COLUMBIA UNIVERSITY

Statistical NLP for the Web

Log Linear Models, MEMM, Conditional Random Fields

Sameer Maskey

Week 13, Nov 28, 2012

Announcements

- Next lecture is the last lecture
- Wrap up of the semester

Final Project

- Final Project Presentation Day
  - December 12th, 10:00 AM to 2pm
  - Wednesday
  - Each team 8 min talk
  - 2 min for Q&A
  - CS Conference room

Final Project Grading

- Final Project : Report+Presentation+Demo
  - 65% of 55 = 35.75 points
  - Final Project Report (30%)
  - Demo (15%)
  - Final Presentation (20%)

Final Project Report

- Same length as your Intermediate report
- Similar to your Intermediate Report I
- The final report instructions will be available through the course website

HW3

- Due Nov 30th (11:59pm)
- Q1 : implementing simple example from the class is ok as well
- Q2 : Look at previous slides, Python code is already in one of the slides
  - Modify to do n-gram counts
  - Compute bigram probabilities

Topics for Today

- Log-linear Models
- Maximum Entropy Markov Models
- Conditional Random Fields
- Applications of CRF for NLP

Naïve Bayes vs. Conditional Random Field

Naïve Bayes Model

- Trained by maximizing likelihood of data and class
- Features are assumed independent
- Feature weights set independently

CRF Model

- Trained by maximizing conditional likelihood of classes
- Dependency on features taken account by feature weights
- Feature weights are set mutually
- Good for sequence prediction

Max Ent Model vs. CRF Model

- Both are types of log-linear models
- Max Ent variation called Max Ent Markov Model is more similar to CRF Model addresses some deficiencies with MEMM
- Training method is different
  - normalization is over all possible state sequence and labels
  - This makes the training bit more complicated
- Can train both models with Iterative Scaling, though stochastic gradient method and other numerical optimization methods are preferred

HMM vs. CRF

- Both have efficient inference algorithms to find the best sequence
- Some differences:

HMM

- maximizing p(x,y)
- Models p(x) as well
- Limited on types of features that can be used
- Per State Normalization

CRF

- maximizing p(y|x)
- No need to model p(x)
- Allows much more set of features to be used
- Normalization over the whole sequence

Relating CRF with Other Models

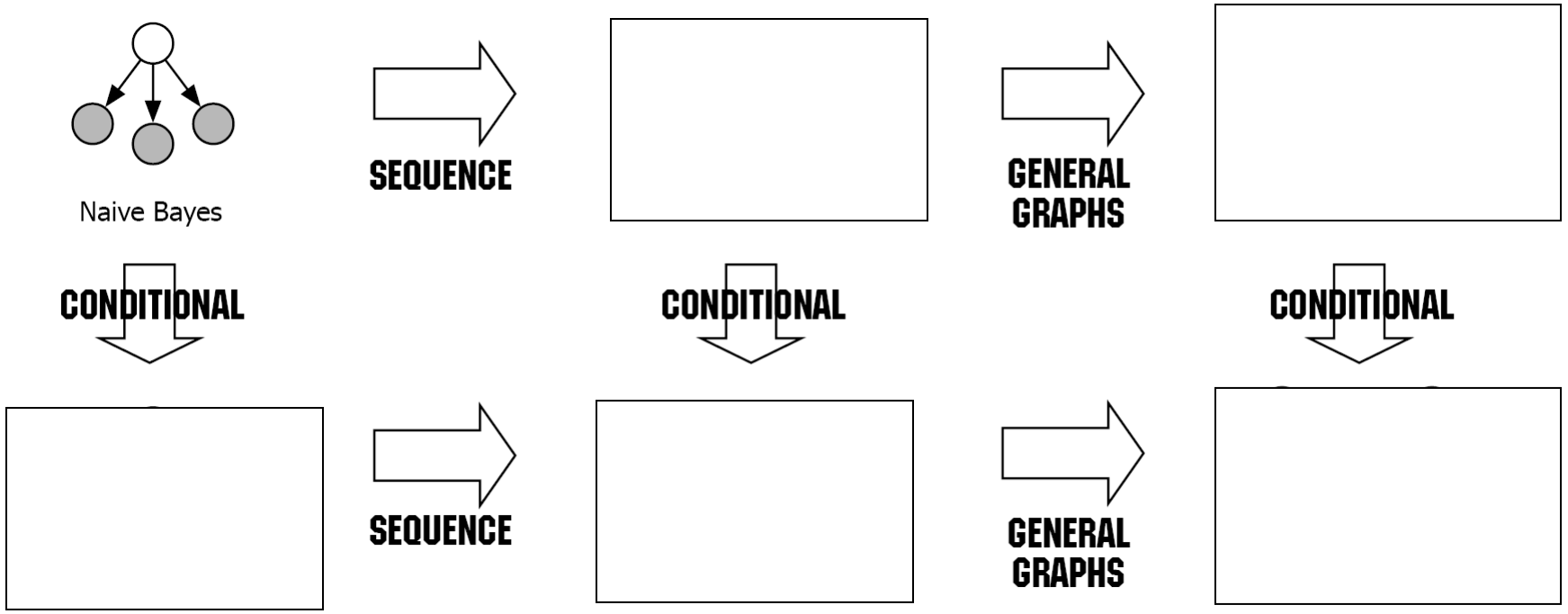


Figure from [1]

