80250993 Machine Learning
@Tsinghua University

Chapter 12
Hidden Markov Models and Graphic Models

Xuegong Zhang
November 11, 2021

12.1
Hidden Markov Models (HMMs)

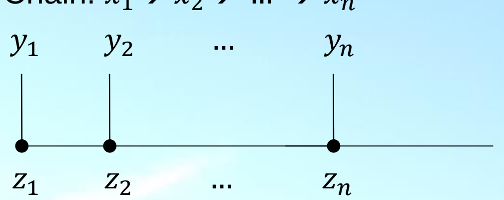
Durbin et al, Biological sequence analysis: probabilistic models of proteins and nucleic acids, Cambridge University Press, 1998

- A motivation from the CpG island example:
  - How can we detect possible CpG islands from a long sequence?
    - Method 1: Cut the sequence into overlapping windows and discriminate the sequence in each window

$$S(x) = \log\frac{P(x|+)}{P(x|-)} = \sum_{i=1}^{L} \log\frac{a^{+}_{x_{i-1}x_i}}{a^{-}_{x_{i-1}x_i}} = \sum_{i=1}^{L} \boldsymbol{\beta}_{x_{i-1}x_i}$$

    - Method 2: Build a single model for the entire sequence that incorporates both Markov chains (of CpG and non-CpG)

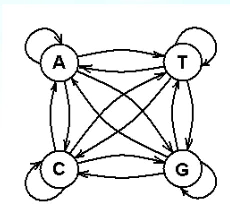
- Markov Chain: $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$
- HMM:

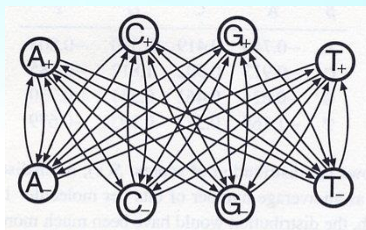


  - Only the y's are observed
  - The z’s are governed by a Markov transition rule
  - The y can be viewed as a "*noisy copy*" of the z
  - Examples:
    - Gene Recognition: z: intron or exon, y: observed sequence
    - CpG island detection: z: A+,C+,T+,G+,A-,C-,T-,G-, y: observed seq: A C T G

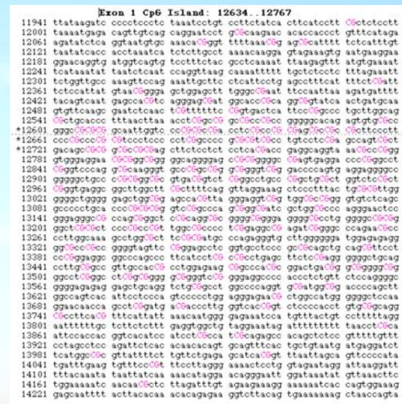
- Model: the "islands" in a "sea" of non-island genomic sequence
  - Both chains in the same model
  - With a small probability of switching
- We now have two states corresponding to each nucleotide symbol
  - Solution: re-labelling the states

  z: A+,C+,T+,G+,A-,C-,T-,G-;   y: observed: A C T G

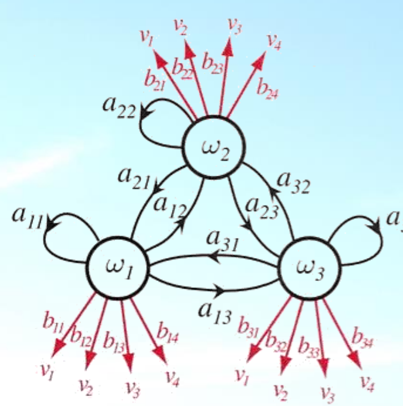




Note: all transitions within the same set are omitted in the plot.



Hidden Markov Models (HMMs)

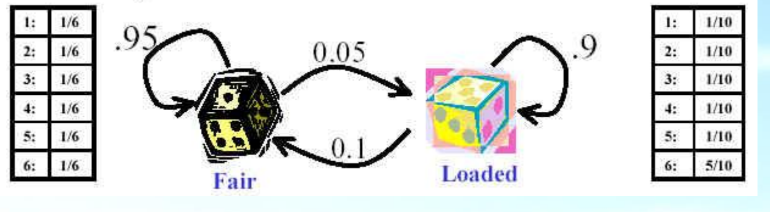


- A sequence of hidden states (of a Markov chain)

→ A sequence of visible symbols emitted by underlying hidden states (of a Markov chain)

Example: the Casino Game

Example:
The occasionally dishonest casino

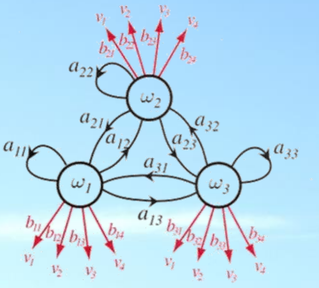


1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 1 6 2 1 6 6 5 6 6 6 3 5 2 3 2 1 2 6 4 6 2 2 5 3 3 3 1 4 3 1 5 1 3 6 1 6 3 5 1 6 3 1 2 3 1 4 6 3 6

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2
5 1 3 3 5 6 1 3 5 5 4 6 3 2 4 1 6 2 5 4 2 4 4 2 1 2 3 2 6 3 6 6 6 4 5 6 2 2 4 6 6 1 4 6 3 4 2 6 4 6

- Hidden states: whether the die is fair or loaded
- Observed sequence: a sequence of numbers of the rolls

