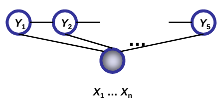
Machine Learning

10-701/15-781, Fall 2011

Conditional Random Fields

Eric Xing

Lecture 12, October 19, 2011



Reading: Chap. 13 CB

Definition (of HMM)

- Observation space
  - **Alphabetic set:** $\mathbb{C} = \{c_1, c_2, \cdots, c_K\}$
  - **Euclidean space:** $\mathbb{R}^d$
- Index set of hidden states

  $\mathrm{I} = \{1, 2, \cdots, M\}$
- Transition probabilities between any two states

  $p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j},$

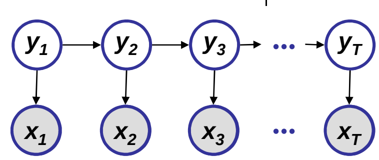
  or $\quad p(y_t \mid y_{t-1}^i = 1) \sim \text{Multinomial}(a_{i,1}, a_{i,2}, \ldots, a_{i,M}), \forall i \in \mathrm{I}.$
- Start probabilities

  $p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$
- Emission probabilities associated with each state

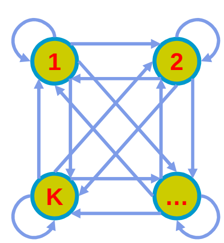
  $p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathrm{I}.$

  or in general:

  $p(x_t \mid y_t^i = 1) \sim \mathrm{f}(\cdot \mid \theta_i), \forall i \in \mathrm{I}.$



Graphical model



State automata

Three Main Questions on HMMs

1. Evaluation

   GIVEN an HMM ***M***, and a sequence ***x***,
   FIND Prob (***x*** | ***M***)
   ALGO. Forward

2. Decoding

   GIVEN an HMM ***M***, and a sequence ***x*** ,
   FIND the sequence **y** of states that maximizes, e.g., P(***y*** | ***x*** , ***M***),
   or the most probable subsequence of states
   ALGO. Viterbi, Forward-backward

3. Learning

   GIVEN an HMM ***M***, with unspecified transition/emission probs.,
   and a sequence ***x***,
   FIND parameters $\theta = (\pi_i, a_{ij}, \eta_{ik})$ that maximize $P(x \mid \theta)$
   ALGO. Baum-Welch (EM)

Learning HMM: two scenarios

- **Supervised learning**: estimation when the "right answer" is known
  - **Examples:**

    GIVEN: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls
- **Unsupervised learning**: estimation when the "right answer" is unknown
  - **Examples:**

    GIVEN: the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition

    GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice
- **QUESTION:** Update the parameters θ of the model to maximize $P(x|\theta)$ --- Maximal likelihood (ML) estimation

MLE

Supervised ML estimation

- Given $x = x_1 \ldots x_N$ for which the true state path $y = y_1 \ldots y_N$ is known,
  - **Define:**

    A_{ij} = # times state transition $i \rightarrow j$ occurs in $\mathbf{y}$

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$

  - **We can show that the maximum likelihood parameters θ are:**

$$a_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}$$

$$b_{ik}^{ML} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T} y_{n,t}^i} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

  - **What if y is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian …**

Supervised ML estimation, ctd.

- **Intuition:**
  - When we know the underlying states, the best estimate of θ is the average frequency of transitions & emissions that occur in the training data
- **Drawback:**
  - Given little data, there may be **overfitting**:
    - $P(x|\theta)$ is maximized, but θ is unreasonable

      0 probabilities – VERY BAD
- **Example:**
  - Given 10 casino rolls, we observe

    `x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3`

    `y = F, F, F, F, F, F, F, F, F, F`
  - Then: $a_{FF} = 1$; $a_{FL} = 0$

    $b_{F1} = b_{F3} = .2$;

    $b_{F2} = .3$; $b_{F4} = 0$; $b_{F5} = b_{F6} = .1$

Pseudocounts

- Solution for small training sets:
  - Add pseudocounts

    A_{ij} = # times state transition $i \rightarrow j$ occurs in $\mathbf{y}$ + R_{ij}