Relating CRF with Other Models

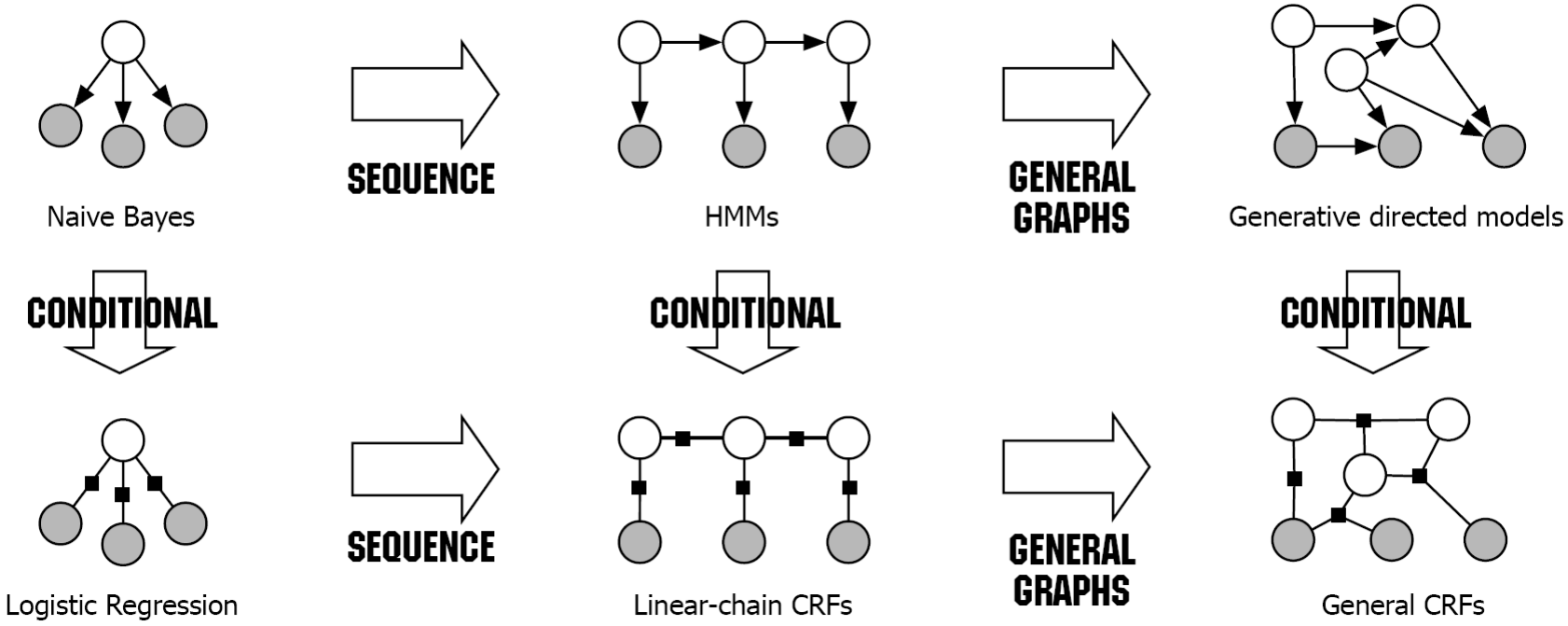
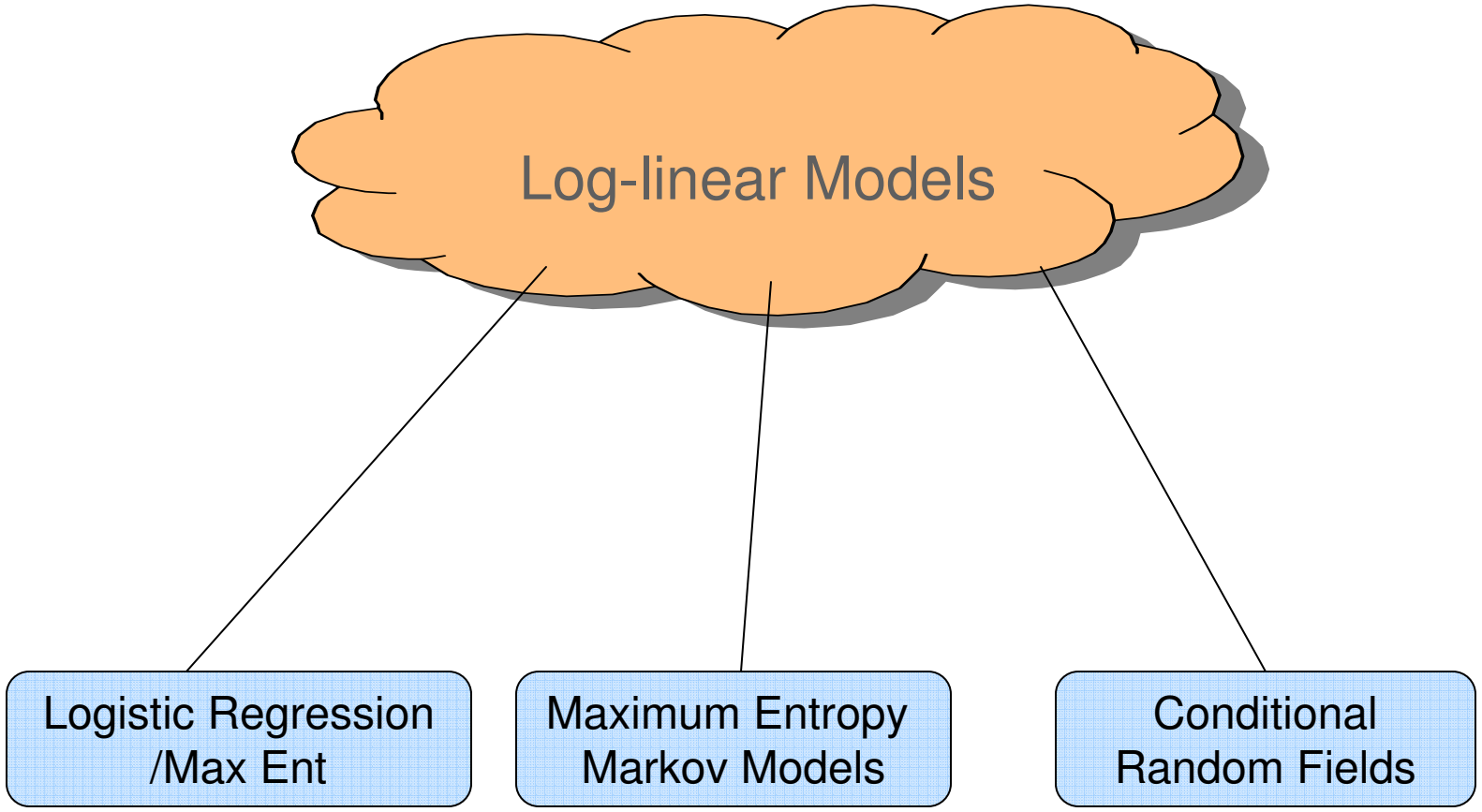


