Towards a more adeguate motif scoring function

CSRE element in yeast

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C	G/C	G/T	T/A	C/T	G/C	C/G	A	T	G/T	C/G	A	T	C/T	C/T	G/T

5 strongly conserved positions and eleven weakly conserved positions each of which features two uncle with similar frequencies.

NF-kB

COUNT(*Motifs*)

A:	2	2	0	0	0	0	9	1	1	1	3	0
C:	1	6	0	0	0	0	0	4	1	2	4	6
G:	0	0	10	10	9	9	1	0	0	0	0	0
T:	7	2	0	0	1	1	0	5	8	7	3	4

1	2	3	4	5	6	7	8	9	10	11	12
T	C	G	G	G	G	A	T/C	T	T	C	C/T

last column is *more conserved* than the second column and should receive a lower score

Entropy

With matrices

Entropy corresponds to a probability distribution, it is a measure of the uncertainty of a probability distribution.

COUNT(*Motifs*)

A:	2	2	0	0	0	0	9	1	1	1	3	0
C:	1	6	0	0	0	0	0	4	1	2	4	6
G:	0	0	10	10	9	9	1	0	0	0	0	0
T:	7	2	0	0	1	1	0	5	8	7	3	4

→ 0.2 log2 0.2+ 0.6 log2 0.6+ 0 log2 0+ 0.2 log2 0.2 = 1.371

→ 0 log2 0+ 0.6 log2 0.6+ 0 log2 0+ 0.4 log2 0.4 = 0.971

→ 0 log2 0+ 0 log2 0+ 0.9 log2 0.9+ 0.1 log2 0.1 = 0.467

The more conserved the columns the smaller its entropy

The motif finding problem - *Score*

With matrices

Motif Finding Problem

given a collection of strings, find a set of k-mers, one from each string, that minimises the score of the resulting motif.

input: A collection of strings DNA and a integer k

output: A collection Motif of k-mers, one from each string in DNA, minimising SCORE(Motifs) among all possible choices of k-mers

	T	C	G	G	G	G	g	T	T	T	t	t
	c	C	G	G	t	G	A	c	T	T	a	C
	a	C	G	G	G	G	A	T	T	T	t	C
	T	t	G	G	G	G	A	c	T	T	t	t
Motifs	a	a	G	G	G	G	A	c	T	T	C	C
	T	t	G	G	G	G	A	c	T	T	C	C
	T	C	G	G	G	G	A	T	T	c	a	t
	T	C	G	G	G	G	A	T	T	c	C	t
	T	a	G	G	G	G	A	a	c	T	a	C
	T	C	G	G	G	t	A	T	a	a	C	C
SCORE(*Motifs*)	3	+ 4	+ 0	+ 0	+ 1	+ 1	+ 1	+ 5	+ 2	+ 3	+ 6	+ 4 = 30

BruteForce algorithm, for every possibile kmer from a DNA it is necessary compute the score value and returns those having the minimum score

The motif finding problem - *Profile*

With matrices

Motifs be a collection of k-mers taken from t strings of DNA

The *entropy* of each column of a PWM is like a four-side dice:

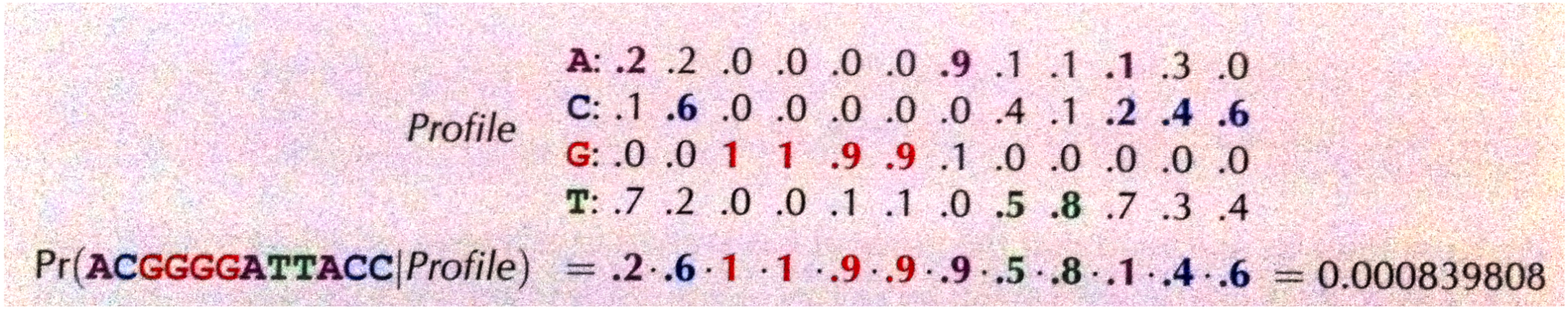
A 0.2
C 0.1
G 0.0
T 0.7

A generated with probability 0.2

C generated with probability 0.1

G generated with probability 0.0

T generated with probability 0.7



Profile-most Probable k-mer Problem

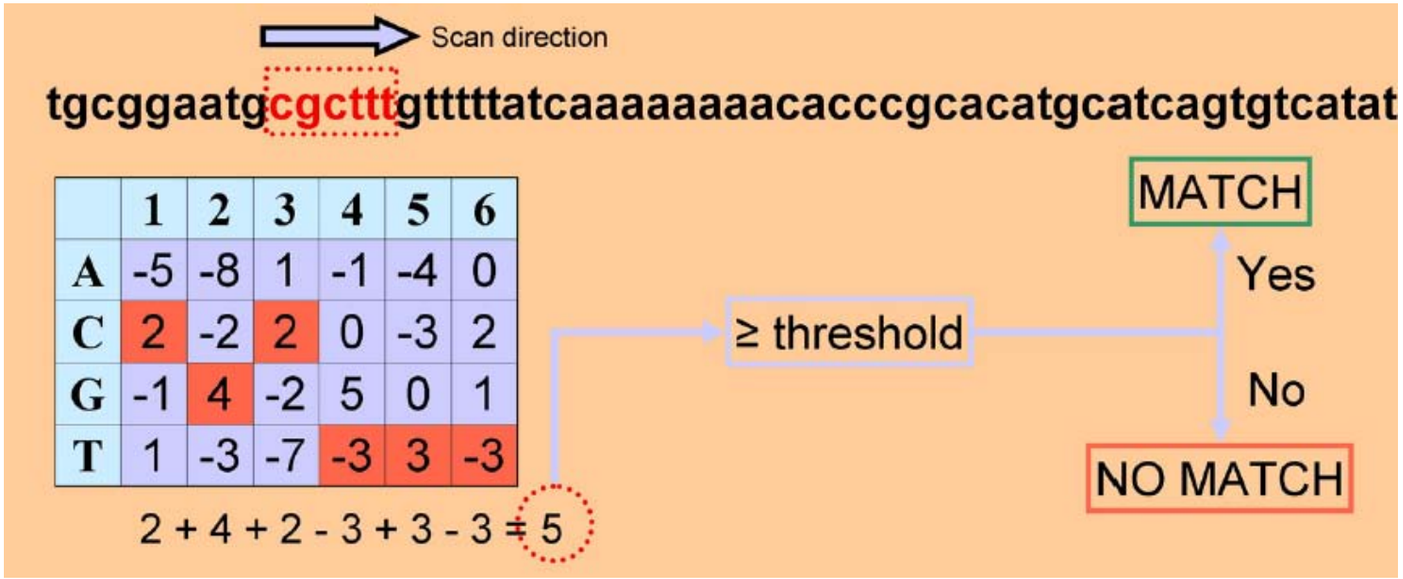
find a Profile-most probable k-mer in a string