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$ + S_{ik}
  - R_{ij}, S_{ij} are pseudocounts representing our prior belief
  - Total pseudocounts: $R_i = \Sigma_j R_{ij}$, $S_i = \Sigma_k S_{ik}$,
    - --- "strength" of prior belief,
    - --- total number of imaginary instances in the prior
- Larger total pseudocounts $\Rightarrow$ strong prior belief
- Small total pseudocounts: just to avoid 0 probabilities --- smoothing

Unsupervised ML estimation

Unsupervised ML estimation

- Given $x = x_1 \dots x_N$ for which the true state path $y = y_1 \dots y_N$ is unknown,

  - <u>**EXPECTATION MAXIMIZATION**</u>

0. Starting with our best guess of a model M, parameters θ:
1. Estimate A_{ij} , B_{ik} in the training data
   - How? $A_{ij} = \sum_{n,t} \langle y^i_{n,t-1} y^j_{n,t} \rangle$ $B_{ik} = \sum_{n,t} \langle y^i_{n,t} \rangle x^k_{n,t}$,
   - Update θ according to A_{ij} , B_{ik}
   - Now a "supervised learning" problem
2. Repeat 1 & 2, until convergence

This is called the Baum-Welch Algorithm

We can get to a provably more (or equally) likely parameter set θ each iteration

The Baum Welch algorithm

- The complete log likelihood

$$\ell_c(\boldsymbol{\theta};\mathbf{x},\mathbf{y}) = \log p(\mathbf{x},\mathbf{y}) = \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t}) \right)$$

- The expected complete log likelihood

$$\langle \ell_c(\boldsymbol{\theta};\mathbf{x},\mathbf{y}) \rangle = \sum_n \left(\langle y_{n,1}^i \rangle_{p(y_{n,1}|\mathbf{x}_n)} \log \pi_i \right) + \sum_n \sum_{t=2}^{T} \left(\langle y_{n,t-1}^i y_{n,t}^j \rangle_{p(y_{n,t-1},y_{n,t}|\mathbf{x}_n)} \log a_{i,j} \right) + \sum_n \sum_{t=1}^{T} \left(x_{n,t}^k \langle y_{n,t}^i \rangle_{p(y_{n,t}|\mathbf{x}_n)} \log b_{i,k} \right)$$

- EM
  - The **E** step

$$\gamma_{n,t}^i = \langle y_{n,t}^i \rangle = p(y_{n,t}^i = 1 \mid \mathbf{x}_n)$$

$$\xi_{n,t}^{i,j} = \langle y_{n,t-1}^i y_{n,t}^j \rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1 \mid \mathbf{x}_n)$$

  - The **M** step ("symbolically" identical to MLE)

$$\pi_i^{ML} = \frac{\sum_n \gamma_{n,1}^i}{N} \qquad a_{ij}^{ML} = \frac{\sum_n \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \qquad b_{ik}^{ML} = \frac{\sum_n \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i}$$

The Baum-Welch algorithm -- comments

Time Complexity:

iterations × O(K²N)

- Guaranteed to increase the log likelihood of the model
- Not guaranteed to find globally best parameters
- Converges to local optimum, depending on initial conditions
- Too many parameters / too large model: Overt-fitting

Summary: the HMM algorithms

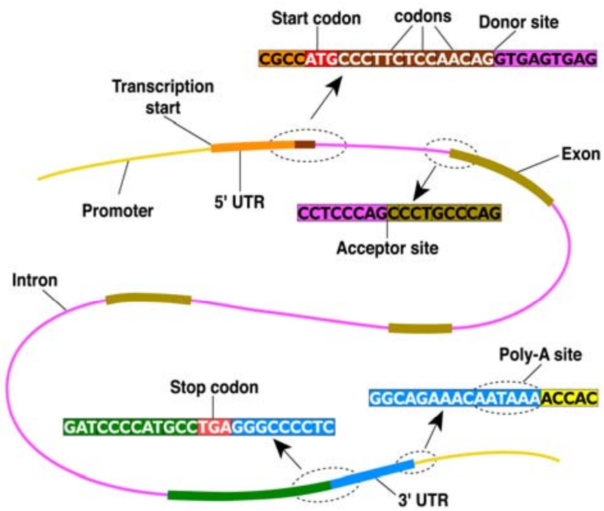
Questions:

- **Evaluation**: What is the probability of the observed sequence? Forward
- **Decoding**: What is the probability that the state of the 3rd roll is loaded, given the observed sequence? Forward-Backward
- **Decoding**: What is the most likely die sequence? Viterbi
- **Learning**: Under what parameterization are the observed sequences most probable? Baum-Welch (EM)

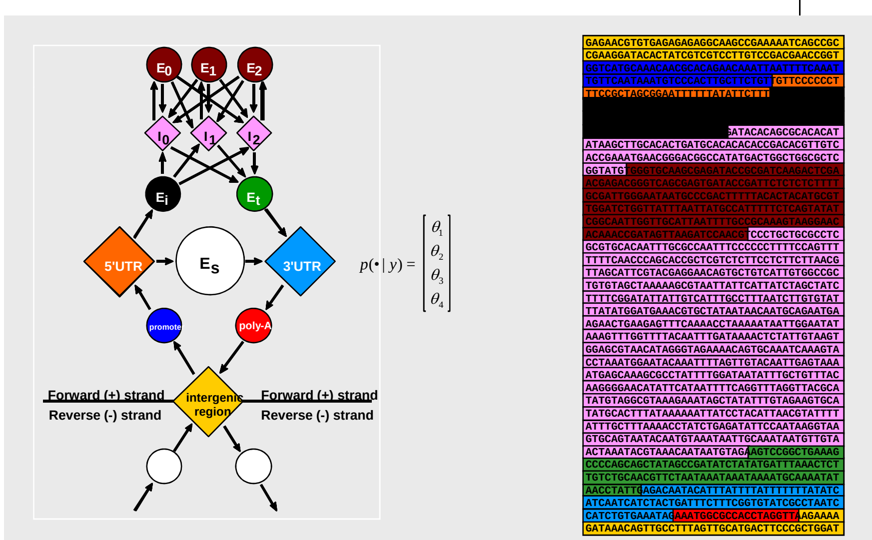
Applications of HMMs

- **Some early applications of HMMs**
  - finance, but we never saw them
  - speech recognition
  - modelling ion channels
- In the mid-late 1980s HMMs entered genetics and molecular biology, and they are now firmly entrenched.
- **Some current applications of HMMs to biology**
  - mapping chromosomes
  - aligning biological sequences
  - predicting sequence structure
  - inferring evolutionary relationships
  - finding genes in DNA sequence

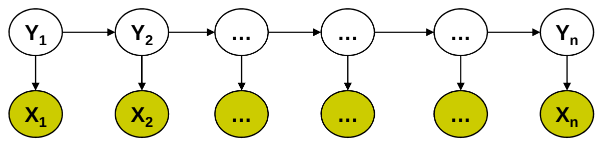
Typical structure of a gene



GENSCAN (Burge & Karlin)



Shortcomings of Hidden Markov Model

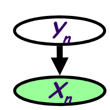


- HMM models capture dependences between each state and only its corresponding observation
  - NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line length, indentation, amount of white space, etc.
- Mismatch between learning objective function and prediction objective function
  - HMM learns a joint distribution of states and observations P(**Y**, **X**), but in a prediction task, we need the conditional probability P(**Y**|**X**)

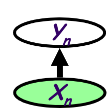
Recall Generative vs. Discriminative Classifiers

- Goal: Wish to learn f: X → Y, e.g., P(Y|X)

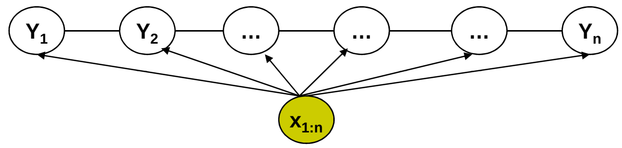
- Generative classifiers (e.g., Naïve Bayes):
  - Assume some functional form for P(X|Y), P(Y)
    This is a ***'generative'*** model of the data!
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X= x)