Figure from [1]

Applications of Conditional Random Fields

- Many uses in NLP
- Noun phrase segmentation [Sha and Pereira, 2003]
- Named Entity Recognition [McCallum and Li, 2003]
- Semantic Roles [Roth and Yih, 2005]
- RNA structure alignment [Liu et. al, 2005]
- Protein structure [Liu et. al, 2005]

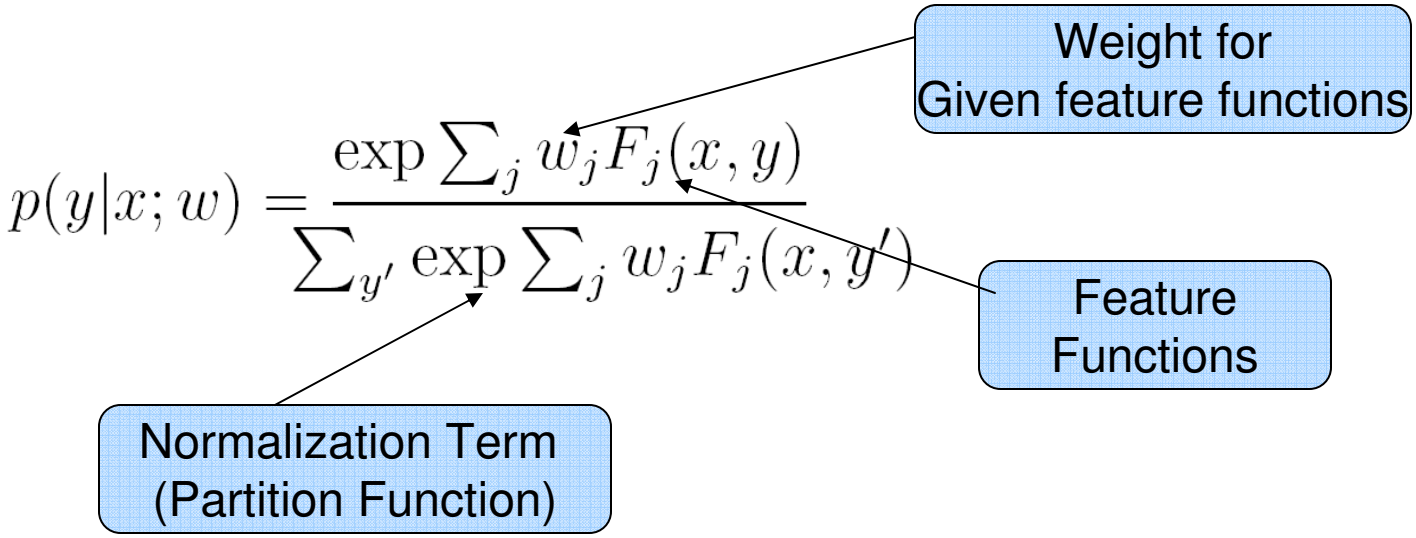
Log-linear Models



All of these models are a type of log-linear models, there are more of them

Log-Linear Model

- If x is any data point and y is the label, general log-linear linear model can be described as follows



Understanding the Equation Form

- Linear combination of features and weights
  - Can be any real value
- Numerator always positive (exponential of any number is +ve)
- Denominator normalizes the output making it valid probability between 0 and 1
- Ranking of output same as ranking of linear values
  - i.e. exponentials magnify the ranking difference but ranking still stay the same
- Why is it called log-linear?
  - Remember the logistic regression derivation?

Inference in Log-Linear Model

$$\hat{y} = \text{argmax}_y p(y|x; w)$$

- Best labels for the data given the model
- Basically saying we can find the best predicted label by doing linear combination of features and their weights and searching over the all label space

Inference in Log-Linear Model

$$\hat{y} = \text{argmax}_y p(y|x; w) = \text{argmax}_y \sum_j w_j F_j(x, y)$$

- Best labels for the data given the model
- Basically saying we can find the best predicted label by doing linear combination of features and their weights and searching over the all label space

Feature Functions

- Feature function can take account of relations between both data and label space
- Can be any real value
- Often feature functions are indicator functions such that they are 0 or 1 depending on absence or presence of the feature
- Weight of feature function captures how closely the given feature function is related with the given label

- $f_1(c,d)$ = { c=NN Λ curword(d)=book Λ prevword(d)=to}
- $f_3(c,d)$ = { c=VB Λ curword(d)=book Λ prevClass(d)=ADJ}

Remember Maximum Entropy Model

- We predicted the class label given a set of features for the given data point
- Inference was just taking the trained weights, doing linear combination and finding the class with highest probability
- Find probability score for each class
- What if we have to predict a sequence of classes?
- Is this method optimal?

$$p(c|\mathbf{x}) = \frac{exp(\sum_{i=0}^{N} \lambda_{ci} f_i)}{\sum_{c' \in C} exp(\sum_{i=0}^{N} \lambda_{c'i} f_i)}$$

Thinking of Classification as Search in Label Space

- We can think of classification as searching the feature and label space to find the correct sequence of labels
- Think about binary classifier for finding the right segmentation of a word

Seg-men-tation or segment-ation ?

