

Introduction: mathematics, statistics and computer science for biologists

Uri Keich

School of Mathematics and Statistics
University of Sydney

Its hopeless...

- I cannot possibly cram 3 well developed disciplines within 85 minutes (of those 5 minutes are for questions at the *end*)
- Instead, I chose to highlight how a few techniques from these fields help us solve specific problems in bioinformatics
- Moreover, I'll try to avoid using formulas
- No, the Winter School does *not* have a satisfaction guarantee policy

Sequence Alignment

- Why align?
 - Durbin et al. (1998):
 - ▶ “Nature is a tinkerer not an inventor” (Jacob 1977)
 - ▶ “New sequences are adapted from pre-existing sequences rather than invented de novo”
- We only discuss pairwise alignments
- Global alignment: the two sequences are aligned in their entirety
- **Example.** $x = \text{HEAGAWGHEE}$, $y = \text{PAWHEAE}$ and

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{array}{l} \text{HEAGAWGHE-E} \\ \text{-PA--W-HEAE} \end{array}$$

Key issues

- How do we assess the quality, or score, a given alignment?
- How do we find an optimal alignment?
- Is the optimal alignment statistically significant?
- The answers to the last two questions depend on whether:
 - we align a sequence against another one
 - we look for an optimal alignment of an input sequence against a database of sequences (BLAST)

Scoring an Alignment

- Penalize insertions/deletions
 - We consider linear gap penalty: $-d$ per gap position
- Reward conservation and penalize substitutions
 - Handled by the substitution matrix (e.g. BLOSUM62)
 - More on its construction later
- Linear score: sum over all aligned positions using the chosen gap penalty and substitution matrix

Global alignment problem

- Given two sequences $\mathbf{x}_{1:m}$ and $\mathbf{y}_{1:n}$ find an optimal global alignment
- Easier problem: find $F(m, n)$, the score of an optimal global alignment

Global alignment problem

- Given two sequences $\mathbf{x}_{1:m}$ and $\mathbf{y}_{1:n}$ find an optimal global alignment
- Easier problem: find $F(m, n)$, the score of an optimal global alignment
- Naive approach: score all possible alignments
 - infeasible for all but very short sequences

Global alignment problem

- Given two sequences $x_{1:m}$ and $y_{1:n}$ find an optimal global alignment
- Easier problem: find $F(m, n)$, the score of an optimal global alignment
- Naive approach: score all possible alignments
 - infeasible for all but very short sequences
- Instead, use *dynamic programming* (DP, Bellman 1940s):
 - Find a recursive formula showing the problem can be trivially solved by combining solutions of the same kind of problem but with smaller sized inputs
 - Compute each smaller problem only once

Needleman-Wunsch Algorithm

- Consider an optimal alignment of $x_{1:m}$ and $y_{1:n}$, it ends with either

$$\begin{bmatrix} x_m \\ y_n \end{bmatrix}, \begin{bmatrix} x_m \\ - \end{bmatrix}, \text{ or } \begin{bmatrix} - \\ y_n \end{bmatrix}$$

Needleman-Wunsch Algorithm

- Consider an optimal alignment of $\mathbf{x}_{1:m}$ and $\mathbf{y}_{1:n}$, it ends with either

$$\begin{bmatrix} x_m \\ y_n \end{bmatrix}, \begin{bmatrix} x_m \\ - \end{bmatrix}, \text{ or } \begin{bmatrix} - \\ y_n \end{bmatrix}$$

- Assume it ends with $\begin{bmatrix} x_m \\ y_n \end{bmatrix}$
 - The score is linear and the alignment is optimal so the rest of the alignment is optimal for $\mathbf{x}_{1:m-1}$ and $\mathbf{y}_{1:n-1}$
 - Therefore, in this case $F(m, n) = F(m - 1, n - 1) + S(x_m, y_n)$

Needleman-Wunsch Algorithm

- Consider an optimal alignment of $\mathbf{x}_{1:m}$ and $\mathbf{y}_{1:n}$, it ends with either

$$\begin{bmatrix} x_m \\ y_n \end{bmatrix}, \begin{bmatrix} x_m \\ - \end{bmatrix}, \text{ or } \begin{bmatrix} - \\ y_n \end{bmatrix}$$

- Assume it ends with $\begin{bmatrix} x_m \\ y_n \end{bmatrix}$
 - The score is linear and the alignment is optimal so the rest of the alignment is optimal for $\mathbf{x}_{1:m-1}$ and $\mathbf{y}_{1:n-1}$
 - Therefore, in this case $F(m, n) = F(m - 1, n - 1) + S(x_m, y_n)$
- Similarly, if the optimal alignment ends with $\begin{bmatrix} x_m \\ - \end{bmatrix}$,
 - $F(m, n) = F(m - 1, n) - d$ where d is the gap penalty

Needleman-Wunsch Algorithm (cont.)

- This yields a recursive formula for $F(m, n)$:

$$F(m, n) = \max \{ F(m-1, n-1) + S(x_m, y_n), F(m-1, n) - d, F(m, n-1) - d \}$$

Needleman-Wunsch Algorithm (cont.)

