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
  - \( p(y_t^i = 1 \mid y_{t-1}^j = 1) = a_{ij} \)
  - or \( p(y_t^i \mid y_{t-1}^j = 1) \sim \text{Multinomial}(a_{i1}, a_{i2}, \ldots, a_{iM}) \) \( \forall i \in I \).
- Start probabilities
  - \( p(y_1^i) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M) \)
- Emission probabilities associated with each state
  - \( p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(\theta_{i1}, \theta_{i2}, \ldots, \theta_{iK}) \) \( \forall i \in I \).
  - or in general:
    - \( p(x_t \mid y_t^i = 1) \sim f(\cdot \mid \theta_i) \forall i \in I. \)
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 \( \theta = (\pi, a_{ij}, \eta_k) \) 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
  \( P(x \mid \theta) \) --- Maximal likelihood (ML) estimation
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_{n=2}^N y_{n,t}^i \cdot y_{n-1,t}^j}{\sum_t \sum_{n=2}^N y_{n,t}^i} = \frac{A_{ij}}{\sum_j A_{ij}}
    
    b_{ik}^{\text{ML}} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_t \sum_{n=2}^N y_{n,t}^i x_{n,t}^k}{\sum_t \sum_{n=2}^N 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 $\theta$ 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
    $A_{ij} = \# \text{times state transition } i \rightarrow j \text{ occurs in } y + R_{ij}$
    $B_{ik} = \# \text{times state } i \text{ in } y \text{ emits } k \text{ in } x + S_{ik}$
  - $R_{ij}, S_{ik}$ are pseudocounts representing our prior belief
  - Total pseudocounts: $R_i = \Sigma_j R_{ij}, S_i = \Sigma_k S_{ik}$
    - "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

- 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 $\theta$.
2. Estimate $A_{ij}, B_d$ in the training data
   - How? $A_{ij} = \sum_x (y_{i+1} = y_{i+1} | x_j) / \sum_x (y_{i+1} = y_{i+1} | x_i)$
   - $B_d = \sum_x (y_{i+1} = y_{i+1} | x_d) x_i$.
3. Update $\theta$ according to $A_{ij}, B_d$
4. Now a "supervised learning" problem
5. 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
  \[ \zeta(\theta; x, y) = \log p(x, y) = \log \prod_{t} \left( p(y_{n_t} | y_{n_{t-1}}) \prod_{t=2}^{T} p(y_{n_{t-1}} | y_{n_{t-2}}) \prod_{t=1}^{T} p(x_{n_t} | x_{n_{t-1}}) \right) \]

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

- EM
  - The E step
    \[ \gamma'_{n,t} = \mathbb{P}(y_{n_t} = 1 | x_{n}) \]
    \[ \xi'_{t,j} = \mathbb{P}(y_{n_{t-1}, n_t} = 1 | y_{n_{t-2}} = 1, x_{n}) \]
  - The M step ("symbolically" identical to MLE)
    \[ \pi_{ij}^{ML} = \frac{\sum_{t=1}^{T} \gamma'_{n,t} \delta_{i,j}}{N} \]
    \[ a_{ij}^{ML} = \frac{\sum_{t=1}^{T} \sum_{n=1}^{N} \gamma'_{n,t} \delta_{i,j}}{\sum_{t=1}^{T} \sum_{n=1}^{N} \gamma'_{n,t}} \]
    \[ b_{ik}^{ML} = \frac{\sum_{t=1}^{T} \sum_{n=1}^{N} \gamma'_{n,t} x_{n,t} \delta_{i,k}}{\sum_{t=1}^{T} \sum_{n=1}^{N} \gamma'_{n,t}} \]

The Baum-Welch algorithm -- comments

- Time Complexity:
  \[ \# \text{ iterations} \times O(K^2N) \]
  - 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:

$$p_y(y \mid x) \propto \exp \left( \sum_{e \in E, k} \lambda_e f_e(y|_e, x) + \sum_{v \in V, k} \mu_k g_k(y|_v, x) \right)$$

  - $x$ 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 $V$
  - $f_e$ and $g_k$ are given and fixed. $g_k$ is a Boolean vertex feature; $f_e$ is a Boolean edge feature
  - $k$ is the number of features
  - $\theta = (\lambda_1, \lambda_2, \cdots, \lambda_n; \mu_1, \mu_2, \cdots, \mu_k)$; $\lambda$ 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

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

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

General parametric form:

$P(y|x) = \frac{1}{Z(x, \lambda, \mu)} \exp(\sum_{i=1}^{n} (\sum_{k} \lambda_k f_k(y_i, y_{i-1}, x) + \sum_{i} \mu_i g_i(y_i, x)))$

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

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

\[ p_y(y \mid x) = \frac{1}{Z(\theta, x)} \exp \left\{ \sum_z \theta_z f_z(x, y) \right\} \]

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

CRFs: Inference

\[ m_{i\to j}(S_j) = \sum_{c \in C} \psi_c \prod_{k \in j} m_{k \to i}(S_k) \]

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

  \[ \alpha^0(y_i, y_{i-1}) = \exp(\lambda^T f(y_i, y_{i-1}, x_d)) + \mu^T g(y_i, x_d) \]

- Initialization:

- After calibration:

  \[ P(y_i, y_{i-1} \mid x_d) \propto \alpha(y_i, y_{i-1}) \]

  \[ \Rightarrow P(y_i, y_{i-1} \mid x_d) = \frac{\alpha(y_i, y_{i-1})}{\sum_{y_{i-1}} \alpha(y_i, y_{i-1})} = \alpha'(y_i, y_{i-1}) \]

© Eric Xing @ CMU, 2006-2011
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 $y$

- Again run a message-passing algorithm called “max-product”:

  ![Diagram of max-product algorithm]

  Same as Viterbi decoding used in HMMs!

CRF learning

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

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

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

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

- Computing the gradient w.r.t $\lambda$:

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

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

- Requires exponentially large number of summations: Is it intractable?

\[
\sum_y (P(y|x_d) \sum_{i=1}^n f(y_{i}, y_{i-1}, x_d)) = \sum_{i=1}^n (\sum_y f(y_{i}, y_{i-1}, x_d) P(y|x_d)) - \sum_{i=1}^n \sum_{y_{i}, y_{i-1}} f(y_{i}, y_{i-1}, x_d) P(y_{i}, y_{i-1}|x_d)
\]

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

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

![Graph showing comparison of CRF and HMM error rates]

- 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 $\geq$ CRF
- Using additional overlapping features: CRF $\gg$ 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.
  ...

© Eric Xing @ CMU, 2006-2011