input: A string DNA, a integer k, and a 4 X k matrix Profile

output: A profile-most probable k-mer in DNA

With matrices

Scoring a Sequence



Courtesy of Kenzie MacIsaac and Ernest Fraenkel. Used with permission. MacIsaac, Kenzie, and Ernest Fraenkel. "Practical Strategies for Discovering Regulatory DNA Sequence Motifs." *PLoS Computational Biology* 2, no. 4 (2006): e36.

Common threshold = 60% of maximum score

MacIsaac & Fraenkel (2006) PLoS Comp Bio

Scoring A Sequence

To score a sequence, we compare to a null model

$$Score = \log \frac{P(S \mid PFM)}{P(S \mid B)} = \log \prod_{i=1}^{N} \frac{P_i(S_i \mid PFM)}{P(S_i \mid B)} = \sum_{i=1}^{N} \underbrace{\log \frac{P_i(S_i \mid PFM)}{P(S_i \mid B)}}$$

Without matrices

Equivalent Motif Finding Problem

given a collection of strings, find a pattern and a collection of k-mers (one for each string) that minimises the distance between all possible patterns and all possibile collections of k-mers.

input: A collection of strings DNA and a integer k

output: A k-mer pattern and a collection of k-mers Motifs, one from each string in DNA, minimising d(Pattern, Motifs) among all possible choices of Pattern and Motifs

d(pattern, motifs) = HAMMING DISTANCE (Pattern, Motif)

BUT given a Pattern, we do not need to explore all possible collection of Motifs in order to minimise d(Pattern, Motifs)

$$d(Pattern, Text) = \min_{\text{all } k\text{-mers } Pattern' \text{ in } Text} \text{HAMMINGDISTANCE}(Pattern, Pattern').$$

$$d(\mathbf{GATTCTCA}, \text{gcaaa}\mathbf{GACGCTGA}\text{ccaa}) = 3.$$

$$\text{MOTIF}(\mathbf{GATTCTCA}, \text{gcaaa}\mathbf{GACGCTGA}\text{ccaa}) = \mathbf{GACGCTGA}.$$

Median string Problem

find a median string

input: A collection of strings DNA and a integer k

output: A k-mer pattern minimising d(Pattern, DNA) among all k-mers Pattern

```
MEDIANSTRING(Dna, k)
    distance ← ∞
    for each k-mer Pattern from AA...AA to TT...TT
        if distance > d(Pattern, Dna)
            distance ← d(Pattern, Dna)
            Median ← Pattern
    return Median
```

Greedy motif search

Greedy algorithm select the most attractive alternative at each iteration

```
GREEDYMOTIFSEARCH(Dna, k, t)
    BestMotifs ← motif matrix formed by first k-mers in each string from Dna
    for each k-mer Motif in the first string from Dna
        Motif_1 ← Motif
        for i = 2 to t
            form Profile from motifs Motif_1, ..., Motif_{i−1}
            Motif_i ← Profile-most probable k-mer in the i-th string in Dna
        Motifs ← (Motif_1, ..., Motif_t)
        if SCORE(Motifs) < SCORE(BestMotifs)
            BestMotifs ← Motifs
    return BestMotifs
```

```
ttACCTtaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtcagAGGT
```

If the algorithm has chosen ACCT from the first string, what is the profile?

Laplace's rule of succession

Profile												
A:	.2	.2	.0	.0	.0	.0	**.9**	.1	.1	.1	.3	.0
C:	.1	**.6**	.0	.0	.0	.0	.0	.4	.1	.2	**.4**	**.6**
G:	.0	.0	**1**	1	**.9**	**.9**	.1	.0	.0	.0	.0	.0
T:	.7	.2	.0	.0	.1	.1	.0	.5	.8	.7	.3	.4

$$Pr(\text{TCGTGGATTTCC}|Profile) = .7 \cdot .6 \cdot 1 \cdot .0 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .7 \cdot .4 \cdot .6 = 0$$

To improve the unfair scoring, often substitute zeros with small number called **pseudo counts**

Motifs				
	T	A	A	C
	G	T	C	T
	A	C	T	A
	A	G	G	T

COUNT(*Motifs*)				
A:	2	1	1	1
C:	0	1	1	1
G:	1	1	1	0
T:	1	1	1	2

PROFILE(*Motifs*)				
	2/4	1/4	1/4	1/4
	0	1/4	1/4	1/4
	1/4	1/4	1/4	0
	1/4	1/4	1/4	2/4

Laplace's rule of succession adds 1 to each element to COUNT(motif)

COUNT(*Motifs*)				
A:	2+1	1+1	1+1	1+1
C:	0+1	1+1	1+1	1+1
G:	1+1	1+1	1+1	0+1
T:	1+1	1+1	1+1	2+1

PROFILE(*Motifs*)				
	3/8	2/8	2/8	2/8
	1/8	2/8	2/8	2/8
	2/8	2/8	2/8	1/8
	2/8	2/8	2/8	3/8

Greedy motif search

Greedy algorithm select the most attractive alternative at each iteration

```
GREEDYMOTIFSEARCH(Dna, k, t)
    BestMotifs ← motif matrix formed by first k-mers in each string from Dna
    for each k-mer Motif in the first string from Dna
        Motif_1 ← Motif
        for i = 2 to t
            form Profile from motifs Motif_1, ..., Motif_i-1
            Motif_i ← Profile-most probable k-mer in the i-th string in Dna
        Motifs ← (Motif_1, ..., Motif_t)
        if SCORE(Motifs) < SCORE(BestMotifs)
            BestMotifs ← Motifs
    return BestMotifs
```

apply **Laplace's Rule of Succession** to form Profile from Motif1… Motif i-1

Greedy motif search - in action

With matrices

Dna	ttACCTtaac
	gATGTctgtc
	acgGCGTtag
	ccctaACGAg
	cgtcagAGGT

		Motifs	ACCT			
COUNT(*Motifs*)	A:	1+1	0+1	0+1	0+1	
	C:	0+1	1+1	1+1	0+1	
	G:	0+1	0+1	0+1	0+1	
	T:	0+1	0+1	0+1	1+1	
PROFILE(*Motifs*)		2/5	1/5	1/5	1/5	
		1/5	2/5	2/5	1/5	
		1/5	1/5	1/5	1/5	
		1/5	1/5	1/5	2/5	

