Definition (of HMM)

- Observation space
  - Alphabetic set: $C = \{c_1, c_2, \ldots, c_K\}$
  - Euclidean space: $\mathbb{R}^d$
- Index set of hidden states
  - $I = \{1, 2, \ldots, M\}$
- Transition probabilities between any two states
  - $p(y^i_t = 1 | y^i_{t-1} = 1) = a_{ij}$, or $p(y^i_t | y^j_{t-1} = 1) \sim \text{Multinomial}(a_{i1}, a_{i2}, \ldots, a_{iM}) \quad \forall i \in I$.
- Start probabilities
  - $p(y^i_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M)$
- Emission probabilities associated with each state
  - $p(x^i_t | y^i_t = 1) \sim \text{Multinomial}(b_{i1}, b_{i2}, \ldots, b_{iK}) \quad \forall i \in I$.
  - or in general:
    - $p(x^i_t | y^i_t = 1) \sim f(\cdot | \theta_i) \quad \forall i \in I$. 

Graphical model

State automata
Three Main Questions on HMMs

1. Evaluation
   GIVEN an HMM $M$ and a sequence $x$.
   FIND $\text{Prob}(x \mid M)$.
   ALGO. Forward

2. Decoding
   GIVEN an HMM $M$ and a sequence $x$.
   FIND the sequence $y$ of states that maximizes, e.g., $P(y \mid x, M)$, 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 $\theta = (\pi, a, \eta)$ that maximize $P(x \mid \theta)$.
   ALGO. Baum-Welch (EM)

Example:

\[ x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4 \]

\[ \alpha_i^t = \begin{cases} \pi, & \text{if } x_1 = i \\ \sum_{j=1}^{k} \alpha_j^{t-1} a_{ij} \end{cases} \]

\[ \beta_t^k = \begin{cases} 1, & \text{if } x_t = k \\ \sum_{j=1}^{k} a_{jk} \beta_{t+1}^j \end{cases} \]

\[ P(y_t^k = 1 \mid x) = \frac{\alpha_t^k \beta_t^k}{P(x)} \]
\[
\begin{align*}
\alpha_i^k &= P(x_i | y_i^k) = 1 \sum_i \alpha_{i+1}^i \beta_{i+1} \\
\beta_i^k &= \sum_d \beta_{d,i} P(x_{d+1} | y_{d+1}^k) = 1 \beta_{i+1} \\
\mathbb{P}(y_i^k = 1 | x) &= \frac{\alpha_i^k \beta_i^k}{\mathbb{P}(x)}
\end{align*}
\]

\[
\begin{array}{cccc}
\text{Alpha (actual)} & \text{Beta (actual)} & P(1|F) & P(1|L) \\
0.0833 & 0.0500 & 1/6 & 1/10 \\
0.0136 & 0.0052 & 1/6 & 1/10 \\
0.0022 & 0.0006 & 1/6 & 1/10 \\
0.0004 & 0.0001 & 1/6 & 1/10 \\
0.0001 & 0.0000 & 1/6 & 1/2 \\
0.0000 & 0.0000 & 1/6 & 1/2 \\
0.0000 & 0.0000 & 1/6 & 1/2 \\
0.0000 & 0.0000 & 1/6 & 1/2 \\
0.0000 & 0.0000 & 1/6 & 1/2 \\
\end{array}
\]
What is the probability of a hidden state prediction?

- A single state:
  \[ P(y_t | \mathbf{X}) = \sum_{k \in \text{values of } k} \frac{\alpha_t^k \beta_t^k}{P(x)} = \frac{\text{exp}(-\gamma t) \text{exp}(\gamma t) + \text{exp}(\gamma t - \gamma t)}{\text{exp}(\gamma t - \gamma t)} \]
- What about a hidden state sequence?
  \[ P(y_1, \ldots, y_T | \mathbf{X}) \]

Posterior decoding

- We can now calculate
  \[ P(y_t^k = 1 | x) = \frac{P(y_t^k = 1, x)}{P(x)} = \frac{\alpha_t^k \beta_t^k}{P(x)} \]
- Then, we can ask
  - What is the most likely state at position \( t \) of sequence \( x \):
    \[ k_t^* = \arg \max_k P(y_t^k = 1 | x) \]
  - Note that this is an MPA of a single hidden state, what if we want to a MPA of a whole hidden state sequence?
  - Posterior Decoding:
    \[ \{ y_t^k = 1 : t = 1 \cdots T \} \]
  - This is different from MPA of a whole sequence of hidden states
  - This can be understood as bit error rate vs. word error rate

