Sequence labeling and hands on

Kan Min-Yen

Day 3 / Morning

(partial slide credits: Inna Weiner, Rongkun Shen, Jie Tang)

Recap

- **Information Retrieval**
  - Weighting words with respect to global and local importance
  - Represent both docs and queries as vectors

- **Introduced NLP as a machine learning problem**
- **Casts the problem as annotation and feature engineering**
  - Annotation requires clear policy and guidelines
  - Evaluation to assess performance and identify sources of error for improvement

Day Outline

Day 1	Day 2	>>Day 3
AM	**AM**	**AM**
– Applications' Input / Output – Resources	– Evaluation – Annotation – Information Retrieval – ML Intro	– Sequence Labeling – CRF++ Hands-on
PM	**PM**	**PM**
– Selected Toolkits – Python Intro – NLTK Hands-on	– Machine Learning – SVM Hands-on	– Dimensionality Reduction – Trends & Issues

Sequence Labeling Models

- **HMM**
  - –Generative model
  - –E.g. Ghahramani (1997), Manning and Schutze (1999)
- **MEMM**
  - –Conditional model
  - –E.g. Berger and Pietra (1996), McCallum and Freitag (2000)
- **CRFs**
  - –Conditional model without label bias problem
  - –Linear-Chain CRFs
    - E.g. Lafferty and McCallum (2001), Wallach (2004)
  - –Non-Linear Chain CRFs
    - Modeling more complex interaction between labels: DCRFs, 2D-CRFs
    - E.g. Sutton and McCallum (2004), Zhu and Nie (2005)

Labeling Sequence Data Problem

- **X is a random variable over data sequences**
- **Y is a random variable over label sequences**
- **Y_i is assumed to range over a finite label alphabet A**
- **The problem:**
  - –Learn how to give labels from a closed set Y to a data sequence **X**

	x_1	x_2	x_3
X:	Thinking	is	being
Y:	noun	verb	noun
	y_1	y_2	y_3

Generative Probabilistic Models

- Learning problem:

  Choose Θ to maximize *joint likelihood*:

$$L(\Theta) = \Sigma \log p_{\Theta}(y_i, x_i)$$

- The goal: maximization of the joint likelihood of training examples

  $y = \text{argmax } p^*(y|x) = \text{argmax } p^*(y,x)/p(x)$

  Joint likelihood
- Needs to enumerate all possible observation sequences

Markov Model

A Markov process or model assumes that we can predict the future based just on the present (or on a limited horizon into the past):

Let $\{X_1,\ldots,X_T\}$ be a sequence of random variables taking values $\{1,\ldots,N\}$ then the Markov properties are:

1. Limited Horizon:

$$P(X_{t+1}|X_1,\ldots,X_t) = P(X_{t+1}|X_t) =$$

2. Time invariant (stationary):

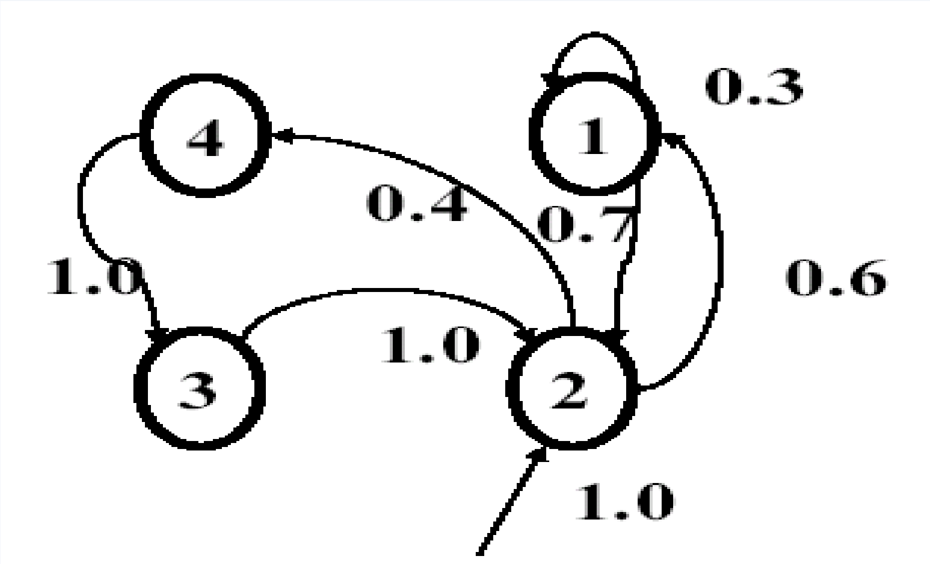
$$= P(X_2|X_1)$$

Describing a Markov Chain

Markov Chains can be described by the transition matrix **A** and the initial (start) probabilities **Q**:

$A_{ij} = P(X_{t+1}=j|X_t=i)$

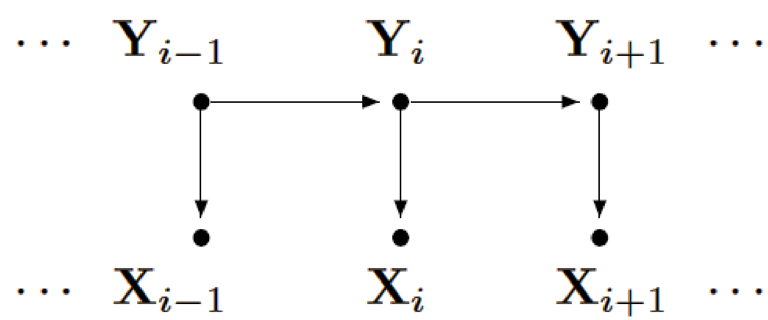
$q_i = P(X_1=i)$



Hidden Markov Model

- Do not observe the sequence that the model passes through (**X**) but only some probabilistic function of it (**Y**). Thus, it is a Markov model with the addition of *emission probabilities*:

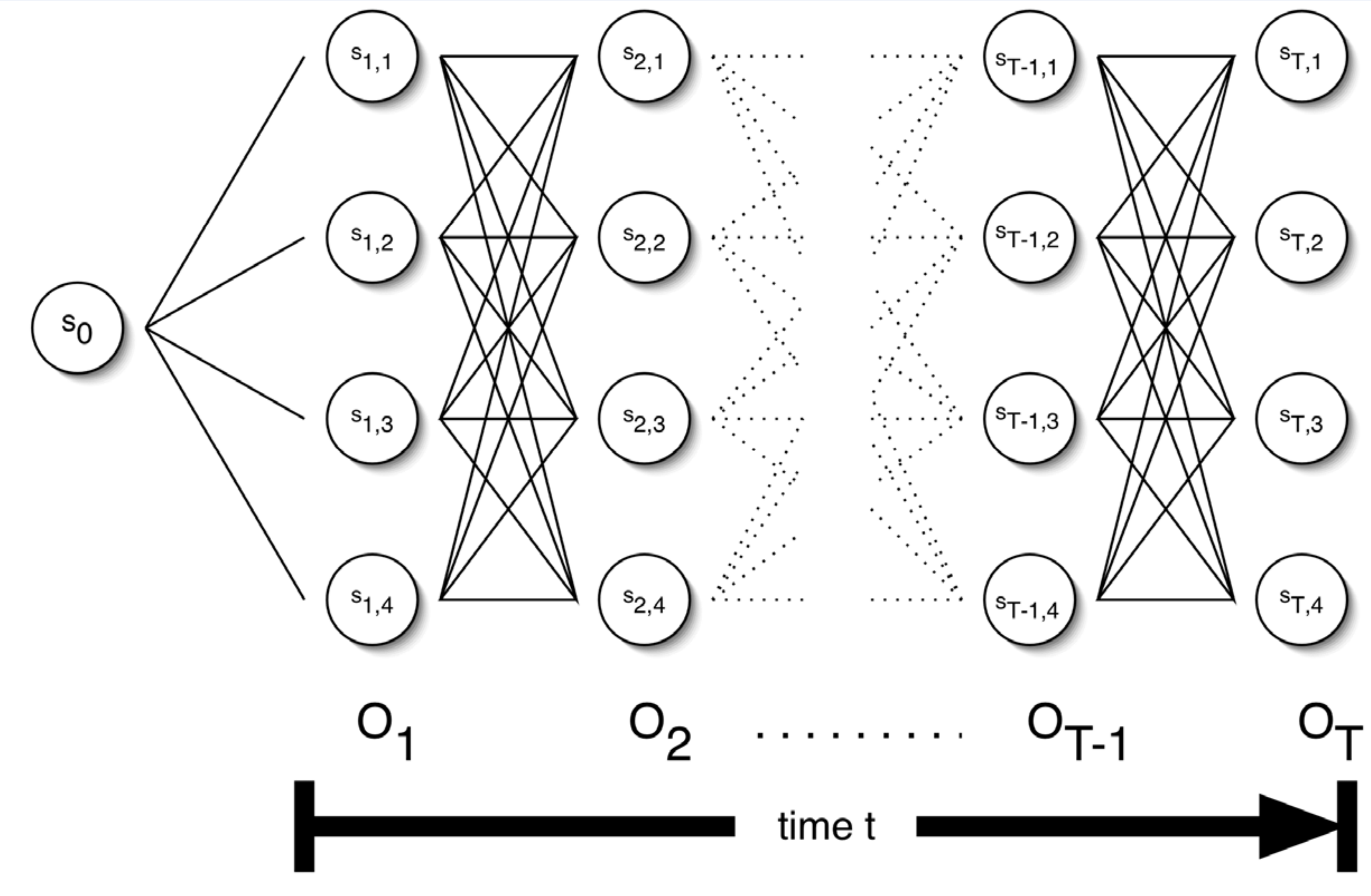
Standard tool is the hidden Markov Model (HMM).