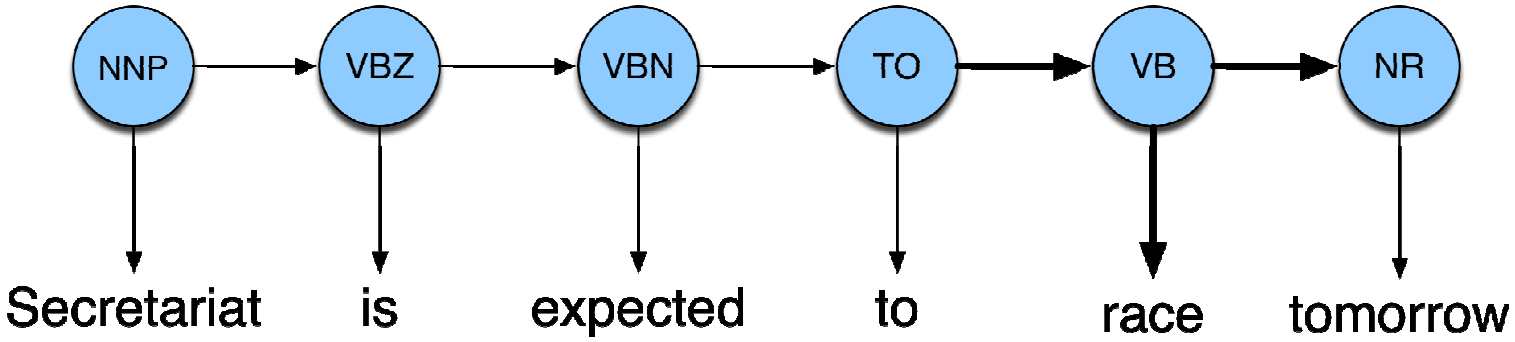
- Can treat as a binary classifier for individual letter
- If we believe that there is dependency between labels then the output label is in fact vector sequence of 0 and 1
  - 2^N possible label vectors
  - Cannot infer using brute force, need to search the label space given the features

HMM to Maximum Entropy Markov Model

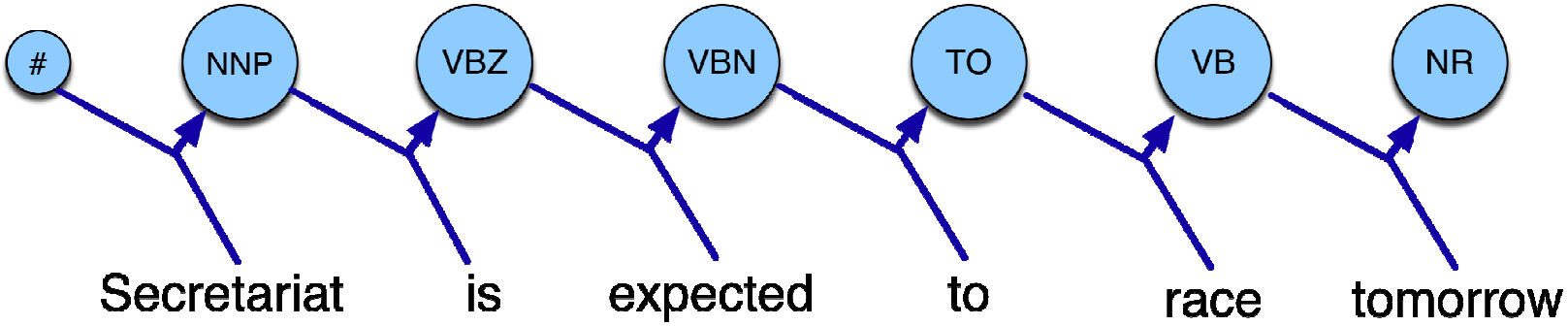
- We have already seen a modeling technique which exploits markov assumption to search over label space : HMM
- Issue with HMM was restrictions on types of features
- We can marry the good things of both HMM and Maximum Entropy models
  - Use Viterbi and Forward-Backward styled algorithm we learned in HMM
  - But use the framework of Maximum Entropy for features and normalization

Maximum Entropy Markov Models

HMM



MEMM



Figures from [2]

