Algorithms in Bioinformatics

Randomized Algorithms
Gibbs sampling
+ motif search

Recap

- Final implementation project

Today: Randomization (Ch. 12)

We've used some probability, but haven't yet really focused on randomized algorithms.

Idea: Use randomness to:

- "foil" an adversary:
  choosing something randomly makes pathological cases less likely
- randomly sample:
  Markov-chain: sample huge space but only small part of it
- hashing / fingerprinting:

- load balancing

Note:

Broad Catergories

1 • Las Vegas algorithms:
  - always return correet answers
  - but varying run time

2 • Monte Carlo algorithms:
  - may produce incorrect or approximate solutions
  - very extensively used in bioinformatics, though

First look: Randomized quicksort.

Algorithm: input: S

```
Choose a random element e ("pivot")
Determine
    $S_L = \{\text{elements of } S < e\}$
    $S_G = \{\text{elements of } S > e\}$
Recursively sort $S_L$ + $S_G$
Return $\{\text{sorted } S_L\}$ ++ e ++ $\{\text{sorted } S_G\}$
```

How to do runtime?

Let $S_{(i)}$ be element of <u>rank</u> i in S

(so $S_{(1)}$ is smallest + $S_{(n)}$ is largest)

Let $X_{ij} = \begin{cases} 1 & \text{if } S_{(i)} + S_{(j)} \text{ are compared} \\ 0 & \text{if not} \end{cases}$

(indicator variable)

Then:

Total # of comparisons

$$= \sum_{i=1}^{n} \sum_{j>i} X_{ij}$$

Our goal: expected # of comparisons

$$E\left[\sum_{i=1}^{n} \sum_{j>i} X_{ij}\right]$$

☆ → use <u>linearity of expectation</u>:

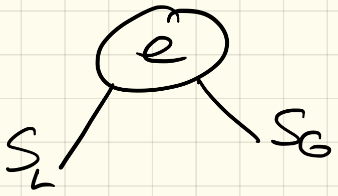
$$= \sum_{i=1}^{n} \sum_{j>i} E[X_{ij}]$$

If we let p_{ij} = prob. i + j are compared, then

$$E[X_{ij}] = 1 \cdot p_{ij} + 0(1-p_{ij})$$
$$= p_{ij}$$

Shifting our view:

View execution as a binary tree, where each node gets labeled with its pivot choice



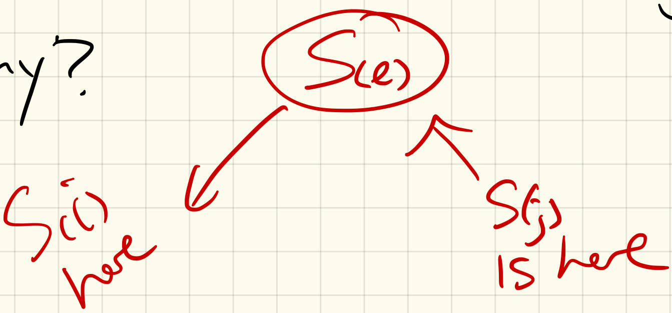
Note: root value s is compared to everything

But: nothing in S_L is ever compared to anything in S_G

2 observations:

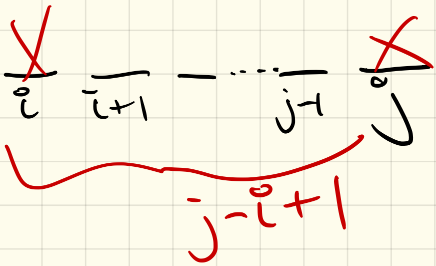
- $S_{(i)}$ & $S_{(j)}$ are compared only if either is chosen as pivot before any $S_{(\ell)}$, with $i < \ell < j$.

Why?



