Three Main Questions on HMMs

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

2. Decoding
   GIVEN an HMM $\mathcal{M}$, and a sequence $\mathbf{x}$,
   FIND the sequence $\mathbf{y}$ of states that maximizes, e.g., $P(\mathbf{y} | \mathbf{x}, \mathcal{M})$, or the most probable subsequence of states
   ALGO. Viterbi, Forward-backward

3. Learning
   GIVEN an HMM $\mathcal{M}$, with unspecified transition/emission probs., and a sequence $\mathbf{x}$
   FIND parameters $\theta = (\pi, a_{ij}, \eta_k)$ that maximize $P(\mathbf{x} | \theta)$
   ALGO. Baum-Welch (EM)
Example:

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

\[ P(1|F) = 1/6 \]
\[ P(2|F) = 1/6 \]
\[ P(3|F) = 1/6 \]
\[ P(4|F) = 1/6 \]
\[ P(5|F) = 1/6 \]
\[ P(6|F) = 1/6 \]

\[ P(1|L) = 1/10 \]
\[ P(2|L) = 1/10 \]
\[ P(3|L) = 1/10 \]
\[ P(4|L) = 1/10 \]
\[ P(5|L) = 1/10 \]
\[ P(6|L) = 1/2 \]

\[ \alpha^k_t = P(x_t | y^k_t = 1)\sum a^i_{t+1} \alpha^i_{t+1} \]

\[ \beta^k_t = \sum a_{k,i} P(x_{t+1} | y^i_{t+1} = 1)\beta^i_{t+1} \]

\[ P(y^k_t = 1 | x) = \frac{P(y^k_t = 1, x)}{P(x)} = \frac{\alpha^k_t \beta^k_t}{P(x)} \]
What is the probability of a hidden state prediction?

- A single state:

\[ P(y_t | X) \]

- What about a hidden state sequence?

\[ P(y_1, \ldots, y_T | X) \]
### Posterior decoding

- We can now calculate
  \[ P(y_t^k = 1 \mid x) = \frac{P(y_t^k = 1, x)}{P(x)} = \frac{\alpha_t^k \beta_t^k}{\hat{P}(x)} \]

- Then, we can ask
  - What is the most likely state at position \( t \) of sequence \( x \):
    \[ k_t^* = \arg \max_k P(y_t^k = 1 \mid x) \]

- 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:
  \[ \{y_t^k = 1 : t = 1 \ldots T \} \]

- This is different from MPA of a whole sequence of hidden states

- This can be understood as bit error rate vs. word error rate

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( P(x,y) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**Example:** MPA of \( X \)? MPA of \( X, Y \)?

### Viterbi decoding

- GIVEN \( x = x_1, ..., x_T \), we want to find \( y = y_1, ..., y_T \), such that \( R(y \mid x) \) is maximized:
  \[ y^* = \arg \max_y R(y \mid x) = \arg \max_y R(y, x) \]

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

- The recursion:
  \[ V_t^k = p(x_t \mid y_t^k = 1) \max_{i} a_{i,k} V_{t-1}^i \]

- Underflows are a significant problem
  \[ p(x_1, ..., x_T, y_1, ..., y_T) = \prod_{t=1}^T a_{y_{t-1},y_t} b_{y_t,x_t} \]

  - These numbers become extremely small – underflow
  - Solution: Take the logs of all values:
    \[ V_t^k = \log p(x_t \mid y_t^k = 1) + \max_{i} \left( \log(a_{i,k}) + V_{t-1}^i \right) \]
The Viterbi Algorithm – derivation

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

The Viterbi Algorithm

- Input: \( x = x_1, \ldots, x_T \)
  
  **Initialization:**
  \[ V_1^k = P(x_1 | y_1^k = 1) \]
  
  **Iteration:**
  \[ V_t^k = P(x_t | y_t^k = 1) \max_k V_{t-1}^k \]
  \[ \text{Ptr}(k, t) = \text{arg} \max_k V_{t-1}^k \]
  
  **Termination:**
  \[ P(x, y^*) = \max_k V_T^k \]
  
  **TraceBack:**
  \[ y_T^* = \text{arg} \max_k V_T^k \]
  \[ y_{T-1}^* = \text{Ptr}(y_T^*, t) \]
Viterbi Vs. MPA (individual)