HMM: definitions



- Two sequences:
  - Sequence of symbols (x) --- observations
  - Sequence of states (the path, π) --- hidden
- The path itself follows a simple Markov chain
  - Transition probability: $a_{kl} = P(\pi_i = l|\pi_{i-1} = k)$
- A state produces a symbol from a distribution over all possible symbols
  - Emission probability: $e_k(b) = P(x_i = b|\pi_i = k)$
- Probability of the sequence and the path

$$P(x,\pi) = a_{0\pi_1}\prod_{i=1}^{L} e_{\pi_i}(x_i)a_{\pi_i\pi_{i+1}}$$

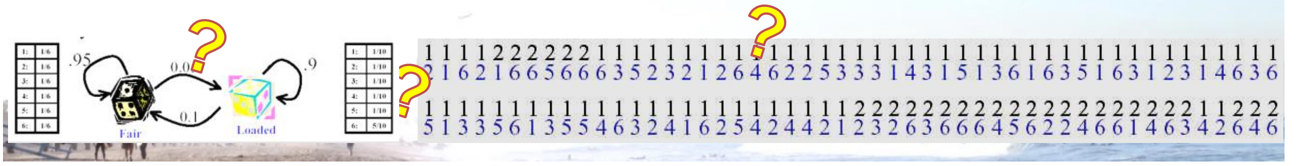
The model

But the path is hidden.

D

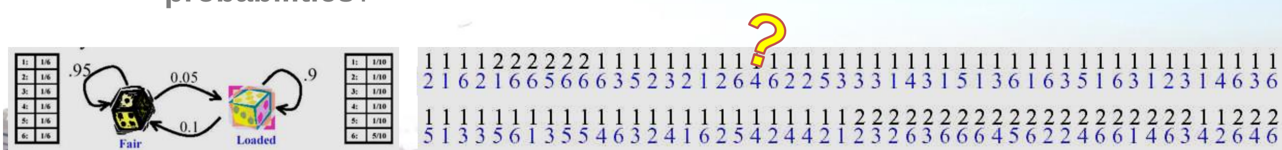
Three Central Problems in HMM

- The Evaluation Problem
  - Given an HMM, complete with transition and emission probabilities, how to determine **the probability of a particular sequence of symbols** being generated by that model?
- The Decoding Problem
  - Given an HMM as well as a set of observations, how to determine the **most likely sequence of hidden states** that led to those observations?
- The Learning Problem
  - Given the coarse structure of the model but ***not*** the probabilities, and given a set of training observations of symbols, how to determine **the probabilities**?



Three Central Problems in HMM

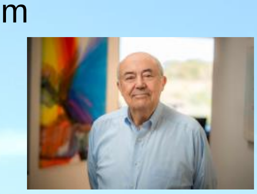
- The Evaluation Problem
  - Given an HMM, complete with transition and emission probabilities, how to determine **the probability of a particular sequence of symbols** being generated by that model?
- The Decoding Problem
  - Given an HMM as well as a set of observations, how to determine the **most likely sequence of hidden states** that led to those observations?
- The Learning Problem
  - Given the coarse structure of the model but ***not*** the probabilities, and given a set of training observations of symbols, how to determine **the probabilities**?



The most probable state path
(the *decoding* problem)
--- the Viterbi algorithm

$\boldsymbol{x}$: CGCG

$\boldsymbol{\pi}$: C+ G+ C+ G+ ?
C− G− C− G− ?
C+ G− C+ G− ?


Andrew J. Viterbi

Given the x and the model, what is the state path π^*?

$\pi^* = \arg\max_{\pi} P(x,\pi)$ --- can be found recursively by DP (**dynamic programming**)

- $v_k(i)$: the probability of the most probable path ending in state k with observation x_i
- If $v_k(i)$ is known, then

$$v_l(i+1) = e_l(x_{i+1}) \max_k (v_k(i) a_{kl})$$

Viterbi Algorithm:

- Initialization $(i = 0)$: $v_0(0) = 1,\ v_k(0) = 0$ for $k > 0$
- Recursion $(i = 1, \cdots, L)$: $v_l(i) = e_l(x_i)\max_k(v_k(i-1)a_{kl})$ for any l

  $ptr_i(l) = \text{argmax}_k(v_k(i-1)a_{kl})$

- Termination: $P(x, \pi^*) = \max_k(v_k(L)a_{k0})$

  $\pi_L^* = \text{argmax}_k(v_k(L)a_{k0})$

- Traceback $(i = L, \cdots, 1)$: $\pi_{i-1}^* = ptr_i(\pi_i^*)$

  --- the most probable hidden state sequence

$ptr_i(l)$: pointer backwards to remember the most probable state at time $i-1$ if the state at i is l.

A computational note:

- too small numbers: numerical problem
- solution: taking log( )

Examples

x: CGCG

π: C+ G+ C+ G+ ?

C− G− C− G− ?

