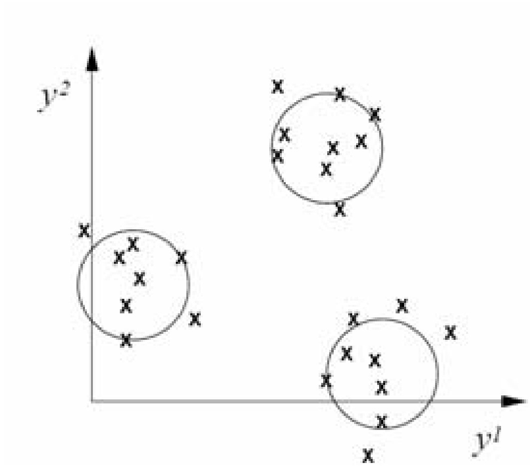
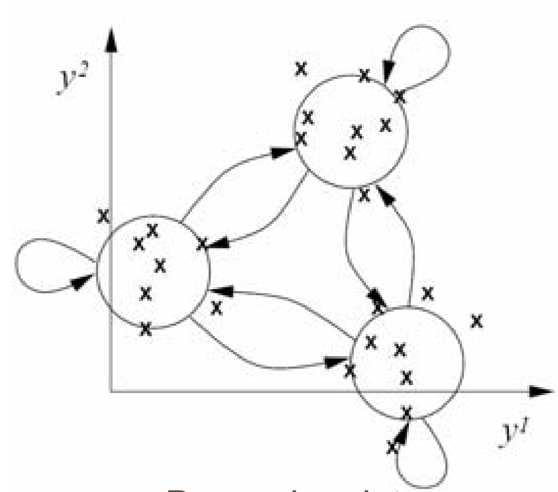
Hidden Markov Models

Adrian Barbu

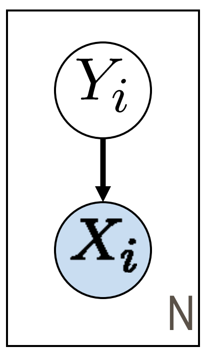
HMM: Modeling Dynamic Mixtures

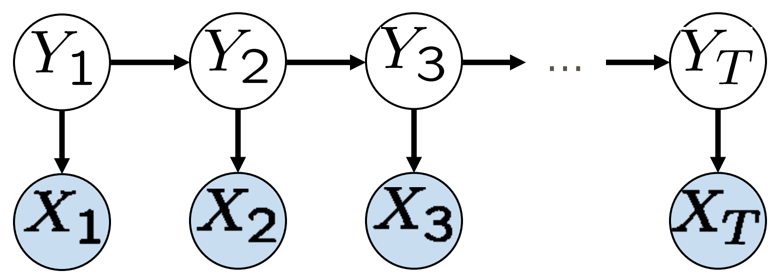


Static mixture



Dynamic mixture





- Dynamic mixture: the mixture model and parameters change in time.

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

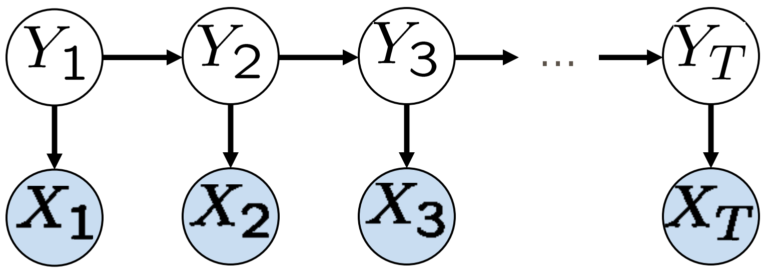
  $P(1) = P(2) = P(3) = P(5) = 1/10$ $P(6) = 1/2$

- Casino player randomly switches between fair and loaded die about once every 20 turns

Game:

1. You bet an amount X
2. You roll
   - always with a fair die
3. Casino player rolls
   - maybe with fair die, maybe with loaded die, you don't know
4. Highest number wins 2X

Hidden Markov Models



Hidden variables: the source generating the observations

Sequence of Observations

- Observations:
  - Can be measured
  - E.g.: Sound waves, sequence of dice numbers, genomic entities
- Hidden Variables:
  - Cannot be measured
  - Are responsible for the observed output
  - E.g. words that are pronounced, type of dice (fair/unfair), parts of the gene

Questions to Ask

Observations:

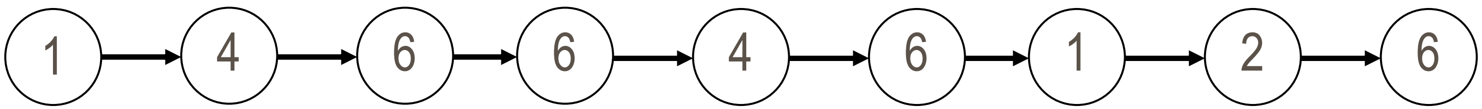
- A sequence of rolls by the casino player
- E.g. 12455264621461461361366616646616366163661636165156151151461235623 44