$$P(\mathbf{X}, \mathbf{Y}) = \prod_i P(\mathbf{X}_i \mid \mathbf{Y}_i)\, P(\mathbf{Y}_i \mid \mathbf{Y}_{i-1})$$

Hidden = Latent

The Trellis


s0
s1,1
s1,2
s1,3
s1,4
s2,1
s2,2
s2,3
s2,4
sT-1,1
sT-1,2
sT-1,3
sT-1,4
sT,1
sT,2
sT,3
sT,4
o1
o2
oT-1
oT
time t

The Three Problems in HMMs

- **Likelihood/Evaluation:** Given a series of observations **y** and a model **λ = {A,B,q}**, compute the likelihood **p(y| λ)**
  >> Forward Algorithm

- **Inference/Decoding:** Given a series of observations **y** and a model **λ = {A,B,q}**, compute the most likely sequence of hidden states **x**
  >> Viterbi Algorithm (like forward algorithm but just do max instead of sum)

- **Learning:** Given a series of observations, learn the best model **λ**
  >> Forward-Backward Algorithm (Baum Welch)
  (Iterative algorithm to re-estimate parameters, like EM)

Likelihood in HMMs

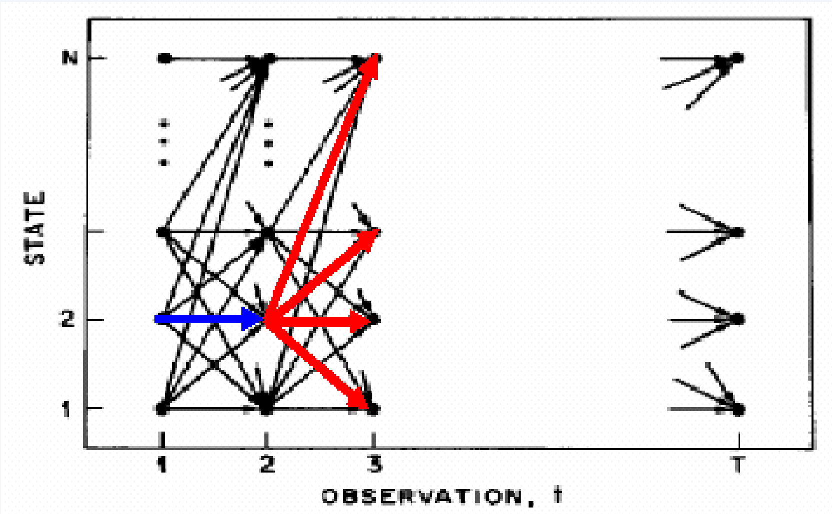
- Given a model **λ = {A,B,q}**, we can compute the likelihood by