C+ G− C+ G− ?

v		C	G	C	G
$\mathcal{B}$	1	0	0	0	0
A_+	0	0	0	0	0
C_+	0	**0.13**	0	**0.012**	0
G_+	0	0	**0.034**	0	**0.0032**
T_+	0	0	0	0	0
A_-	0	0	0	0	0
C_-	0	0.13	0	0.0026	0
G_-	0	0	0.010	0	0.00021
T_-	0	0	0	0	0

The occasionally dishonest casino:

2 1 6 2 1 6 6 5 6 6 6 3 5 2 3 2 1 2 6 4 6 2 2 5 3 3 3 1 4 3 1 5 1 3 6 1 6 3 5 1 6 3 1 2 3 1 4 6 3 6
2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

5 1 3 3 5 6 1 3 5 5 4 6 3 2 4 1 6 2 5 4 2 4 4 2 1 2 3 2 6 3 6 6 6 4 5 6 2 2 4 6 6 1 4 6 3 4 2 6 4 6
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2

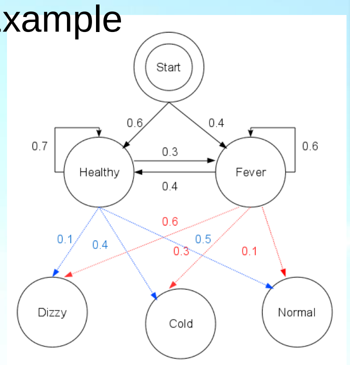
清华大学

```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls   651166453132651245636664631636663162326455236266666625151631
Die     LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls   222555441666566563564324364131513465146353411126414626253356
Die     FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLL
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466662632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLFFFFFFFFFFFF
```

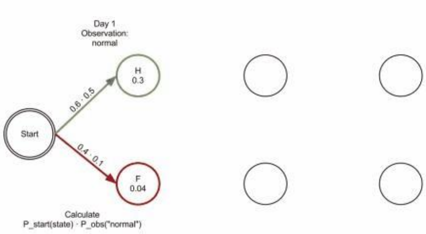
Figure 3.5 *The numbers show 300 rolls of a die as described in the example. Below is shown which die was actually used for that roll (F for fair and L for loaded). Under that the prediction by the Viterbi algorithm is shown.*

清华大学

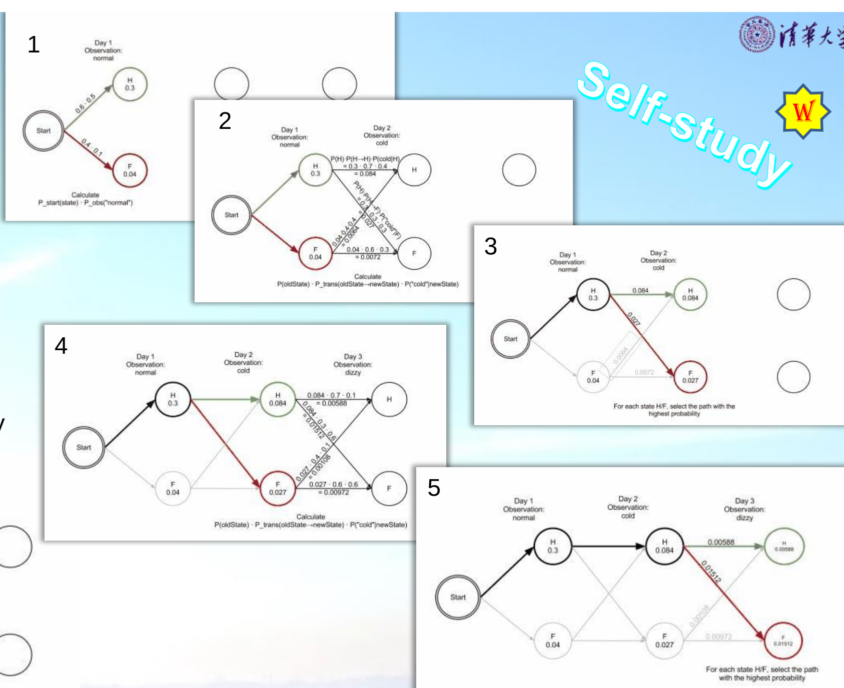
Example



Observation: Normal→Cold→Dizzy
States: ? → ? → ?



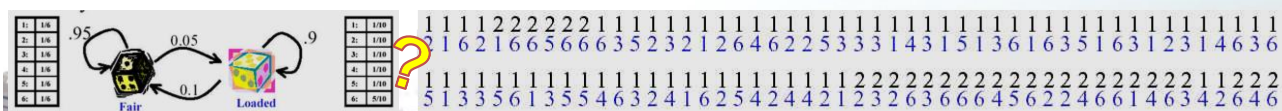
Self-study



https://en.wikipedia.org/wiki/Viterbi_algorithm

Three Central Problems in HMM

- The Evaluation Problem
  - Given an HMM, complete with transition and emission probabilities, how to determine **the probability of a particular sequence of symbols** being generated by that model?
- The Decoding Problem
  - Given an HMM as well as a set of observations, how to determine the **most likely sequence of hidden states** that led to those observations?
- The Learning Problem
  - Given the coarse structure of the model but ***not*** the probabilities, and given a set of training observations of symbols, how to determine **the probabilities**?



The probability of a sequence
(the *evaluation* problem)

- The Viterbi algorithm: *the most probable state path*
- The evaluation problem: the probability $P(x)$ of a sequence x under the model
  - Many different state paths can give rise to the same sequence

$$P(x) = \sum_{\pi} P(x,\pi) \qquad P(x,\pi) = a_{0\pi_1}\prod_{i=1}^{L} e_{\pi_i}(x_i)a_{\pi_i\pi_{i+1}}$$

  - Enumerating all paths is not practical in many cases
  - One simplified approximation:

    To use the joint probability at *the most probable path* π^* as an approximation to $P(x)$:

$$P(x,\pi^*) = a_{0\pi_1^*}\prod_{i=1}^{L} e_{\pi_i^*}(x_i)a_{\pi_i^*\pi_{i+1}^*}$$

  - Implicit assumption: π^* is the only path with significant probability

--- "startling but surprisingly good."

Can we calculate the full probability?