If we use profile matrix to compute the probabilities of all 4-mres in the second string of DNA

gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
$1/5^4$	$4/5^4$	$1/5^4$	$4/5^4$	$2/5^4$	$2/5^4$	$1/5^4$

Motifs	ACCT
	ATGT

We get lucky and we choose the implanted 4-mers ATGT

	Motifs	ACCT				
		ATGT				
COUNT(*Motifs*)	A:	2+1	0+1	0+1	0+1	
	C:	0+1	1+1	1+1	0+1	
	G:	0+1	0+1	1+1	0+1	
	T:	0+1	1+1	0+1	2+1	
PROFILE(*Motifs*)		3/6	1/6	1/6	1/6	
		1/6	2/6	2/6	1/6	
		1/6	1/6	2/6	1/6	
		1/6	2/6	1/6	3/6	

Greedy motif search - in action

Dna	ttACCTtaac
	gATGTctgtc
	acgGCGTtag
	ccctaACGAg
	cgtcagAGGT

acgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$12/6^4$	$2/6^4$	$2/6^4$	$12/6^4$	$3/6^4$	$2/6^4$	$2/6^4$

This time we will assume that ACGG is selected instead the GCGT

Motifs		ACCT ATGT acgG
COUNT(*Motifs*)	A: 3+1 0+1 0+1 1+1 C: 0+1 2+1 1+1 0+1 G: 0+1 0+1 2+1 1+1 T: 0+1 1+1 0+1 2+1	
PROFILE(*Motifs*)		4/7 1/7 1/7 1/7 1/7 3/7 2/7 1/7 1/7 1/7 3/7 2/7 1/7 2/7 1/7 3/7

ccct	ccta	ctaA	taAC	aACG	ACGA	CGAg
$18/7^4$	$3/7^4$	$2/7^4$	$1/7^4$	$16/7^4$	$36/7^4$	$2/7^4$

While in this case the profile-most probable 4-mer is ACGA

Motifs		ACCT ATGT acgG ACGA
COUNT(*Motifs*)	A: 4+1 0+1 0+1 0+1 C: 0+1 3+1 1+1 0+1 G: 0+1 0+1 3+1 1+1 T: 0+1 1+1 0+1 2+1	
PROFILE(*Motifs*)		5/8 1/8 1/8 2/8 1/8 4/8 2/8 1/8 1/8 1/8 4/8 2/8 1/8 2/8 1/8 3/8

Greedy motif search - in action

cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
$1/8^4$	$8/8^4$	$8/8^4$	$8/8^4$	$10/8^4$	$8/8^4$	$60/8^4$

Dna	ttACCTtaac
	gATGTctgtc
	acgGCGTtag
	ccctaACGAg
	cgtcagAGGT

	ACCT
	ATGT
Motifs	acgG
	ACGA
	AGGT
CONSENSUS(*Motifs*)	ACGT

Laplace's Rule of Succession has provided
a great improvement over the original
GreedyMotifSearch

Randomised Motif Search

Randomised algorithms may be nonintuitive because they lack the control of traditional algorithms

These algorithms are not guaranteed to return exact solutions, but they quickly find approximate solutions

Given a collection of strings DNA and an arbitrary 4 x k matrix profile, we define MOTIF(Profile,DNA) as the collections of k-mers formed by the Profile-most probable k-mers in each sequence from DNA.

Considering profile and DNA

Profile					*Dna*
A:	4/5	0	0	1/5	ttaccttaac
C:	0	3/5	1/5	0	gatgtctgtc
G:	1/5	1/5	4/5	0	acggcgttag
T:	0	1/5	0	4/5	ccctaacgag
					cgtcagaggt

the profile-most 4-mer from each row of DNA produces the following 4-mers:

MOTIFS(*Profile*, *Dna*)

ttaccttaac
gatgtctgtc
acggcgttag
ccctaacgag
cgtcagaggt

The **general idea** is that we can begin from a collection of randomly chosen k-mers Motif in DNA construct the profile and use this profile to generate a new collection of k-mers.

Randomised Motif Search

```
RANDOMIZEDMOTIFSEARCH(Dna, k, t)
    randomly select k-mers Motifs = (Motif_1, ..., Motif_t) in each string from Dna
    BestMotifs ← Motifs
    while forever
        Profile ← PROFILE(Motifs)
        Motifs ← MOTIFS(Profile, Dna)
        if SCORE(Motifs) < SCORE(BestMotifs)
            BestMotifs ← Motifs
        else
            return BestMotifs
```

A single run may generate a poor set of motifs, then usually it is necessary run this algorithm thousand of times. In each run it begin from a new randomly selected set of k-mers and finally we select the best set of k-mers found in all these runs.

Randomised Motif Search - in action

	ttACCT**taac**
	gAT**GTct**gtc
Dna	**ccgG**CGTtag
	c**acta**ACGAg
	cgtcag**AGGT**

We construct the profile matrix of the chosen 4-mers

Motifs			
t	a	a	c
G	T	c	t
c	c	g	G
a	c	t	a
A	G	G	T

PROFILE(*Motifs*)				
A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.2	0.4	0.2
T:	0.2	0.2	0.2	0.4

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	**.0128**	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	**.0128**	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	**.0128**	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	**.0128**	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	**.0128**

We compute the probabilities of every 4-mer in DNA based on this profile matrix.

	tt**ACCT**taac
	g**ATGT**ctgtc
Dna	ccg**GCGT**tag
	cacta**ACGA**g
	cgtcag**AGGT**

Randomised Motif Search

Starting from a **uniform distribution** as the first profile matrix is useless because no string is more probable than any other according to this profile and it does not provide any clues on what an implanted motif looks like.

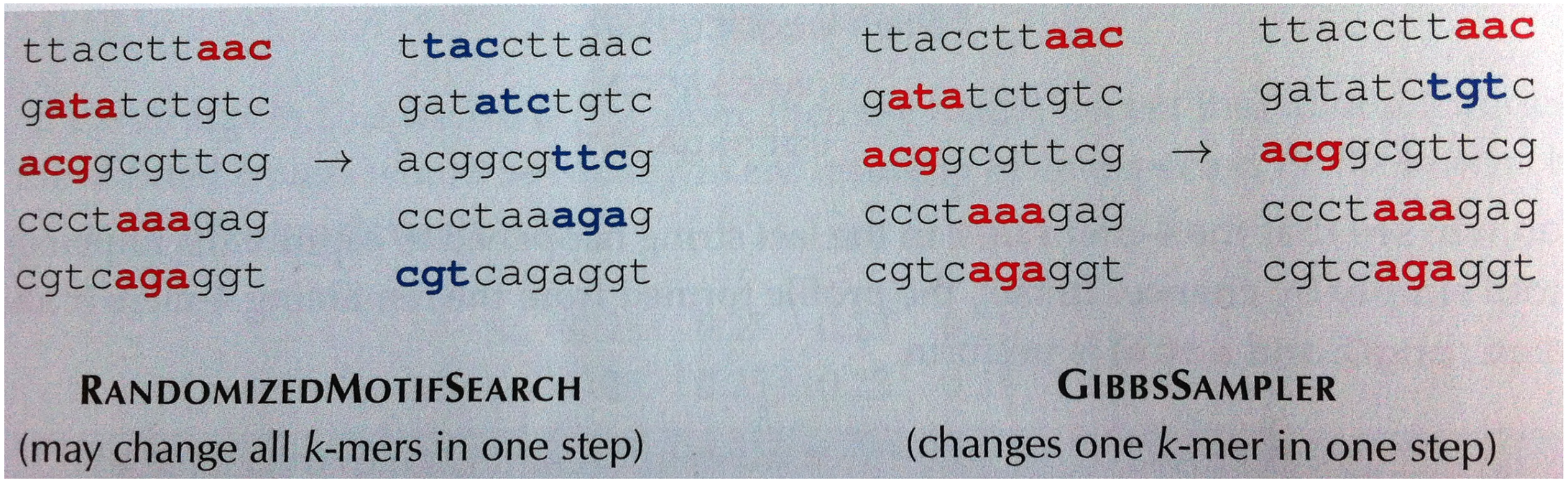
Considering that if the strings in DNA were random, then the algorithm would start form a nearly uniform profile, and there would be nothing to work with. **The KEY observation is that the strings in DNA are NOT random**.