$$\begin{aligned} \mathbf{P(y)} &= \mathbf{p(y|\,\lambda)} \\ &= \mathbf{\Sigma\, p(x)p(y|x)} \\ &= \mathbf{q(x_1)\ \Pi A(x_{t+1}|x_t)\ \Pi B(y_t|x_t)} \end{aligned}$$

- But … this computation complexity is $O(N^T)$, when $|x_i| = N$ → impossible in practice

Forward-Backward algorithm

- **To compute likelihood:**
  - Need to enumerate over all paths in the lattice (all possible instantiations of $X_1 \ldots X_T$). But … some starting subpath (blue) is common to many continuing paths (blue+red)

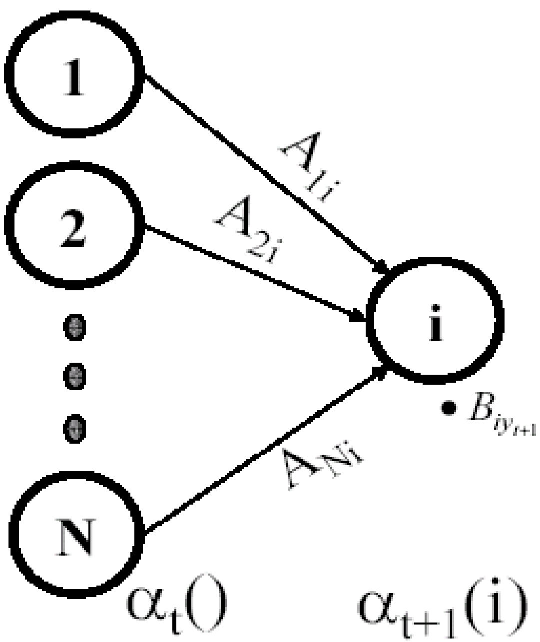


The idea:

Use **dynamic programming**, calculate a path in terms of shorter sub-paths

Forward-Backward algorithm (cont'd)

- We build a matrix of the probability of being at time t at state i: $\boldsymbol{\alpha_t(i) = P(x_t=i, y_1\, y_2 \ldots y_t)}$. This is a function of the previous column (forward procedure):



$$\alpha_1(i) = q_i B_{jy_1}$$

$$\alpha_{t+1}(i) = B_{iy_{t+1}} \sum_{j=1}^{N} \alpha_t(j) A_{ji}$$

$$P(Y) = \sum_{i=1}^{N} \alpha_T(i)$$

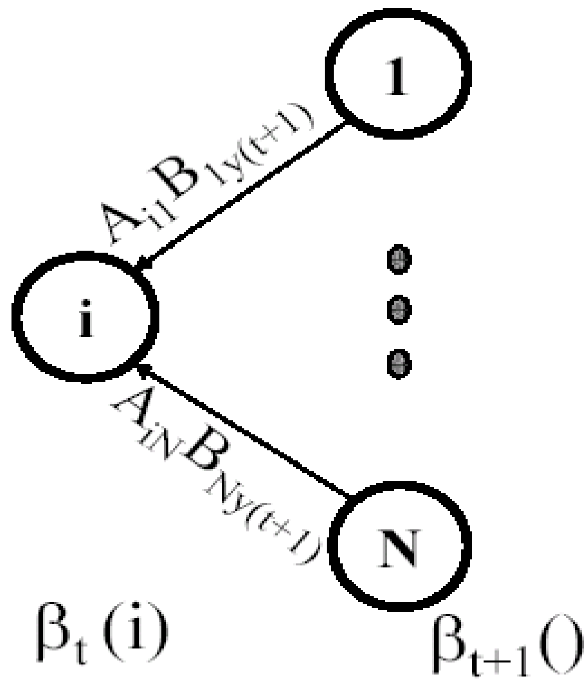
Forward-Backward algorithm (cont'd)

