Hidden Markov Models

Ivan Gesteira Costa Filho
IZKF Research Group Bioinformatics
RWTH Aachen
Adapted from: www.ioalgorithms.info

Outline

- CG-islands
- The “Fair Bet Casino”
- Hidden Markov Model
- Decoding Algorithm
- Forward-Backward Algorithm
- HMM Parameter Estimation
- Viterbi training
- Baum-Welch algorithm

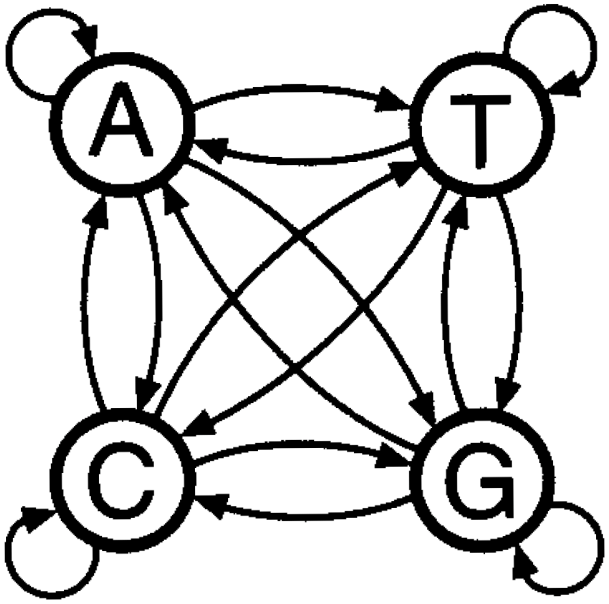
CG-Islands

- Given 4 nucleotides: probability of occurrence is ~ 1/4. Thus, probability of occurrence of a dinucleotide is ~ 1/16.
- However, the frequencies of dinucleotides in DNA sequences vary widely.
- In particular, *CG* is typically underepresented (frequency of *CG* is typically < 1/16)

Why CG-Islands?

- *CG* is the least frequent dinucleotide because *C* in *CG* is easily *methylated and* has the tendency to mutate into T afterwards
- However, the methylation is suppressed around genes in a genome. So, *CG* appears at relatively high frequency within these *CG* islands
- So, finding the *CG* islands in a genome is an important problem

Markov Chains



- Given a sequence

x = AGCTCTC...T

- and transition probabilities

$a_{AG}=p(x_i=G|x_{i-1}=A)$

- What is $p(x)$?

Markov Chains

$$p(x)=p(x_L,x_{L-1},x_{L-2},\ldots,x_1)$$

$$=p(x_L|x_{L-1},x_{L-2},\ldots,x_1).p(x_{L-1}|x_{L-2},\ldots,x_1)\ldots p(x_1)$$

Markov Assumption

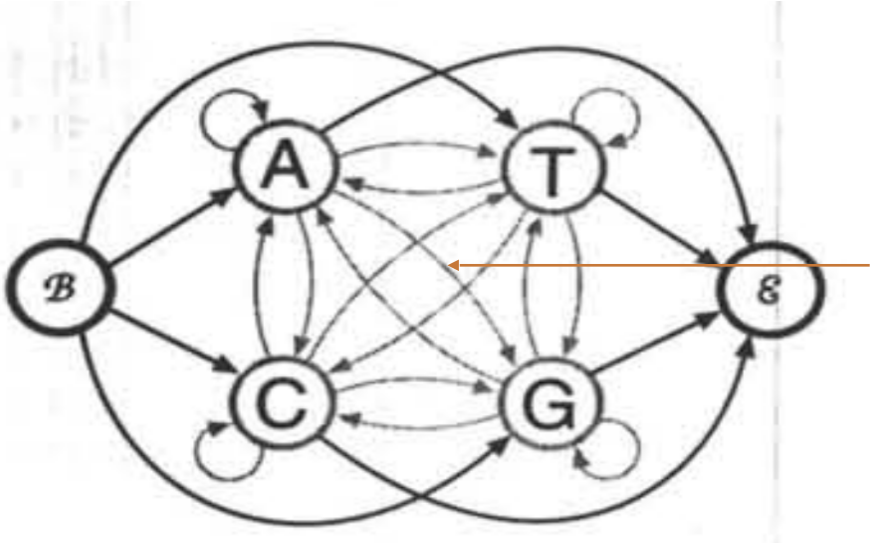
$$p(x_i|x_{i-1},x_{i-2},\ldots,x_1) = p(x_i|x_{i-1})$$

then

$$p(x)=p(x_L|x_{L-1}).p(x_{L-1}|x_{L-2})\ldots p(x_1)$$

$$=p(x_1)\Pi_{i=1}p(x_{i-1}|x_{i-2})$$

Markov Chains – Start State



- Given a sequence
  x = *b*AGCTCTC...T*e*
- Transition probabilities

$a_{AG}=p(x_i=G|x_{i-1}=A)$

- $p(x) = p(x_1)\Pi_{i=1} a_{x_i x_{i-1}}$
- $p(x_1)=1$
- $p(x_2|x_1)=1/4$

Using Markov Model for CG discrimination

- Use curated sequences of CG islands (non-CG islands) to build two markov models (*model*$^+$ *and model*$^-$)
- Estimate $a_{st} = c_{st} / \Sigma_{t'} c_{st'}$