Gibbs Sampling

AlignACE
Bioprospector

The randomise strategy can discarded all k-mers in each iteration, while Gibbs sampling is more conservative.

The algorithm starts from randomly chosen k-mers in each DNA, and select an integer between 1 and t and randomly changes a single k-mer.



Gibbs Sampling

```
GIBBSSAMPLER(Dna, k, t, N)
    randomly select k-mers Motifs = (Motif_1, ..., Motif_t) in each string from Dna
    BestMotifs ← Motifs
    for j ← 1 to N
        i ← RANDOM(t)
        Profile ← profile matrix formed from all strings in Motifs except for Motif_i
        Motif_i ← Profile-randomly generated k-mer in the i-th sequence
        if SCORE(Motifs) < SCORE(BestMotifs)
            BestMotifs ← Motifs
    return BestMotifs
```

Profile-randomly generated k-mer in the i-th sequence

Gibbs sampling uses an advice random number generator.

Gibbs Sampling - in action

	ttACCT**taac**		ttACCT**taac**
	gAT**GTct**gtc		gAT**GTct**gtc
Dna	**ccgG**CGTtag	⟶	----------
	c**acta**ACGAg		c**acta**ACGAg
	cgtcag**AGGT**		cgtcag**AGGT**

Initial step, from a set of random k-mers, the algorithm select the third string for removal.

Motifs

t a a c
G T c t
a c t a
A G G T

COUNT(*Motifs*)

A:	2	1	1	1
C:	0	1	1	1
G:	1	1	1	0
T:	1	1	1	2

PROFILE(*Motifs*)

A:	2/4	1/4	1/4	1/4
C:	0	1/4	1/4	1/4
G:	1/4	1/4	1/4	0
T:	1/4	1/4	1/4	2/4

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
0	0	0	1/128	0	1/256	0

Gibbs Sampling - in action

$$\text{COUNT}(\textit{Motifs}) = \begin{matrix} \text{A:} & 3 & 2 & 2 & 2 \\ \text{C:} & 1 & 2 & 2 & 2 \\ \text{G:} & 2 & 2 & 2 & 1 \\ \text{T:} & 2 & 2 & 2 & 3 \end{matrix} \qquad \text{PROFILE}(\textit{Motifs}) = \begin{matrix} \text{A:} & 3/8 & 2/8 & 2/8 & 2/8 \\ \text{C:} & 1/8 & 2/8 & 2/8 & 2/8 \\ \text{G:} & 2/8 & 2/8 & 2/8 & 1/8 \\ \text{T:} & 2/8 & 2/8 & 2/8 & 3/8 \end{matrix}$$

application of Laplace's Rule to the count matrix

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$4/8^4$	$8/8^4$	$8/8^4$	$24/8^4$	$12/8^4$	$16/8^4$	$8/8^4$

Gibbs sampling is not deterministic, then create a seven-face dice in which the total sum is equal to $80/8^4$

$$\text{RANDOM}\left(\frac{4/8^4}{80/8^4}, \frac{8/8^4}{80/8^4}, \frac{8/8^4}{80/8^4}, \frac{24/8^4}{80/8^4}, \frac{12/8^4}{80/8^4}, \frac{16/8^4}{80/8^4}, \frac{8/8^4}{80/8^4}\right)$$
$$= \text{RANDOM}\left(\frac{4}{80}, \frac{8}{80}, \frac{8}{80}, \frac{24}{80}, \frac{12}{80}, \frac{16}{80}, \frac{8}{80}\right).$$

The deleted string is now added but instead to use the **ccgG** mer, we roll the die and we obtain the mer **GCGT**

Gibbs Sampling - in action

	ttACCTtaac		----------
	gATGTctgtc		gATGTctgtc
Dna	ccgGCGTtag	⟶	ccgGCGTtag
	cactaACGAg		cactaACGAg
	cgtcagAGGT		cgtcagAGGT

Randomly selection of the deletion of the first DNA string

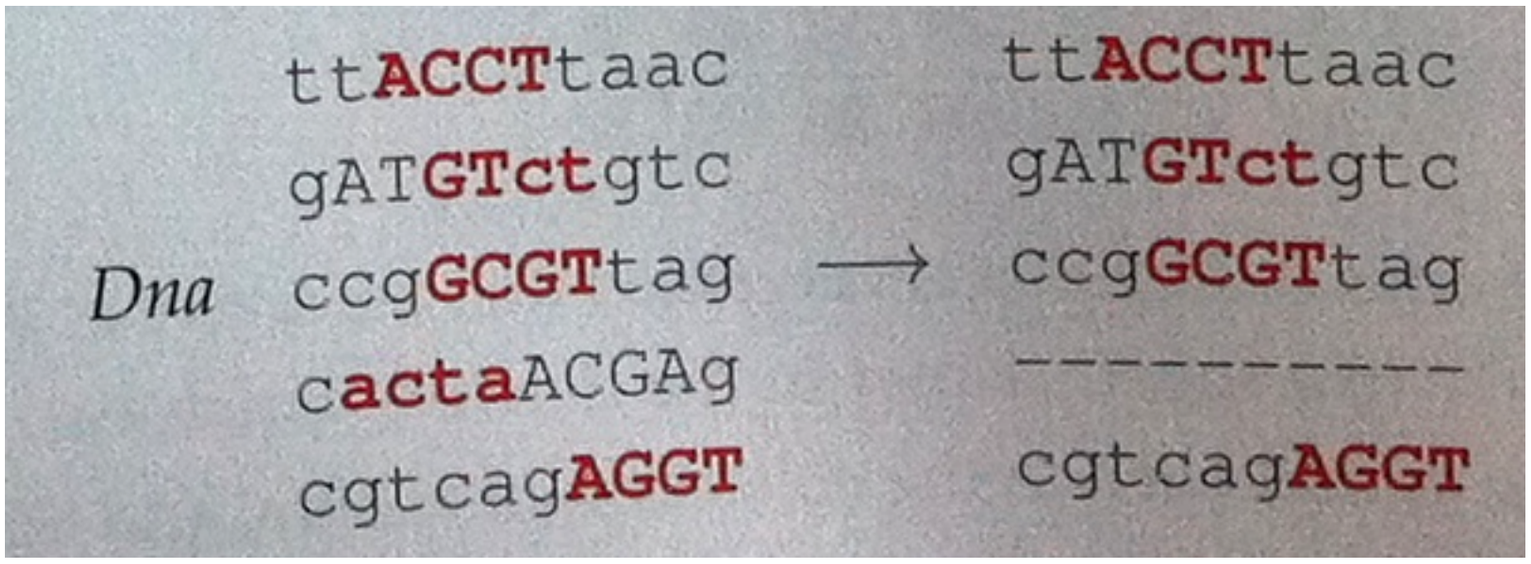
Motifs		PROFILE(*Motifs*)				
	G T c t	A:	2/4	0	0	1/4
	G C G T	C:	0	2/4	1/4	0
	a c t a	G:	2/4	1/4	2/4	0
	A G G T	T:	0	1/4	1/4	3/4

