Machine Learning

10-701/15-781, Spring 2010

Hidden Markov Model (II)

Eric Xing

Lecture 1~~1~~2, February 24, 2010

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

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

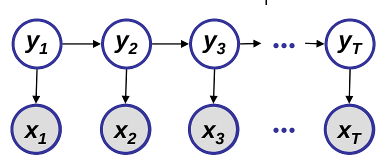
  or $p(y_t \mid y_{t-1}^i = 1) \sim \text{Multinomial}(a_{i,1}, a_{i,2}, \ldots, a_{i,M}), \forall i \in \mathrm{I}.$
- Start probabilities

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

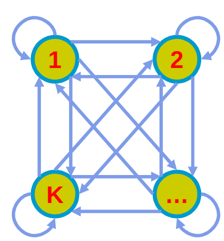
  $p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathrm{I}.$

  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 ($\boldsymbol{x}$ | ***M***)
   ALGO. Forward

2. Decoding

   GIVEN an HMM ***M***, and a sequence ***x*** ,
   FIND the sequence **y** of states that maximizes, e.g., P($\boldsymbol{y}$ | $\boldsymbol{x}$, ***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($\boldsymbol{x}$ | θ)
   ALGO. Baum-Welch (EM)

© Eric Xing @ CMU, 2006-2010 3

Example:

$\boldsymbol{x}$ = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

α_1^1
$= \pi^1 P(x_1 | y_1^1)$
$= 0.5 \times 1/6 = 0.083.$

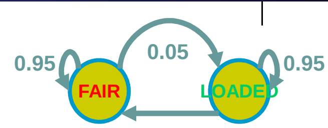
α_1^2
$= \pi^1 P(x_1 | y_1^2)$
$= 0.5 \times 1/10 = 0.05$

α_2^1
$= 1/6 \times (0.083 \times .95 + .05 \times .05)$
$= 0.013$

α_2^2
$= 1/10 \times (0.083 \times 0.05 + 0.05 \times .95)$
$= 0.005$

α_3^1
⋮

α_3^2
⋮



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

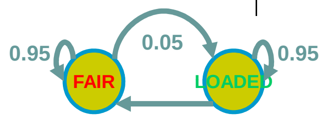
$$\alpha_t^k = P(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

© Eric Xing @ CMU, 2006-2010 4

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



Alpha (actual)		Beta (actual)	
0.0833	0.0500	0.0000	0.0000
0.0136	0.0052	0.0000	0.0000
0.0022	0.0006	0.0000	0.0000
0.0004	0.0001	0.0000	0.0000
0.0001	0.0000	0.0001	0.0001
0.0000	0.0000	0.0007	0.0006
0.0000	0.0000	0.0045	0.0055
0.0000	0.0000	0.0264	0.0112
0.0000	0.0000	0.1633	0.1033
0.0000	0.0000	1.0000	1.0000

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

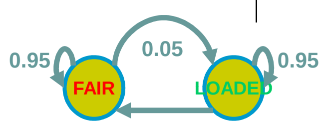
P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

$$\alpha_t^k = P(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1)\beta_{t+1}^i$$

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

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



Alpha (logs)		Beta (logs)	
-2.4849	-2.9957	-16.2439	-17.2014
-4.2969	-5.2655	-14.4185	-14.9922
-6.1201	-7.4896	-12.6028	-12.7337
-7.9499	-9.6553	-10.8042	-10.4389
-9.7834	-10.1454	-9.0373	-9.7289
-11.5905	-12.4264	-7.2181	-7.4833
-13.4110	-14.6657	-5.4135	-5.1977
-15.2391	-15.2407	-3.6352	-4.4938
-17.0310	-17.5432	-1.8120	-2.2698
-18.8430	-19.8129	0	0

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

$$\alpha_t^k = P(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1)\beta_{t+1}^i$$

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

What is the probability of a hidden state prediction?

- A single state:

$$P(y_t|\mathbf{X})$$

- What about a hidden state sequence ?

$$P(y_1, \ldots, y_T|\mathbf{X})$$

Posterior decoding

- We can now calculate

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

- Then, we can ask
  - What is the most likely state at position *t* of sequence $\mathbf{x}$:

$$k_t^* = \arg\max_k P(y_t^k = 1 \mid \mathbf{x})$$

  - 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: $\left\{ y_t^{k_t^*} = 1 : t = 1 \cdots T \right\}$
  - 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 X ?
MPA of (X, Y) ?

