CS481: Bioinformatics Algorithms

Can Alkan
EA224
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/

Heuristic Similarity Searches

- Genomes are huge: Smith-Waterman quadratic alignment algorithms are too slow
- Alignment of two sequences usually has short identical or highly similar fragments
- Many heuristic methods (i.e., FASTA) are based on the same idea of *filtration*
  - Find short exact matches, and use them as seeds for potential match extension
  - “Filter” out positions with no extendable matches

PatternHunter: faster and even more sensitive

- BLAST: matches short consecutive sequences (consecutive seed)
- Length = *k*
- Example (*k* = 11):

11111111111

Each 1 represents a “match”

- PatternHunter: matches short non-consecutive sequences (spaced seed)
- Increases sensitivity by locating homologies that would otherwise be missed
- Example (a spaced seed of length 18 w/ 11 “matches”):

111010010100110111

Each 0 represents a “don’t care”, so there can be a match or a mismatch

Spaced seeds

Example of a hit using a spaced seed:

```
GAGTACTCAACACCAACATTAGTGGCAATGGAAAAT...
||  |||||||||| ||||| || |||||    ||||||
GAATACTCAACAGCAACACTAATGGCAGCAGAAAAT...
        11101001010011011 1
```

Why is PH better?

- BLAST: redundant hits

```
TTGACCTCACC?
||||||||||||?
TTGACCTCACC?
11111111111
 11111111111
```

This results in > 1 hit and creates clusters of redundant hits

- **PatternHunter**

```
CAA?A??A?C??TA?TGG?
|||?|??|?|??||?|||?
CAA?A??A?C??TA?TGG?
11101001010011011 1
 11101001010011011 1
```

This results in very few redundant hits

Why is PH better?

BLAST may also miss a hit

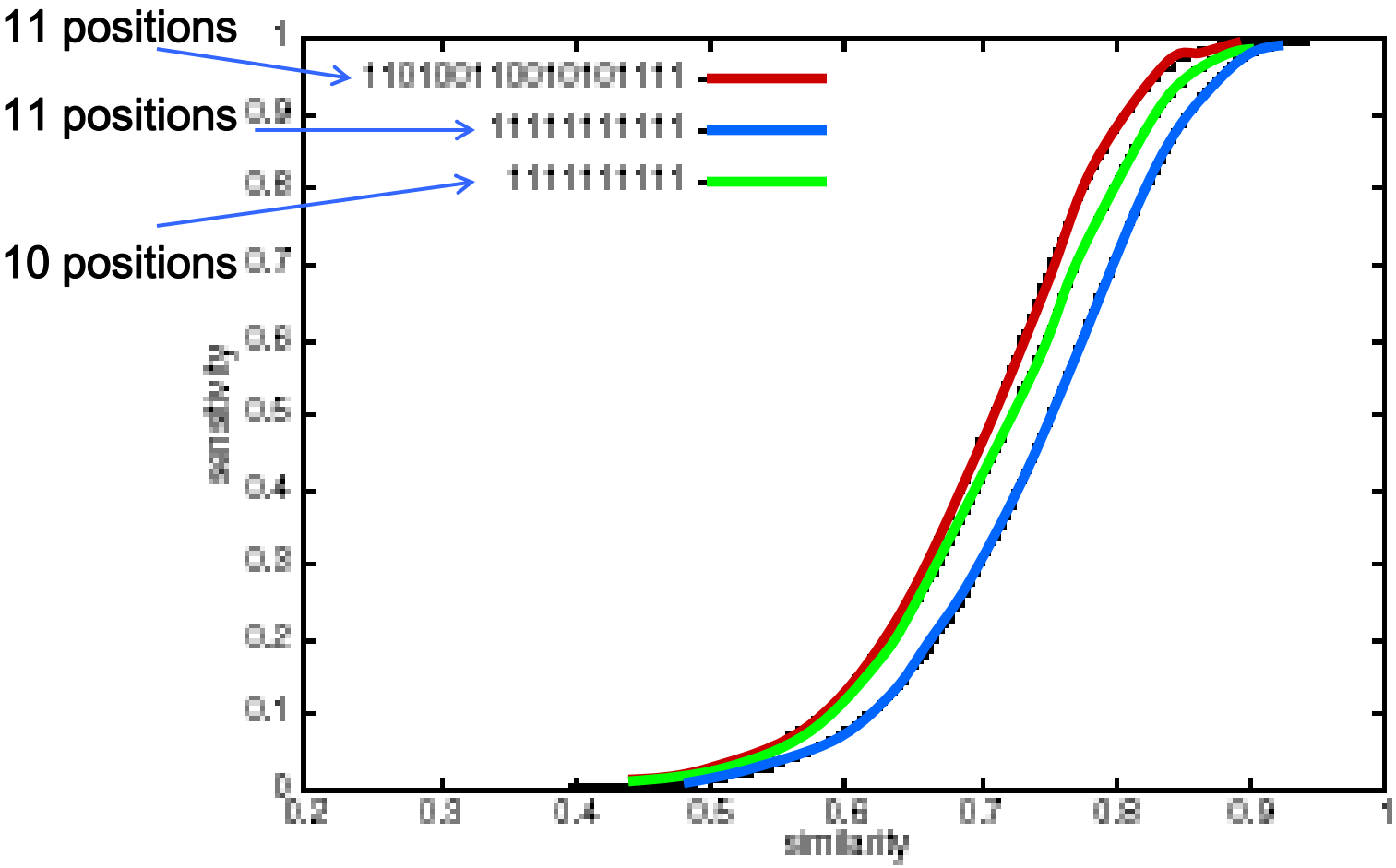
```
GAGTACTCAACACCAACATTAGTGGGCAATGGAAAAT
||  |||||||||  ||||||  |  ||||||    ||||||
GAATACTCAACAGCAACATCAATGGGCAGCAGAAAAT
```