The Forward Algorithm

- The full probability can be calculated through **dynamic programming**

$$P(x) = \sum_{\pi} P(x, \pi)$$

$f_k(i)$: the probability of the observed sequence up to and including x_i, requiring $\pi_i = k$

$$f_k(i) = P(x_1 \cdots x_i, \pi_i = k)$$

$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}$$

Forward Algorithm:

Initialization $(i = 0)$: $f_0(0) = 1,\ f_k(0) = 0$ for $k > 0$

Recursion$(i = 1, \cdots, L)$: $f_l(i) = e_l(x_i) \sum_k (f_k(i-1) a_{kl})$ for any l

Termination: $P(x) = \sum_k f_k(L) a_{k0}$

Questions answered

- What is the most likely state sequence (given the HMM)?
  - Viterbi Algorithm: the decoding problem
- What is the probability of the observed sequence (under the HMM)?
  - Forward Algorithm: the evaluation problem

One more decoding question

- What is the probability that the ith state is k given the observed sequence (*the posterior probability*), i.e. $P(\pi_i = k|x)$?

(a special version of the decoding problem)

→ Backward Algorithm

The Backward Algorithm

$$
\begin{aligned}
P(x, \pi_i = k) &= P(x_1 \cdots x_i, \pi_i = k) P(x_{i+1} \cdots x_L | x_1 \cdots x_i, \pi_i = k) \\
&= P(x_1 \cdots x_i, \pi_i = k) P(x_{i+1} \cdots x_L | \pi_i = k) \\
&= f_k(i)\ b_k(i)
\end{aligned}
$$

$$f_k(i) = P(x_1 \cdots x_i, \pi_i = k)$$

$$b_k(i) = P(x_{i+1} \cdots x_L | \pi_i = k)$$

Backward Algorithm:

- Initialization $(i = L)$: $b_k(L) = a_{k0}$ for all k
- Recursion$(i = L-1, \cdots, 1)$: $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$ for any k
- Termination: $P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$

$$P(\pi_i = k | x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) b_k(i)}{P(x)}$$

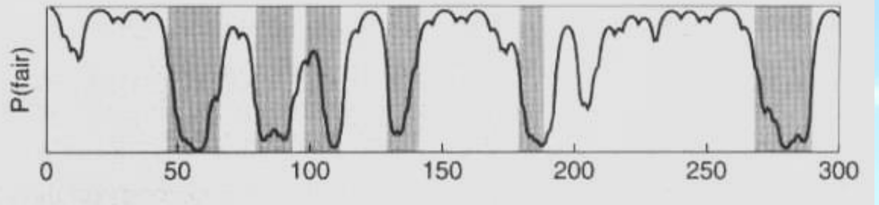


Figure 3.6 *The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.*

Posterior Decoding

- If it is not the state sequence itself which is of interest, but some other property derived from it, new decoding approach can arise.
- Assume we have a function $g(k)$ defined on the states, then we can look at

$$G(i|x) = \sum_k P(\pi_i = k|x)g(k) \quad \text{: posterior probability of } g(i)$$

- An important special case:
  - $g(k) \in \{0,1\}$ represents two subsets of the states
  - $G(i|x)$ is the posterior probability of the symbol i coming from a state of one of the two classes
    - e.g. for the CpG island problem, we can define $g(k) = 1$ for {A+,C+,G+,T+} and $g(x) = 0$ for {A-,C-,G-,T-} . Then $G(i|x)$ is the posterior probability according to the model that base i is in a CpG island.

1-minute break

Three Central Problems in HMM

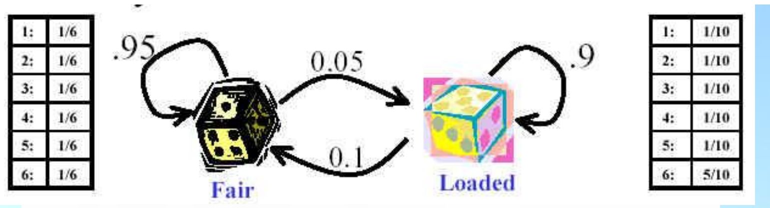
- The Evaluation Problem
  - Given an HMM, complete with transition and emission probabilities, how to determine **the probability of a particular sequence of symbols** being generated by that model?
- The Decoding Problem
  - Given an HMM as well as a set of observations, how to determine the **most likely sequence of hidden states** that led to those observations?
- The Learning Problem
  - <u>Given the coarse structure of the model but ***not*** the probabilities, and given a set of training observations of symbols</u>, how to determine **the probabilities**?

1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 1 6 2 1 6 6 5 6 6 6 3 5 2 3 2 1 2 6 4 6 2 2 5 3 3 3 1 4 3 1 5 1 3 6 1 6 3 5 1 6 3 1 2 3 1 4 6 3 6

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2
5 1 3 3 5 6 1 3 5 5 4 6 3 2 4 1 6 2 5 4 2 4 4 2 1 2 3 2 6 3 6 6 6 4 5 6 2 2 4 6 6 1 4 6 3 4 2 6 4 6

Modeling HMM
(the *learning* problem)

- The most difficult problem faced when using HMMs is that of specifying the model in the first place.
  - The assignment of parameter values: the transition and emission probabilities a_{kl}, $e_k(b)$
  - The design of the structure, i.e. what states there are and how they are connected