X	Y	P(X,Y)
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Viterbi decoding

$P(y_1 \quad y_T|X)$

$p(y_1^* \quad y_T^*|X)$

- GIVEN $\mathbf{x} = x_1, \ldots, x_T$, we want to find $\mathbf{y} = y_1, \ldots, y_T$, such that $P(\mathbf{y}|\mathbf{x})$ is maximized:

$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{y_\pi} P(\mathbf{y},\mathbf{x}) = P(y|x)P(x)$$

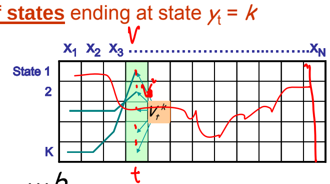
  - Let

$$V_t^k = \max_{\{y_1,\ldots,y_{t-1}\}} P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t^k=1) \qquad V_t^k = ($$

$=$ 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

$$p(x_1,\ldots,x_t,y_1,\ldots,y_t) = \pi_{y_1} a_{y_1,y_2} \cdots a_{y_{t-1},y_t} b_{y_1,x_1} \cdots b_{y_t,x_t}$$

    - 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:

$$\begin{aligned}
V_{t+1}^k &= \max_{\{y_1,\ldots,y_t\}} P(x_1,\ldots,x_t,y_1,\ldots,y_t,x_{t+1},y_{t+1}^k=1) \\
&= \max_{\{y_1,\ldots,y_t\}} P(x_{t+1},y_{t+1}^k=1 \mid x_1,\ldots,x_t,y_1,\ldots,y_t) P(x_1,\ldots,x_t,y_1,\ldots,y_t) \\
&= \max_{\{y_1,\ldots,y_t\}} P(x_{t+1},y_{t+1}^k=1 \mid y_t) P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t) \\
&= \max_i P(x_{t+1},y_{t+1}^k=1 \mid y_t^i=1) \max_{\{y_1,\ldots,y_{t-1}\}} P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t^i=1) \\
&= \max_i P(x_{t+1} \mid y_{t+1}^k=1) a_{i,k} V_t^i \\
&= P(x_{t+1} \mid y_{t+1}^k=1) \max_i a_{i,k} V_t^i
\end{aligned}$$

The Viterbi Algorithm

- Input: $\mathbf{x} = x_1, \ldots, x_T$,

Initialization:

$$V_1^k = P(x_1 \mid y_1^k = 1)\pi_k$$

Iteration:

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

$$\mathrm{Ptr}(k,t) = \arg\max_i a_{i,k} V_{t-1}^i$$

Termination:

$$P(\mathbf{x}, \mathbf{y}^*) = \max_k V_T^k$$

TraceBack:

$$y_T^* = \arg\max_k V_T^k$$

$$y_{t-1}^* = \mathrm{Ptr}(y_t^*, t)$$

Viterbi Vs. MPA (individual)

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

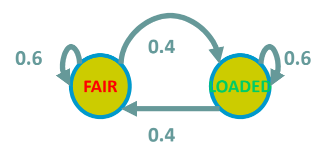
V_t^k (log)		$ptr(k,t)$		Seq	Veterbi	MPA	$p(y_t^k = 1 \mid x)$	
--2.4849	-2.9957	N/A		1	1	1	0.8128	0.1872
-4.3280	-5.3496	1	2	2	1	1	0.8238	0.1762
-6.1710	-7.7035	1	2	1	1	1	0.8176	0.1824
-8.0141	-10.0574	1	2	5	1	1	0.7925	0.2075
-9.8571	-10.8018	1	2	6	1	1	0.7415	0.2585
-11.7002	-13.1557	1	2	2	1	1	0.7505	0.2495
-13.5432	-15.5096	1	2	1	1	1	0.7386	0.2614
-15.3863	-16.2540	1	2	6	1	1	0.7027	0.2973
-17.2293	-18.6079	1	2	2	1	1	0.7251	0.2749
-19.0724	-20.9618	1	2	4	1	1	0.7251	0.2749

Another Example

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

V_t^k (log)		$ptr(k,t)$		Seq	Veterbi	MPA	$p(y_t^k = 1 \mid x)$	
-2.4849	-1.3863	N/A		6	2	2	0.2733	0.7267
-4.0943	-4.1997	2	2	2	1	1	0.6040	0.3960
-6.3969	-7.0131	1	2	3	1	1	0.6538	0.3462
-8.6995	-9.6158	1	1	5	1	1	0.6062	0.3938
-11.0021	-10.3090	1	1	6	2	2	0.2861	0.7139
-13.0170	-13.1224	2	2	2	2	1	0.5342	0.4658
-15.3196	-14.3263	1	2	6	2	2	0.2734	0.7266
-17.0344	-17.1397	2	2	3	2	1	0.5226	0.4774
-19.3370	-18.3437	1	2	6	2	2	0.2252	0.7748
-21.0518	-19.5477	2	2	6	2	2	0.2159	0.7841

max



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 Prob (x | M)
ALGO. Forward

2. Decoding

GIVEN an HMM M, and a sequence x ,
FIND the sequence $\mathbf{y}$ 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

$$P(X,y) = \prod_{i=1}^{N} P(x^{(i)}, y^{(i)})$$

$$\{\pi^*, a^*_{ij}, \eta^*_{ik}\} = \underline{\arg\max \log P(X,y)}.$$

marginal count.