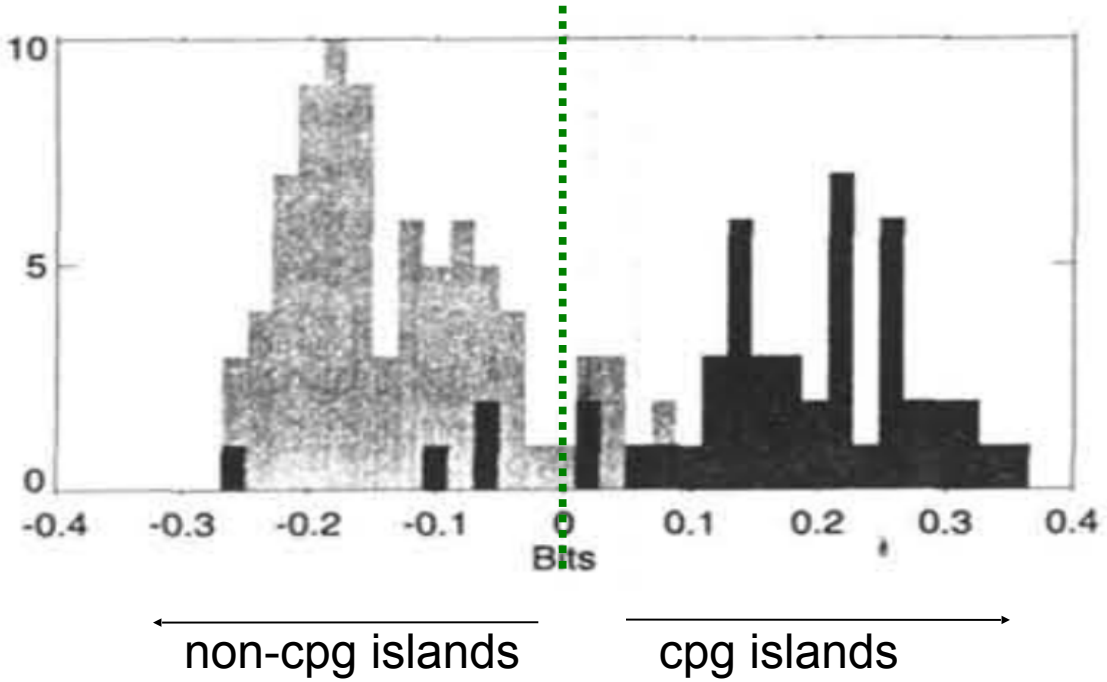
+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

−	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

Markov Model for CG discrimination

- For a given sequence *x*

calculate log ($p(x|model^{+})/ p(x|model^{-})$)



What if we don't known any CG islands?

CG Islands and the "Fair Bet Casino"

- The *CG* islands problem can be modeled after a problem named ***"The Fair Bet Casino"***
- The game is to flip coins, which results in only two possible outcomes: **H**ead or **T**ail.
- The **F**air coin will give **H**eads and **T**ails with same probability ½.
- The **B**iased coin will give **H**eads with prob. ¾.

The "Fair Bet Casino" (cont'd)

- Thus, we define the probabilities:
  - P(H|F) = P(T|F) = ½
  - P(H|B) = ¾, P(T|B) = ¼
  - The crooked dealer changes between Fair and Biased coins with probability 10%

The Fair Bet Casino Problem

- **Input:** A sequence $x = x_1x_2x_3\ldots x_n$ of coin tosses made by two possible coins (***F*** or ***B***).

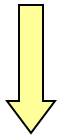
- **Output:** A sequence $\pi = \pi_1\ \pi_2\ \pi_3\ldots\ \pi_n$, with each π_i being either *F* or *B* indicating that x_i is the result of tossing the Fair or Biased coin respectively.

Problem…

Fair Bet Casino Problem

Any observed outcome of coin tosses could have been generated by any sequence of dices π!

Need to incorporate a way to grade different sequences differently.