application of Laplace's Rule to the count matrix

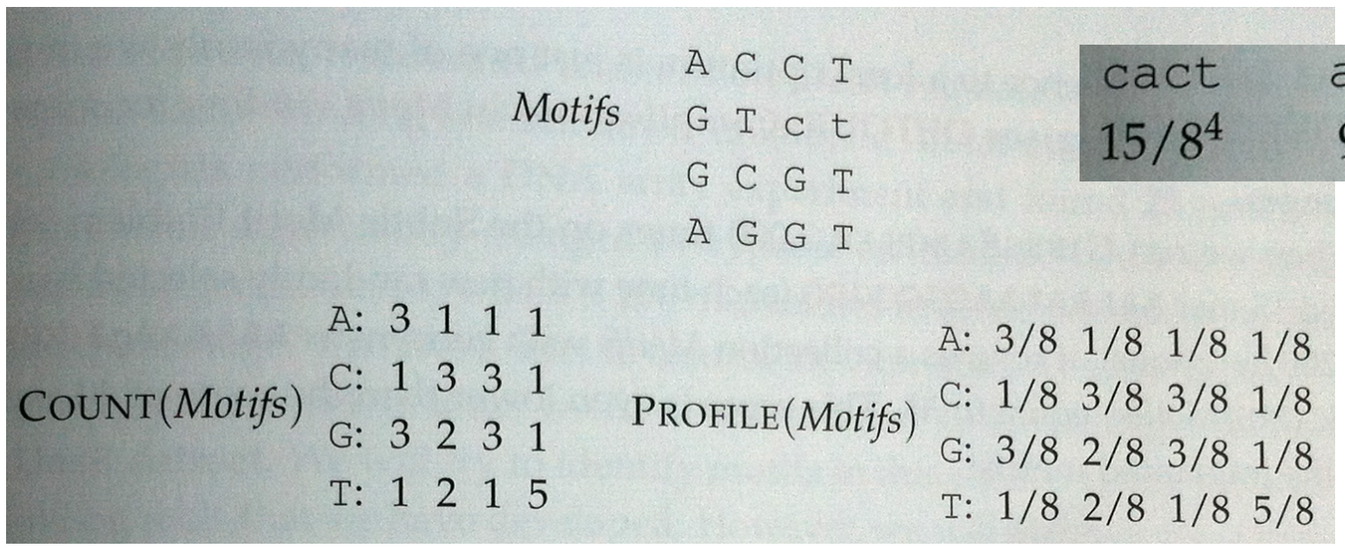
COUNT(*Motifs*)					PROFILE(*Motifs*)				
A:	3	1	1	2	A:	3/8	1/8	1/8	2/8
C:	1	3	2	1	C:	1/8	3/8	2/8	1/8
G:	3	2	3	1	G:	3/8	2/8	3/8	1/8
T:	1	2	2	4	T:	1/8	2/8	2/8	4/8

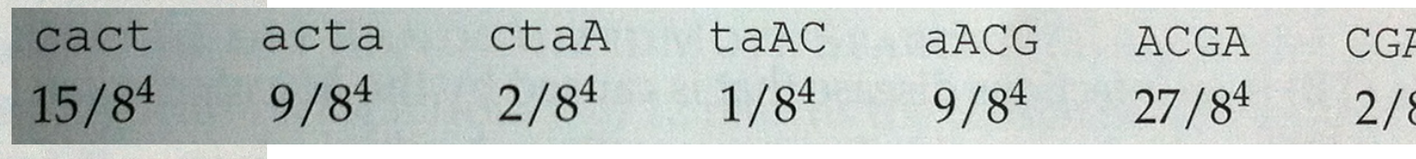
ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
$2/8^4$	$2/8^4$	$72/8^4$	$24/8^4$	$8/8^4$	$4/8^4$	$1/8^4$