<table>
<thead>
<tr>
<th>( V_i^k ) (log)</th>
<th>( p(r(k,t)) )</th>
<th>( \text{Seq} )</th>
<th>( \text{Viterbi} )</th>
<th>( \text{MPA} )</th>
<th>( p(y_i^k = 1 \mid x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.4849</td>
<td>-2.39959</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>0.8128</td>
</tr>
<tr>
<td>-4.3280</td>
<td>-5.3496</td>
<td>1 2</td>
<td>2</td>
<td>1</td>
<td>0.8238</td>
</tr>
<tr>
<td>-6.1710</td>
<td>-7.7035</td>
<td>1 2 1</td>
<td>1</td>
<td>1</td>
<td>0.8176</td>
</tr>
<tr>
<td>-8.0141</td>
<td>-10.0574</td>
<td>1 2 5</td>
<td>1</td>
<td>1</td>
<td>0.7925</td>
</tr>
<tr>
<td>-9.8571</td>
<td>-10.8018</td>
<td>1 2 6</td>
<td>1</td>
<td>1</td>
<td>0.7415</td>
</tr>
<tr>
<td>-11.7002</td>
<td>-13.1557</td>
<td>1 2 2</td>
<td>1</td>
<td>1</td>
<td>0.7505</td>
</tr>
<tr>
<td>-13.5432</td>
<td>-15.5096</td>
<td>1 2 1</td>
<td>1</td>
<td>1</td>
<td>0.7386</td>
</tr>
<tr>
<td>-15.3863</td>
<td>-16.2540</td>
<td>1 2 6</td>
<td>1</td>
<td>1</td>
<td>0.7272</td>
</tr>
<tr>
<td>-19.0724</td>
<td>-20.9618</td>
<td>1 2 4</td>
<td>1</td>
<td>1</td>
<td>0.7251</td>
</tr>
</tbody>
</table>

Another Example

<table>
<thead>
<tr>
<th>( V_i^k ) (log)</th>
<th>( p(r(k,t)) )</th>
<th>( \text{Seq} )</th>
<th>( \text{Viterbi} )</th>
<th>( \text{MPA} )</th>
<th>( p(y_i^k = 1 \mid x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.4849</td>
<td>-1.3863</td>
<td>N/A</td>
<td>6</td>
<td>2</td>
<td>0.2733</td>
</tr>
<tr>
<td>-4.0943</td>
<td>-4.1997</td>
<td>2 2</td>
<td>2</td>
<td>1</td>
<td>0.6040</td>
</tr>
<tr>
<td>-6.3969</td>
<td>-7.0131</td>
<td>1 2 3</td>
<td>1</td>
<td>1</td>
<td>0.6538</td>
</tr>
<tr>
<td>-8.6995</td>
<td>-9.6158</td>
<td>1 1 5</td>
<td>1</td>
<td>1</td>
<td>0.6062</td>
</tr>
<tr>
<td>-11.0021</td>
<td>-10.3090</td>
<td>1 1 6</td>
<td>2</td>
<td>2</td>
<td>0.2861</td>
</tr>
<tr>
<td>-13.0170</td>
<td>-13.1224</td>
<td>2 2 2</td>
<td>2</td>
<td>1</td>
<td>0.5342</td>
</tr>
<tr>
<td>-15.3196</td>
<td>-14.3263</td>
<td>1 2 6</td>
<td>2</td>
<td>2</td>
<td>0.2734</td>
</tr>
<tr>
<td>-17.0344</td>
<td>-17.1397</td>
<td>2 2 3</td>
<td>2</td>
<td>1</td>
<td>0.5226</td>
</tr>
<tr>
<td>-19.3370</td>
<td>-18.3437</td>
<td>1 2 6</td>
<td>2</td>
<td>2</td>
<td>0.2252</td>
</tr>
<tr>
<td>-21.0518</td>
<td>-19.5477</td>
<td>2 2 6</td>
<td>2</td>
<td>2</td>
<td>0.2159</td>
</tr>
</tbody>
</table>

Same transition probabilities
Computational Complexity and implementation details

- What is the running time, and space required, for Forward, and Backward?

\[
\alpha_i^k = p(x_i \mid y_i^k = 1) \sum_j \alpha_{i-1}^j a_{j,k}
\]

\[
\beta_i^k = \sum_j a_{k,j} p(x_{i+1} \mid y_{i+1}^k = 1) \beta_{i+1}^j
\]

\[
V_i^k = p(x_i \mid y_i^k = 1) \max_j a_{i,j} V_{i+1}^j
\]

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

- 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 \(\text{Prob}(x \mid M)\);
   - 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)
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

© Eric Xing @ CMU, 2006-2010
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:
  
  \[ a_{ij}^{\text{ML}} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow *)} \frac{\sum_{n=1}^{T} \sum_{t=2}^{T} y_{n,t-1} y_{n,t}}{\sum_{j} A_{ij}} \]
  
  \[ b_{ik}^{\text{ML}} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow *)} \frac{\sum_{n=1}^{T} \sum_{t=1}^{T} y_{n,t-1} x_{n,t} y_{n,t}}{\sum_{k} \sum_{j} B_{ik}} \]

- What if $y$ is continuous? We can treat $\{(x_{nt}, y_{nt}) : t = 1 \ldots T, n = 1 \ldots N\}$ 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

- **Example:**
  - Given 10 casino rolls, we observe
    
    \[ x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3 \]
    

  - Then:
    
    \[ a_{FL} = 0; \quad a_{FL} = 0 \]
    
    \[ b_{F1} = b_{F3} = .2; \]
    
    \[ b_{F2} = .3; b_{F4} = 0; b_{F5} = b_{F6} = .1 \]
Pseudocounts

Solution for small training sets:
- Add pseudocounts
  \[ A_{ij} = \text{# times state transition } i \rightarrow j \text{ occurs in } y + R_{ij} \]
  \[ B_{ik} = \text{# times state } i \text{ in } y \text{ emits } k \text{ in } x + S_{ik} \]
- \[ R_{ij}, S_{ik} \] 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
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**