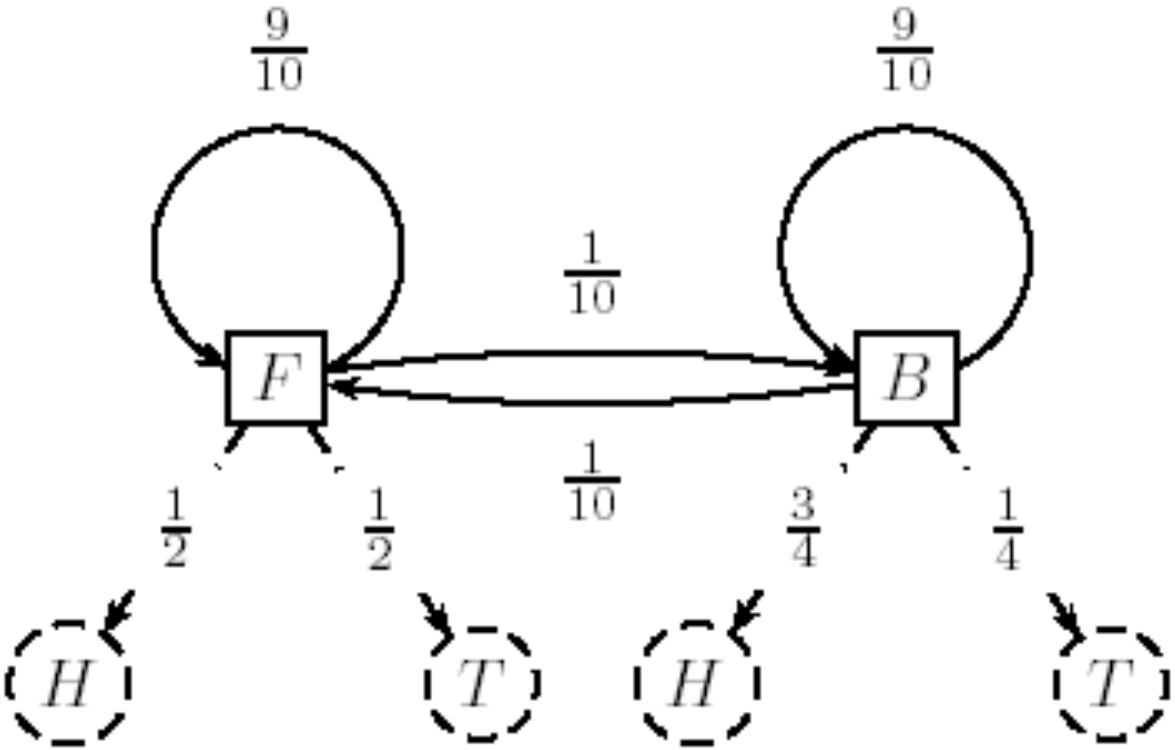
Decoding Problem

Find sequence π with maximum probability

Hidden Markov Model (HMM)

- Can be viewed as an abstract machine with *k hidden* states that emits symbols from an alphabet Σ.
- Each state has its own probability distribution, and the machine switches between states according to this probability distribution.
- While in a certain state, the machine makes 2 decisions:
  - What state should I move to next?
  - What symbol - from the alphabet Σ - should I emit?

HMM for Fair Bet Casino (cont'd)



HMM model for the *Fair Bet Casino* Problem

Why "Hidden"?

- Observers can see the emitted symbols of an HMM but have *no ability to know which state the HMM is currently in*.
- Thus, the goal is to infer the most likely hidden states of an HMM based on the given sequence of emitted symbols.

HMM Parameters

Σ: set of emission characters.

Ex.: Σ = {H, T} for coin tossing

Σ = {1, 2, 3, 4, 5, 6} for dice tossing

Q: set of hidden states, each emitting symbols from Σ.

Q={F,B} for coin tossing

HMM Parameters (cont'd)

A = (a_{kl}): a |Q| x |Q| matrix of probability of changing from state *k* to state *l*.

$a_{FF} = 0.9$ $a_{FB} = 0.1$

$a_{BF} = 0.1$ $a_{BB} = 0.9$

E = $(e_k(b))$: a |Q| x |Σ| matrix of probability of emitting symbol *b* while being in state *k*.

$e_F(0) = ½$ $e_F(1) = ½$

$e_B(0) = ¼$ $e_B(1) = ¾$

HMM for Fair Bet Casino

- The *Fair Bet Casino* in *HMM* terms:

  Σ = {0, 1} (0 for ***T***ails and 1 ***H***eads)

  Q = {*F,B*} – *F* for Fair & *B* for Biased coin.
- Transition Probabilities *A* *** Emission Probabilities *E*

	Fair	Biased
Fair	$a_{FF} = 0.9$	$a_{FB} = 0.1$
Biased	$a_{BF} = 0.1$	$a_{BB} = 0.9$

	Tails(0)	Heads(1)
Fair	$e_F(0) = ½$	$e_F(1) = ½$
Biased	$e_B(0) = ¼$	$e_B(1) = ¾$

Hidden Paths

- A *path* $\pi = \pi_1 \ldots \pi_n$ in the HMM is defined as a sequence of states.
- Consider path π = FFFBBBBBFF and sequence x = 0101110100

Probability that x_i was emitted from state π_i

X	0	1	0	1	1	1	0	1	0	0	1
π =	F	F	F	B	B	B	B	B	F	F	F
$P(x_i \mid \pi_i)$	½	½	½	¾	¾	¾	¼	¾	½	½	½
$P(\pi_i \mid \pi_{i-1})$	9/10	9/10	1/10	9/10	9/10	9/10	9/10	1/10	9/10	9/10	9/10

Transition probability from state π_{i-1} to state π_i

P(x,π) Calculation

- P(*x*,π): Probability that sequence *x* was generated by the path *π:*

$$P(x,\pi) = \prod_{i=1}^{n} P(x_i|\ \pi_i) \cdot P(\pi_{i+1}|\ \pi_i)$$

$$= \prod_{i=1}^{n} e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$

HMM - Algorithms

We will describe algorithms that allow us to compute:

$\text{argmax}_{\pi} P(\pi, x)$ - Most probable path for a given string x (Viterbi path)

$P(\pi_i = k \mid x)$ - "Posterior" probability that the i^{th} state is k, given x