9 matches

In this example, despite a clear homology, there is no sequence of continuous matches longer than length 9. BLAST uses a length 11 and because of this, BLAST does not recognize this as a hit!

Resolving this would require reducing the seed length to 9, which would have a damaging effect on speed

Advantage of Gapped Seeds



Why is PH better?

- Higher hit probability
- Lower expected number of random hits

Use of Multiple Seeds

<u>Basic Searching Algorithm</u>

1. Select a group of spaced seed models
2. For each hit of each model, conduct extension to find a homology.

Another method: BLAT

- BLAT (BLAST-Like Alignment Tool)
- Same idea as BLAST - locate short sequence hits and extend

BLAT vs. BLAST: Differences

- BLAT builds an index of the database and scans linearly through the query sequence, whereas BLAST builds an index of the query sequence and then scans linearly through the database
- Index is stored in RAM which is memory intensive, but results in faster searches

BLAT: Fast cDNA Alignments

Steps:

1. Break cDNA into 500 base chunks.
2. Use an index to find regions in genome similar to each chunk of cDNA.
3. Do a detailed alignment between genomic regions and cDNA chunk.
4. Use dynamic programming to stitch together detailed alignments of chunks into detailed alignment of whole.

BLAT: Indexing

- An index is built that contains the positions of each *k*-mer in the genome
- Each *k*-mer in the query sequence is compared to each *k*-mer in the index
- A list of ‘hits’ is generated - positions in cDNA and in genome that match for *k* bases

Indexing: An Example

Here is an example with *k* = 3:

```
Genome: cacaattatcacgaccgc
3-mers (non-overlapping): cac aat tat cac gac cgc
Index:  aat 3       gac 12
        cac 0,9     tat 6
        cgc 15

cDNA (query sequence): aattctcac
3-mers (overlapping): aat att ttc tct ctc tca cac
                       0   1   2   3   4   5   6
```

Multiple instances map to single index

Position of 3-mer in query, genome

```
Hits: aat 0,3
      cac 6,0
      cac 6,9
clump: cacAATtatCACgaccgc
```

However…

- BLAT was designed to find sequences of 95% and greater similarity of length >40; may miss more divergent or shorter sequence alignments

PatternHunter and BLAT vs. BLAST

- PatternHunter is 5-100 times faster than Blastn, depending on data size, at the same sensitivity
- BLAT is several times faster than BLAST, but best results are limited to closely related sequences

HIDDEN MARKOV MODELS

Outline

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

CG-Islands (=*CpG 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 underrepresented (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

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 states!

Need to incorporate a way to grade different sequences differently.

⇩

Decoding Problem

P(x| fair coin) vs. P(x| biased coin)

- Suppose first that dealer never changes coins. Some definitions:
  - P(x|fair coin): prob. of the dealer using the *F* coin and generating the outcome *x.*
  - P(x|biased coin): prob. of the dealer using the *B* coin and generating outcome *x.*

$P(x|\text{fair coin})$ vs. $P(x|\text{biased coin})$

- $P(x|\text{fair coin})=P(x_1 \ldots x_n|\text{fair coin})$

  $\Pi_{i=1,n}\, p\,(x_i|\text{fair coin})= (1/2)^n$

- $P(x|\text{biased coin})= P(x_1 \ldots x_n|\text{biased coin})=$

$\Pi_{i=1,n}\, p\,(x_i|\text{biased coin})=(3/4)^k(1/4)^{n-k}= 3^k/4^n$

  - k - the number of ***H***eads in x.

P(x| fair coin) vs. P(x| biased coin)

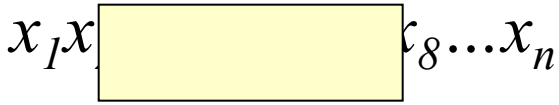
- P(x|fair coin) = P(x|biased coin)
- $1/2^n = 3^k/4^n$
- $2^n = 3^k$
- $n = k \log_2 3$
- when $k = n / \log_2 3$ ($k \sim 0.67n$)

Log-odds Ratio

- We define ***log-odds ratio*** as follows:

$$\log_2(P(x|\text{fair coin}) / P(x|\text{biased coin}))$$
$$= \Sigma^{k}_{i=1} \log_2(p^{+}(x_i) / p^{-}(x_i))$$
$$= n - k \log_2 3$$

Computing Log-odds Ratio in Sliding Windows



Consider a *sliding window* of the outcome sequence. Find the log-odds for this short window.