We can similarly define a backwards procedure for filling the matrix $\boldsymbol{\beta_t(i) = P(y_{t+1} \ldots y_T | x_t = i)}$

$$\beta_T(i) = 1$$

$$\beta_t(i) = \sum_{j=1}^{n} A_{ij} B_{jy_{t+1}} \beta_{t+1}(j)$$

$$P(Y) = \sum_{i=1}^{N} q_i B_{iy_1} \beta_1(i)$$



Combine both …

- Combine both processes to arrive at likelihood:

$$P(y,x_t=i) = P(x_t=i,y_1y_2\ldots y_t)^* \; P(y_{t+1}\ldots y_T|x_t=i)$$
$$= \alpha_t(i)\ \beta_t(i)$$

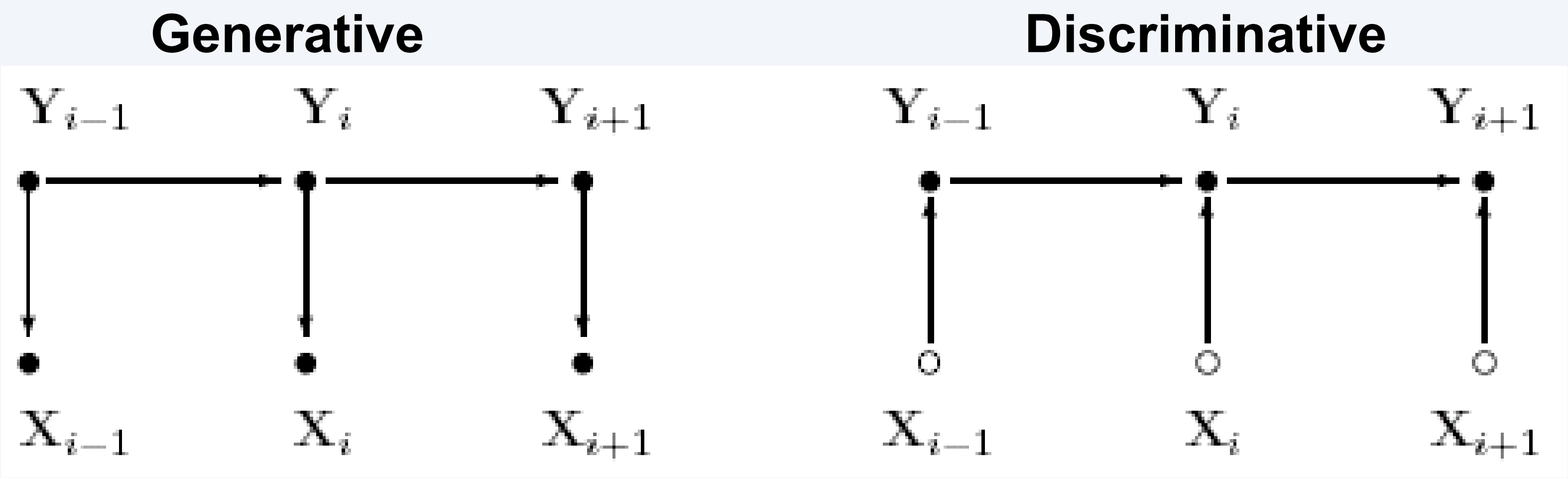
- And then we get:

$$P(y) = \Sigma\ P(y,x_t=i) = \Sigma\ \alpha_t(i)\ \beta_t(i)$$

HMM Summary

- **Advantages:**
  - – Estimation very easy
  - – Closed form solution
  - – The parameters can be estimated with relatively high confidence from small samples
- **But:**
  - – The model represents all possible (x,y) sequences and defines **joint probability** over all possible observation and label sequences
  - – Need to enumerate all possible observation sequences
  - – Impossible to represent multiple interacting features
  - – Difficult to model long-range dependencies of the observations
  - – Very strict independence assumptions on the observations

Discriminative Probabilistic Models



"Solve the problem you need to solve":

The traditional approach inappropriately uses a generative *joint* model in order to solve a *conditional* problem in which the observations are given.