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

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

- 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

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

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

Viterbi in MEMM

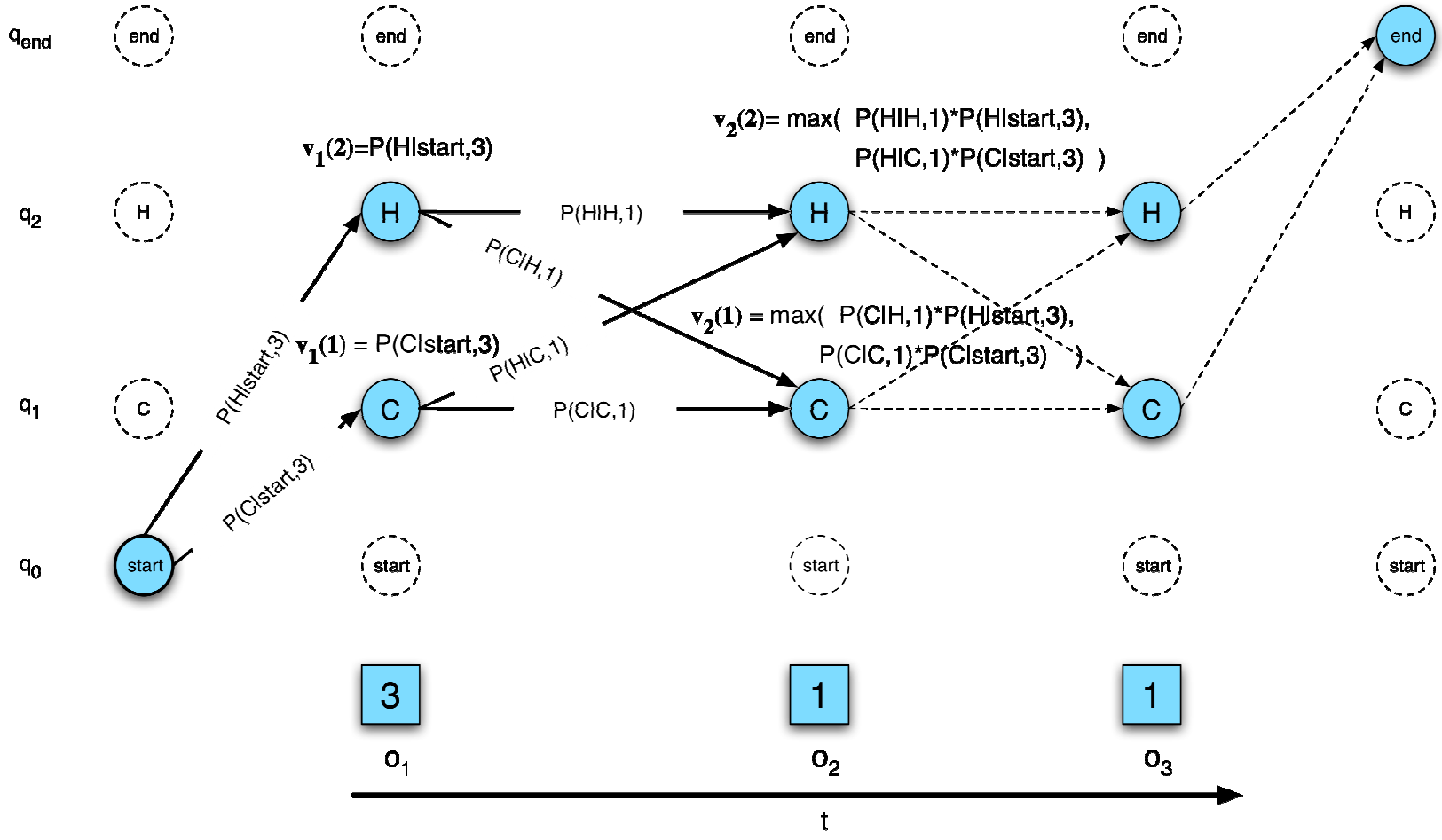


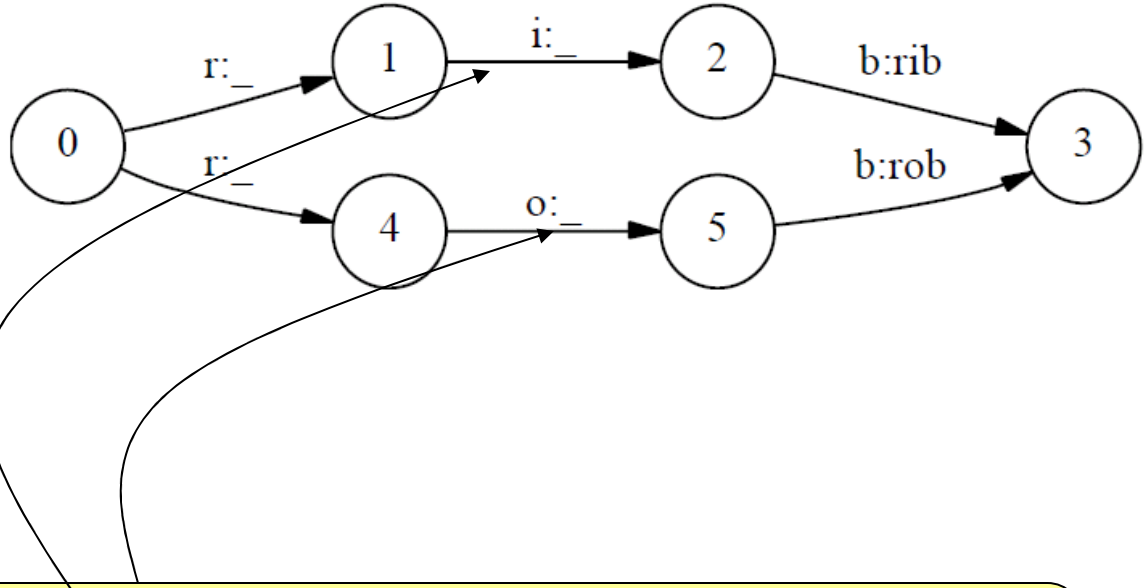
Figure from [2]

Label Bias Problem in MEMM

- We saw that with markov assumption but with transitions conditioned on both state and observation where transitions are computed based on exponential model on state allowed us to do Viterbi Inference in MEMM
- There is a weakness in MEMM though
  - "Label Bias Problem"
  - Transitions from a given state are competing against each other only
  - Per state normalization, i.e. sum of transition probability for any state has to sum to 1
  - Causes bias: states with fewer arcs are preferred
  - What happens if there is only one outgoing arc? Does it matter what the observation is?

Label Bias Example

In generative model we could have learned
state 1 generates i lot more and state 4 generates
o lot less thus preferring 1-2 path more



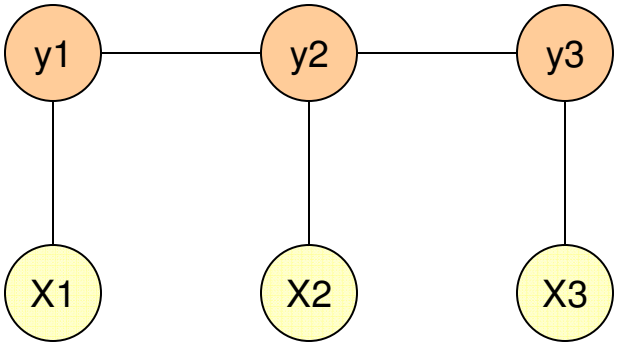
Even if the training data may have seen
a lot more ri example vs ro example state 1
and state 4 passes all probability mass to next state

MEMMs to Conditional Random Fields

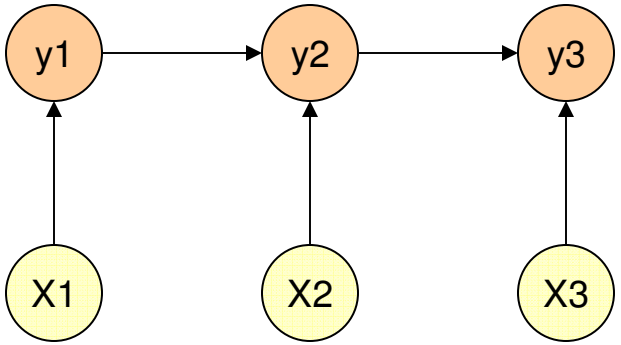
- MEMM
  - We have exponential model for each state to tell us the conditional probability of the next states