Disadvantages:
- the length of CG-island is not known in advance
- different windows may classify the same position differently

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?

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.: $\Sigma = \{H, T\}$ for coin tossing

$\Sigma = \{1, 2, 3, 4, 5, 6\}$ 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*.

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

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

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

$e_F(0) = ½ \quad e_F(1) = ½$

$e_B(0) = ¼ \quad e_B(1) = ¾$

HMM for Fair Bet Casino

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

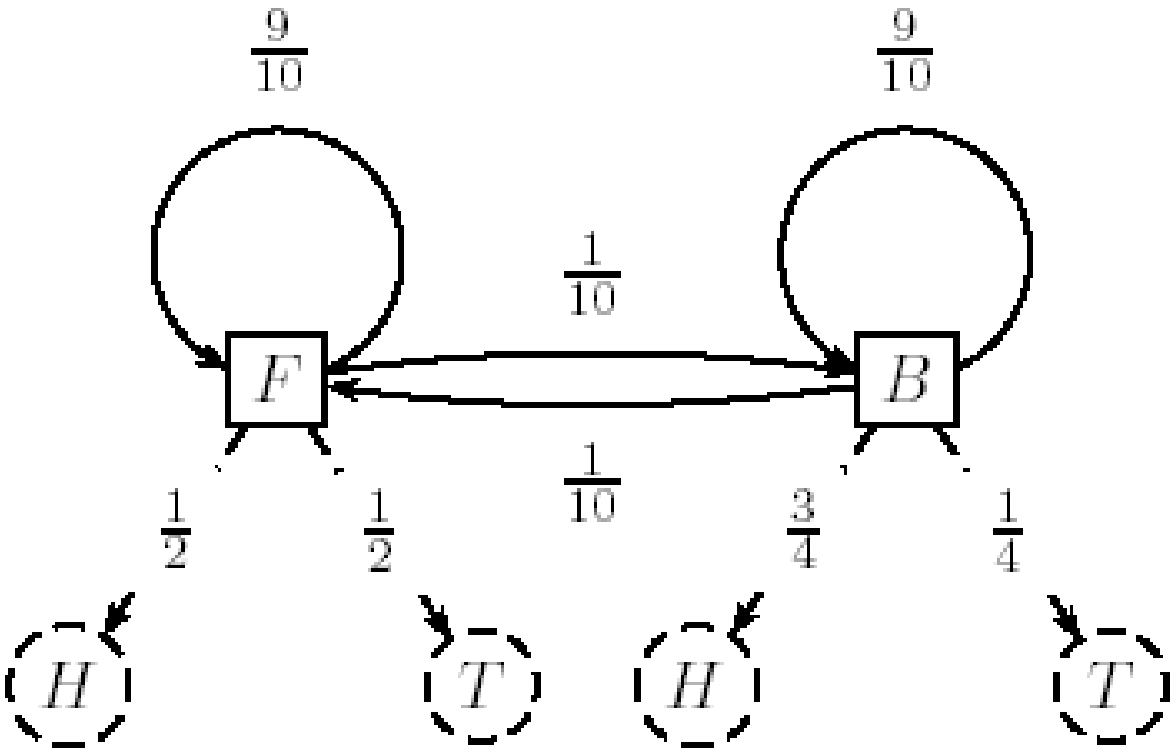
  Σ = {0, 1} (0 for ***T***ails and 1 ***H***eads)

  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	$e_B(0) = ¼$	$e_B(1) = ¾$

HMM for Fair Bet Casino (cont'd)



HMM model for the *Fair Bet Casino* Problem

Hidden Paths

- A *path* $\pi = \pi_1 \ldots \pi_n$ in the HMM is defined as a sequence of states.
- Consider path π = FFFBBBBBFFF and sequence x = 01011101001

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
$P(\pi_{i-1} \rightarrow \pi_i)$	½	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$

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

$P(x|\pi)$ Calculation

- $P(x|\pi)$: Probability that sequence x was generated by the path π:

$$P(x|\pi) = P(\pi_0 \rightarrow \pi_1) \cdot \prod_{i=1}^{n} P(x_i|\pi_i) \cdot P(\pi_i \rightarrow \pi_{i+1})$$

$$= a_{\pi_0, \pi_1} \cdot \prod e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$

$P(x|\pi)$ Calculation

- $P(x|\pi)$: Probability that sequence x was generated by the path π:

$$P(x|\pi) = P(\pi_0 \rightarrow \pi_1) \cdot \prod_{i=1}^{n} P(x_i|\pi_i) \cdot P(\pi_i \rightarrow \pi_{i+1})$$

$$= a_{\pi_0, \pi_1} \cdot \prod e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$

$$= \prod e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$

if we count from $i=0$ instead of $i=1$

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.