- Discriminative classifiers (e.g., logistic regression)
  - Directly assume some functional form for P(Y|X)
    This is a ***'discriminative'*** model of the data!
  - Estimate parameters of P(Y|X) directly from training data

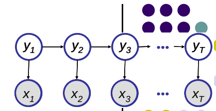


Structured Conditional Models



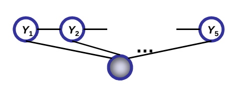
- Conditional probability P(label sequence $\mathbf{y}$ | observation sequence $\mathbf{x}$) rather than joint probability $P(\mathbf{y}, \mathbf{x})$
  - Specify the probability of possible label sequences given an observation sequence
- Allow arbitrary, non-independent features on the observation sequence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

Conditional Distribution



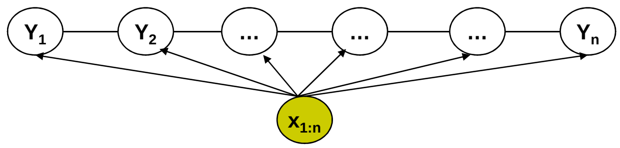
- If the graph $G = (V, E)$ of $\mathbf{Y}$ is a tree, the conditional distribution over the label sequence $\mathbf{Y} = \mathbf{y}$, given $\mathbf{X} = \mathbf{x}$, by the Hammersley Clifford theorem of random fields is:

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



- x is a data sequence
- y is a label sequence
- *v* is a vertex from vertex set V = set of label random variables
- *e* is an edge from edge set E over V
- f_k and g_k are given and fixed. g_k is a Boolean vertex feature; f_k is a Boolean edge feature
- *k* is the number of features
- $\theta = (\lambda_1, \lambda_2, \cdots, \lambda_n; \mu_1, \mu_2, \cdots, \mu_n); \lambda_k$ and μ_k are parameters to be estimated
- $\mathrm{y}|_e$ is the set of components of y defined by edge *e*
- $\mathrm{y}|_v$ is the set of components of y defined by vertex *v*

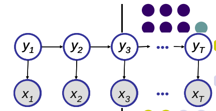
Conditional Random Fields



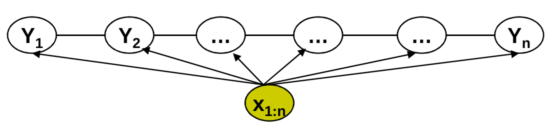
$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\phi(y_i, y_{i-1}, \mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n}, \mathbf{w})}\prod_{i=1}^{n}\exp(\mathbf{w}^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_{1:n}))$$

- CRF is a partially directed model
  - Discriminative model
  - Usage of global normalizer Z(**x**)
  - Models the dependence between each state and the entire observation sequence

Conditional Random Fields



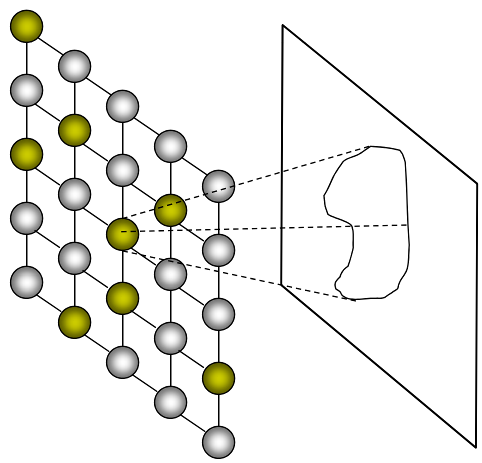
- General parametric form:



$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x},\lambda,\mu)}\exp(\sum_{i=1}^{n}(\sum_{k}\lambda_k f_k(y_i, y_{i-1}, \mathbf{x}) + \sum_{l}\mu_l g_l(y_i, \mathbf{x}))) \\ &= \frac{1}{Z(\mathbf{x},\lambda,\mu)}\exp(\sum_{i=1}^{n}(\lambda^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T\mathbf{g}(y_i, \mathbf{x}))) \end{aligned}$$

$$\text{where } Z(\mathbf{x},\lambda,\mu) = \sum_{\mathbf{y}}\exp(\sum_{i=1}^{n}(\lambda^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T\mathbf{g}(y_i, \mathbf{x})))$$

Conditional Random Fields



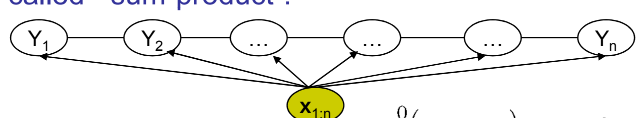
$$p_\theta(y \mid x) = \frac{1}{Z(\theta, x)} \exp\left\{\sum_c \theta_c f_c(x, y_c)\right\}$$

- Allow arbitrary dependencies on input
- Clique dependencies on labels
- Use approximate inference for general graphs

CRFs: Inference

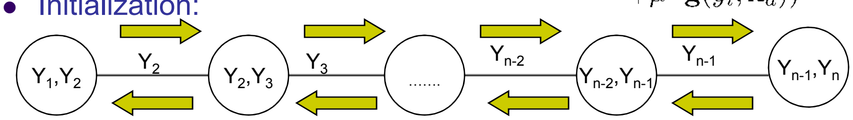
$$m_{i \to j}(S_{ij}) = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k \neq j} m_{k \to i}(S_{ki})$$

- Computing marginals using a message passing algorithm called "sum-product":



$$\alpha^0(y_i, y_{i-1}) = \exp(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_i, \mathbf{x}_d))$$

- Initialization:



Also called forward-backward algorithm

- After calibr

$$P(y_i, y_{i-1} | \mathbf{x}_d) \propto \alpha(y_i, y_{i-1})$$

$$\Rightarrow P(y_i, y_{i-1} | \mathbf{x}_d) = \frac{\alpha(y_i, y_{i-1})}{\sum_{y_i, y_{i-1}} \alpha(y_i, y_{i-1})} = \alpha'(y_i, y_{i-1})$$

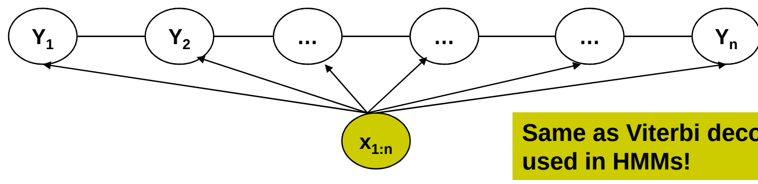
CRFs: Inference

$$m_{i\to j}(S_{ij}) = \max_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k\neq j} m_{k\to i}(S_{ki})$$

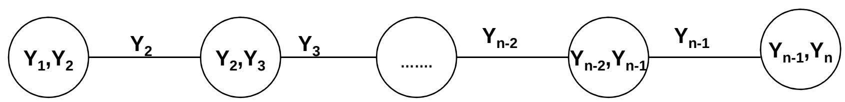
- Given CRF parameters λ and μ, find the $\mathbf{y}^*$ that maximizes P(**y**|**x**)

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \exp\left(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x}))\right)$$

  - Can ignore Z(**x**) because it is not a function of **y**
- Again run a message-passing algorithm called "max-product":



Same as Viterbi decoding used in HMMs!



CRF learning

- Given $\{(\mathbf{x}_d, \mathbf{y}_d)\}_{d=1}^N$, find λ^*, μ^* such that

$$\begin{aligned}
\lambda*, \mu* &= \arg\max_{\lambda,\mu} L(\lambda,\mu) = \arg\max_{\lambda,\mu} \prod_{d=1}^{N} P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) \\
&= \arg\max_{\lambda,\mu} \prod_{d=1}^{N} \frac{1}{Z(\mathbf{x}_d,\lambda,\mu)} \exp\left(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d))\right) \\
&= \arg\max_{\lambda,\mu} \sum_{d=1}^{N}\left(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)) - \log Z(\mathbf{x}_d, \lambda, \mu)\right)
\end{aligned}$$

- Computing the gradient w.r.t λ:

Gradient of the log-partition function in an exponential family is the expectation of the sufficient statistics.