- CRF
  - No per state normalization
  - Per sequence normalization

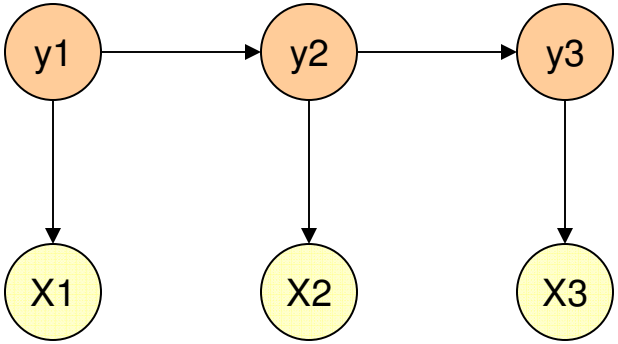
CRF, MEMM, HMM



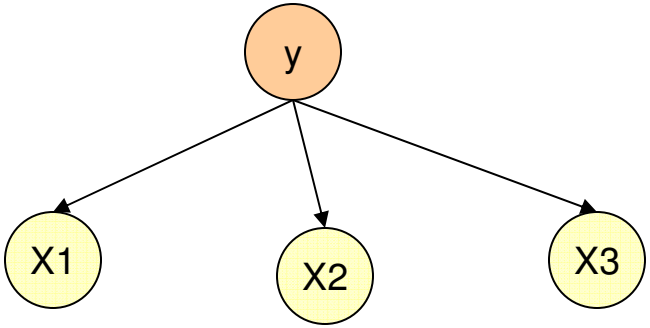
CRF



MEMM

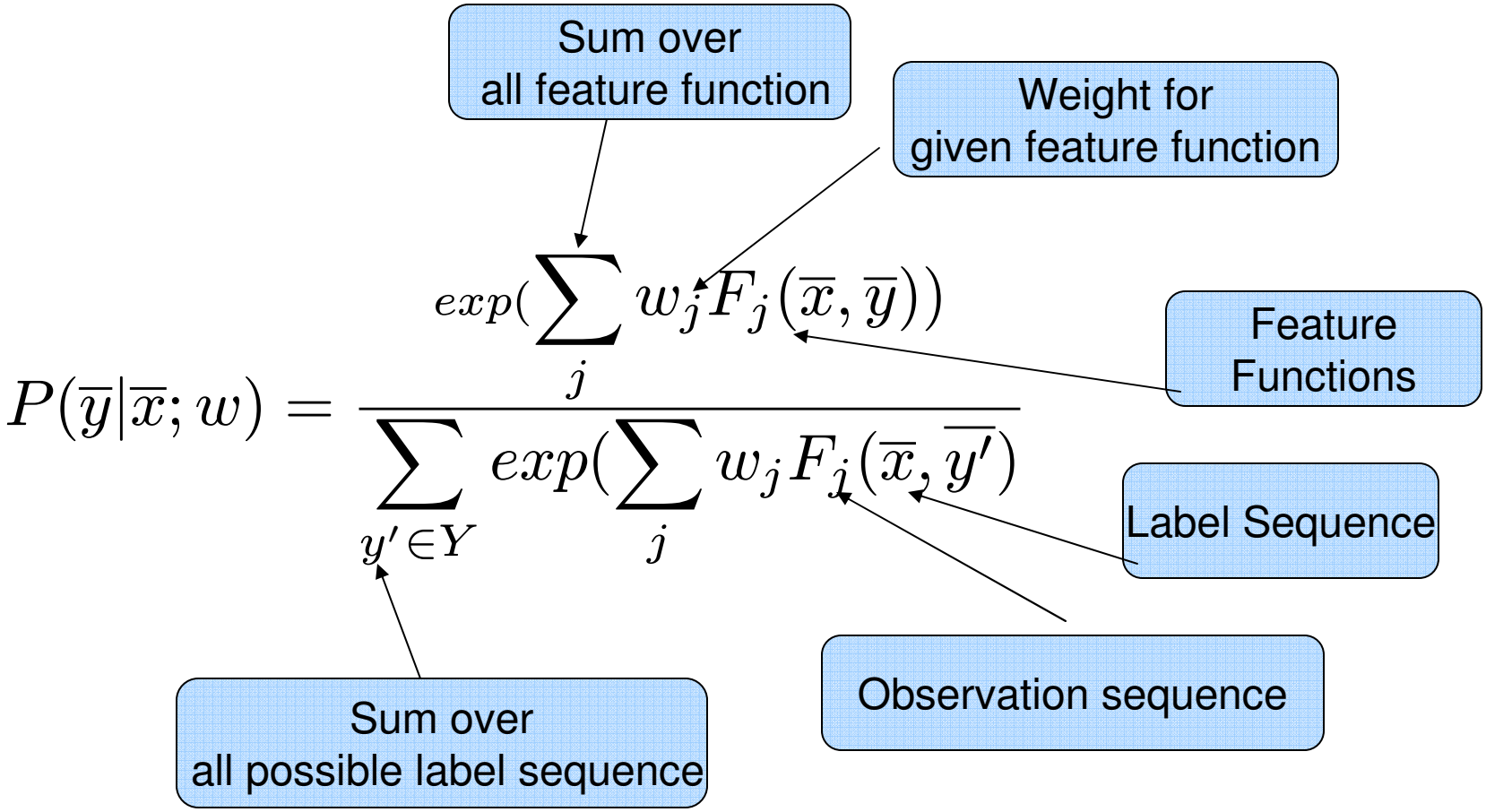


HMM



Naïve Bayes

Conditional Random Field



Model log linear on Feature functions

Feature Function

$$\mathrm{F}_j(\overline{x}, \overline{y})$$

- Entire sequence of observation and label are paired
- Each feature function Fj is sum along the sentence i = 1 to n where n is length of the sentence
- Assuming binary class labels and a training example of length n
  - sequence y can be any of the 2^n possible sequence (state space huge)
  - Computing denominator is worse as sum is across all 2^n state space
- How to address this?

