Machine Learning

10-701/15-781, Fall 2008

Hidden Markov Model

Eric Xing

Lecture 17, March 24, 2008

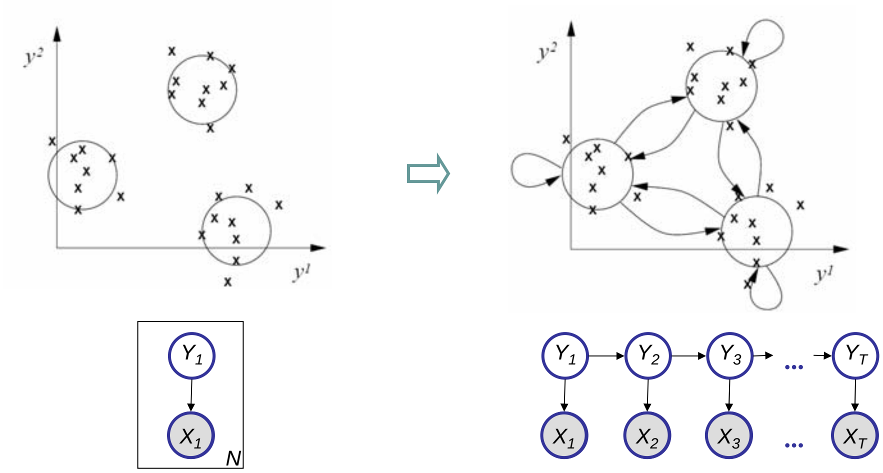
Reading: Chap. 13, C.B book

Hidden Markov Model:

from static to dynamic mixture models

Static mixture

Dynamic mixture



Hidden Markov Models

The underlying source:

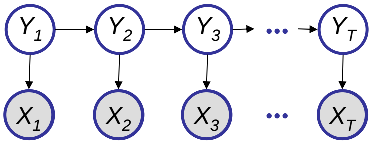
genomic entities,

dice,

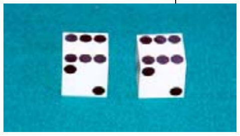
The sequence:

Ploy NT,

sequence of rolls,



Example: The Dishonest Casino



A casino has two dice:

- Fair die
  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6
- Loaded die
  P(1) = P(2) = P(3) = P(5) = 1/10
  P(6) = 1/2

Casino player switches back-&-forth between fair and loaded die once every 20 turns

Game:

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Puzzles Regarding the Dishonest Casino

GIVEN: A sequence of rolls by the casino player

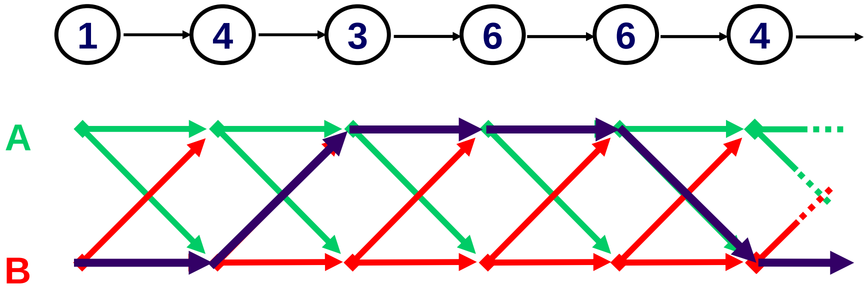
124552**6**4**6**214**6**14**6**13**6**13**666**1**66**4**66**1**6**3**66**1**6**3**66**1**6**3**6**1**6**515**6**1511514**6**1235**6**2344

QUESTION

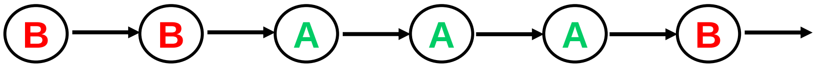
- How likely is this sequence, given our model of how the casino works?
  - This is the **EVALUATION** problem in HMMs
- What portion of the sequence was generated with the fair die, and what portion with the loaded die?
  - This is the **DECODING** question in HMMs
- How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?
  - This is the **LEARNING** question in HMMs

A Stochastic Generative Model

- Observed sequence:



- Hidden sequence (a parse or segmentation):



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

- Start probabilities

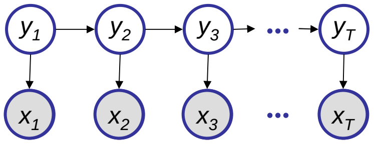
  $p(y_1) \sim \mathrm{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$

- Emission probabilities associated with each state

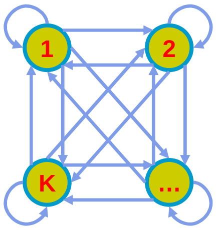
  $p(x_t \mid y_t^i = 1) \sim \mathrm{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathrm{I}.$

  or in general:

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



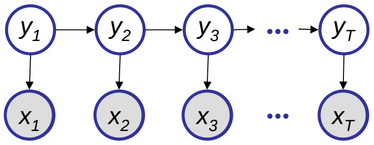
Graphical model



State automata

Probability of a Parse

- Given a sequence $\mathbf{x} = x_1 \ldots\ldots x_T$
  and a parse $\mathbf{y} = y_1, \ldots\ldots, y_T$,
- To find how likely is the parse:
  (given our HMM and the sequence)



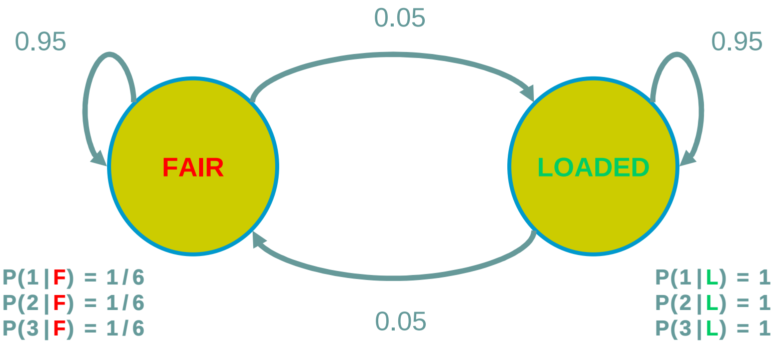
$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) &= p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \qquad \text{(Joint probability)} \\ &= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T) \\ &= p(y_1)\, \mathrm{P}(y_2 \mid y_1) \ldots p(y_T \mid y_{T-1}) \times p(x_1 \mid y_1)\, p(x_2 \mid y_2) \ldots p(x_T \mid y_T) \\ &= p(y_1, \ldots\ldots, y_T)\, p(x_1 \ldots\ldots x_T \mid y_1, \ldots\ldots, y_T) \end{aligned}$$

Let $\pi_{y_1} \overset{\text{def}}{=} \prod_{i=1}^{M} [\pi_i]^{y_1^i}$, $\quad a_{y_t, y_{t+1}} \overset{\text{def}}{=} \prod_{i,j=1}^{M} [a_{ij}]^{y_t^i y_{t+1}^j}$, $\quad$ and $\quad b_{y_t, x_t} \overset{\text{def}}{=} \prod_{i=1}^{M} \prod_{k=1}^{K} [b_{ik}]^{y_t^i x_t^k}$,

$$= \pi_{y_1} a_{y_1, y_2} \cdots a_{y_{T-1}, y_T}\; b_{y_1, x_1} \cdots b_{y_T, x_T}$$

- Marginal probability: $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_N} \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1}, y_t} \prod_{t=1}^{T} p(x_t \mid y_t)$
- Posterior probability: $p(\mathbf{y} \mid \mathbf{x}) = p(\mathbf{x}, \mathbf{y}) / p(\mathbf{x})$

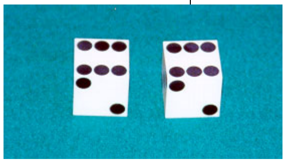
The Dishonest Casino Model



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

Example: the Dishonest Casino



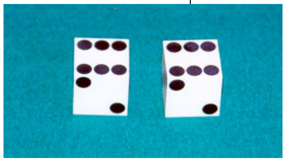
- Let the sequence of rolls be:
  - x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4
- Then, what is the likelihood of
  - y = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

    (say initial probs $a_{0Fair} = ½$, $a_{0Loaded} = ½$)

½ × P(1 | Fair) P(Fair | Fair) P(2 | Fair) P(Fair | Fair) … P(4 | Fair) =

$½ \times (1/6)^{10} \times (0.95)^{9} = .00000000521158647211 = 5.21 \times 10^{-9}$

Example: the Dishonest Casino



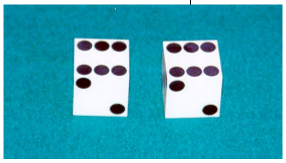
- So, the likelihood the die is fair in all this run is just 5.21×10^{-9}
- OK, but what is the likelihood of
  - π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

½ × P(1 | Loaded) P(Loaded | Loaded) … P(4 | Loaded) =

$½ \times (1/10)^{8} \times (1/2)^{2} (0.95)^{9} = .00000000078781176215 = 0.79 \times 10^{-9}$

- Therefore, it is after all 6.59 times more likely that the die is fair all the way, than that it is loaded all the way

Example: the Dishonest Casino



- Let the sequence of rolls be:
  - x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6
- Now, what is the likelihood π = F, F, …, F?
  - $\frac{1}{2} \times (1/6)^{10} \times (0.95)^9 = 0.5 \times 10^{-9}$, same as before
- What is the likelihood y = L, L, …, L?