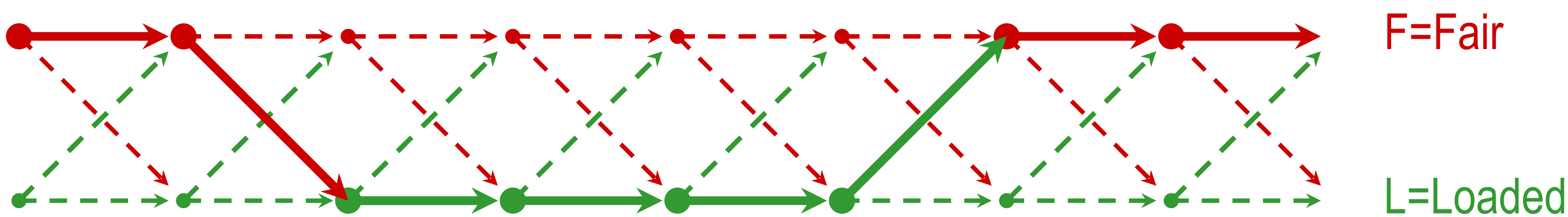
Questions

- How likely is this sequence, given our model of how the casino works?
  - This is the EVALUATION problem in HMMs
  - Evaluation of the quality of the model
- 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
  - Decoding the sequence of data
- 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
  - Learning of the model parameters

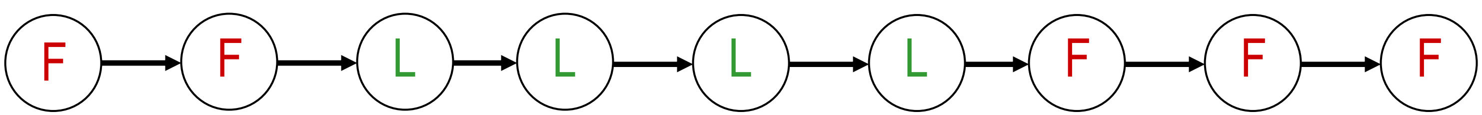
A Stochastic Generative Model

- Observed Sequence



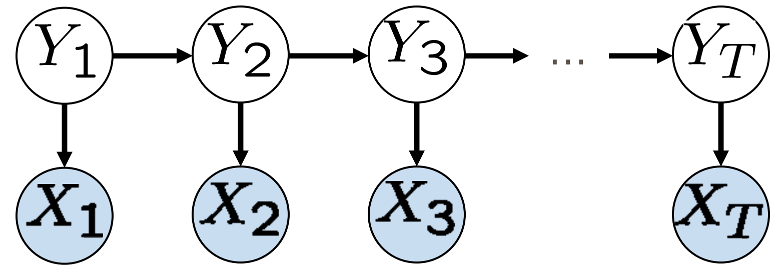


- Hidden Sequence – one (most likely) of the possible parses (segmentations)



Definition of HMM

- Observation space
  - Discrete: $C=\{c_1,\ldots,c_k\}$
  - Continuous: R^d



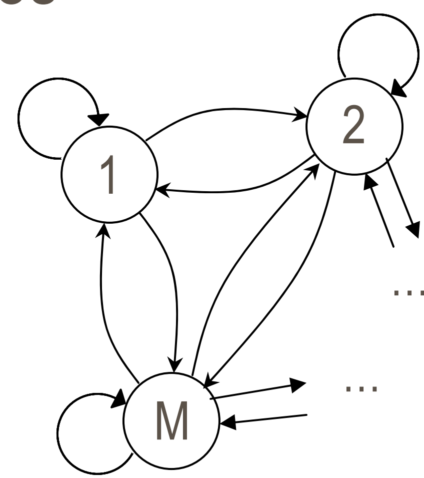
Graphical Model

- Hidden states
  - Use again indicator vectors

$$y_t = [y_t^1, \ldots, y_t^M], \sum_k y_t^k = 1, y_t^k \in \{0,1\}$$

- Transition probabilities between any two states

$$p(y_t^j = 1 | y_{t-1}^i = 1) = a_{ij}$$

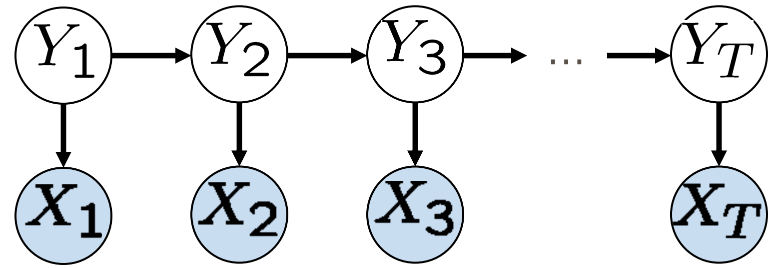


- Start probabilities $p(y_1^i = 1) = \pi_i$
- Emission probabilities associated with each state:

$$p(x_t = k | y_t^i = 1) = b_k^i$$

Probability of a Parse

- Parse=Hidden sequence
- Use the GM to find the likelihood of the parse:



$$
\begin{aligned}
p(x,y) &= p(x_1,\dots,x_T,y_1,\dots,y_T)\\
&= p(y_1)p(x_1|y_1)p(y_2|y_1)p(x_2|y_2)\dots p(y_T|y_{T-1})p(x_T|y_T)\\
&= p(y_1)p(y_2|y_1)\dots p(y_T|y_{T-1})p(x_1|y_1)p(x_2|y_2)\dots p(x_T|y_T)\\
&= p(y_1,y_2,\dots,y_T)p(x_1,x_2,\dots,x_T|y_1,y_2,\dots,y_T)
\end{aligned}
$$

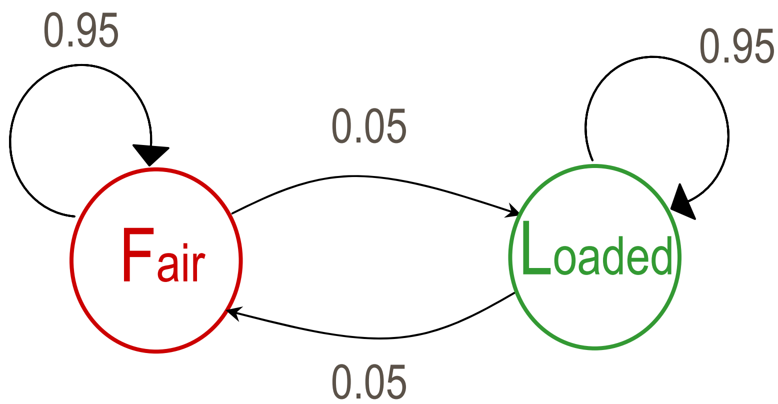
- Let $\pi_{y_1} = \prod_{i=1}^{M} \pi_i^{y_1^i},\ a_{y_t y_{t+1}} = \prod_{i,j=1}^{M} a_{ij}^{y_t^i y_{t+1}^j},\ b_{y_t x_t} = \prod_{i=1}^{M}\prod_{k=1}^{K} (b_k^i)^{y_t^i x_t^k}$
- Then

$$p(x,y) = \pi_{y_1} a_{y_1 y_2} \dots a_{y_{T-1} y_T} b_{y_1 x_1} \dots b_{y_T x_T}$$

  - Marginal probability: $p(x) = \sum_{y_1,\dots,y_T} p(x,y) = \sum_{y_1,\dots,y_T} \pi_{y_1} \prod_{t=1}^{T-1} a_{y_t y_{t+1}} \prod_{t=1}^{T} b_{y_t x_t}$
  - Posterior probability:

$$p(y|x) = p(x,y)/p(x)$$

The Dishonest Casino Model



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

- Say initial probabilities

  $$\pi_F = 1/2, \pi_L = 1/2$$

- The likelihood of

  y = (F, F, F, F, F, F, F, F, F, F)

  is $p(x,y) = 0.5 \cdot P(1|F)P(F|F)P(2|F)P(F|F)...P(4|F)$

  $$= 0.5 \cdot (1/6)^{10} \cdot (0.95)^9 = 5.21 \cdot 10^{-9}$$

- To compute p(x) we would need to sum over all 2^{10}=1024 possible parses

Example: the Dishonest Casino

- The likelihood the die is fair all the time is

  5.21 × 10-9

- The likelihood of

  Y=(L, L, L, L, L, L, L, L, L, L)

  is

$$p(x,y) = 0.5 \cdot P(1|L)P(L|L)P(2|L)P(L|L)...P(4|L)$$
$$= 0.5 \cdot (1/10)^8 \cdot (1/2)^2 \cdot (0.95)^9 = 0.79 \cdot 10^{-9}$$

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

Example: the Dishonest Casino

- Remember

  x = (1, 2, 1, 5, 6, 2, 1, 6, 2, 4)
- How about

  Y= (F, F, F, F, L, L, L, L, F, F)?