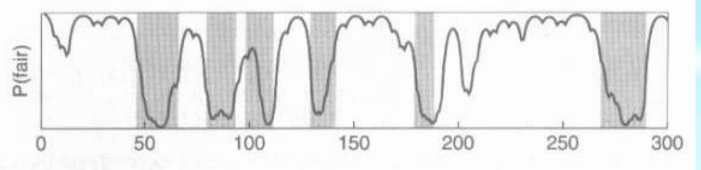
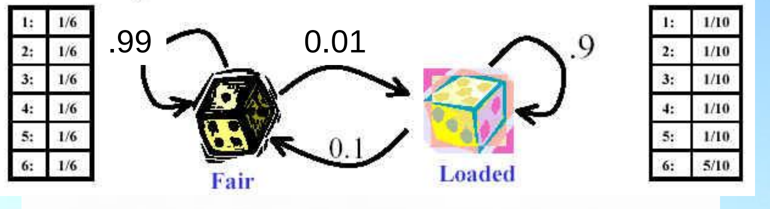


Figure 3.6 *The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.*



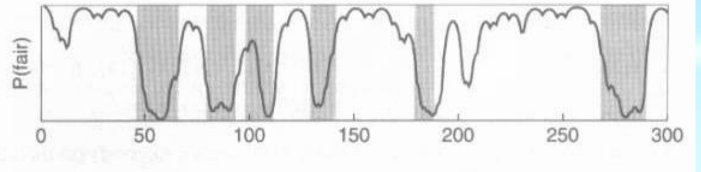


Figure 3.6 *The posterior probability of being in the state corresponding to the fair die in the casino example. The x axis shows the number of the roll. The shaded areas show when the roll was generated by the loaded die.*

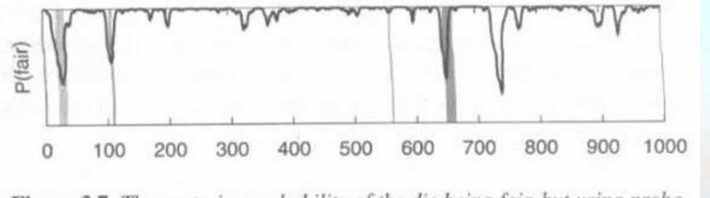


Figure 3.7 *The posterior probability of the die being fair, but using probability 0.01 for switching to the loaded die (cf. Figure 3.6).*

Parameter estimation for HMMs
(the *learning* problem)

- The framework:
  - A set of example sequences (training sequences)

    $$x^1, \ldots, x^n$$

    which are independent.
  - Maximum Likelihood Method:
    - The joint probability of all the sequences given a set of parameters
  - Log-likelihood of the sequences given the model:

$$l(x^1, \cdots, x^n|\theta) = \log P\,(x^1, \cdots, x^n|\theta) = \sum_{j=1}^{n} \log P\,(x^j|\theta)$$

If the state sequences are known

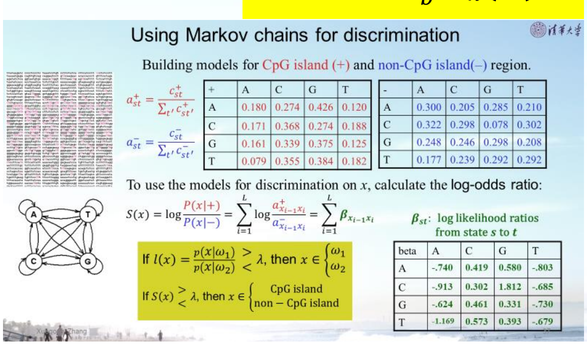
$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

Number of transitions k to l in training data + r_{kl}

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

Number of emissions of b from k in training data + $r_k(b)$

The r_{kl} and $r_k(b)$ are *pseudocounts* that reflect our prior biases about the probability values, or 0s.


Using Markov chains for discrimination

Building models for CpG island (+) and non-CpG island(–) region.

$$a^+_{st} = \frac{c^+_{st}}{\sum_{t'} c^+_{st'}} \qquad a^-_{st} = \frac{c^-_{st}}{\sum_{t'} c^-_{st'}}$$

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

-	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

To use the models for discrimination on x, calculate the log-odds ratio:

$$S(x) = \log\frac{P(x|+)}{P(x|-)} = \sum_{i=1}^{L} \log\frac{a^+_{x_{i-1}x_i}}{a^-_{x_{i-1}x_i}} = \sum_{i=1}^{L} \beta_{x_{i-1}x_i}$$

β_{st}: log likelihood ratios from state s to t

If $l(x) = \frac{p(x|\omega_1)}{p(x|\omega_2)} \gtrless \lambda$, then $x \in \begin{cases}\omega_1\\ \omega_2\end{cases}$

If $S(x) \gtrless \lambda$, then $x \in \begin{cases}\text{CpG island}\\ \text{non}-\text{CpG island}\end{cases}$

beta	A	C	G	T
A	-.740	0.419	0.580	-.803
C	-.913	0.302	1.812	-.685
G	-.624	0.461	0.331	-.730
T	-1.169	0.573	0.393	-.679

When paths are unknown

- Basic idea:
  - Iterative procedure:
    - First estimate the A_{kl} and $E_k(b)$ by considering probable paths for the training sequences using the current model
    - Then derive new values of the parameters
    - Iterating until some stopping criterion is reached
  - Two approaches:
    - Baum-Welch Algorithm
    - Viterbi Learning

Baum-Welch Algorithm ---- an EM algorithm

$f_k(i) = P(x_1 \cdots x_i, \pi_i = k)$
$b_k(i) = P(x_{i+1} \cdots x_L | \pi_i = k)$