Example:
- MPA of \( \mathbf{X} \)?
- MPA of \( \mathbf{X}, \mathbf{Y} \)?
Viterbi decoding

Given \( x = x_1, \ldots, x_T \), we want to find \( y = y_1, \ldots, y_T \), such that \( P(y|x) \) is maximized:

\[
y^* = \text{argmax}_y P(y|x) = \text{argmax}_y P(y|x) = \frac{P(x|y)P(y)}{P(x)}
\]

Let

\[
V_t^k = \max_{y_1, \ldots, y_{t-1}} P(x_1, \ldots, x_t, y_1, \ldots, y_{t-1}, y_t = k) = \text{Probability of most likely sequence of states ending at state } y_t = k
\]

The recursion:

\[
V_t^k = P(x_t | y_t^k = 1) \max_i a_{i,k} V_{t-1}^i
\]

Underflows are a significant problem

\[
p(x_1, \ldots, x_T, y_1, \ldots, y_T) = \pi_{y_1} a_{y_1,y_2} \ldots a_{y_{T-1},y_T} b_{y_T}
\]

- These numbers become extremely small – underflow
- Solution: Take the logs of all values:
  \[
  V^k_t = \log p(x_t | y_t^k = 1) + \max_i \left( \log(a_{i,k}) + V^i_{t-1} \right)
  \]

The Viterbi Algorithm – derivation

- Define the viterbi probability:
  \[
  V_{t-1}^k = \max_{y_1, \ldots, y_{t-2}} P(x_1, \ldots, x_{t-1}, y_1, \ldots, y_{t-1}, y_t^k = 1)
  \]

  \[
  = \max_{y_1, \ldots, y_{t-2}} P(x_1, x_2, y_1^k = 1 | x_1, x_2, y_1, \ldots, y_{t-1}) P(x_1, x_2, \ldots, x_{t-1}, y_1, y_{t-1})
  \]

  \[
  = \max_{y_1, \ldots, y_{t-2}} P(x_1, x_2, y_1^k = 1 | y_1) P(x_1, x_2, \ldots, x_{t-1}, y_1, y_{t-1})
  \]

  \[
  = \max_P p(x_1, x_2, y_1^k = 1 | y_1) \max_{y_1, \ldots, y_{t-2}} P(x_1, x_2, y_1^k = 1, y_1) P(x_1, x_2, \ldots, x_{t-1}, y_1, y_{t-1})
  \]

  \[
  = \max_P p(x_1, x_2, y_1^k = 1 | y_1) a_{y_1} V_{t-1}^i
  \]

  \[
  = p(x_1, x_2, y_1^k = 1) a_{y_1} V_{t-1}^i
  \]

  \[
  = p(x_1, x_2, y_1^k = 1) a_{y_1} V_{t-1}^i
  \]

\[© Eric Xing @ CMU, 2006-2010\]
The Viterbi Algorithm

- **Input:** $x = x_1, ..., x_T$
  
  **Initialization:**
  
  $V_1^k = P(x_1 | y_1^k = 1) \pi_k$

  **Iteration:**
  
  $V_t^k = P(x_t | y_t^k = 1) \max, a_{tk} V_{t-1}^j$
  
  $\text{Ptr}(k, t) = \arg \max, a_{tk} V_{t-1}^j$

  **Termination:**
  
  $P(x, y^\ast) = \max_k V_T^k$

  **TraceBack:**
  
  $y_T^\ast = \arg \max_k V_T^k$
  
  $y_{T-1}^\ast = \text{Ptr}(y_T^\ast, t)$

Viterbi Vs. MPA (individual)

$x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4$

$V_t^k (\log)$  $\text{Ptr}(k, t)$  $\text{Seq}$  $\text{Viterbi}$  $\text{MPA}$  $p(y_t^k = 1 | x)$