$$p(x,y) = 0.5 \cdot P(1|F)P(F|F)P(2|F)P(F|F)...P(L|F)P(6|L)P(2|L)...$$
$$= 0.5 \cdot (1/6)^6 \cdot (1/10)^2 \cdot (1/2)^2 \cdot 0.95^7 \cdot 0.05^2 = 4.68 \cdot 10^{-11}$$

which is very unlikely

Example: the Dishonest Casino

- Say now we observed:

  x = (1, 6, 4, 5, 6, 2, 6, 6, 3, 6)

- The likelihood of

  Y = (F, F, F, F, F, F, F, F, F, F) is

$$p(x, y) = 0.5 \cdot (1/6)^{10} \cdot (0.95)^9 = 5.21 \cdot 10^{-9}$$

- The likelihood of

  Y=(L, L, L, L, L, L, L, L, L, L) is

$$0.5 \cdot (1/10)^5 \cdot (1/2)^5 \cdot 0.95^9 = 9.84 \cdot 10^{-8}$$

- So, it is 20 times more likely that the die is loaded

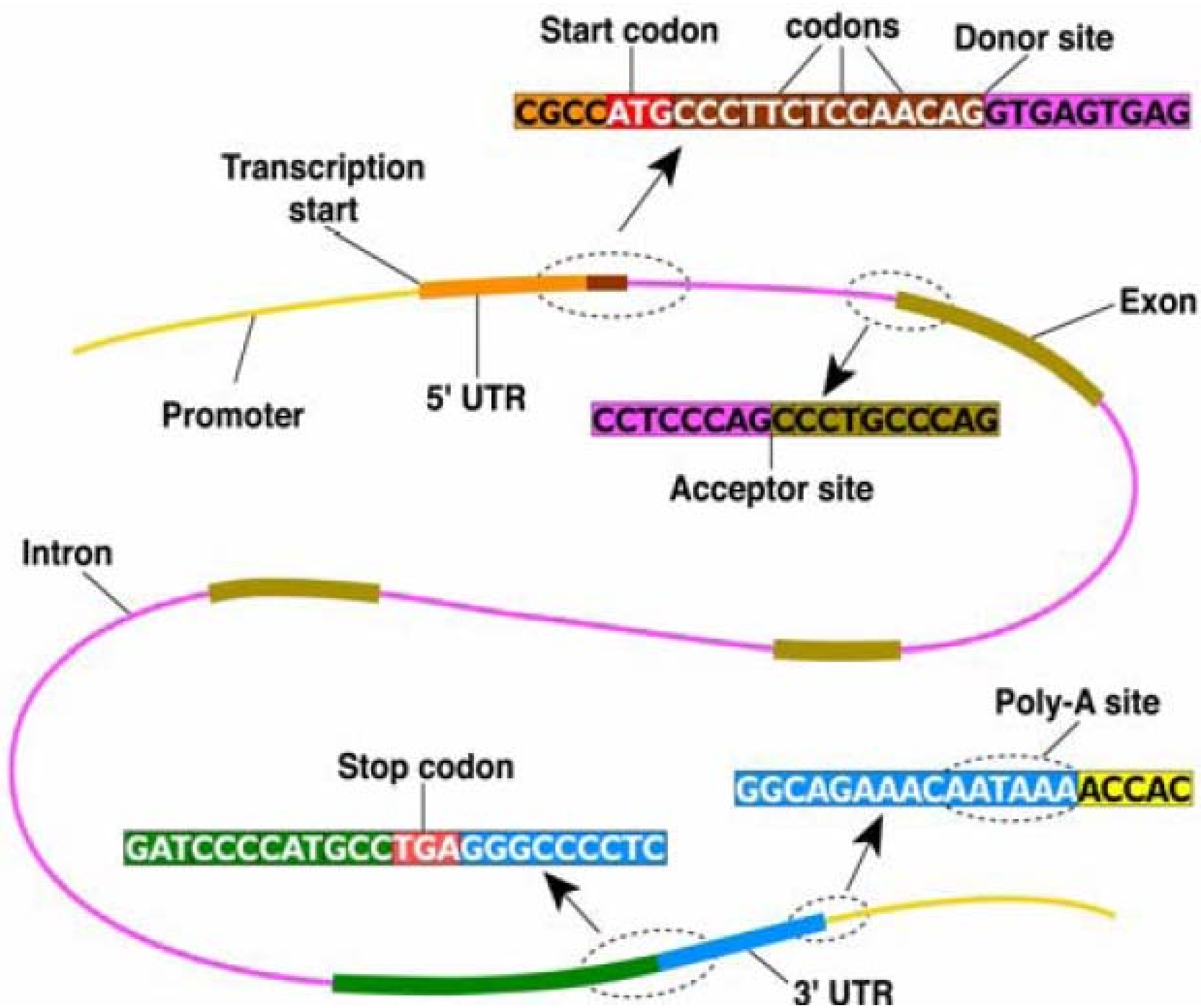
The Main Questions of HMM

1. Evaluation
   - Given: an HMM M and a sequence x
   - Find: P(x | M)
   - Algorithm: Forward
2. Decoding
   - Given: an HMM M and a sequence x
   - Find: the sequence y maximizing P(y | x, M)
   - Algorithm: Viterbi, Forward-backward
3. Learning
   - Given: an HMM M with unknown transition/emission probabilities and a sequence x
   - Find: parameters $\theta = (\pi_i, a_{ij}, b_{ik})$ that maximize P(x | θ)
   - Algorithm: Baum-Welch (EM)