- Initialization: pick arbitrary model parameters
- Recurrence:
  - Set all *A* and *E* variables to their *pseudocount* values *r* (or 0)
  - For each sequence *j*=1,…,*n*:
    - Calculate $f_k(i)$ for sequence *j* using **the Forward Algorithm**
    - Calculate $b_k(i)$ for sequence *j* using **the Backward Algorithm**
    - Add the contribution of sequence *j* to *A* and *E*

The probability that a_{kl} is used (all positions, all sequences)

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1)$$

\# of times the letter *b* appears in state *k*

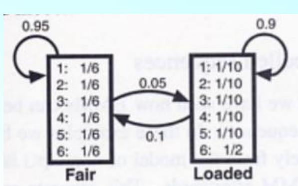
$$E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i|x_i^j=b\}} f_k^j(i) b_k^j(i)$$

  - Calculate the new model parameters $a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}, \quad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$
  - Calculate the new log likelihood of the model
- Termination:
  - Stop if the change in log-likelihood is less than some predefined threshold, or the maximum number of iterations is exceeded.

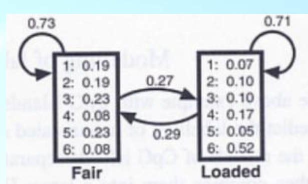
Viterbi Training

- The most probable paths for the training sequences (obtained with Viterbi algorithm) are used in the re-estimation process
- The process is iterated when the new parameter values are obtained
  - It does not maximize the true likelihood,
  - but maximizes the contribution to the likelihood from the most probable paths for all the sequences.
  - performs less well in general than Baum-Welch
  - *but is frequently used in practice, esp. the primary use of HMM is decoding*

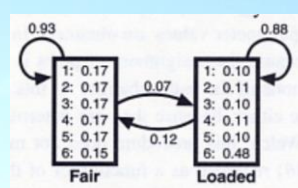
Example:
unveiling the secret of the dishonest casino



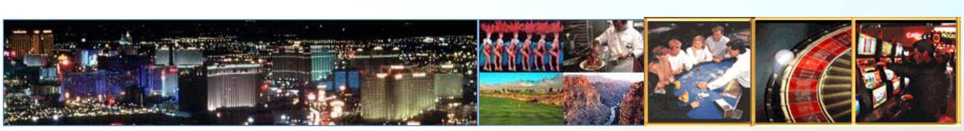
the secret truth



unveiled with 300 observations

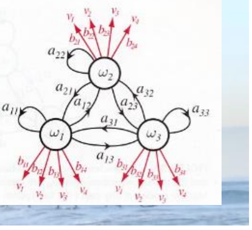


unveiled with 30 000 observations



Three Central Issues in HMM?

- The Evaluation Problem
  - Given an HMM, complete with transition and emission probabilities, how to determine the probability that a particular sequence of symbols was generated by that model?
- The Decoding Problem
  - Given an HMM as well as a set of observations, how to determine the most likely sequence of hidden states that led to those observations?
- The Learning Problem
  - Given the coarse structure of the model but *not* the probabilities, and given a set of training observations of symbols, how to determine the probabilities?
- **A more challenging issue**
  - **The Structure-Learning Problem**



Learning the HMM Structure

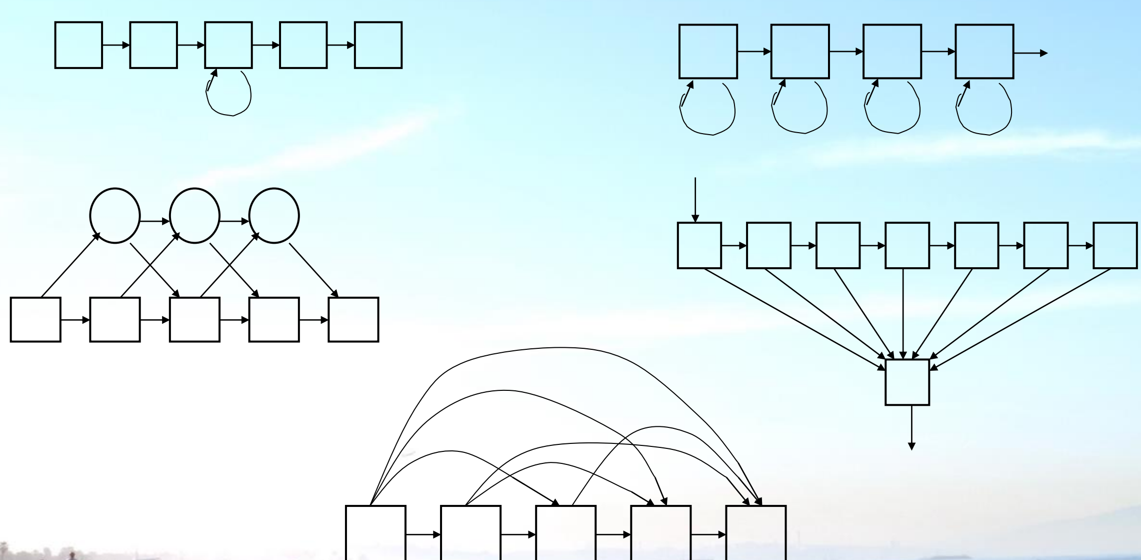
- The idea *"start with a fully connected model and let the model find out for itself"* does not work in most cases.
  - "It almost never works in practice."
  - For problems of any realistic size, it will usually lead to very bad models, even with plenty of training data.
  - The problem is not overfitting, but local maxima.
    - The less constrained the model is, the more severe the local maximum problem becomes.