Gibbs Sampling - in action

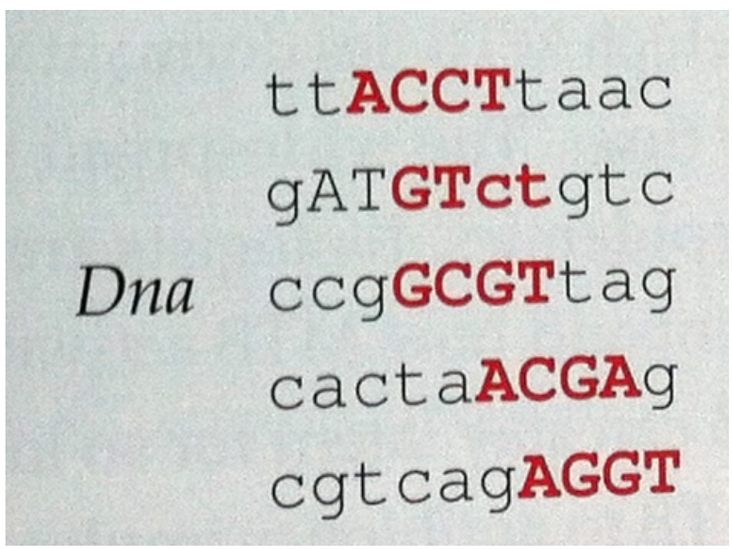


Randomly selection of the deletion of the fourth DNA string





Profile-randomly generated k-mer

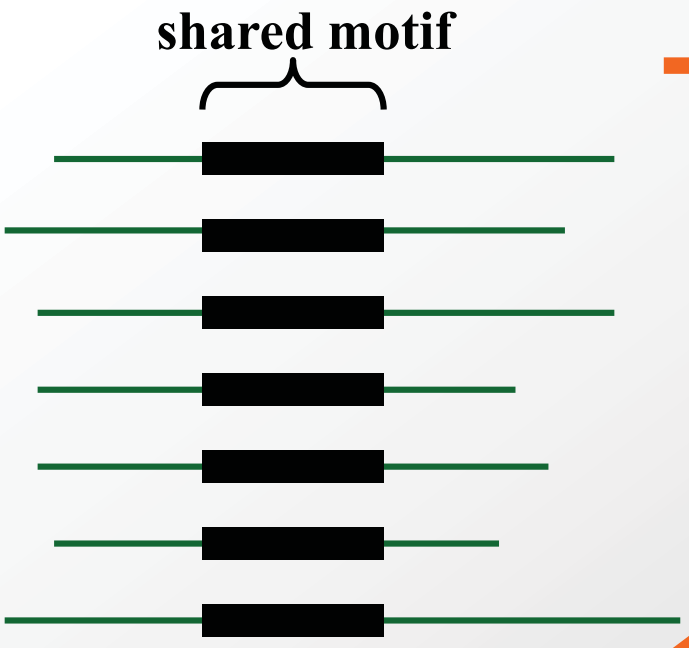


The algorithm starts to converge, but in same cases the algorithm can converge in a suboptimal solution.

A **local optimum** si a solution that is optimal witting a small neighbouring set of solutions, which is in contact to the **global optimum** or the optimal solutions among all possible solutions.

Starting positions ⇔ Motif matrix

- given aligned sequences ➔ easy to compute profile matrix



sequence positions

	1	2	3	4	5	6	7	8
A	0.1	0.3	0.1	0.2	0.2	0.4	0.3	0.1
C	0.5	0.2	0.1	0.1	0.6	0.1	0.2	0.7
G	0.2	0.2	0.6	0.5	0.1	0.2	0.2	0.1
T	0.2	0.3	0.2	0.2	0.1	0.3	0.3	0.1

given profile matrix

- easy to find starting position probabilities

Key idea: Iterative procedure for estimating both, given uncertainty

(learning problem with hidden variables: the starting positions)

Basic Iterative Approach

Given: length parameter *W*, training set of sequences

set initial values for **motif**

do

➔ re-estimate ***starting-positions*** from ***motif***

➔ re-estimate ***motif*** from ***starting-positions***

until convergence (change < ε)

return: ***motif, starting-positions***

Representing Motif M(k,c) and Background B(c)

- Assume motif has fixed width, *W*
- Motif represented by matrix of probabilities: ***M(k,c)***
  the probability of character ***c*** in column ***k***

$$M = \begin{array}{c|ccc} & 1 & 2 & 3 \\ A & 0.1 & 0.5 & 0.2 \\ C & 0.4 & 0.2 & 0.1 \\ G & 0.3 & 0.1 & 0.6 \\ T & 0.2 & 0.2 & 0.1 \end{array}$$

(~CAG)

- **Background represented by B(c), frequency of each base**

$$B = \begin{array}{c|c} A & 0.26 \\ C & 0.24 \\ G & 0.23 \\ T & 0.27 \end{array}$$

(near uniform)

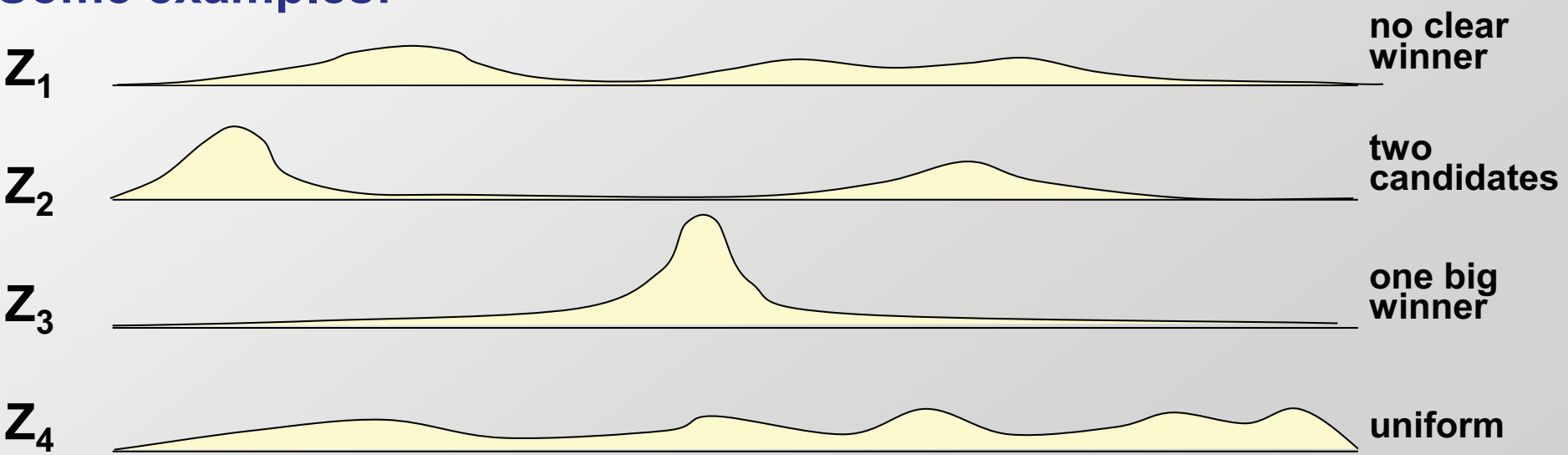
(see also: di-nucleotide etc)

Representing the starting position probabilities (Z_{ij})

- the element Z_{ij} of the matrix Z represents the probability that the motif starts in position *j* in sequence *i*

$$Z = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline \texttt{seq1} & 0.1 & 0.1 & 0.2 & 0.6 \\ \texttt{seq2} & 0.4 & 0.2 & 0.1 & 0.3 \\ \texttt{seq3} & 0.3 & 0.1 & 0.5 & 0.1 \\ \texttt{seq4} & 0.1 & 0.5 & 0.1 & 0.3 \end{array}$$

Some examples:



Starting positions (Z_{ij}) ⇔ Motif matrix M(k,c)

M-step

E-step

	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8
c=A	0.1	0.3	0.1	0.2	0.2	0.4	0.3	0.1
c=C	0.5	0.2	0.1	0.1	0.6	0.1	0.2	0.7
c=G	0.2	0.2	0.6	0.5	0.1	0.2	0.2	0.1
c=T	0.2	0.3	0.2	0.2	0.1	0.3	0.3	0.1

Starting positions: Z_{ij} **Motif: M(k,c)**

- Z_{ij}: Probability that on sequence i, motif start at position j
- M(k,c): Probability that k^{th} character of motif is letter c

- **Computing Z_{ij} matrix from M(k,c) is straightforward**
  - **At each position, evaluate start probability by multiplying across the matrix**