$y_1 \rightarrow x_1$

$$\pi^1 = \frac{n^1}{N}.$$

$$\eta^1_1 = \frac{n^1_1}{N_1}$$

conditional count.

17

Supervised ML estimation

- Given $x = x_1 \ldots x_N$ for which the true state path $\cancel{y = y_1 \ldots y_N}$ 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:**

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

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

  - **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 …**

18

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

$X = (\qquad)$

$y = \quad ?$

$a_{ij} = \frac{A_{ij}}{\sum_j A_{ij}}$ $= \#_y (i \to j)$

$b_{ik} = \frac{\#(y=i \to x=k)}{\#(y=i)}$

$p(y_t^i \mid X)$?

Unsupervised ML estimation

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

$p(y_t \mid X)$

$p(y_t, y_{t+1} \mid X)$

- **EXPECTATION MAXIMIZATION**

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

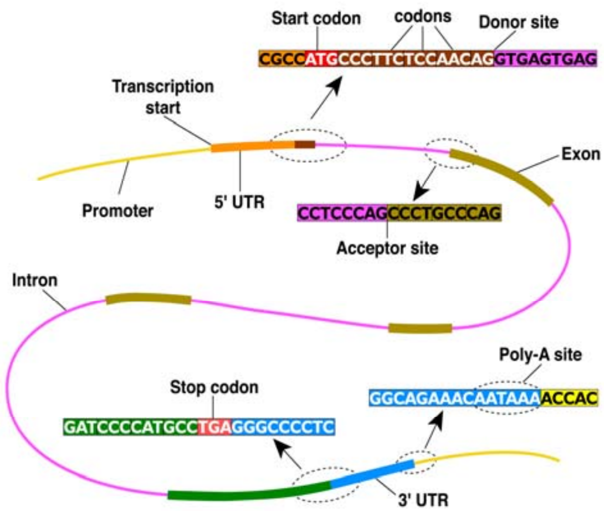
Questions:

- **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)

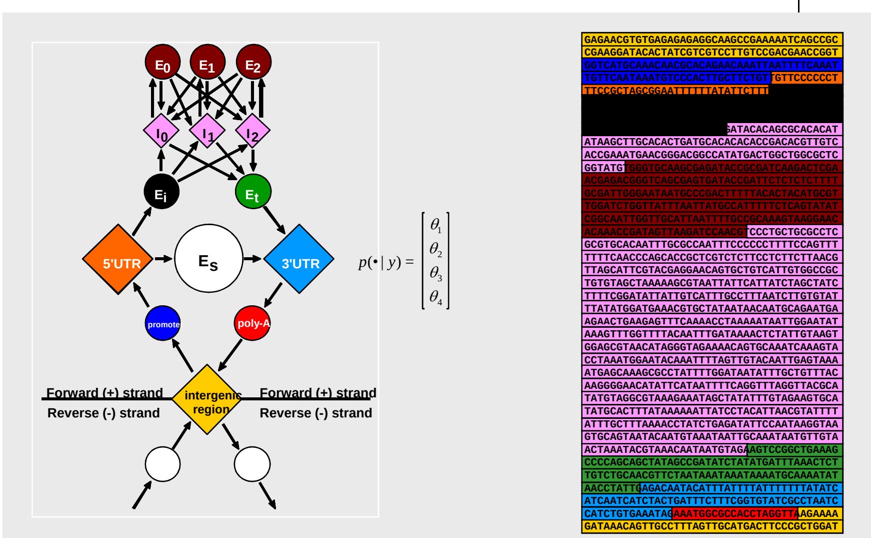
Applications of HMMs

- **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

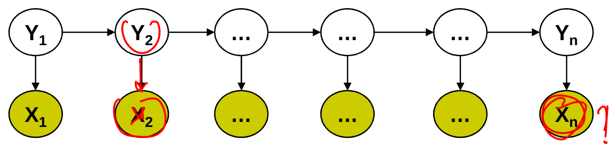


GENSCAN (Burge & Karlin)



Shortcomings of Hidden Markov Model

Structured I/O



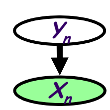
- HMM models capture dependences between each state and only its corresponding observation
  - NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line 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**)

p(x|y) p(y)
↓
p(x,y) → P(y|x)

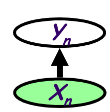
Recall Generative vs. Discriminative Classifiers