Learning the HMM Structure

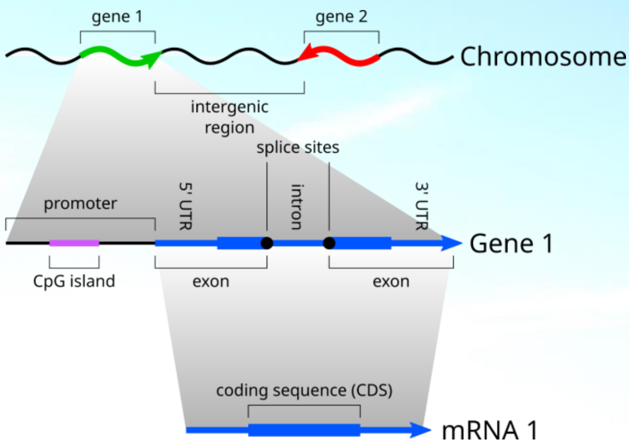
- There are methods that attempt to adapt the model topology based on the data by adding and removing transitions and states.

- However, in practice successful HMMs are constructed by *carefully deciding which transitions are to be allowed in the model*, based on <u>knowledge about the problem</u> under investigation.
  - To disable the transition from state k to state l, set $a_{kl} = 0$
  - All the mathematics in the learning algorithm is unchanged.

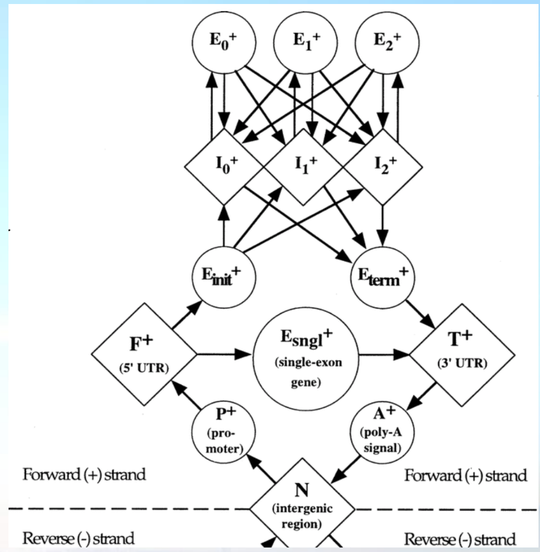
Some examples of HMM structures



The HMM of GENSCAN



https://www.cs.rice.edu/~ogilvie/assets/gene-model.png



1-minute break

12.2
Basic Ideas of Graphic Models

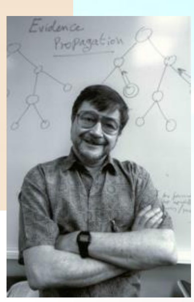
Bayesian Networks & Graphic Models

R-43
April 1985

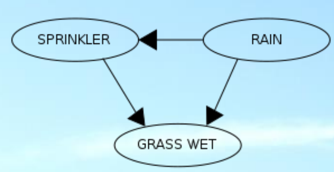
BAYESIAN NETWORKS: A MODEL OF SELF-ACTIVATED MEMORY FOR EVIDENTIAL REASONING*

Judea Pearl
Computer Science Department
University of California
Los Angeles, CA 90024
(judea@UCLA-locus)
(213) 825-3243

Topics: Memory Models
Belief Systems
Inference Mechanisms
Knowledge Representation



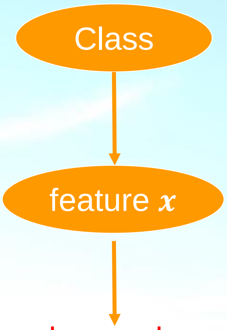
RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99



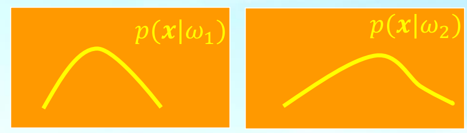
RAIN T	RAIN F
0.2	0.8

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

A graphic representation of the model behind Bayesian Decision



Class	$\boldsymbol{\omega_1}$	$\boldsymbol{\omega_2}$
Probability (prior)	$P(\omega_1)$	$P(\omega_2)$



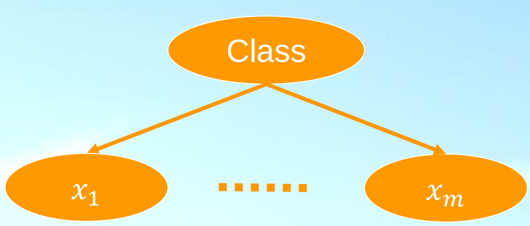
observed $\boldsymbol{x}$

$P(\boldsymbol{x}|\omega_1)P(\omega_1)$ $P(\boldsymbol{x}|\omega_2)P(\omega_2)$

$$P(\boldsymbol{x}, \omega_i) = p(\boldsymbol{x}|\omega_i)P(\omega_i)$$

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$$

A graphic representation of Naïve Bayesian



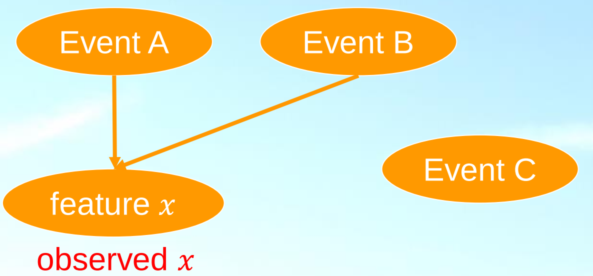
observed $\boldsymbol{x} = [x_1, \cdots, x_m]^T$

$$P(\omega_i, \boldsymbol{x}) = P(\omega_i) \prod_{j=1}^{m} p(x_j|\omega_i)$$

$$P(\omega_i|\boldsymbol{x}) \propto P(\omega_i) \prod_{j=1}^{m} p(x_j|\omega_i)$$