- **Three variations for re-computing motif M(k,c) from Z_{ij} matrix**
  - **Expectation maximization** ➔ **All starts weighted by Z_{ij} prob distribution**
  - **Gibbs sampling** ➔ **Single start for each seq X_i by sampling Z_{ij}**
  - **Greedy approach** ➔ **Best start for each seq X_i by maximum Z_{ij}**

E-step:
Estimate Z_{ij} positions from matrix

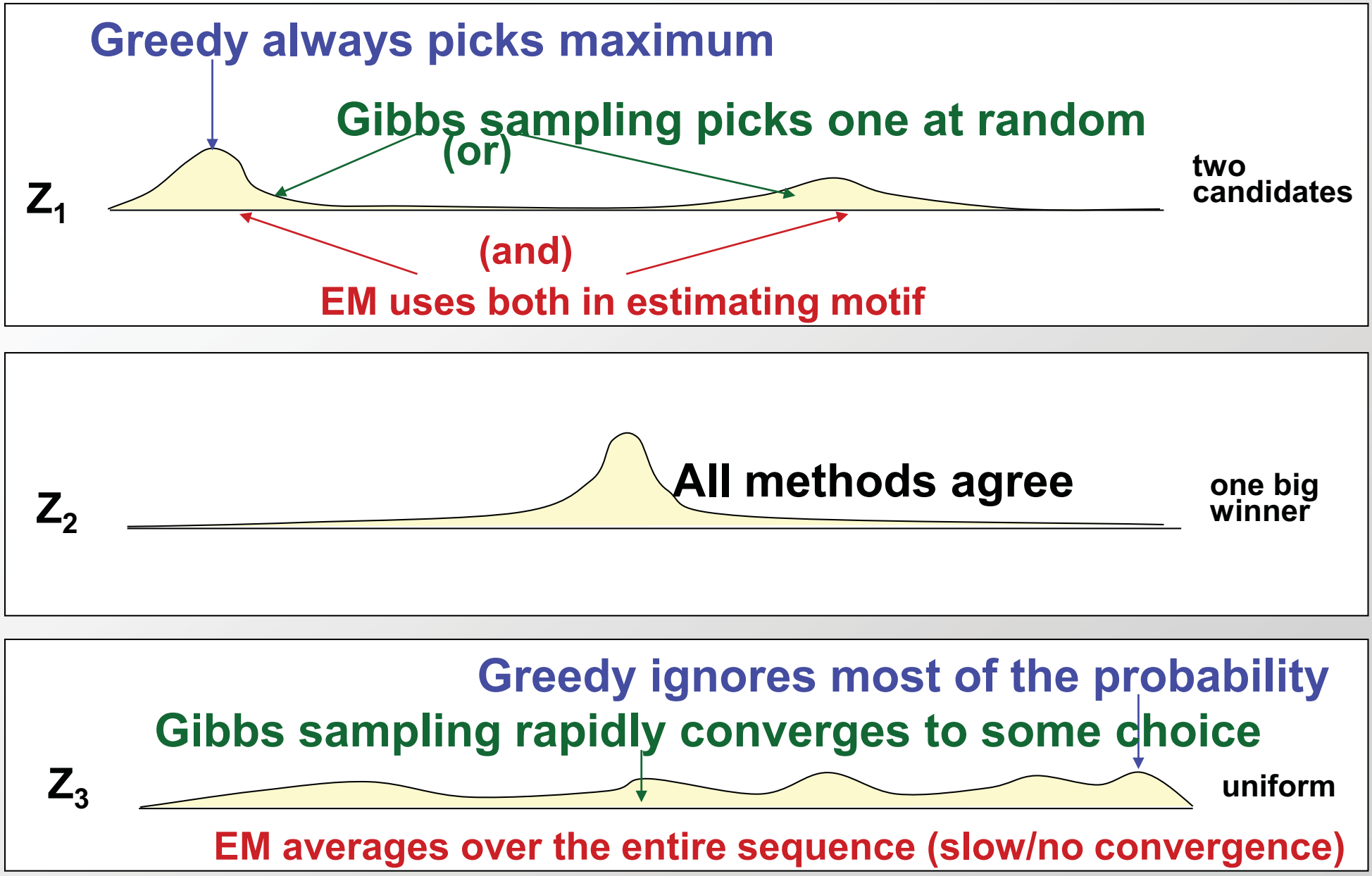
E-step

	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8
c=A	0.1	0.3	0.1	0.2	0.2	0.4	0.3	0.1
c=C	0.5	0.2	0.1	0.1	0.6	0.1	0.2	0.7
c=G	0.2	0.2	0.6	0.5	0.1	0.2	0.2	0.1
c=T	0.2	0.3	0.2	0.2	0.1	0.3	0.3	0.1

Starting positions: Z_{ij}

Motif: M(k,c)

Three examples for Greedy, Gibbs Sampling, EM



Calculating $P(X_i)$ when motif position is known

- Probability of training sequence X_i, given hypothesized start position j

$$\Pr(X_i \mid Z_{ij} = 1, M, B) = \underbrace{\prod_{k=1}^{j-1} B(X_{i,k})}_{\text{before motif}} \underbrace{\prod_{k=j}^{j+W-1} M(k-j+1, X_{i,k})}_{\text{motif}} \underbrace{\prod_{k=j+W}^{L} B(X_{i,k})}_{\text{after motif}}$$

- **Example:**

$X_i =$ G C **T G T** A G

$B =$

A	0.25
C	0.25
G	0.25
T	0.25

$M =$

	1	2	3
A	0.1	0.5	0.2
C	0.4	0.2	0.1
G	0.3	**0.1**	0.6
T	**0.2**	0.2	**0.1**

$$\Pr(X_i \mid Z_{i3} = 1, M, B) =$$

$$B(G) \times B(C) \times M(1,T) \times M(2,G) \times M(3,T) \times B(A) \times B(G) =$$

$$0.25 \times 0.25 \times \boxed{0.2 \times 0.1 \times 0.1} \times 0.25 \times 0.25$$

Calculating the Z vector (using M)

- **To estimate the starting positions in *Z* at step *t***

$$Z_{ij}^{(t)} = \Pr(Z_{ij}=1 \mid X_i, M^{(t)}) = \frac{\Pr(X_i \mid Z_{ij}=1, M^{(t)})\Pr(Z_{ij}=1)}{\Pr(X_i)}$$

posterior; likelihood; prior; evidence

(Bayes' rule)

- At iteration t, calculate $Z_{ij}^{(t)}$ based on $M^{(t)}$
  - We just saw how to calculate $\Pr(X_i \mid Z_{ij}=1, M^{(t)})$
  - To obtain total probability $\Pr(X_i)$, sum over all starting positions

$$Z_{ij}^{(t)} = \frac{\Pr(X_i \mid Z_{ij}=1, M^{(t)})\Pr(Z_{ij}=1)}{\sum_{k=1}^{L-W+1} \Pr(X_i \mid Z_{ik}=1, M^{(t)})\Pr(Z_{ik}=1)}$$