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

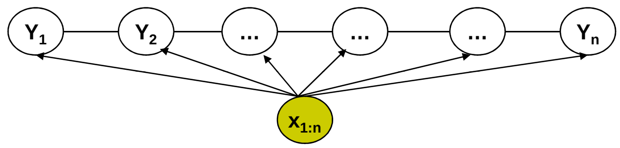
- Generative classifiers (e.g., Naïve Bayes):
  - Assume some functional form for P(X|Y), P(Y)
    This is a '***generative***' model of the data!
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X= x)



- 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

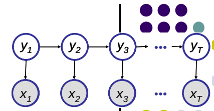


Structured Conditional Models



- 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

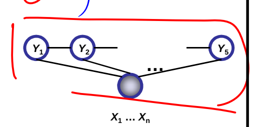
Conditional Distribution



- 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:

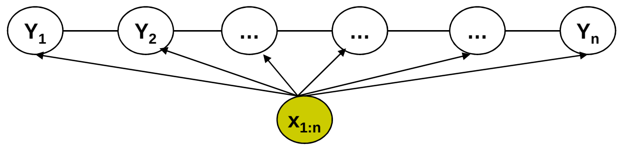
$$p_\theta(\mathrm{y} \mid \mathrm{x}) \propto \exp\left(\sum_{e \in E, k} \lambda_k f_k(e, \mathrm{y}|_e, \mathrm{x}) + \sum_{v \in V, k} \mu_k g_k(v, \mathrm{y}|_v, \mathrm{x}) \right)$$

$f(y_e, x_1, -)$



- 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

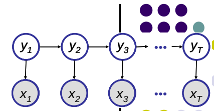


$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\phi(y_i, y_{i-1}, \mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n}, \mathbf{w})}\prod_{i=1}^{n}\exp(\mathbf{w}^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_{1:n}))$$

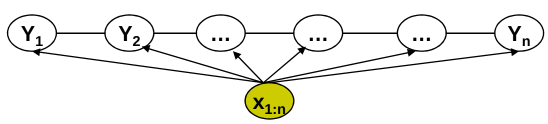
$Z = \sum_{y_1 \cdots y_n}$

- 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



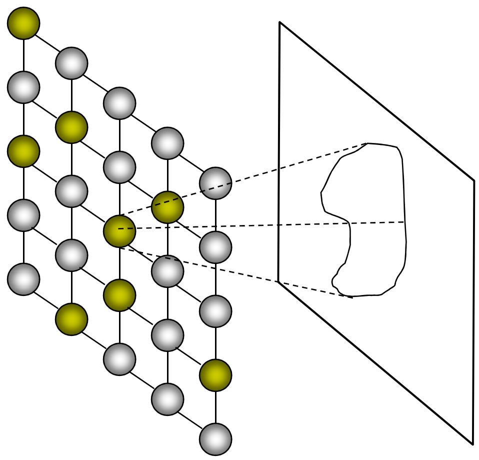
- General parametric form:



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

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

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

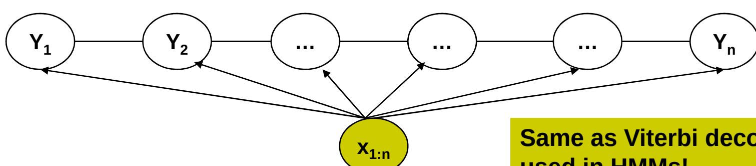
35

CRFs: Inference

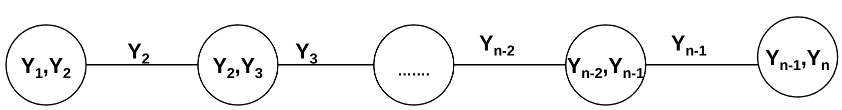
- Given CRF parameters λ and μ, find the $\mathbf{y}^*$ that maximizes P(**y**|**x**)

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

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



Same as Viterbi decoding used in HMMs!



36

CRF learning

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

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

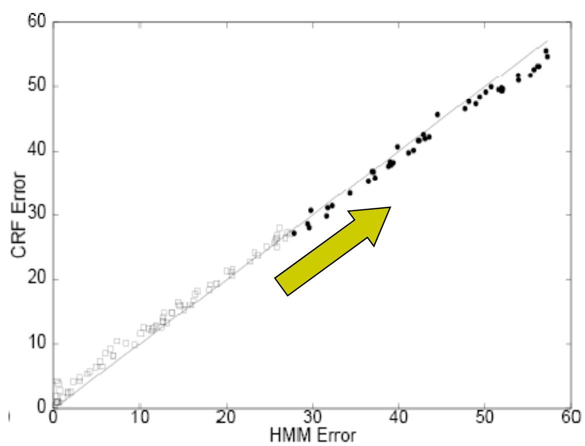
- Computing the gradient w.r.t λ:

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

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

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

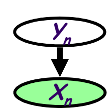
model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$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.
  …