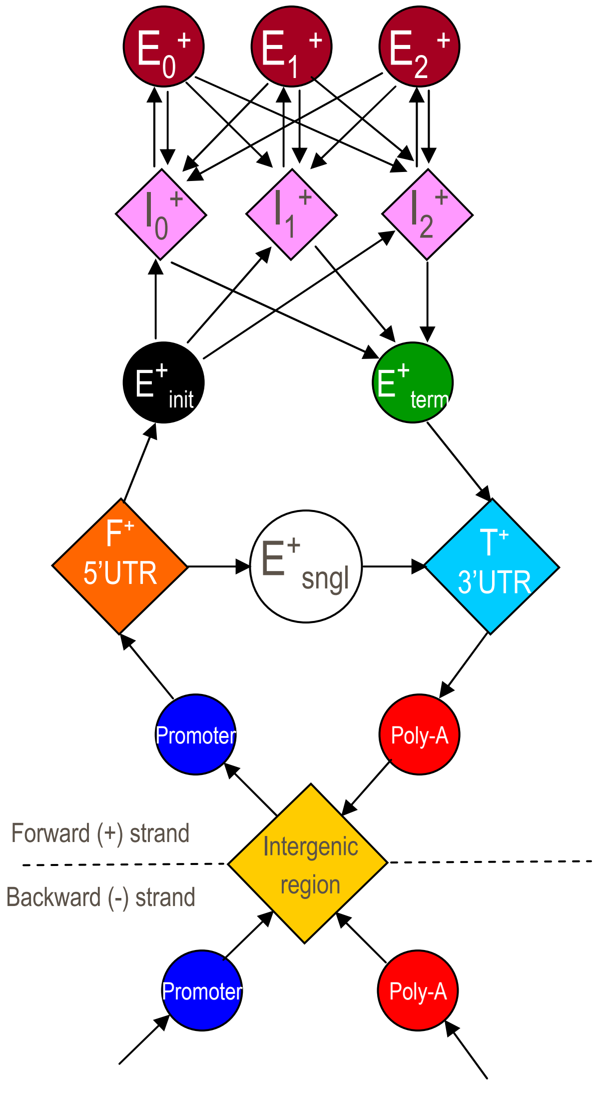
Applications of HMM

- Machine translation
- Recognition:
  - Speech recognition (since the 70s)
  - Optical character recognition
  - Sign language recognition
  - Gesture and body motion recognition
- Bioinformatics and genomics (since mid 80s)
  - Mapping chromosomes
  - Aligning biological sequences
  - Predicting sequence structure
  - Inferring evolutionary relationships
  - Finding genes in DNA sequence

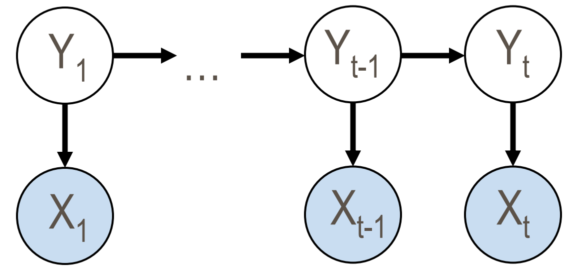
Structure of a Gene



GENSCAN (Burge & Karlin, 97)



The Forward Algorithm



- We want to calculate the likelihood p(x), given the HMM
- Marginalize over all sequences y:

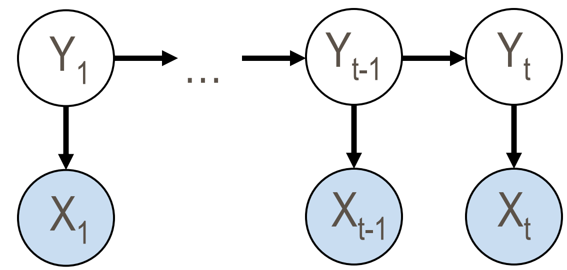
$$p(x) = \sum_{y_1,\ldots,y_T} p(x,y) = \sum_{y_1,\ldots,y_T} \pi_{y_1} \prod_{t=1}^{T-1} a_{y_t y_{t+1}} \prod_{t=1}^{T} p(x_t|y_t)$$