A more refined measure of which states x may be in

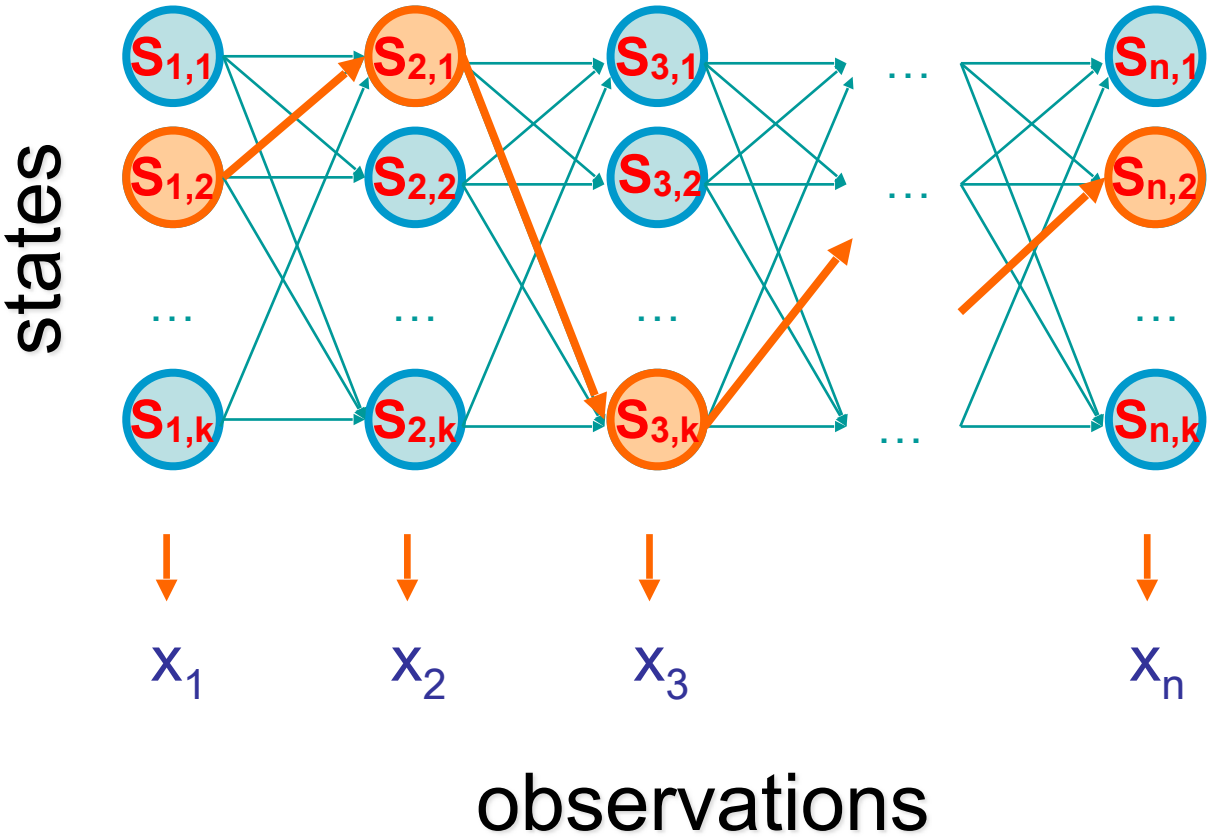
Decoding Problem

- **Goal:** Find an optimal hidden path of states given observations.
- **Input:** Sequence of observations $x = x_1 \ldots x_n$ generated by an HMM $M(\Sigma, Q, A, E)$
- **Output:** A path that maximizes $P(x,\pi)$ over all possible paths π.

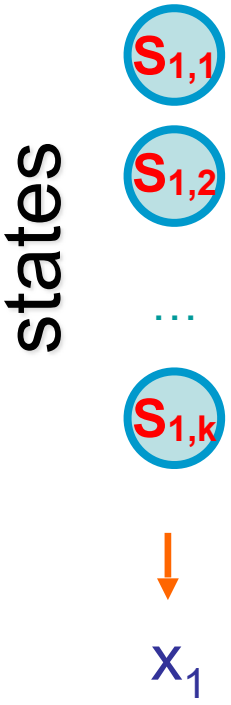
Building Manhattan for Decoding Problem

- Andrew Viterbi used the Manhattan grid model to solve the *Decoding Problem*.
- Explore the fact that the solution for $(\pi_1 \ldots \pi_n)$ is based on the solution from $(\pi_1 \ldots \pi_{n-1})$
- A node $S_{i,k}$ in the graph encodes the probability that $P(x_1,..x_i,\pi_{i1},\ldots,\pi_i=k)$.
- The Viterbi algorithm finds the path that maximizes $P(x,\pi)$ among all possible paths.

Edit Graph for Decoding Problem



Edit Graph for Decoding Problem



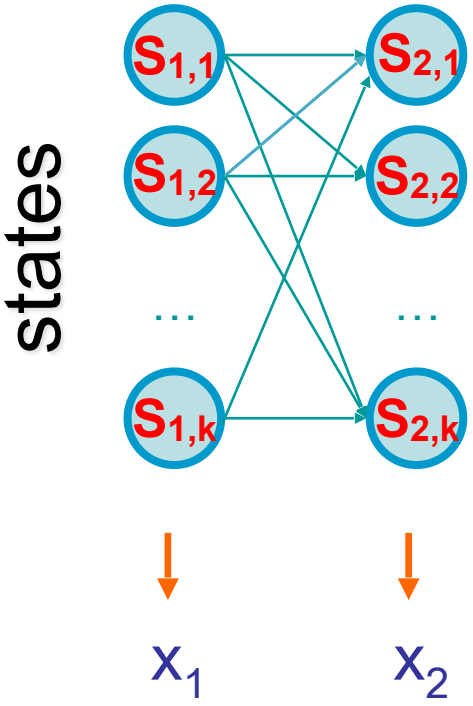
Initialization:

$s_{1,l} = e_l(x_i)$

Recursions:

$s_{i+1,l} = e_l(x_{i+1}) \cdot \max_{j \in Q} \{s_{ji} \cdot a_{jl}\}$

Edit Graph for Decoding Problem



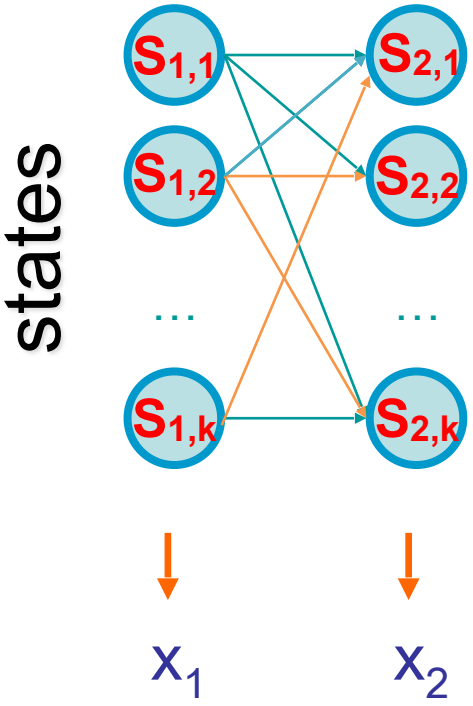
Initialization:

$s_{1,l} = e_l(x_i)$

Recursions:

$s_{i+1,l} = e_l(x_{i+1}) \cdot \max_{j \in Q} \{s_{ji} \cdot a_{jl}\}$

Edit Graph for Decoding Problem



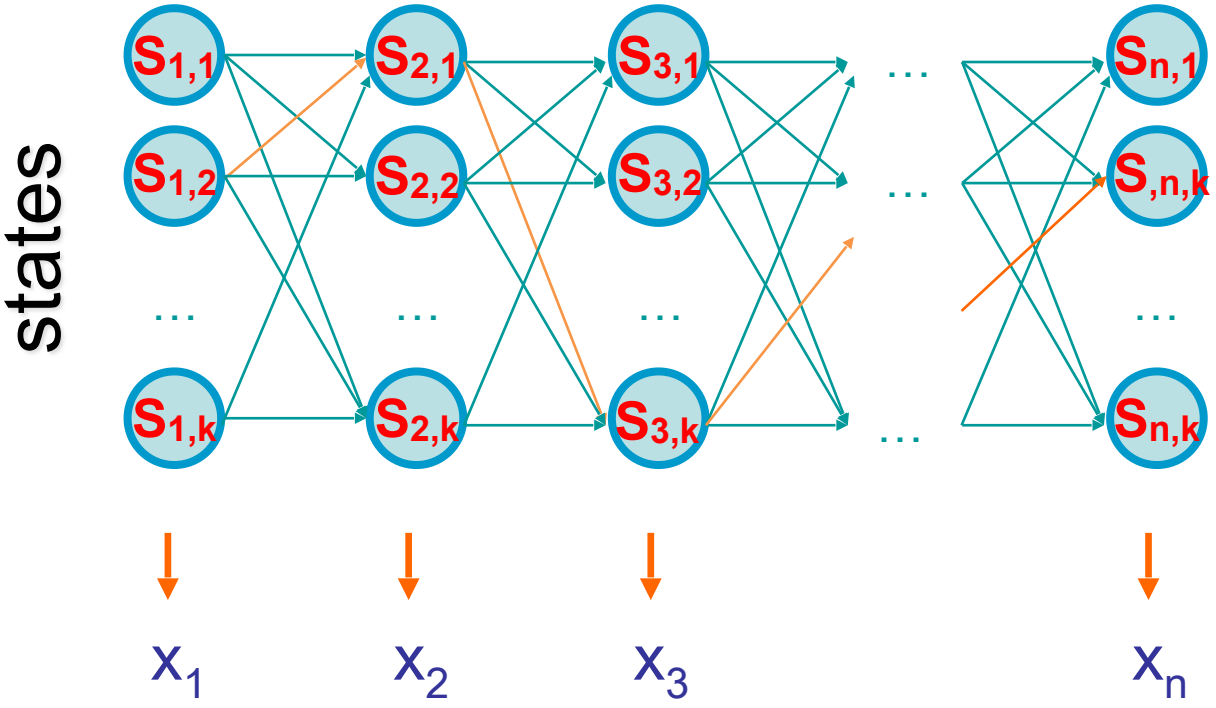
Initialization:

$s_{1,l} = e_l(x_i)$

Recursions:

$s_{i+1,l} = e_l(x_{i+1}) \cdot \max_{j \in Q} \{s_{ji} \cdot a_{jl}\}$* (keep pointer to max)

Edit Graph for Decoding Problem



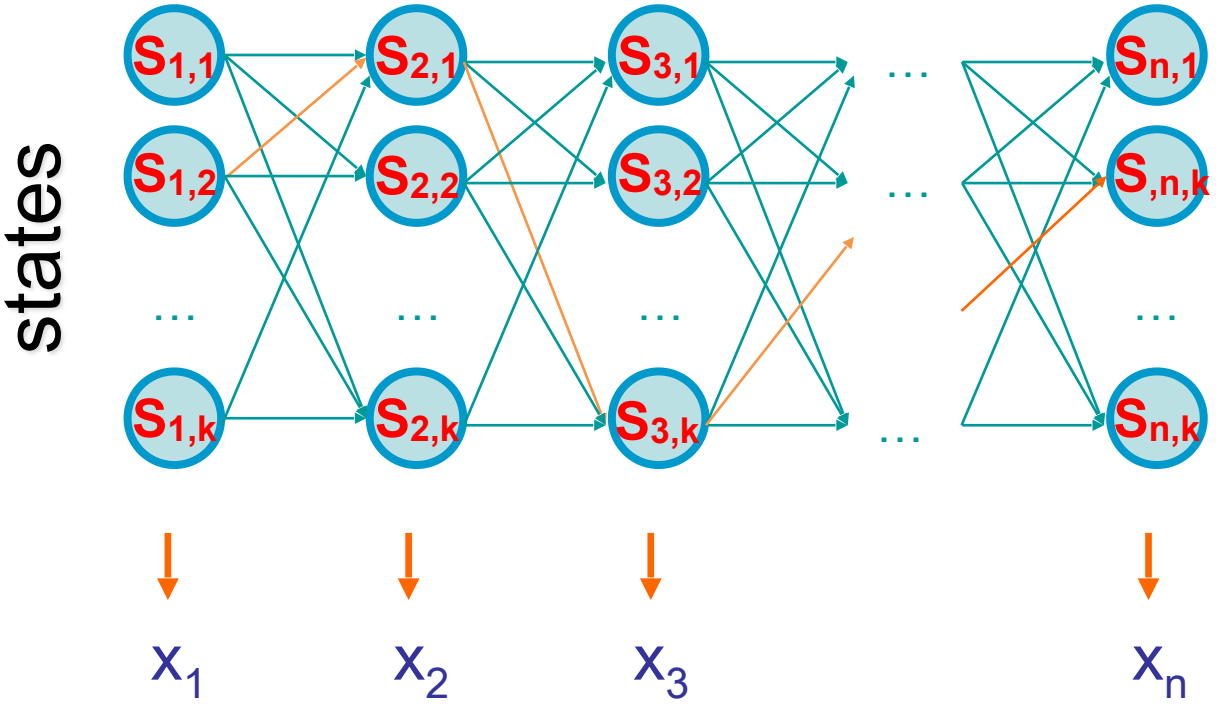
Initialization:

$s_{1,l} = e_l(x_i)$

Recursions:

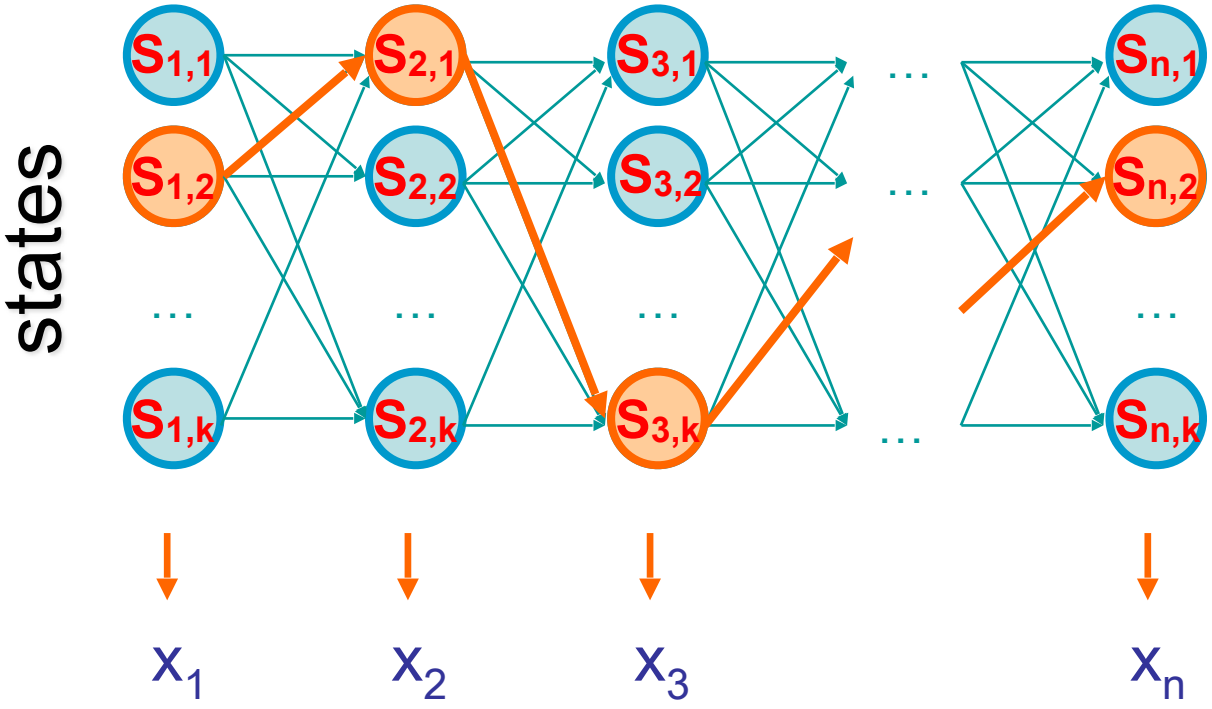
$s_{i+1,l} = e_l(x_{i+1}) \cdot \max_{j \in Q} \{s_{ji} \cdot a_{jl}\}$* (keep pointer to max)

Edit Graph for Decoding Problem



- Let π^* be the optimal path. Then,
  1. find $P(x,\pi^*) = \max_{l \in Q} \{s_{n,l}\}$
  2. backtrack on pointers

Edit Graph for Decoding Problem



- Let π^* be the optimal path. Then,
  1. find $P(x,\pi^*) = \max_{l \in Q} \{s_{n,l}\}$
  2. backtrack on pointers

Viterbi Algorithm

- Computational complexity $n*k^2$

- The value of the product can become extremely small, which leads to overflowing.
- To avoid overflowing, use log value instead.

$$s_{i+1,k} = \log e_l(x_{i+1}) + \max_{k \in Q} \{s_{i,k} + \log(a_{kl})\}$$

Posterior Problem

Given: a sequence of coin tosses generated by an HMM.

Goal: find the probability that the dealer was using a biased coin at a particular time (posterior problem).