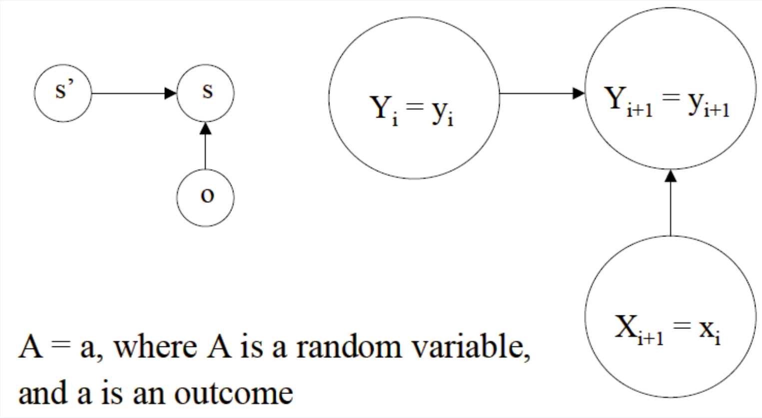
To classify we actually need **p(y|x)** – there's no need to implicitly approximate **p(x,y)**.

Discriminative / Conditional Models

- Conditional probability **P(label seq y | observed seq x)** rather than joint probability P(y, x)
  - Specify the probability of possible label sequences given an observation sequence
- Allow arbitrary, non-independent features on the observation sequence X
- The probability of a transition between labels may depend on **past** and **future** observations
  - Relax strong independence assumptions in HMM

Maximum Entropy Markov Models (MEMMs)

a.k.a. Conditional Markov Models CMMs



- **Models probability of a state given an observation and just the previous state**
  - Conditional probs are represented as exponential models based on arbitrary observation features

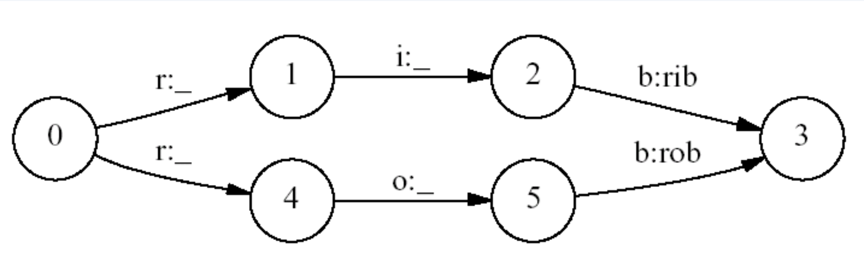
- **Given training set X with label sequence Y:**
  - Train a model θ that maximizes P(Y|X, θ)
  - For a new data sequence x, predict label y that maximizes P(y|x, θ)

$$P(y' \mid y, x) = \frac{1}{Z(y,x)} \exp\left(\sum_k \underbrace{\lambda_k}_{\text{weight}} \underbrace{f_k(x, y, y')}_{\text{feature}} \right)$$

Per state normalization: all prob mass that arrives is distributed among its successor states

The Label Bias Problem

- In MaxEnt's formulation, the prob mass that arrives at the state must be distributed among the possible successor states



- If one of transitions leaving state 0 occurs more frequently in training, its transition prob is greater, irrespective of the observation sequence
  - Especially in cases where there are few outgoing transitions (as in states 1, 2, 4 and 5).
- In the example, say that 'rib' is slightly more common that 'rob' in the training data. Then in the test data, if 'rob' occurs it will be classified as 'rib' as the the transition to 1 is more likely than to 4; the observation of the 'o' is effectively ignored as that it is only observed later at state 1.