- Variable elimination:
  - Avoid exponential summation by memorizing:
    $\alpha_t^k = p(x_1, \ldots, x_t, y_t^k = 1)$ (the forward probability)
- Then:

$$p(x) = \sum_k p(x_1, \ldots, x_T, y_t^k = 1) = \sum_k \alpha_T^k$$

The Forward Algorithm

- We have:



$$
\begin{aligned}
\alpha_t^k &= p(x_1, ..., x_t, y_t^k = 1) \\
&= \sum_i p(x_1, ..., x_t, y_{t-1}^i = 1, y_t^k = 1) \\
&= \sum_i p(x_1, ..., x_{t-1}, y_{t-1}^i = 1, y_t^k = 1) p(x_t | x_1, ..., x_{t-1}, y_{t-1}^i = 1, y_t^k = 1) \\
&= \sum_i p(x_1, ..., x_{t-1}, y_{t-1}^i = 1, y_t^k = 1) p(x_t | y_t^k = 1) \\
&= \sum_i p(x_1, ..., x_{t-1}, y_{t-1}^i = 1) p(y_t^k = 1 | x_1, ..., x_{t-1}, y_{t-1}^i = 1) p(x_t | y_t^k = 1) \\
&= p(x_t | y_t^k = 1) \sum_i p(x_1, ..., x_{t-1}, y_{t-1}^i = 1) p(y_t^k = 1 | y_{t-1}^i = 1) \\
&= p(x_t | y_t^k = 1) \sum_i \alpha_{t-1}^i a_{ik}
\end{aligned}
$$

- Obtain recursion formula:

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

The Forward Algorithm

- Based on Dynamic Programming
- Initialization:
  - For all k

$$\alpha_1^k = p(x_1, y_1^k = 1) = p(x_1 | y_1^k = 1) p(y_1^k = 1) = b_{x_1}^k \pi_k$$

- For t=2 to T
  - Compute for all k

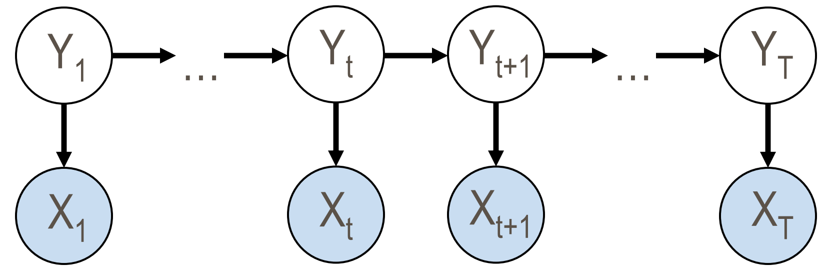
$$\alpha_t^k = b_{x_t}^k \sum_i \alpha_{t-1}^i a_{ik}$$

- Obtain the final result

$$p(x) = \sum_k \alpha_T^k$$

The Backward Algorithm

- We want $p(y_t^k = 1|x)$
- We compute



$$\begin{aligned} p(y_t^k = 1, x) &= p(x_1, ..., x_t, y_t^k = 1, x_{t+1}, ..., x_T) \\ &= p(x_1, ..., x_t, y_t^k = 1) p(x_{t+1}, ..., x_T | x_1, ..., x_t, y_t^k = 1) \\ &= p(x_1, ..., x_t, y_t^k = 1) \cdot p(x_{t+1}, ..., x_T | y_t^k = 1) \end{aligned}$$

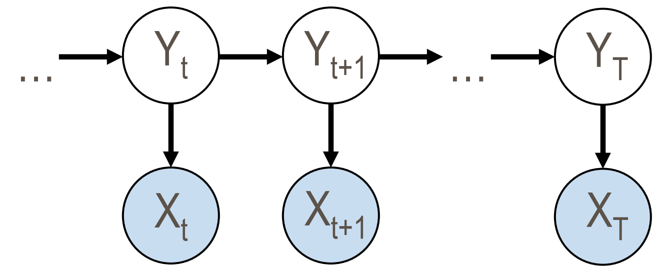
Forward α_t^k Backward β_t^k

- Define the backward probabilities

$$\beta_t^k = p(x_{t+1}, ..., x_T | y_t^k = 1)$$

The Backward Algorithm

- Recursion:



$$
\begin{aligned}
\beta_t^k &= p(x_{t+1}, \ldots, x_T | y_t^k = 1) \\
&= \sum_i p(x_{t+1}, \ldots, x_T, y_{t+1}^i = 1 | y_t^k = 1) \\
&= \sum_i p(x_{t+1}, \ldots, x_T | y_{t+1}^i = 1, y_t^k = 1) p(y_{t+1}^i = 1 | y_t^k = 1) \\
&= \sum_i p(x_{t+1}, \ldots, x_T | y_{t+1}^i = 1) p(y_{t+1}^i = 1 | y_t^k = 1) \\
&= \sum_i p(x_{t+2}, \ldots, x_T | x_{t+1}, y_{t+1}^i = 1) p(x_{t+1} | y_{t+1}^i = 1) p(y_{t+1}^i = 1 | y_t^k = 1) \\
&= \sum_i p(x_{t+2}, \ldots, x_T | y_{t+1}^i = 1) p(x_{t+1} | y_{t+1}^i = 1) p(y_{t+1}^i = 1 | y_t^k = 1) \\
&= \sum_i a_{ki} \beta_{t+1}^i p(x_{t+1} | y_{t+1}^i = 1)
\end{aligned}
$$