Posterior Algorithm

We want to compute $P(\pi_i = k \mid x)$, the probability distribution on the i^{th} position, given x

We start by computing

$$P(\pi_i = k, x) = P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_N)$$

$$= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid x_1 \ldots x_i, \pi_i = k)$$

$$= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid \pi_i = k)$$

Forward, $f_{i\,k}$ Backward, $b_{k\,i}$

Then, $P(\pi_i = k \mid x) = P(\pi_i = k, x) / P(x)$

Backward Algorithm (derivation)

- Define *backward probability* $b_{k,i}$ as the probability of being in state $\pi_i = k$ and emitting the *suffix* $x_{i+1}...x_n$

$b_{i\,k} = P(x_{i+1}\ldots x_N \mid \pi_i = k)$ *"starting from i^{th} state = k, generate rest of x"*

$= \sum_{\pi_{i+1}\ldots\pi_N} P(x_{i+1},x_{i+2}, \ldots, x_N, \pi_{i+1}, \ldots, \pi_N \mid \pi_i = k)$

$= \sum_l \sum_{\pi_{i+1}\ldots\pi_N} P(x_{i+1},x_{i+2}, \ldots, x_N, \pi_{i+1} = l, \pi_{i+2}, \ldots, \pi_N \mid \pi_i = k)$

$= \sum_l e_l(x_{i+1})\, a_{kl} \sum_{\pi_{i+1}\ldots\pi_N} \mathbf{P(x_{i+2}, \ldots, x_N, \pi_{i+2}, \ldots, \pi_N \mid \pi_{i+1} = l)}$

$= \sum_l e_l(x_{i+1})\, a_{kl}\, \mathbf{b_l(i+1)}$

Backward Algorithm

We can compute $b_{i\,k}$ for all k, i, using dynamic programming/edit graph

Initialization:

$b_{n,k} = 1$, for all k

Iteration:

$b_k(i) = \sum_l e_l(x_{i+1})\, a_{kl}\, b_l(i+1)$

Termination:

$P(x) = \sum_l a_{0l}\, e_l(x_1)\, b_l(1)$

Forward Algorithm

- Define *forward probability* $f_{k,i}$ as the probability of being in state $\pi_i = k$ and emitting the *prefix* $x_1 \ldots x_i$.

Initialization:

$f_0(0) = 1$

Iteration:

$f_k(i) = e_k(x_i) \sum_l f_l(i-1)\, a_{lk}$

Termination:

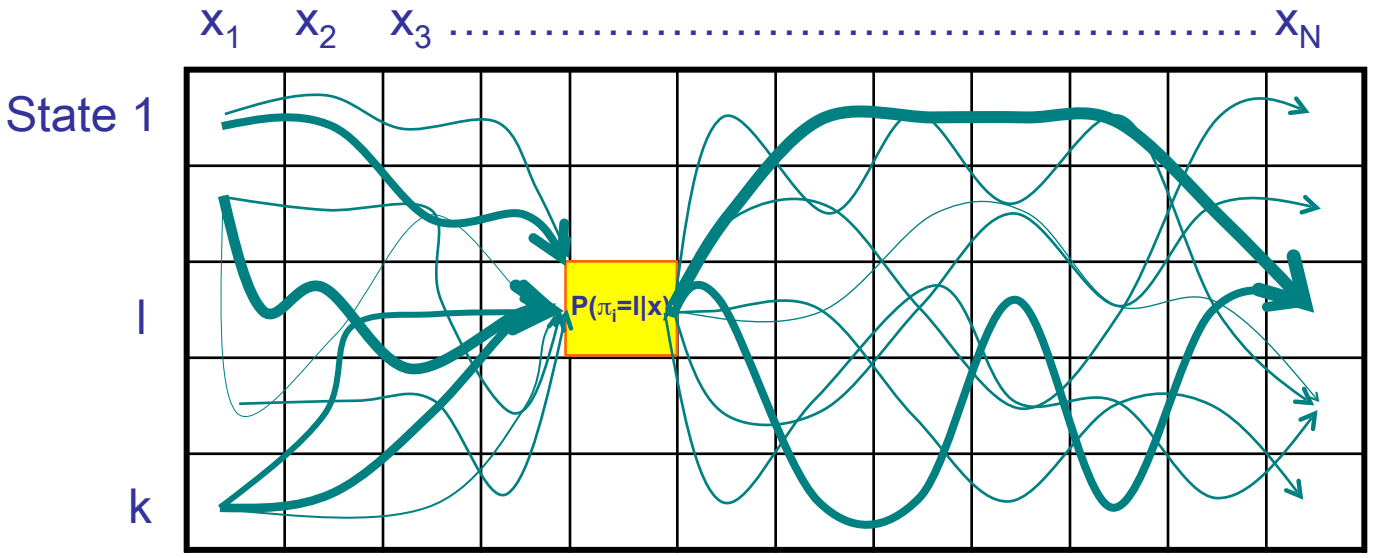
$P(x) = \sum_k f_k(N)$

Posterior Decoding

We can now calculate $P(\pi_i = k \mid x)$ and perform the poster decoding

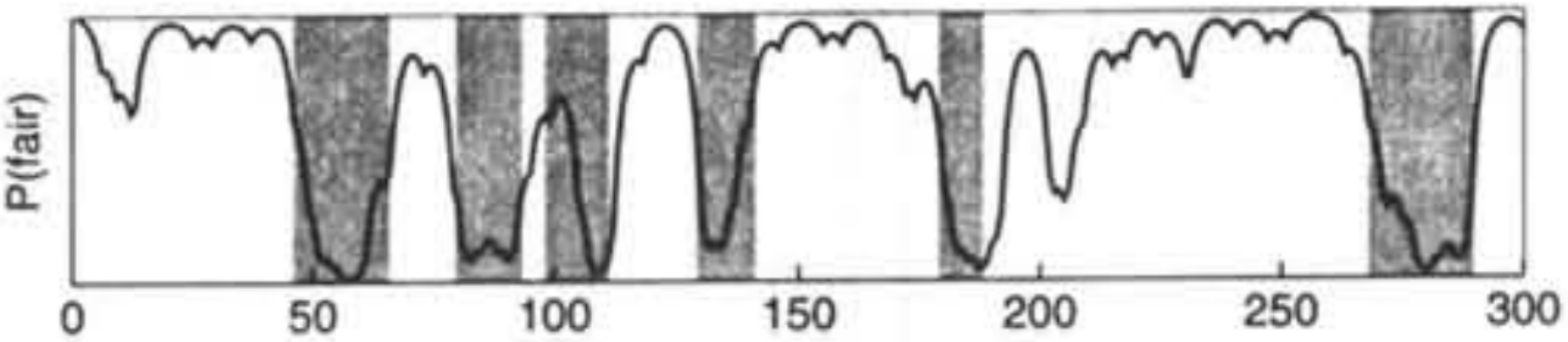
$$P(\pi_i = k \mid x) = \frac{f_k(i)\, b_k(i)}{P(x)}$$