Conditional Random Fields (CRFs)

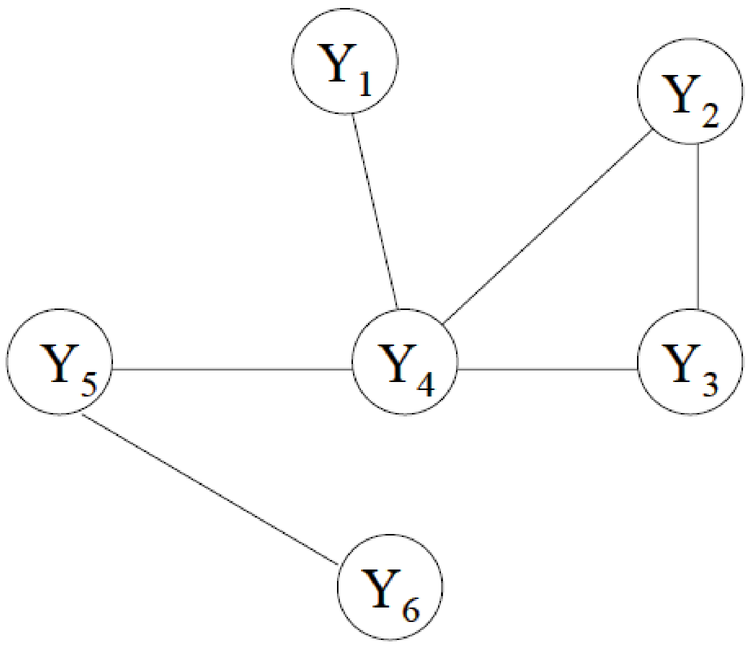
- **CRFs have all the advantages of MEMMs without label bias problem**
  - –MEMM uses **per-state exponential** model for the conditional probabilities of next states given the current state
  - –CRF has a **single exponential** model for the joint probability of the entire sequence of labels given the observation sequence
  - – This difference means that some transitions have more influence than others depending on the corresponding observations in our previous example
- **Undirected acyclic graph**

Random Fields – Undirected Graphical Models

Let $G = (Y, E)$ be a graph where each vertex Y_v is a random variable

Suppose $P(Y_v \mid \text{all other } Y) = P(Y_v \mid \text{neighbors}(Y_v))$ then Y is a random field

Example:



- $P(Y_5 \mid \text{all other } Y) = P(Y_5 \mid Y_4, Y_6)$

Conditional Random Field: Definition

- **X – random variable over data sequences**
- **Y – random variable over label sequences**
- **Y_i is assumed to range over a finite label alphabet A**

- **Discriminative approach:**
  – We construct a conditional model p(y|x) and do not explicitly model marginal p(x)

CRF Distribution Function

$$p_\theta(\mathrm{y} \mid \mathrm{x}) = \frac{1}{Z(\mathrm{x})} \exp\left(\sum_{e \in E, k} \lambda_k f_k(e, \mathrm{y}|_e, \mathrm{x}) + \sum_{v \in V, k} \mu_k g_k(v, \mathrm{y}|_v, \mathrm{x}) \right)$$

Where :

V = Set of random variables (observed and hidden)
f_k and g_k = features
g_k = State feature
f_k = Edge feature

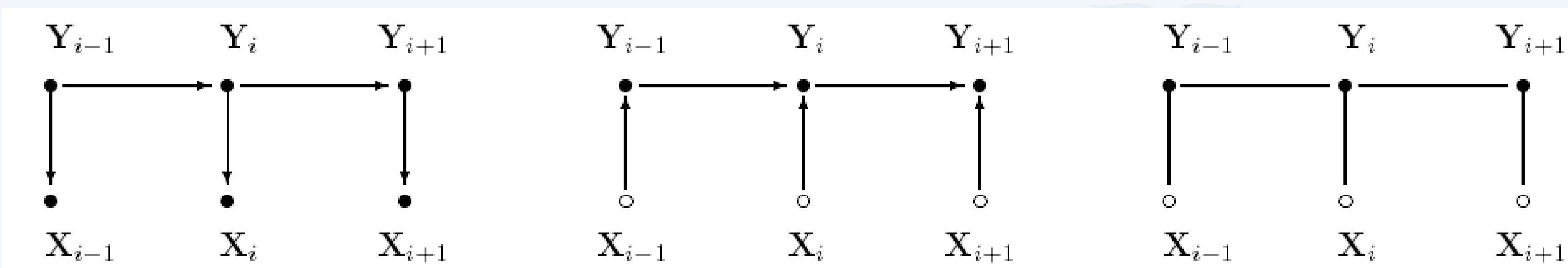
$\theta = (\lambda_1, \lambda_2, \cdots, \lambda_n; \mu_1, \mu_2, \cdots, \mu_n)$; λ_k and μ_k

are parameters to be estimated
y|$_e$ = Set of components of y defined by edge *e*
y|$_v$ = Set of components of y defined by vertex *v*