- **Assume uniform priors (motif eq likely to start at any position)**

Calculating the Z vector: Example

$X_i =$ G C T G T A G

$p =$

	0	1	2	3
A	0.25	0.1	0.5	0.2
C	0.25	0.4	0.2	0.1
G	0.25	0.3	0.1	0.6
T	0.25	0.2	0.2	0.1

$$Z_{i1} = 0.3 \times 0.2 \times 0.1 \times 0.25 \times 0.25 \times 0.25 \times 0.25$$

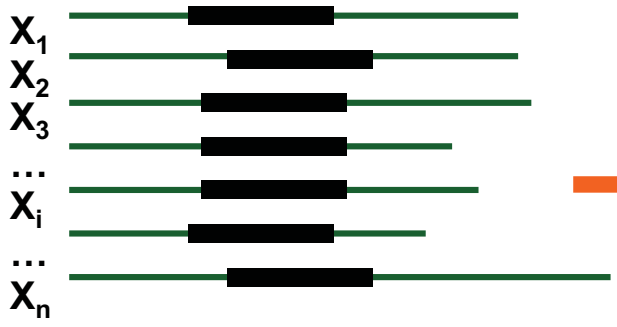
$$Z_{i2} = 0.25 \times 0.4 \times 0.2 \times 0.6 \times 0.25 \times 0.25 \times 0.25$$

$\vdots$

- then normalize so that

$$\sum_{j=1}^{L-W+1} Z_{ij} = 1$$

M-step:
Max-likelih motif from Z_{ij} positions



	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8
c=A	0.1	0.3	0.1	0.2	0.2	0.4	0.3	0.1
c=C	0.5	0.2	0.1	0.1	0.6	0.1	0.2	0.7
c=G	0.2	0.2	0.6	0.5	0.1	0.2	0.2	0.1
c=T	0.2	0.3	0.2	0.2	0.1	0.3	0.3	0.1

Starting positions: Z_{ij}

Motif: M(k,c)

M-step example: Estimating M(k,c) from Z_{ij}

X_1 = A **C A G** C A

Z_1 = 0.1 0.7 0.1 0.1

X_2 = A G G **C A G**

Z_2 = 0.4 0.1 0.1 0.4

X_3 = T **C A G** T C

Z_3 = 0.2 0.6 0.1 0.1

$$M(1,A) = \frac{Z_{1,1} + Z_{1,3} + Z_{2,1} + Z_{3,3} + 1}{Z_{1,1} + Z_{1,2} \ldots + Z_{3,3} + Z_{3,4} + 4}$$

Em approach: Avg'em all
Gibbs sampling: Sample one
Greedy: Select max

- EM: sum over full probability
  - $n_{1,A}$= 0.1+0.1+0.4+0.1 = 0.7
  - $n_{1,C}$= 0.7+0.4+0.6 = 1.7
  - $n_{1,G}$= 0.1+0.1+0.1+0.1= 0.4
  - $n_{1,T}$= 0.2 = 0.2
  - Total: T=0.7+1.7+0.4+0.2 = 3.0
- Normalize and add pseudo-counts
  - M(1,A) = (0.7+1)/(T+4) = 1.7/7=0.24
  - M(1,C) = (1.7+1)/(T+4) = 2.7/7=0.39
  - M(1,G) = (0.4+1)/(T+4) = 1.4/7=0.2
  - M(1,T) = (0.2+1)/(T+4) = 1.2/7=0.17
- M(k,c) =

	1	2	3
A	0.24	0.39	0.21
C	0.39	0.21	0.18
G	0.2	0.24	0.44
T	0.17	0.16	0.16

The EM Algorithm

- EM converges to a local maximum in the likelihood of the data given the model:

$$\prod_i \Pr(X_i \mid M, B)$$

- **Deterministic iterations max direction of ascent**
- **Usually converges in a small number of iterations**
- **Sensitive to initial starting point (i.e. values in *M*)**

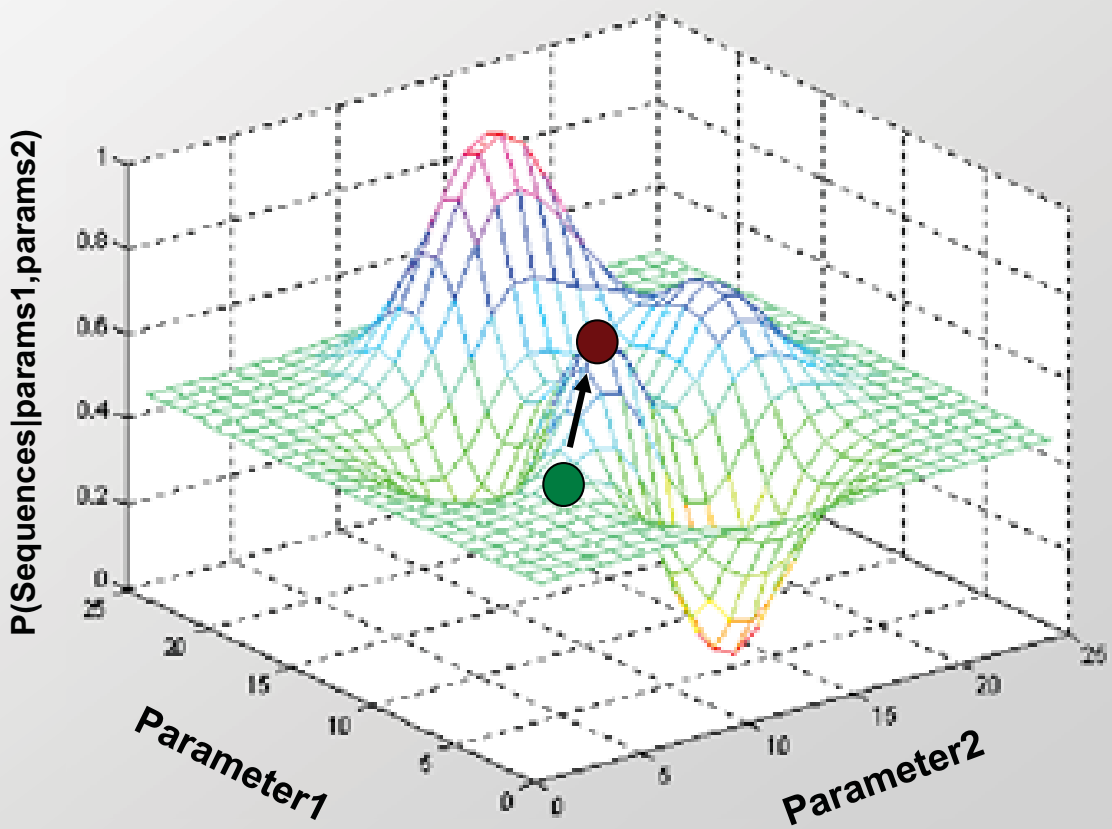
P(Seq|Model) Landscape

EM searches for parameters to increase P(seqs|parameters)

Useful to think of P(seqs|parameters) as a function of parameters

EM starts at an initial set of parameters

And then "climbs uphill" until it reaches a local maximum



Where EM starts can make a big difference