Feature Function

$$\text{We get } p(\overline{y}|\overline{x};w) = \frac{1}{Z(\overline{x},w)} exp \sum_j w_j F_j(\overline{x},\overline{y})$$

$$\text{where} Z(\overline{x},w) = \sum_{y' \in Y} exp \sum_j w_j F_j(\overline{x},\overline{y'})$$

Costly operation in CRF

Feature Function Markov Assumption

$$\text{Let } F_j(\overline{x}, \overline{y}) = \sum_i f_j(y_{i-1}, y_i, \overline{x}, i)$$

$$\text{We get } p(\overline{y}|\overline{x}; w) = \frac{1}{Z(\overline{x}, w)} exp \sum_j w_j F_j(\overline{x}, \overline{y})$$

$$\text{where} Z(\overline{x}, w) = \sum_{y' \in Y} exp \sum_j w_j F_j(\overline{x}, \overline{y'})$$

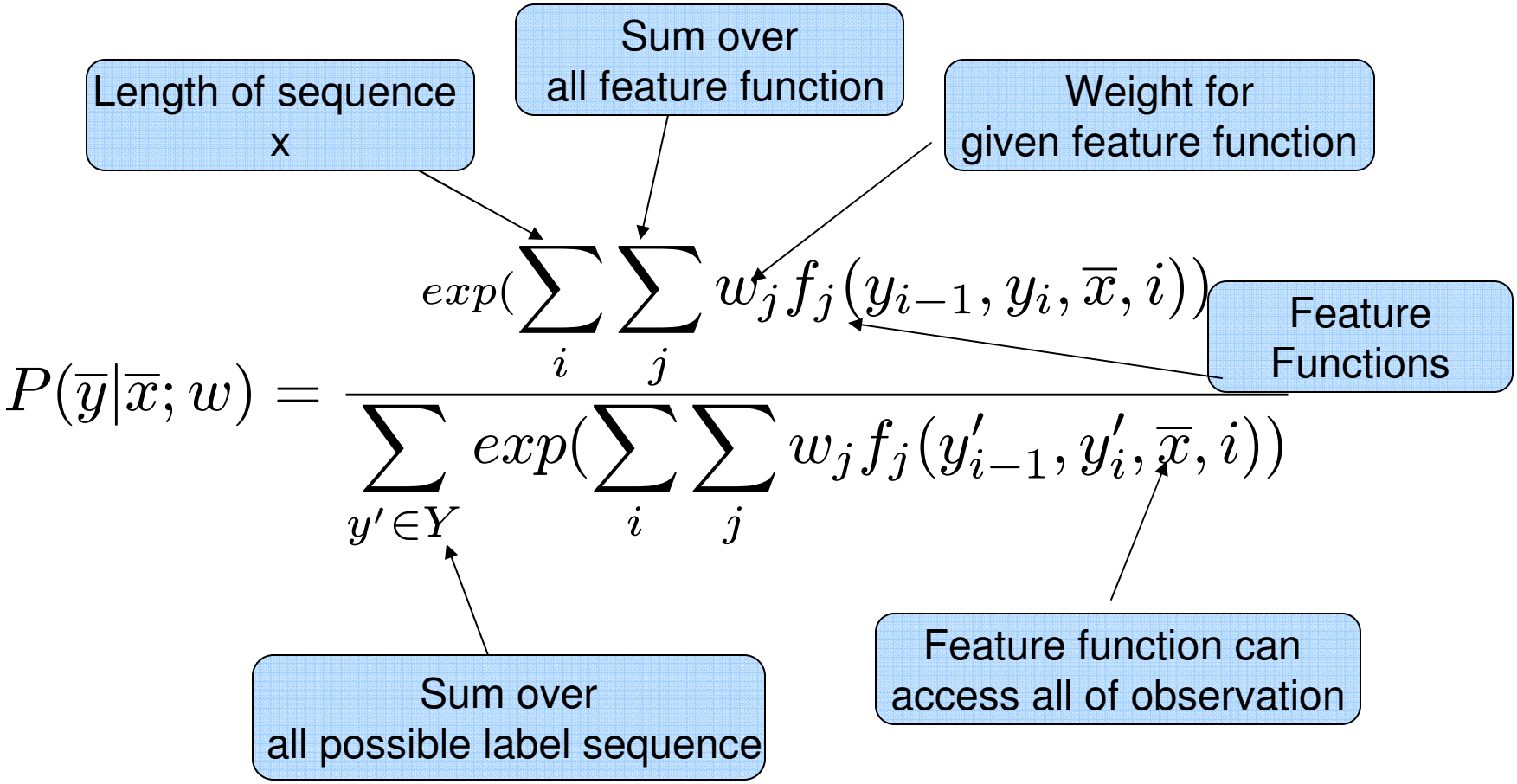
Feature Function Markov Assumption

$$\text{Let } F_j(\overline{x}, \overline{y}) = \sum_i f_j(y_{i-1}, y_i, \overline{x}, i)$$

$$\text{We get } p(\overline{y}|\overline{x}; w) = \frac{1}{Z(\overline{x}, w)} exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \overline{x}, i))$$

$$\text{where} Z(\overline{x}, w) = \sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \overline{x}, i)))$$

Conditional Random Field



Model log linear on Feature functions

Hyphenation Feature Function Example

$$\text{Let } F_j(\overline{x}, \overline{y}) = \sum_i f_j(y_{i-1}, y_i, \overline{x}, i)$$

Sum of low level feature function becomes a real value

$$f_{18}(y_{i-1}, y_i, \overline{x}, i) = y_{i-1}, y_i = 00 \& x_{i-1} x_i = th$$

- Example for hyphenation
- Two consecutive tags are not hyphens and letters are ‘th’
- If 0 means no hyphen after the letter, Positive weight means no hyphen after t and after h
- Can you look at multiple y beyond 2?
- How about adjacency on y?

3 Problems

- Inference

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

- Need to compute normalization constant to compute probabilities

- Parameter Estimation

Inference Problem

- Inference

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

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

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?