CRF on the linear chain graph

- **We will handle the case when G is a simple chain: G = (V = {1,…,m}, E={ (I,i+1) })**

HMM (Generative) MEMM (Discriminative) CRF

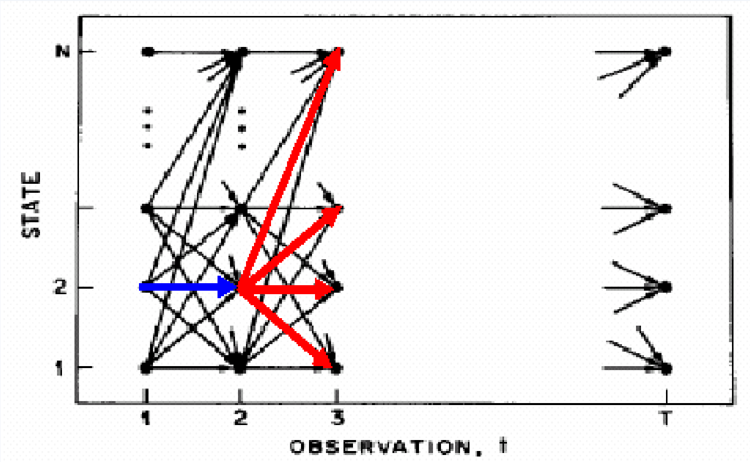


CRF – the Learning Problem

- **Assumption: the *features* f_k and g_k are given and fixed.**
  – For example, a boolean feature g_k is TRUE if the word X_i is upper case and the label Y_i is a “noun”.
- **The learning problem**
  – We need to determine the parameters $\Theta = (\lambda_1, \lambda_2, . . . ;$ μ1, μ2, . . .) from training data D = {(x(i), y(i))} with empirical distribution p˜(x, y).

Parameter Estimation for CRFs

- The parameter vector Θ that maximizes the log-likelihood is found using a iterative scaling algorithm.
- We define standard HMM-like forward and backward vectors α and β, which allow polynomial time calculations.



- However as the normalization is conditioned over the entire CRF (not over single vertices), it is expensive to compute → slow training time

Experiment Validation of the models

- Part-of-speech (POS) tagging experiments

model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%

MEMM^{+}	4.81%	26.99%
CRF^{+}	4.27%	23.76%

$^{+}$Using spelling features

Summary

- **HMM:**
  - – Basic sequence labeling framework where labels are considered hidden and generate the observed words, and are just dependent on the previous state (Markovian assumption)
  - – Fast algorithms for three classic problems

- **CRF:**
  - – Discriminatively trained models P(y|x) as compared to modeling joint P(x,y) probability
  - – Allows combination of arbitrary and overlapping observation features from both the past and future
  - – main current limitation is the slow convergence of the training algorithm relative to MEMMs or HMMs, for which training is efficient.

Hands on with CRF++

Reference string labeling over the Cora dataset

NUS National University of Singapore

Cora Dataset

- 200 reference strings taken from articles in Computer Science
- Labeled with fields in XML style

Isaac G. Councill, C. Lee Giles, Min-Yen Kan. (2008) *ParsCit: An open-source CRF reference string parsing package.* To appear in the proceedings of the Language Resources and Evaluation Conference (LREC 08), Marrakesh, Morrocco, **May**.

CRF++

- Implementation of CRFs in C++
- Built with multithreading
- Generates individual binary feature functions from feature templates (each which describe a class of features)
- Uses `conlleval.pl` script to assess performance

NUS National University of Singapore

Six Steps

1. Convert data to CRF++ format
2. Create basic data and template file
3. Create basic word features
4. Integrate lexicon features
5. Error analysis – Inspect results more closely
6. Create punctuation, numeric features
7. Create global features