- Any of $S_{(i)}, S_{(i+1)}, \ldots, S_{(j)}$ is equally likely to be chosen as a first partition from this range

So prob. that $S_{(i)}$ or $S_{(j)}$ is first is $= \dfrac{2}{j-i+1}$



So: $P_{ij} = \frac{2}{j-i+1}$

$$E[\#\text{ comparisons}] = \sum_{i=1}^{n} \sum_{j>i} \cancel{P_{ij}} \frac{2}{j-i+1}$$

$$= 2\sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{1}{k}$$

$$\leq 2\sum_{i=1}^{n} \underbrace{\left(\sum_{k=1}^{n} \frac{1}{k}\right)}_{=}$$

Dfn: The nth harmonic number, H_n, is defined as $\sum_{k=1}^{n} \frac{1}{k}$

This is $\approx \underbrace{\ln n + \Theta(1)}_{=}$

$\Rightarrow$ Quick sort runs in $O(n \log n)$ <u>expected</u> time.

Key thing:

- This expected running time holds for every input.

  Randomness depends upon the algorithm making random choices.

- Exact runtime will still vary - we are treating runtime as the random variable.

- This one is still always correct, though.

  (not an approximation)

  (Las Vegas)

Gibbs Sampling:

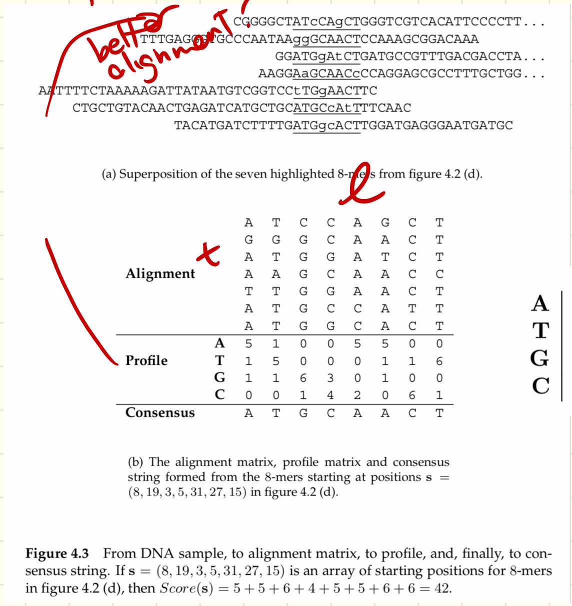
An older method, based on Markov chain Monte Carlo methods - 1953-ish.

First applied to motifs in 1993 by Lawrence et al.

Recall: Motif Finding

Find an l-mer from each of t input sequences s.t. similarity is maximized

find a better alignment?



```
                    CGGGGCTATcCAgCTGGGTCGTCACATTCCCCTT...
              TTTGAGGGTGCCCAATAAggGCAACTCCAAAGCGGACAAA
                              GGATGgAtCTGATGCCGTTTGACGACCTA...
                           AAGGAaGCAACcCCAGGAGCGCCTTTGCTGG...
AATTTTCTAAAAAGATTATAATGTCGGTCCtTGgAACTTC
      CTGCTGTACAACTGAGATCATGCTGCATGCcAtTTTCAAC
                        TACATGATCTTTTGATGgcACTTGGATGAGGGAATGATGC
```

(a) Superposition of the seven highlighted 8-mers from figure 4.2 (d).

l

t

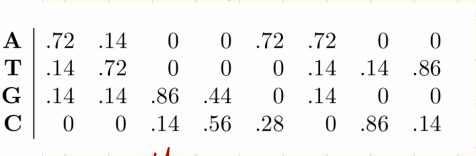
Alignment		A	T	C	C	A	G	C	T
		G	G	G	C	A	A	C	T
		A	T	G	G	A	T	C	T
		A	A	G	C	A	A	C	C
		T	T	G	G	A	A	C	T
		A	T	G	C	C	A	T	T
		A	T	G	G	C	A	C	T
Profile	**A**	5	1	0	0	5	5	0	0
	T	1	5	0	0	0	1	1	6
	G	1	1	6	3	0	1	0	0
	C	0	0	1	4	2	0	6	1
Consensus		A	T	G	C	A	A	C	T

(b) The alignment matrix, profile matrix and consensus string formed from the 8-mers starting at positions $\mathbf{s} = (8, 19, 3, 5, 31, 27, 15)$ in figure 4.2 (d).