- Recursion formula:

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

The Backward Algorithm

- Dynamic Programming again

- Initialization
  - For all k, $\beta_T^k = 1$
- For t=T-1,…,1
  - For all k compute

$$\beta_t^k = \sum_i a_{ki} \beta_{t+1}^i b_{x_{t+1}}^i$$

- Obtain: $p(y_t^k = 1, x) = \alpha_t^k \beta_t^k$

$$p(y_t^k = 1 | x) = \frac{p(y_t^k = 1, x)}{p(x)} = \frac{\alpha_t^k \beta_t^k}{\sum_i \alpha_T^i}$$

Posterior Decoding

- We have $p(y_t^k = 1, x) = \alpha_t^k \beta_t^k$

$$p(y_t^k = 1|x) = \frac{p(y_t^k = 1, x)}{p(x)} = \frac{\alpha_t^k \beta_t^k}{\sum_i \alpha_T^i}$$

- We can ask:
  - What is the most likely state at position t of sequence x?

$$k_t^* = \arg\max_k p(y_t^k = 1|x)$$

  - MPA (Most Probable Assignment) of a single hidden state
  - Posterior Decoding:

$$\{y_t^{k_t^*} = 1, t = 1, ..., T\}$$

  - Different from MPA of a whole sequence of hidden states

Viterbi Decoding

- Given x, we want the most probable y

$$y^* = \arg\max_y p(y|x) = \arg\max_\pi p(x, y)$$

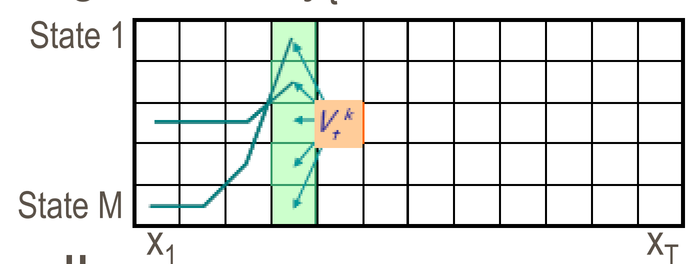
- Let

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

  - Probability of most likely sequence y ending at state y_t=k

- Recursion formula:

$$V_t^k = p(x_t|y_t^k = 1) \max_i a_{ik} V_{t-1}^i$$



- Since products become smaller and smaller
  - Use log to get additive recursion

$$C_t^k = \log V_t^k$$

$$C_t^k = \log p(x_t|y_t^k = 1) + \max_i(\log a_{ik} + C_{t-1}^i)$$

Viterbi Recursion Formula

- We have:

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

The Viterbi Algorithm

- Again similar to Dynamic Programming
- Initialization

$$C_1^k = \log(b_{x_1}^k \pi_k)$$

- For t=2 to T
  - For all k

$$C_t^k = \log(b_{x_t}^k) + \max_i[\log(a_{ik}) + C_{t-1}^i]$$

$$Ptr_t^k = \arg\max_i[\log(a_{ik}) + C_{t-1}^i]$$

- Obtain: $P(x, y^*) = \exp(\max_k C_T^k)$

$$y_T^* = \arg\max_i C_T^k$$

- Trace back to obtain the full y*

$$y_{t-1}^* = Ptr_t^{y_t^*}$$

Computational Complexity

- Algorithms:
  - Forward
    $$\alpha_t^k = b_{x_t}^k \sum_i \alpha_{t-1}^i a_{ik}$$
  - Backward
    $$\beta_t^k = \sum_i a_{ki} \beta_{t+1}^i b_{x_{t+1}}^i$$
  - Viterbi
    $$V_t^k = b_{x_t}^k \max_i a_{ik} V_{t-1}^i$$
- Running time: $O(M^2T)$
- Memory: $O(MT)$
- Implementation tricks:
  - Multiply all α_t^k ,β_t^k by the same constant at each time step t for the Forward and Backward algorithms to avoid floating point underflow
  - Use log probability for Viterbi