1. Starting with our best guess of a model \( M \), parameters \( \theta \).
   - Estimate \( A_{ij} \), \( B_{ik} \) in the training data.
     - How? \( A_{ij} = \sum_{k} \langle y'_n, y'_t \rangle \cdot x'_{t+1} \).
     - 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
  \[
  \ell(\theta; x, y) = \log p(x, y) = \log \prod_n p(y_{n,1}) \prod_{t=2}^T p(y_{n,t} | y_{n,t-1}) \prod_{t=1}^T p(x_{n,t} | x_{n,t})
  \]

- The expected complete log likelihood
  \[
  \langle \ell(\theta; x, y) \rangle = \sum_n \langle y_{n,1} \rangle_{p(y_{n,1})} \log \pi_1 + \sum_n \sum_{t=2}^T \langle y_{n,t}, y_{n,t-1} \rangle_{p(y_{n,t} | y_{n,t-1})} \log a_{ij} + \sum_n \sum_{t=2}^T \langle y_{n,t} \rangle_{p(y_{n,t} | x_{n,t})} \log b_{ik}
  \]

- EM
  - The E step
    \[
    y'_{n,t} = \langle y_{n,t} \rangle = p(y_{n,t} = 1 | x_n)
    \]
    \[
    z'_{n,t} = \langle y_{n,t-1}, y'_{n,t} \rangle = p(y_{n,t-1} = 1, y'_{n,t} = 1 | x_n)
    \]
  - The M step ("symbolically" identical to MLE)
    \[
    \pi'_{M} = \frac{\sum_{t=1}^T y'_{n,t}}{N}
    \]
    \[
    a'_{ij} = \frac{\sum_{t=1}^T z'_{n,t} y'_{n,t+1}}{\sum_{t=1}^T z'_{n,t} y'_{n,t+1}}
    \]
    \[
    b'_{ik} = \frac{\sum_{t=1}^T z'_{n,t} x'_{n,t+1}}{\sum_{t=1}^T z'_{n,t} x'_{n,t+1}}
    \]
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

© Eric Xing @ CMU, 2006-2010
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:

$$p_y(y | x) \propto \exp \left( \sum_{e \in E} \lambda_k f_k(e, y|e, x) + \sum_{v \in V} \mu_k g_k(v, y|v, x) \right)$$

- $x$ 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 $V$
- $f_k$ 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
- $\theta = (\lambda_1, \lambda_2, \ldots, \lambda_k; \mu_1, \mu_2, \ldots, \mu_k)$; $\lambda_k$ 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

\[ P(y_1:n|x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, x_{1:n}) = \frac{1}{Z(x_{1:n}, w)} \prod_{i=1}^{n} \exp(w^T f(y_i, y_{i-1}, x_{1:n})) \]

- 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:

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

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

where 

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

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 \(P(y|x)\)
  \[
  y^* = \arg \max_y \exp\left(\sum_{i=1}^{n} (\lambda^T f(y_i, y_{i-1}, x) + \mu^T g(y_i, x))\right)
  \]
  - Can ignore \(Z(x)\) because it is not a function of \(y\)
  - Run the max-product algorithm on the junction-tree of CRF:

\[\begin{array}{c}
Y_1 \quad Y_2 \quad \ldots \quad \ldots \quad \ldots \quad Y_n \\
Y_{n-1} \quad Y_n
\end{array}\]

CRF learning

- Given \(\{(x_d, y_d)\}_{d=1}^{N}\), find \(\lambda^*, \mu^*\) such that
  \[
  \lambda^*, \mu^* = \arg \max_{\lambda, \mu} L(\lambda, \mu) = \arg \max_{\lambda, \mu} \prod_{d=1}^{N} P(y_d|x_d, \lambda, \mu)
  \]
  \[
  = \arg \max_{\lambda, \mu} \frac{1}{N} \prod_{d=1}^{N} \exp\left(\sum_{i=1}^{n} (\lambda^T f(y_{d,i}, y_{d,i-1}, x_d) + \mu^T g(y_{d,i}, x_d))\right)
  \]
  \[
  = \arg \max_{\lambda, \mu} \sum_{d=1}^{N} \left(\sum_{i=1}^{n} (\lambda^T f(y_{d,i}, y_{d,i-1}, x_d) + \mu^T g(y_{d,i}, x_d)) - \log Z(x_d, \lambda, \mu)\right)
  \]

- Computing the gradient w.r.t \(\lambda\):
  \[
  \nabla_\lambda L(\lambda, \mu) = \sum_{d=1}^{N} \sum_{i=1}^{n} f(y_{d,i}, y_{d,i-1}, x_d) - \sum_{y} \left(P(y|x_d) \sum_{i=1}^{n} f(y_{d,i}, y_{d,i-1}, x_d))\right)
  \]
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 spelling features

- Using same set of features: HMM >= CRF > MEMM
- Using additional overlapping features: CRF+ > MEMM+ >> 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. 
  ...