Figure 4.3 From DNA sample, to alignment matrix, to profile, and, finally, to consensus string. If $\mathbf{s} = (8, 19, 3, 5, 31, 27, 15)$ is an array of starting positions for 8-mers in figure 4.2 (d), then $Score(\mathbf{s}) = 5 + 5 + 6 + 4 + 5 + 5 + 6 + 6 = 42$.

We'll view it this way.

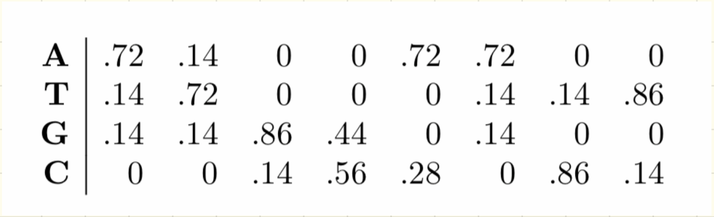
$\downarrow$



A	.72	.14	0	0	.72	.72	0	0
T	.14	.72	0	0	0	.14	.14	.86
G	.14	.14	.86	.44	0	.14	0	0
C	0	0	.14	.56	.28	0	.86	.14

ch 11

Given a profile P:



A	.72	.14	0	0	.72	.72	0	0
T	.14	.72	0	0	0	.14	.14	.86
G	.14	.14	.86	.44	0	.14	0	0
C	0	0	.14	.56	.28	0	.86	.14

+ arbitrary ℓ-mer: $a_1 \dots a_\ell$,

let $$Pr(a|P) = \prod_{i=1}^{\ell} p_{a_i,i}$$

This is the probability that a was generated by P.

Example: $a = ATGCAACT$ consensus string

$$P(a|P) = .72 \times .72 \times .14 \cdots$$

$$\approx 9.6 \times 10^{-2}$$

$$P(TACGCGTC|P)$$

$$=$$

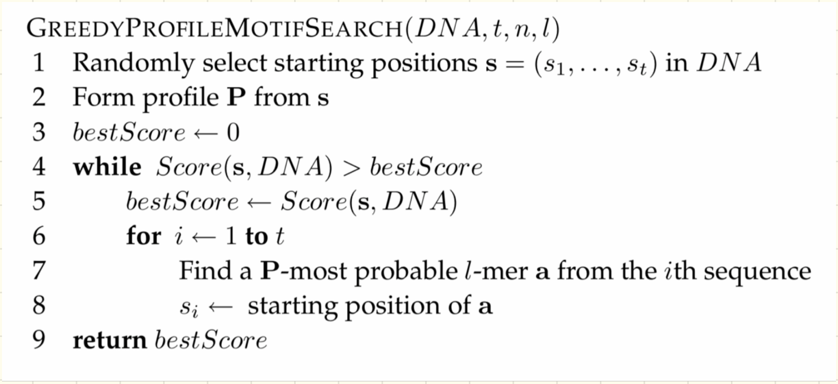
$$\approx 9.3 \times 10^{-7}$$

So: you can evaluate the probability of each l-mer and find the most likely one.

- called the P-most probable l-mer we don't know this

Motivates a random approach:

- Start with random starting positions
- Try to greedily improve



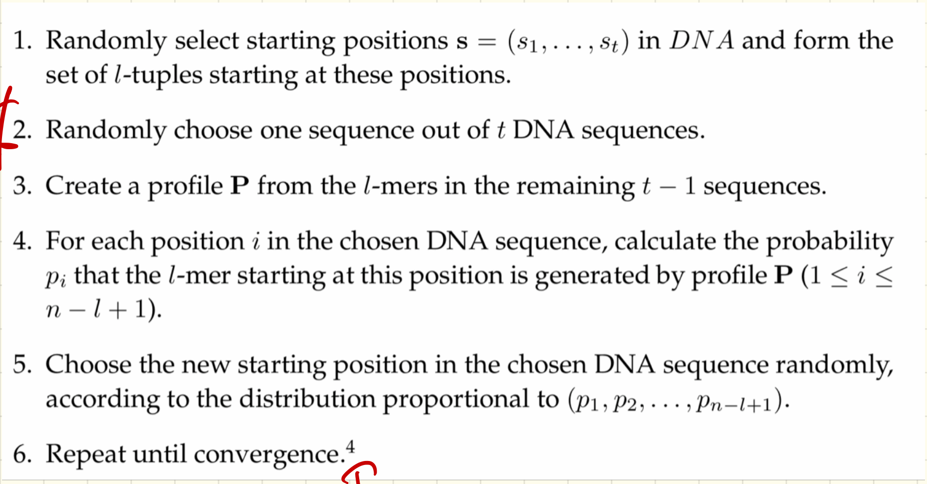
```
GREEDYPROFILEMOTIFSEARCH($DNA, t, n, l$)
Randomly select starting positions $\mathbf{s} = (s_1, \ldots, s_t)$ in $DNA$
Form profile P from s
$bestScore \leftarrow 0$
while $Score(\mathbf{s}, DNA) > bestScore$
    $bestScore \leftarrow Score(\mathbf{s}, DNA)$
    for $i \leftarrow 1$ to $t$
        Find a P-most probable $l$-mer a from the $i$th sequence
        $s_i \leftarrow$ starting position of a
return $bestScore$
```

Problem: It jumps around in large search space.

The last algorithm changes the starting positions in each iteration

Gibbs sampling moves more slowly:

In each iteration, discard <u>one</u> l-mer + replace with a new one.



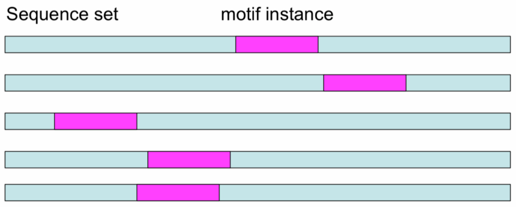
1. Randomly select starting positions $\mathbf{s} = (s_1, \ldots, s_t)$ in DNA and form the set of l-tuples starting at these positions.
2. Randomly choose one sequence out of t DNA sequences.
3. Create a profile $\mathbf{P}$ from the l-mers in the remaining $t-1$ sequences.
4. For each position i in the chosen DNA sequence, calculate the probability p_i that the l-mer starting at this position is generated by profile $\mathbf{P}$ $(1 \leq i \leq n-l+1)$.
5. Choose the new starting position in the chosen DNA sequence randomly, according to the distribution proportional to $(p_1, p_2, \ldots, p_{n-l+1})$.
6. Repeat until convergence.[4]

we don't address this

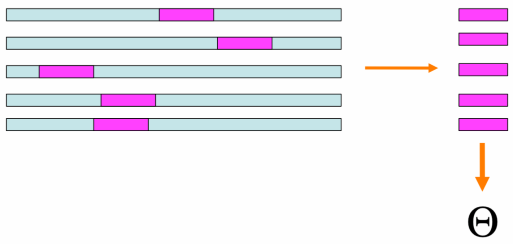
Illustration (from MIT demo):

Gibbs Sampling Algorithm I

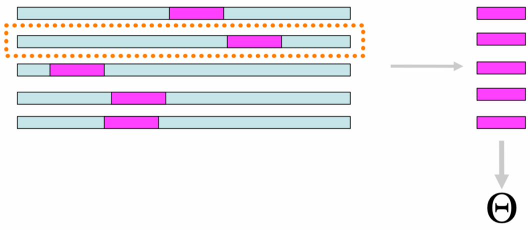
1. Select a random position in each sequence