$$\hat{\pi}_i = \text{argmax}_k\, P(\pi_i = k \mid x)$$



Example: Fair dice

$P(\pi_i = \text{“}fair\text{”}|x)$ for a given sequence of dices x



HMM Parameter Estimation

- So far, we have assumed that the transition and emission probabilities are known.
- However, in most HMM applications, the probabilities are not known. It's very hard to estimate the probabilities.

HMM Parameter Estimation Problem

- Given
  - HMM with states and alphabet (emission characters)
  - Independent training sequences $x^1, \ldots x^m$
- Find HMM parameters Θ (that is, a_{kl}, $e_k(b)$) that maximize

$$P(x^1, \ldots, x^m \mid \Theta)$$

the joint probability of the training sequences.

Maximize the likelihood

$P(x^1, \ldots, x^m \mid \Theta)$ as a function of Θ is called the likelihood of the model.

The training sequences are assumed independent, therefore

$$P(x^1, \ldots, x^m \mid \Theta) = \Pi_i P(x^i \mid \Theta)$$

The parameter estimation problem seeks Θ that realizes

$$\max_{\Theta} \prod_i P(x^i \mid \Theta)$$

In practice the log likelihood is computed to avoid underflow errors

Two situations

Known paths for training sequences

- CpG islands marked on training sequences
- One evening the casino dealer allows us to see when he changes dice

Unknown paths

- CpG islands are not marked
- Do not see when the casino dealer changes dice

Known paths

A_{kl} = # of times each $k \rightarrow l$ is taken in the training sequences

$E_k(b)$ = # of times b is emitted from state k in the training sequences

Compute a_{kl} and $e_k(b)$ as maximum likelihood estimators:

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

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

Pseudocounts

- Some state *k* may not appear in any of the training sequences. This means $A_{kl} = 0$ for every state *l* and a_{kl} cannot be computed with the given equation.
- To avoid this overfitting use predetermined pseudocounts r_{kl} and $r_k(b)$.

  A_{kl} = # of transitions $k \rightarrow l$ + r_{kl}

  $E_k(b)$ = # of emissions of *b* from *k* + $r_k(b)$

The pseudocounts reflect our prior biases about the probability values.

Unknown paths: Viterbi training

Idea: use Viterbi decoding to compute the most probable path for training sequence *x*

- Start with some guess for initial parameters and compute π^* the most probable path for *x* using initial parameters.
- Iterate until no change in π^* :

1. Determine A_{kl} and $E_k(b)$ as before
2. Compute new parameters a_{kl} and $e_k(b)$ using the same formulas as before
3. Compute new π^* for *x* and the current parameters

Viterbi training analysis

- The algorithm converges precisely

  There are finitely many possible paths.

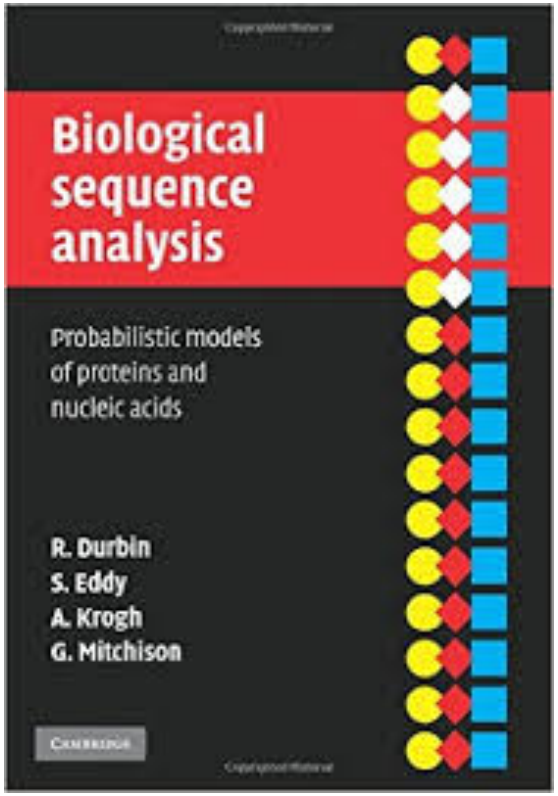
  New parameters are uniquely determined by the current π^*.

  There may be several paths for x with the same probability, hence must compare the new π^* with all previous paths having highest probability.
- Does not maximize the likelihood $\Pi_x P(x \mid \Theta)$ but the contribution to the likelihood of the most probable path $\Pi_x P($ $\mid \Theta, \pi^*)$
- In general performs less well than Baum-Welch

Overview

- introduction to Hidden Markov models
- basic evaluation algorithms
  - Viterbi Path, Posterior Decoding
- basic training methods
  - supervised case
  - unsupervised case: Viterbi & Baum Welch (not seem today)

References



- Original slides from bio algorithms
- Some slides were adapted from CS 262 course at Stanford given by Serafim Batzoglou