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.4849</td>
<td>-2.9957</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8128</td>
</tr>
<tr>
<td>-4.3280</td>
<td>-5.3496</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.8238</td>
</tr>
<tr>
<td>-6.1710</td>
<td>-7.7035</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.8176</td>
</tr>
<tr>
<td>-8.0141</td>
<td>-10.0574</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>0.7925</td>
</tr>
<tr>
<td>-9.8571</td>
<td>-10.8018</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0.7415</td>
</tr>
<tr>
<td>-11.7002</td>
<td>-13.1557</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.7505</td>
</tr>
<tr>
<td>-13.5432</td>
<td>-15.5096</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.7386</td>
</tr>
<tr>
<td>-15.3863</td>
<td>-16.2540</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0.7027</td>
</tr>
<tr>
<td>-17.2293</td>
<td>-18.6079</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.7251</td>
</tr>
<tr>
<td>-19.0724</td>
<td>-20.9618</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0.7251</td>
</tr>
</tbody>
</table>
### Another Example

X = 6, 2, 3, 5, 6, 2, 6, 3, 6, 6

| V_t^k (log) | p||t(r^k,t) | Seq | Viterbi | MPA | p(y_t^k = 1 | x) |
|-------------|--------|--------|--------|--------|------|--------------------|
| -2.4849     | -1.3863 | N/A    | 6      | 2      | 2    | 0.2733 0.7267      |
| -4.0943     | -4.1997 | 2      | 2      | 2      | 1    | 0.6090 0.3909      |
| -6.3969     | -7.0131 | 1      | 2      | 3      | 1    | 0.6538 0.3462      |
| -8.6995     | -9.6158 | 1      | 1      | 5      | 1    | 0.6062 0.3938      |
| -11.0021    | -10.3090| 1      | 1      | 6      | 2    | 0.2861 0.7139      |
| -13.0170    | -13.1224| 2      | 2      | 2      | 1    | 0.5342 0.4658      |
| -15.3196    | -14.3263| 1      | 2      | 6      | 2    | 0.2734 0.7266      |
| -17.0344    | -17.1397| 2      | 2      | 3      | 2    | 0.5226 0.4774      |
| -19.3370    | -18.3437| 1      | 2      | 6      | 2    | 0.2252 0.7748      |
| -21.0518    | -19.5477| 2      | 2      | 6      | 2    | 0.2159 0.7841      |

Same transition probabilities

### Computational Complexity and Implementation Details

- What is the running time, and space required, for Forward, and Backward?
  
  \[
  \alpha_t^k = p(x_t \mid y_t^k = 1) \sum_{i} \alpha_{t-1}^i a_{i,k} \\
  \beta_t^k = \sum_{i} a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i \\
  V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t+1}^i
  \]

  Time: \( O(K^2N) \); Space: \( O(KN) \).

- Useful implementation technique to avoid underflows
  - Viterbi: sum of logs
  - Forward/Backward: rescaling at each position by multiplying by a constant
Three Main Questions on HMMs

1. Evaluation
   GIVEN an HMM $M$ and a sequence $x$.
   FIND $\text{Prob}(x|\mathcal{M})$.
   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 $\theta = (\pi, a, \eta)$ that maximize $P(x|\theta)$.
   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 $\theta$ of the model to maximize
  $P(x|\theta)$ --- Maximal likelihood (ML) estimation
MLE

$$P(x, y) = \sum_{i=1}^{N} P(x_i, y_i)$$

$$\{x^*, A_{ij}, B_{ik}\} = \operatorname{argmax} \log P(x, y) .$$

$$y^*_t = \frac{y_t^1}{N},$$

$$x^*_t = \frac{x_t^1}{N_1} .$$

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:
  $$A_{ij} = \text{# times state transition } i \rightarrow j \text{ occurs in } y$$
  $$B_{ik} = \text{# times state } i \text{ in } y \text{ emits } k \text{ in } x$$
- We can show that the maximum likelihood parameters $\theta$ are:
  $$a_{ij}^{\text{ML}} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_t \sum_{t-1} x_{t-1,i} y_{t,j}}{\sum_t \sum_{t-1} x_{t-1,i}} = \frac{A_{ij}}{\sum_j A_{ij}},$$
  $$b_{ik}^{\text{ML}} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_t \sum_{t-1} x_{t-1,i} y_{t,k}}{\sum_t \sum_{t-1} y_{t,k}} = \frac{B_{ik}}{\sum_k B_{ik}} .$$
- What if $y$ is continuous? We can treat $$\left(x_{t,i}, y_{t,i}\right)_{t=1:T, i=1:N}$$ 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 \( \theta \) 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 \( \theta \) is unreasonable
    - 0 probabilities – VERY BAD