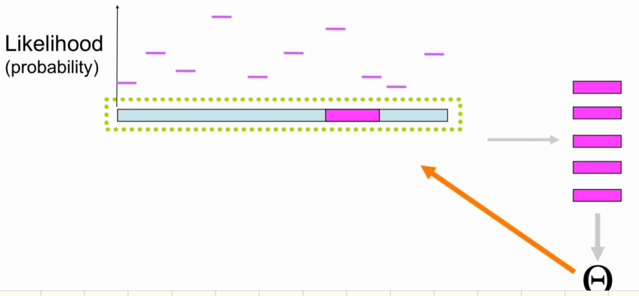
2. Build a weight matrix



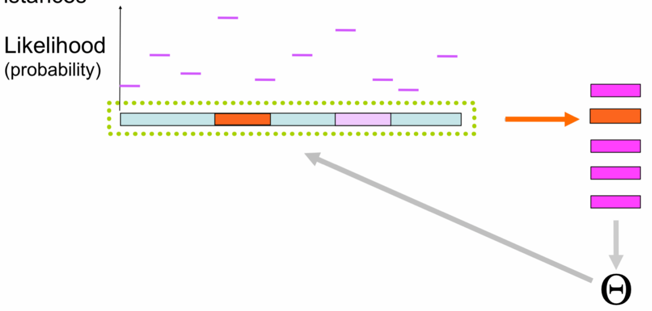
3. Select a sequence at random



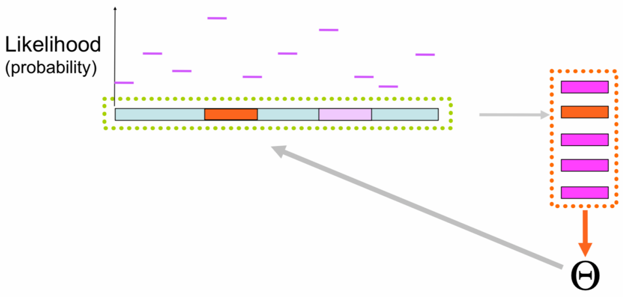
4. Score possible sites in the sequence using weight matrix



5. Sample a new site proportional to likelihood and update motif instances



6. Update weight matrix



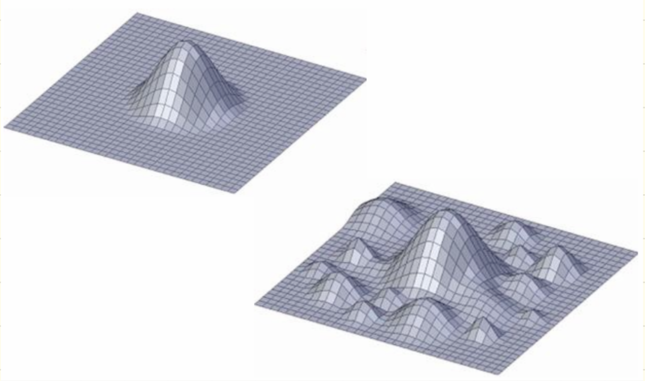
Convergence: $\Delta \text{sites} = 0$ or $\Delta \Theta \approx 0$

The hope:

By moving around (& running several times), we can find a good maximum.

Problem (as always):

What the motif landscape might look like



So - this is a Monte Carlo.

A particular weakness:

If the nucleotide distribution is skewed, ie

- A has 70% frequency

then seach may lead to patterns composed of most frequent, which may not be biologically relevant.

Some use relative entropy to address this:

$$\text{entropy} = \sum_{j=1}^{l} \sum_{r \in \{A,T,C,G\}} P_{rj} \log \frac{P_{rj}}{b_r}$$

where $P_{r,j}$ = frequency of nucleotide $r_{\circ}$ in position j of the alignment

Random Projections

If ℓ-mer is in each DNA strand with mutations, one idea is to try to focus on un-mutated spots.

Gaps may be in different spots, though!

$$\mathrm{CG}\cancel{\_}\mathrm{CAT}\cancel{\_}\mathrm{G}$$

$$\mathrm{CG}\cancel{\_}\mathrm{CA}\cancel{\_}\mathrm{AG}$$

$$\mathrm{C}\cancel{\_}\mathrm{TCA}\cancel{\_}\mathrm{AG}$$

$\hookrightarrow$ "Consensus" gapped pattern is

$$\mathrm{C}\xcancel{\_\,\_}\mathrm{CA}\xcancel{\_\,\_}\mathrm{G}$$

However, these 4 spots are not known!