Learning HMM

- **Supervised learning:** learn the model parameters θ when given a sequence of observations and hidden variables (x,y)
  - E.g: Learn the model parameters when the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls
- **Unsupervised learning:** learn the model parameters θ when only the observations x are given
  - E.g: Learn the model parameters when observed 10,000 rolls of the casino player, but we don't see when he changes dice
- Approach: MLE (Maximum likelihood estimation)
  - Update the parameters θ of the model to maximize $P(x,y|\theta)$ or $P(x|\theta)$

Supervised Learning for HMM

- Know $x=(x_1,\ldots,x_n)$ and $y=(y_1,\ldots,y_n)$
- Define:
  - A_{ij} = # times y changes from state i to state j
  - B_{ik} = # times y is in state i while x is in state k
- We can show that the ML parameters θ are:

$$a_{ij}^{ML} = \frac{A_{ij}}{\sum_l A_{il}}$$

$$b_{ik}^{ML} = \frac{B_{ik}}{\sum_l B_{il}}$$

Supervised Learning for HMM

Intuition:

- When we know the hidden states, the MLE estimate of θ is the average frequency of transitions & emissions that occur in the training data

Drawback:

- Need a lot of training data to populate all entries
  - Prone to overfitting.
  - If not enough data → 0 probabilities – 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)
- Get:
  - $a_{FF} = 1$, $a_{FL} = 0$
  - $b_{F1} = b_{F3} = .2$, $b_{F2} = .3$, $b_{F4} = 0$, $b_{F5} = b_{F6} = .1$
- From 10 observations we learned that die is always fair and that 4 is never rolled

Pseudocounts

- To avoid ovefitting, add pseudocounts:
  - A_{ij} = # times y changes from state i to state j $+R_{ij}$
  - B_{ik} = # times y is in state i while x is in state k $+S_{ik}$
  - R_{ij}, S_{ij} are pseudocounts representing our prior belief
- Total pseudocounts:

$$R_i = \sum_j R_{ij}, S_i = \sum_k S_{ik}$$

  - strength of the bias (prior belief)
- Larger total pseudocounts → strong prior belief
- Small total pseudocounts, to avoid 0 probabilities→smoothing

Unsupervised HMM Learning

- Given x = ($x_1,\ldots,x_N$) for which y is unknown

Expectation Maximization

- Define the soft assignment probabilities for y_i

$$\xi_t^{ij} = p(y_t^i = 1, y_{t+1}^j = 1 | x)$$
$$\gamma_t^i = p(y_t^i = 1 | x) = \sum_j \xi_t^{ij}$$

1. Initialization: Guess the model parameters θ=(π,A,B)
2. E step: Update ξ_t^{ij}, γ_t^i based on A,B
3. M step: Update θ based on ξ_t^{ij}, γ_t^i
   - MLE estimation as in the supervised learning HMM
4. Repeat 2 & 3 until convergence

This is the Baum-Welch Algorithm

The Baum-Welch Algorithm

- E step:
  - Compute α^i_t, β^j_{t+1} using the forward-backward algorithms
  - It can be proved that:

$$\xi_t^{ij} = \frac{\alpha_t^i a_{ij} \beta_{t+1}^j b_{x_{t+1}}^j}{\sum_{k,l} \alpha_t^k a_{kl} \beta_{t+1}^l b_{x_{t+1}}^l}, \quad \gamma_t^i = \frac{\alpha_t^i \beta_t^i}{\sum_k \alpha_t^k \beta_t^k}$$

- M step:
  - MLE update of parameters:

$$\pi_i = \gamma_1^i$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t^{ij}}{\sum_{t=1}^{T-1} \gamma_t^i}$$

$$b_k^i = \frac{\sum_{t=1}^{T} \delta(x_t = k) \gamma_t^i}{\sum_{t=1}^{T} \gamma_t^i}$$

The Baum-Welch Algorithm

- Time Complexity:
  - # iterations × $O(M^2T)$
  - M= nr labels for Y
  - T= sequence length
- EM Algorithm:
  - Not guaranteed to find globally optimal solution
  - Converges to local optimum, depending on initial conditions
- Too many parameters / too few examples → overfitting

Conclusions

- Hidden Markov Models
  - A simple, directed Graphical Model for dynamic mixtures
    - Observations change in time according to some stochastic rules
  - Intensely applied in
    - Speech recognition
    - Bioinformatics and genomics
  - Issues:
    - Model evaluation = how well the model fits the data
    - Decoding = finding most likely hidden states
    - Learning = finding the model parameters
      - Supervised
      - Unsupervised

Discussion Points

- What is the difference between a HMM and a mixture model?
- What are the parameters of a HMM?
- What is the forward algorithm used for?
- What is the backward algorithm used for?
- What is the Viterbi algorithm used for?
- How do we do MPA for a single state?
- How do we learn HMM?