- **Example:**
  - Given 10 casino rolls, we observe
    \[
    x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3 \\
    \]
  - 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
    \[
    \begin{align*}
    A_{ij} &= \text{# times state transition } i \rightarrow j \text{ occurs in } y \\
    B_{ik} &= \text{# times state } i \text{ in } y \text{ emits } k \text{ in } x
    \end{align*}
    \]
  - \( R_{ij}, S_{jk} \) are pseudocounts representing our prior belief
  - Total pseudocounts: \( R_i = \sum_j R_{ij}, S_i = \sum_k S_{ik} \)
    - "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

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

**EXPECTATION MAXIMIZATION**

0. Starting with our best guess of a model $M$, parameters $\theta$.
1. Estimate $A_{ij}, B_{ik}$ in the training data
   - How? $A_{ij} = \sum_{n}(y_i = j, y_{i+1} = k)$, $B_k = \sum_n(y_n = k) x_n$.
   - Update $\theta$ 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 $\theta$ each iteration
The Baum-Welch algorithm

- The complete log likelihood
  \[ \zeta(\theta; x, y) = \log p(x, y) = \log \prod_{t=1}^{T} p(y_{t+1} \mid y_{t}, x_{t}) \prod_{t=2}^{T} p(y_{t} \mid y_{t-1}, x_{t-1}) \prod_{t=1}^{T} p(x_{t} \mid x_{t-1}) \]

- The expected complete log likelihood
  \[ E(\theta; x, y) = \sum_{x} \left( \log p(x_{t} \mid y_{t}) \right) + \sum_{t=2}^{T} \left( \log p(y_{t} \mid y_{t-1}, x_{t-1}) \right) + \sum_{t=1}^{T} \left( \log p(x_{t} \mid x_{t-1}) \right) \]

- EM
  - The E step
    \[ y'_{n,t} = \langle Y_{n,t} \rangle = p(y_{n,t} = 1 \mid x_{n}) \]
    \[ z'_{n,t} = \langle Y_{n,t-1}; y'_{n,t} \rangle = p(y_{n,t-1} = 1, y'_{n,t} = 1 \mid x_{n}) \]
  - The M step ("symbolically" identical to MLE)
    \[ \pi_{j}^{MC} = \frac{\sum_{n=1}^{N} y'_{n,j}}{N} \]
    \[ a_{ij}^{MC} = \frac{\sum_{n=1}^{N} \sum_{t=2}^{T} y'_{n,i} \cdot z'_{n,t-1} \cdot y'_{n,t} \cdot y_{n,t-1}}{\sum_{n=1}^{N} \sum_{t=2}^{T} y'_{n,i} \cdot z'_{n,t-1} \cdot y'_{n,t}} \]
    \[ b_{k}^{MC} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} y'_{n,j} \cdot x'_{n,t} \cdot y'_{n,t} \cdot y_{n,t}}{\sum_{n=1}^{N} \sum_{t=1}^{T} y'_{n,j} \cdot x'_{n,t} \cdot y'_{n,t}} \]

The Baum-Welch algorithm -- comments

Time Complexity:

- # iterations \times O(K^2N)
- 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 $f: X \rightarrow Y$, 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 $P(\text{label sequence } y \mid \text{observation sequence } x)$ 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 generative models