$\frac{1}{2} \times (1/10)^4 \times (1/2)^6 (0.95)^9 = .00000049238235134735 = 5 \times 10^{-7}$

- So, it is 100 times more likely the die is loaded

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

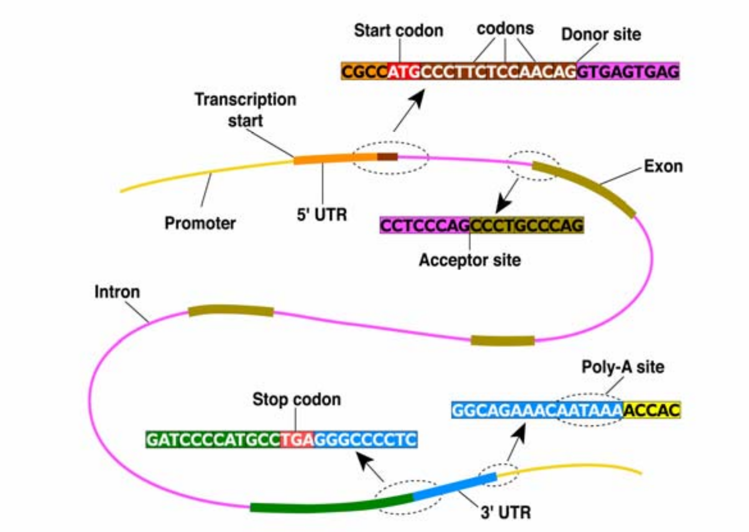
3. Learning

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 | θ)
ALGO.	Baum-Welch (EM)

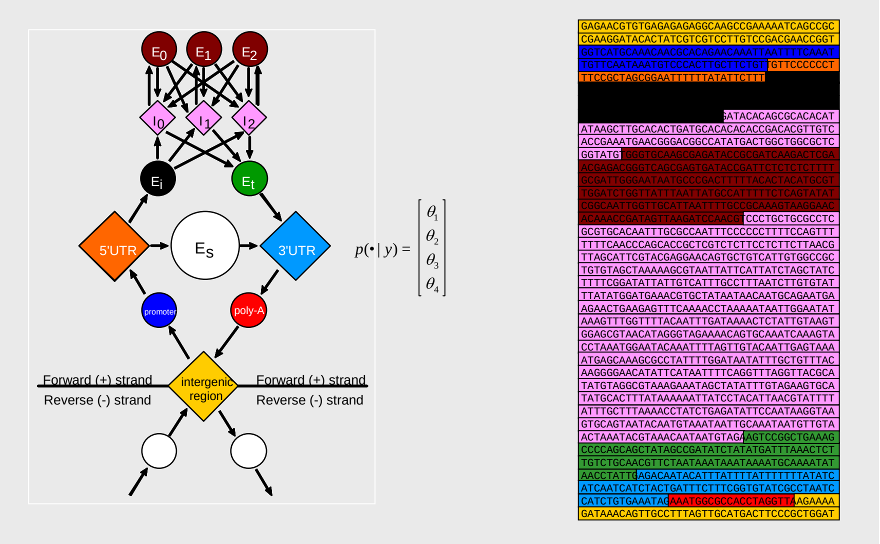
Applications of HMMs

- **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)



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)

The Forward Algorithm

- We want to calculate $P(\mathbf{x})$, the likelihood of $\mathbf{x}$, given the HMM
  - Sum over all possible ways of generating $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) = \sum_{y_1}\sum_{y_2}\cdots\sum_{y_N} \pi_{y_1}\prod_{t=2}^{T} a_{y_{t-1},y_t}\prod_{t=1}^{T} p(x_t \mid y_t)$$

  - To avoid summing over an exponential number of paths $\mathbf{y}$, define

$$\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1,\ldots,x_t, y_t^k = 1)$$

(the forward probability)

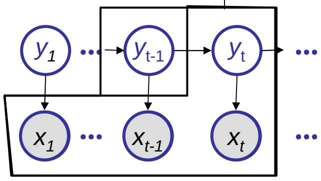
  - The recursion:

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

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Forward Algorithm – derivation

- Compute the forward probability:



$$\begin{aligned}
\alpha_t^k &= P(x_1,\ldots,x_{t-1},x_t,y_t^k=1)\\
&= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},x_t,y_{t-1},y_t^k=1)\\
&= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1 \mid y_{t-1},x_1,\ldots,x_{t-1})P(x_t \mid y_t^k=1,x_1,\ldots,x_{t-1},y_{t-1})\\
&= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1 \mid y_{t-1})P(x_t \mid y_t^k=1)\\
&= P(x_t \mid y_t^k=1)\sum_i P(x_1,\ldots,x_{t-1},y_{t-1}^i=1)P(y_t^k=1 \mid y_{t-1}^i=1)\\
&= P(x_t \mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}
\end{aligned}$$

Chain rule: $P(A,B,C) = P(A)P(B \mid C)P(C \mid A,B)$

The Forward Algorithm

- We can compute α_t^k for all k, t, using dynamic programming!

