - where

$$f_{m+1}(d,c) = M - \sum_{j=1}^{m} f_j(d,c)$$

$$M = max_{i,c} \sum_{j=1}^{m} f_j(d_i,c)$$

How to Cluster Documents with No Labeled Data?

- Treat cluster IDs or class labels as hidden variables
- Maximize the likelihood of the unlabeled data
- Cannot simply count for MLE as we do not know which point belongs to which class
  - User Iterative Algorithm such as K-Means, EM

Document Clustering with K-means

- We can estimate parameters by doing 2 step iterative process
  - Minimize J with respect to r_{nk}
    - Keep μ_k fixed

Step 1

  - Minimize J with respect to μ_k
    - Keep r_{nk} fixed

Step 2

- Minimize J with respect to r_{nk}
  - Keep μ_k fixed

Step 1

- Optimize for each n separately by choosing r_{nk} for k that gives minimum $||x_n - r_{nk}||^2$

$$r_{nk} = 1 \text{ if } k = argmin_j ||x_n - \mu_j||^2$$
$$= 0 \text{ otherwise}$$

- Assign each data point to the cluster that is the closest
- Hard decision to cluster assignment

- Minimize J with respect to μ_k
  - Keep r_{nk} fixed

Step 2

- J is quadratic in μ_k. Minimize by setting derivative w.rt. μ_k to zero

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

- Take all the points assigned to cluster K and re-estimate the mean for cluster K

Expectation Maximization for Gaussian Mixture Models

$$\gamma(z_{nk}) = E(z_{nk}|x_n) = p(z_k = 1|x_n)$$

- E-Step

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

Estimating Parameters

- M-step

$$\mu'_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\sum'_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu'_k)(x_n - \mu'_k)^T$$

$$\pi'_k = \frac{N_k}{N}$$

$$\text{where } N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

- Iterate until convergence of log likelihood

$$log\ p(X|\pi, \mu, \sum) = \sum_{n=1}^{N} log\ (\sum_{k=1}^{k} \mathcal{N}(x|\mu_k, \sum_k))$$

Maximum Entropy Markov Model

MEMM Inference

$$\hat{T} = argmax_T P(T|W)$$

$$= \boxed{argmax_T \prod_i P(t_i|t_{i-1}, w_i)}$$

HMM Inference

$$\hat{T} = argmax_T P(T|W)$$

$$\hat{T} = argmax_T P(W|T)P(T)$$

$$= \boxed{argmax_T \prod_i P(w_i|t_i)p(t_i|t_{i-1})}$$

Transition Matrix Estimation

MEMM Inference

$$\hat{T} = argmax_T P(T|W)$$

$$= \boxed{argmax_T \prod_i P(t_i|t_{i-1}, w_i)}$$

- Transition is dependent on the state and the feature
- These features do not have to be just word id, it can be any features functions
- If q are states and o are observations we get

$$\boxed{P(q_i|q_{i-1}, o_i) = \frac{1}{Z(o,q')} exp(\sum_i w_i f_i(o,q))}$$

Viterbi in HMM vs. MEMM

- HMM decoding:

$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) P(s_j|s_i) P(o_t|s_j) \; 1 \leq j \leq N, 1 < t \leq T$$

- MEMM decoding:

$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) P(s_j|s_i, o_t) \; 1 \leq j \leq N, 1 < t \leq T$$

This computed in maximum entropy framework
but has markov assumption in states thus its name MEMM

Conditional Random Field

Sum over
all data points

Sum over
all feature function

Weight for
given feature function

Feature
Functions

$$P(\overline{y}|\overline{x};w) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \overline{x}, i))}{\sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \overline{x}, i))}$$

Sum over
all possible label sequence

Feature function can
access all of observation

Model log linear on Feature functions

Inference in Linear Chain CRF

- We saw how to do inference in HMM and MEMM
- We can still do Viterbi dynamic programming based inference

$$\overline{y}^* = argmax_{\overline{y}} p(\overline{y}|\overline{x}; w)$$

$$= argmax_{\overline{y}} \sum_j w_j F_j(\overline{x}, \overline{y})$$

$$= argmax_{\overline{y}} \sum_j w_j \sum_i f_j(y_{i-1}, y_i, \overline{x}, i)$$

$$= argmax_{\overline{y}} \sum_i g_i(y_{i-1}, y_i)$$

Denominator?

$$\text{where} g_i(y_{i-1}, y_i) = \sum_j w_j f_j(y_{i-1}, y_i, \overline{x}, i)$$

x and i arguments of f_j dropped in definition of g_i
g_i is different for each I, depends on w, x and i

Computing Expected Counts

$$P(\overline{y}|\overline{x};w) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \overline{x}, i))}{\sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \overline{x}, i))}$$

Need to compute denominator

Optimization: Stochastic Gradient Ascent

For all training (x,y)

For all j

Compute $E_{y' \sim p(y'|x;w)}[F_j(x, y')]$

$$w_j := w_j + \alpha(F_j(x, y) - E_{y' \sim p(y'|x;w)}[F_j(x, y')])$$

End For

End For

References

- [1] Christopher Bishop, “Pattern Recognition and Machine Learning” 2006
- [2] David Smith and Jason Eisner, “Dependency Parsing by Belief Propagation,” Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 145–156, Honolulu, October 2008.