Conditional Distribution

- If the graph $G = (V, E)$ of $Y$ is a tree, the conditional distribution over the label sequence $Y = y$, given $X = x$, by the Hammersley Clifford theorem of random fields is:
  $$p(y \mid x) \propto \exp \left( \sum_{e \in E} \lambda_e \phi_e(e, y \mid y, x) + \sum_{v \in V} \mu_k g_k(v, y \mid x) \right)$$
  - $x$ is a data sequence
  - $y$ is a label sequence
  - $v$ is a vertex from vertex set $V = \text{set of label random variables}$
  - $e$ is an edge from edge set $E$ over $V$
  - $\phi_e$ and $g_k$ are given and fixed. $g_k$ is a Boolean vertex feature; $\phi_e$ is a Boolean edge feature
  - $k$ is the number of features
  - $\theta = (\lambda_1, \lambda_2, \ldots, \lambda_n; \mu_1, \mu_2, \ldots, \mu_k)$; $\lambda_e$ and $\mu_k$ are parameters to be estimated
  - $y_e$ is the set of components of $y$ defined by edge $e$
  - $y_v$ is the set of components of $y$ defined by vertex $v$
Conditional Random Fields

\[ P(y_{1:n} | x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, x_{1:n}) = \frac{1}{Z(x_{1:n}, w)} \prod_{i=1}^{n} \exp(w^T \Gamma(y_i, y_{i-1}, x_{1:n})) \]

- 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:

\[ P(y|x) = \frac{1}{Z(x, \lambda, \mu)} \exp\left( \sum_{i=1}^{n} \sum_{k} \lambda_k f_k(y_i, y_{i-1}, x) + \sum_{i} \mu_i g_i(y_i, x) \right) \]
\[ = \frac{1}{Z(x, \lambda, \mu)} \exp\left( \sum_{i=1}^{n} (\lambda^T \Gamma(y_i, y_{i-1}, x) + \mu^T g(y_i, x)) \right) \]

where \( Z(x, \lambda, \mu) = \sum_{y} \exp\left( \sum_{i=1}^{n} (\lambda^T \Gamma(y_i, y_{i-1}, x) + \mu^T g(y_i, x)) \right) \)
Conditional Random Fields

\[ p_y(y | x) = \frac{1}{Z(\theta, x)} \exp \left\{ \sum_{c} \theta_c f(x, y) \right\} \]

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

CRFs: Inference

- Given CRF parameters \( \lambda \) and \( \mu \), find the \( y^* \) that maximizes \( P(y | x) \)

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

- Can ignore \( Z(x) \) because it is not a function of \( y \)
- Run the max-product algorithm on the junction-tree of CRF:

Same as Viterbi decoding used in HMMs!
CRF learning

- Given \( \{(x_d, y_d)\}_{d=1}^N \), find \( \lambda^*, \mu^* \) such that

\[
\lambda^*, \mu^* = \arg \max_{\lambda, \mu} L(\lambda, \mu) - \arg \max_{\lambda, \mu} \prod_{d=1}^N P(y_d|x_d, \lambda, \mu)
\]

\[
= \arg \max_{\lambda, \mu} \prod_{d=1}^N \frac{1}{Z(x_d, \lambda, \mu)} \exp \left( \sum_{i=1}^n (\lambda^T f(y_{d,i}, y_{d,i-1}, x_d) + \mu^T g(y_{d,i}, x_d)) \right)
\]

\[
= \arg \max_{\lambda, \mu} \sum_{d=1}^N \sum_{i=1}^n (\lambda^T f(y_{d,i}, y_{d,i-1}, x_d) + \mu^T g(y_{d,i}, x_d)) - \log Z(x_d, \lambda, \mu)
\]

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

- Computing the gradient w.r.t \( \lambda \):

\[
\nabla_\lambda L(\lambda, \mu) = \sum_{d=1}^N \sum_{i=1}^n f(y_{d,i}, y_{d,i-1}, x_d) - \sum_{y} P(y|x_d) \sum_{i=1}^n f(y_{d,i}, y_{d,i-1}, x_d)
\]

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

<table>
<thead>
<tr>
<th>model</th>
<th>error</th>
<th>oov error</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>5.69%</td>
<td>45.99%</td>
</tr>
<tr>
<td>MEMM</td>
<td>6.37%</td>
<td>54.61%</td>
</tr>
<tr>
<td>CRF</td>
<td>5.55%</td>
<td>48.05%</td>
</tr>
<tr>
<td>MEMM+</td>
<td>4.81%</td>
<td>26.99%</td>
</tr>
<tr>
<td>CRF+</td>
<td>4.27%</td>
<td>23.76%</td>
</tr>
</tbody>
</table>

- 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. 
  ...