Random projection: pick them randomly

(k,l)-template t:
any k integers $1 \le t_1 < \cdots < t_k \le l$

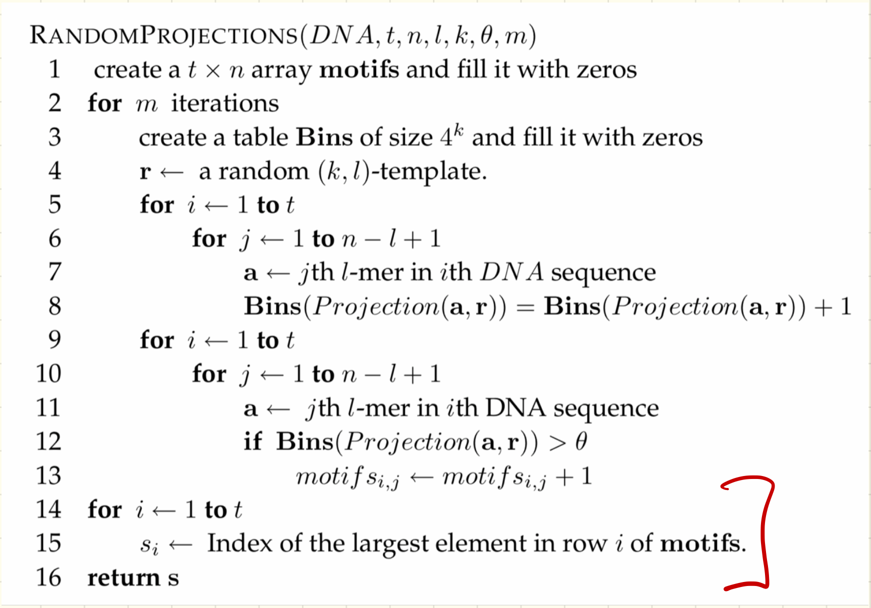
For an l-mer $a_1, \dots, a_l$,

Projection(a, t)
= Concatenation of nucleotides from the template

Example: $a = A\underline{T}GC\underline{A}T\underline{T}$
$t = (2,5,7)$

Proj$(a, t) = TAT$

Idea: choose a random t
project every l-mer with it
record via hash table
Expect likely motif will lead to higher counts.



```
RANDOMPROJECTIONS($DNA, t, n, l, k, \theta, m$)
 create a $t \times n$ array **motifs** and fill it with zeros
for $m$ iterations
    create a table **Bins** of size $4^k$ and fill it with zeros
    $\mathbf{r} \leftarrow$ a random $(k, l)$-template.
    for $i \leftarrow 1$ to $t$
        for $j \leftarrow 1$ to $n - l + 1$
            $\mathbf{a} \leftarrow j$th $l$-mer in $i$th $DNA$ sequence
            $\mathbf{Bins}(Projection(\mathbf{a}, \mathbf{r})) = \mathbf{Bins}(Projection(\mathbf{a}, \mathbf{r})) + 1$
    for $i \leftarrow 1$ to $t$
        for $j \leftarrow 1$ to $n - l + 1$
            $\mathbf{a} \leftarrow$ $j$th $l$-mer in $i$th DNA sequence
            if $\mathbf{Bins}(Projection(\mathbf{a}, \mathbf{r})) > \theta$
                $motifs_{i,j} \leftarrow motifs_{i,j} + 1$
for $i \leftarrow 1$ to $t$
    $s_i \leftarrow$ Index of the largest element in row $i$ of **motifs**.
return **s**
```

Notes:

- Paramaters:
  Selects m random (k,l)-templates

  Aggregates data for all m of them

- θ is significance threshold
- Also need table: Bins(x)
  counts # of l-mers
  s.t. Proj$(a,r) = X$

No guarantee.

However, can show that it 'works with high probability.
(given good parameters)

Practical version by Buhler & Tompa is a bit more complex, but uses a heuristic method to choose the final $(s_1, \ldots, s_t)$