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N}\left(\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}}\left(P(\mathbf{y}|\mathbf{x}_d) \sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d)\right)\right)$$

CRF learning

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N}\Big(\sum_{i=1}^{n}\mathbf{f}(y_{d,i},y_{d,i-1},\mathbf{x}_d) - \sum_{\mathbf{y}}\big(P(\mathbf{y}|\mathbf{x_d})\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\big)\Big)$$

- Computing the model expectations:
  - Requires exponentially large number of summations: Is it intractable?

$$\sum_{\mathbf{y}}\big(P(\mathbf{y}|\mathbf{x}_d)\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\big) = \sum_{i=1}^{n}\big(\sum_{\mathbf{y}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(\mathbf{y}|\mathbf{x}_d)\big)$$

$$= \sum_{i=1}^{n}\sum_{y_i,y_{i-1}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(y_i,y_{i-1}|\mathbf{x}_d)$$

Expectation of **f** over the corresponding marginal probability of neighboring nodes!!

- Tractable!
  - Can compute marginals using the sum-product algorithm on the chain

CRF learning

- Computing feature expectations using calibrated potentials:

$$\sum_{y_i,y_{i-1}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(y_i,y_{i-1}|\mathbf{x}_d) = \sum_{y_i,y_{i-1}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\alpha'(y_i,y_{i-1})$$

- Now we know how to compute $\nabla_\lambda L(\lambda,\mu)$:

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N}\Big(\sum_{i=1}^{n}\mathbf{f}(y_{d,i},y_{d,i-1},\mathbf{x}_d) - \sum_{\mathbf{y}}\big(P(\mathbf{y}|\mathbf{x_d})\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\big)\Big)$$

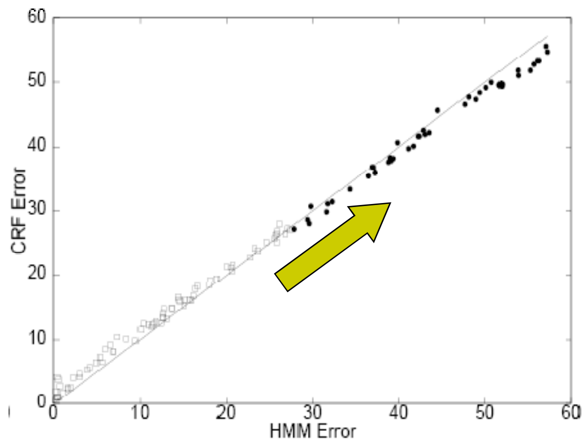
$$= \sum_{d=1}^{N}\Big(\sum_{i=1}^{n}\big(\mathbf{f}(y_{d,i},y_{d,i-1},\mathbf{x}_d) - \sum_{y_i,y_{i-1}}\alpha'(y_i,y_{i-1})\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\big)\Big)$$

- Learning can now be done using gradient ascent:

$$\lambda^{(t+1)} = \lambda^{(t)} + \eta\nabla_\lambda L(\lambda^{(t)},\mu^{(t)})$$
$$\mu^{(t+1)} = \mu^{(t)} + \eta\nabla_\mu L(\lambda^{(t)},\mu^{(t)})$$

CRFs: some empirical results

- Comparison of error rates on synthetic data



Data is increasingly higher order in the direction of arrow

CRFs achieve the lowest error rate for higher order data

CRFs: some empirical results

- Parts of Speech tagging

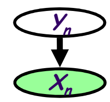
model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$Using spelling features

- Using same set of features: HMM >=< CRF
- Using additional overlapping features: CRF$^+$ >> HMM

Summary

- Conditional Random Fields is a discriminative Structured Input Output model!



- HMM is a generative structured I/O model

- Complementary strength and weakness:

1.
2.
3.

…