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

What data structure to represent gi ?

g function is still dependent on y(i-1) and y(i)
i as a subscript, specific g function for specific i
Maximization for specific x and w, so dropped from notation

Score of Tag Sequence

$$score(y_1, , y_n) = \sum_i^n g_i(y_{i-1}, y_i)$$

Score of tag sequence of length n

U(k) = score of best sequence y1.. Yk
U(k,v) = score of best sequence y1 .. Yk and yk= v

Recurrence Relation for CRF

Let $U(k, v)$ score of best sequence from tags position 1 to K

Where tag number K needs to be v

With this definition maximization of U(k,v) is maximization over k-1 tags because tag K is already fixed as v

$$U(k,v) = max_{y_1,y_2,\ldots,y_{k-1}}[\sum_{i=1}^{k-1} g_i(y_{i-1}, y_i) + g_k(y_{k-1}, v)]$$

$$U(k,v) =$$

$$max_{y_1,y_2,\ldots,y_{k-2}}[\sum_{i=1}^{k-2} g_i(y_{i-1}, y_i) + g_{k-1}(y_{k-2}, y_{k-1}) + g_k(y_{k-1}, v)]$$

$$U(k,v) = max_{y_{k-1}}[U(k-1, y_{k-1}) + g_k(y_{k-1}, v)]$$

Runtime? What was runtime for HMM viterbi?

Inference in CRF

- Our inference algorithm is again based on Viterbi algorithm
- Output transition and observation probabilities are not modeled separately
- Output transition dependent on the state and the observation as one conditional probability
- Build lattice like we did before for decoding

Parameter Estimation for CRF

- Mostly supervised learning
- No EM like we have for HMM
- Introduction of hidden variable makes the problem very hard
- Can be interpreted in maximum entropy framework

Conditional Likelihood

- Given the data we want to maximize the conditional likelihood
- Like we have done previously we can set the derivative of the conditional likelihood function to zero

$$p(\overline{y}|\overline{x};w) = \frac{1}{Z(\overline{x},w)} exp \sum_j w_j F_j(\overline{x},\overline{y})$$

$$L(w,D) = log(\prod_{k=1}^{m} p(\overline{y}^k|\overline{x}^k,w))$$

Taking the Gradient

$$= F_j(x,y) - \frac{1}{Z(x,w)} \sum_{y'} \frac{\partial}{\partial w_j} exp \sum_{j'} w_{j'} F_{j'}(x,y')$$

$$= F_j(x,y) - \sum_{y'} F_j(x,y') p(y'|x;w)$$

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

Derivative w.r.t to the weight wi = (value of feature function i for true y) – (average value of feature function for all possible y')

Maximum Entropy Interpretation

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

Empirical count

Predicted count

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

Most computationally expensive part

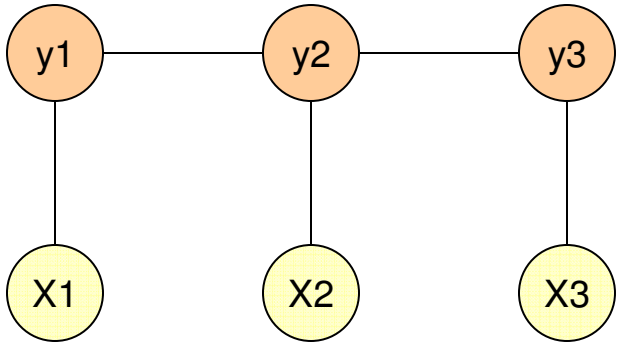
Optimization Methods

- Iterative Scaling
- Gradient Descent
- Newton's Method
- Many optimization packages are available that can treated as blackbox replacement
- Second order methods have shown to be faster

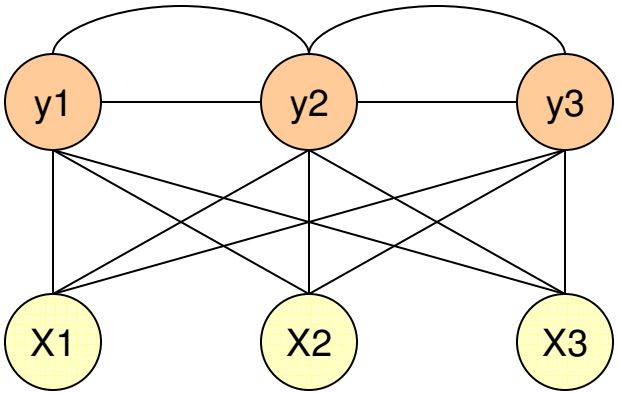
General CRF

- We just showed how to do inference and training of linear structure CRF
- But we can add varying level of dependencies across states
- Training more complicated if we add more dependencies
- We can use more general graphical model approximate algorithms such as belief propagation

General CRF



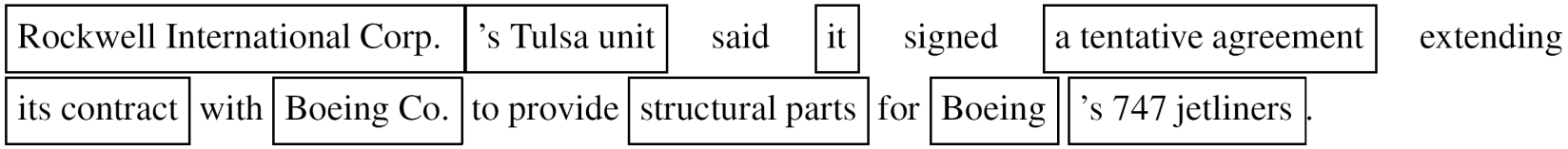
Linear Chain CRF



More General CRF

CRF can be used trees, graphs, etc but can be expensive to train the model

CRF for Shallow Parsing



- [Sha and Pereira, 2004]
- NP chunking
- Used many overlapping features that took account of word interaction

Model	F score
SVM combination (Kudo and Matsumoto, 2001)	94.39%
CRF	94.38%
Generalized winnow (Zhang et al., 2002)	93.89%
Voted perceptron	94.09%
MEMM	93.70%

References

- [1] Sutton, C and McCallum A “An Introduction to Conditional Random Fields for Relational Learning”
- [2] Jurafsky, D and Martin, J, “Speech and Language Processing,” 2009
- [3] Elkan, C “Log-linear Models and Conditional Random Fields” CIKM Tutorial 2008