Initialization:

$$\alpha_1^k = P(x_1 \mid y_1^k = 1)\pi_k$$

$$\begin{aligned} \alpha_1^k &= P(x_1, y_1^k = 1) \\ &= P(x_1 \mid y_1^k = 1)P(y_1^k = 1) \\ &= P(x_1 \mid y_1^k = 1)\pi_k \end{aligned}$$

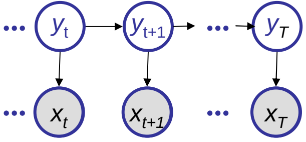
Iteration:

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

Termination:

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Backward Algorithm



- We want to compute $P(y_t^k = 1 \mid \mathbf{x})$,

  the posterior probability distribution on the t th position, given $\mathbf{x}$

  - We start by computing

$$\begin{aligned} P(y_t^k = 1, \mathbf{x}) &= P(x_1, \dots, x_t, y_t^k = 1, x_{t+1}, \dots, x_T) \\ &= P(x_1, \dots, x_t, y_t^k = 1)P(x_{t+1}, \dots, x_T \mid x_1, \dots, x_t, y_t^k = 1) \\ &= P(x_1 \dots x_t, y_t^k = 1)P(x_{t+1} \dots x_T \mid y_t^k = 1) \end{aligned}$$

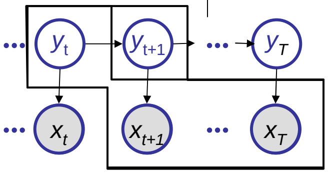
Forward, α_t^k **Backward,** $\beta_t^k = P(x_{t+1}, \dots, x_T \mid y_t^k = 1)$

  - The recursion:

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

The Backward Algorithm – derivation

- Define the backward probability:



$$\begin{aligned}
\beta_t^k &= P(x_{t+1},\ldots,x_T \mid y_t^k = 1) \\
&= \sum\nolimits_{y_{t+1}} P(x_{t+1},\ldots,x_T, y_{t+1} \mid y_t^k = 1) \\
&= \sum\nolimits_i P(y_{t+1}^i = 1 \mid y_t^k = 1)\, p(x_{t+1} \mid y_{t+1}^i = 1, y_t^k = 1)\, P(x_{t+2},\ldots,x_T \mid x_{t+1}, y_{t+1}^i = 1, y_t^k = 1) \\
&= \sum\nolimits_i P(y_{t+1}^i = 1 \mid y_t^k = 1)\, p(x_{t+1} \mid y_{t+1}^i = 1)\, P(x_{t+2},\ldots,x_T \mid y_{t+1}^i = 1) \\
&= \sum\nolimits_i a_{k,i}\, p(x_{t+1} \mid y_{t+1}^i = 1)\, \beta_{t+1}^i
\end{aligned}$$

$$\text{Chain rule: } P(A,B,C \mid \alpha) = P(A,\alpha)P(B \mid C,\alpha)P(C \mid A,B,\alpha)$$

The Backward Algorithm

- We can compute β_t^k for all k, t, using dynamic programming!

Initialization:

$$\beta_T^k = 1, \ \forall k$$

Iteration:

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

Termination:

$$P(\mathbf{x}) = \sum_k \alpha_1^k \beta_1^k$$

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

Eric Xing

25

Viterbi decoding

- 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:

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

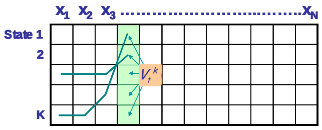
  - Let

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

= 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: $V_t^k = \log p(x_t \mid y_t^k = 1) + \max_i \left(\log(a_{i,k}) + V_{t-1}^i\right)$

Eric Xing

26

The Viterbi Algorithm – derivation

- Define the viterbi probability:

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

The Viterbi Algorithm

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

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:

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

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

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

---

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

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:**

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

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

(Homework!)

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

(Homework!)

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|θ) 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 ⇒ strong prior belief

- Small total pseudocounts: just to avoid 0 probabilities --- smoothing

---

Unsupervised ML estimation

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

  - **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? $A_{ij} = \sum_{n,t} \langle y^i_{n,t-1} y^j_{n,t} \rangle$ $B_{ik} = \sum_{n,t} \langle y^i_{n,t} \rangle x^k_{n,t}$, How? (homework)
2. Update θ according to A_{ij} , B_{ik}
   - Now a "supervised learning" problem
3. 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

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

- EM
  - The **E** step

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

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

  - 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(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