When a node has more parents in the network

- **Node:** event/variable
- **Edge:** probabilistic dependency



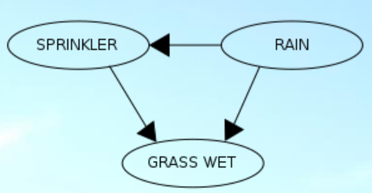
$$P(x, A, B) = p(x|A, B)P(A)P(B)$$

$$P(x, A, B, C) = p(x|A, B)P(A)P(B)P(C)$$

$$P(A|x) = \sum_B P(x, A, B, C)/p(x) \propto P(A) \sum_B p(x|A, B)P(B)$$

The standard example of Bayesian Networks

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

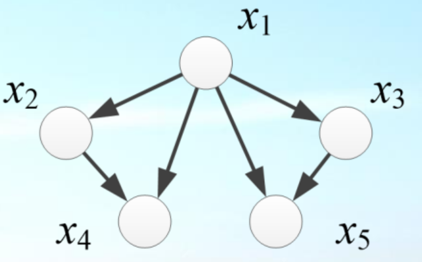
$$P(G, S, R) = p(G|S, R)P(S|R)P(R)$$

$$\Pr(R=T|G=T) = \frac{\Pr(G=T, R=T)}{\Pr(G=T)} = \frac{\sum_{S\in\{T,F\}} \Pr(G=T, S, R=T)}{\sum_{S,R\in\{T,F\}} \Pr(G=T, S, R)}$$

Bayesian Networks

- Present conditional independencies via a DAG (*directed acyclic graph*)

$$p(x_1, x_2, x_3, x_4, x_5)$$
$$= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4)$$
$$= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_1, x_2)p(x_5|x_1, x_3)$$



- General form:

$$p(\mathbf{x}) = \prod_{i=1}^{m} p(x_i \mid \mathrm{pa}_i)$$

Learning tasks

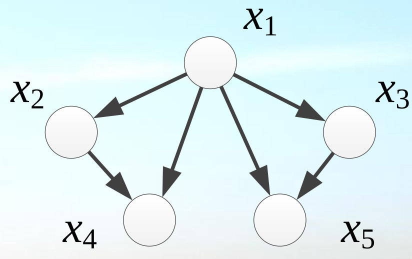
- Parameter learning
  - Discrete: Conditional distribution table
  - Continuous: Conditional Gaussian
  - Maximum likelihood (MLE) or Maximum *a posterior* Probability (MAP)
  - EM algorithm
- Structure learning
  - Specify the structure by an expert
  - Optimization and heuristic search
    Use posterior probability as a score and then do Markov Chain Monte Carlo (MCMC) or simulated annealing

Inference tasks

- Inferring unobserved variables
  - "What is the probability that it is raining, given the grass is wet?"

$$p(x_1|x_5) = \frac{p(x_1,x_5)}{p(x_5)} = \frac{\sum_{x_2,x_3,x_4} p(x_1,x_2,x_3,x_4,x_5)}{\sum_{x_1,x_2,x_3,x_4} p(x_1,x_2,x_3,x_4,x_5)}$$

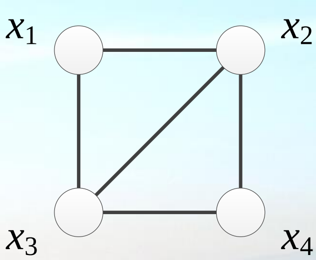
- Exact inference
  - Variable elimination: Joint → marginal → conditional
  - Clique tree propagation
  - Recursive conditioning and AND/OR search
- Approximate inference
  - Importance sampling
  - MCMC simulation
  - Belief propagation



Markov Random Field (MRF)

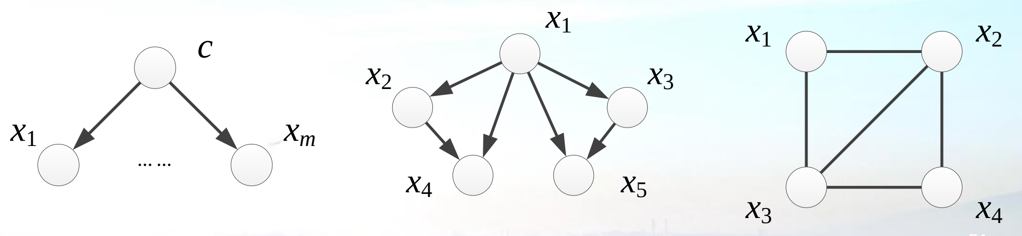
- Present conditional independencies via an *undirected graph*
- Clique factorization

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \{\text{cliques}\}} \psi_c(\mathbf{x}_c)$$



Probabilistic Graphical Models

- Present the joint distribution of a set of random variables using a graph
  - Node: random variable
  - Edge: conditional dependence
  - Non-edge: conditional independence
- Applications:
  - computer vision, speech recognition
  - gene regulatory network inference
  - …



Homework

- Problem Set (Pr.6b)
  - Formulate the Covid-19 case tracing issue into a Bayesian framework.
- Deadline:
  - Nov. 17 (Wednesday), 23:00


See you next week
for
Unsupervised Learning
Xuegong Zhang
53