- This yields a recursive formula for $F(m, n)$:

$$F(m, n) = \max \{ F(m-1, n-1) + S(x_m, y_n), F(m-1, n) - d, F(m, n-1) - d \}$$

- We can still mess things up: if we implement it as a standard recursion (no memory)
- However, if
 - our recursion function has memory
 - or, we compute $F(m, n)$ by filling a table row by row (or column by column)

then we have an efficient DP algorithm

Example (Durbin et al. 1998)

	H	E	A	G	A	W	G	H	E	E	
P	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
A	-8	-2	-9	-17	-25	-33	-42	-49	-57	-65	-73
W	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
H	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
E	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
A	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
E	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

The alignment:

```
HEAGAWGHE-E
--P-AW-HEAE
```

The substitution matrix:

	H	E	A	G	A	W
P	-2	-1	-1	-2	-1	-4
A	-2	-1	5	0	5	-3
W	-3	-3	-3	-3	-3	15
H	10	0	-2	-2	-2	-3
E	0	6	-1	-3	-1	-3

Complexity

- The complexity of an algorithm describes how the required resources of time and memory (space) grow with the size of the input
- Although constants can make a difference in the actual implementation, we do not care about them when describing complexity
- The complexity of the Needleman-Wunsch Algorithm is:
 - Runtime $O(mn)$: a constant number of operations for computing each of the $m \times n$ table entries
 - Space $O(mn)$: need to keep the pointers table till the very end so we can trace back to find the optimal alignment

How do we construct a substitution matrix?

- Consider global *gapless* alignments $\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix}$
- Our scoring function should help us distinguish random alignments from functional (“real”) ones
- We have two simplistic models for generating alignments
 - Both are *iid* models for generating aligned pairs $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$

How do we construct a substitution matrix?

- Consider global *gapless* alignments $\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix}$
- Our scoring function should help us distinguish random alignments from functional (“real”) ones
- We have two simplistic models for generating alignments
 - Both are *iid* models for generating aligned pairs $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$
 - Under the null (random) model (H_0): both x_i and y_i are independently drawn from the marginal distribution q_a
 - Under the alternative model (H_1): $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ is drawn from a joint distribution of aligned homologous pairs p_{ab}
- Was the observed alignment $\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix}$ generated under H_0 or H_1 ?

The Neyman-Pearson Lemma

- The optimal test statistic for two simple hypotheses is the ratio of the likelihoods
- In the context of our alignment this translates to:

$$\frac{P \left(\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix} \mid H_1 \right)}{P \left(\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix} \mid H_0 \right)} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

The Neyman-Pearson Lemma

- The optimal test statistic for two simple hypotheses is the ratio of the likelihoods
- In the context of our alignment this translates to:

$$\frac{P \left(\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix} \mid H_1 \right)}{P \left(\begin{bmatrix} x_{1:n} \\ y_{1:n} \end{bmatrix} \mid H_0 \right)} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

- Since \log is a monotone function, the LLR $\sum_i s(x_i, y_i)$ where $s(a, b) = \log \frac{p_{ab}}{q_a q_b}$ is just as optimal as the LR statistic
- This yields a recipe for defining a substitution matrix
 - The matrices mostly differ in how they estimate the probabilities of aligned homologous pairs p_{ab}

Aligning a query sequence against a large database

- A *local alignment* of sequences x and y is a GA of contiguous subsequences of x and of y

- Example.

AWGHE
AW-HE

 where $x = \text{HEAGAWGHEE}$, $y = \text{PAWHEAE}$

Aligning a query sequence against a large database

- A *local alignment* of sequences x and y is a GA of contiguous subsequences of x and of y
 - Example.

AWGHE
AW-HE

 where $x = \text{HEAGAWGHEE}$, $y = \text{PAWHEAE}$
- The Smith-Waterman algorithm is a DP algorithm (very similar to NW) that finds an optimal local alignment in time $O(nm)$
- Biologists often search databases for similarities to a given query
 - Technically they look for optimal/good local alignments to the query sequence
- NCBI's RefSeq database has as of May 2011 over $157e+09$ nucleotides and over $4e+09$ amino acids
- Given that a guaranteed alignment costs $O(nm)$ it is impractical for frequent large db searches

BLAST (Altschul et al. 1990)

- The most popular database similarity search tool is BLAST
- BLAST is a heuristic tool which is much faster than SW but might (very rarely) miss the optimal alignment
- BLAST achieves its speed up by applying a full DP approach only to small selected regions that “look promising”

BLAST (Altschul et al. 1990)

- The most popular database similarity search tool is BLAST
- BLAST is a heuristic tool which is much faster than SW but might (very rarely) miss the optimal alignment
- BLAST achieves its speed up by applying a full DP approach only to small selected regions that “look promising”
- The first step in finding these is to find seed matches
 - blastn: a database string of 11bp that exactly matches a query string of the same length
- Such matches can be found efficiently using an automaton or hashing

Hashing

- Hashing is a technique for efficient storage and retrieval of data indexed by “unwieldy” keys
 - For example, the key can be an 11bp string and the data can be its location(s) in the db and/or the query
- The hash function maps the, typically very large, space of possible keys to the set of indices of the hash table that actually stores the data
 - The function is often structured as

$$f(\text{key}) \bmod p$$

where f translates the alphanumeric key into an integer and p is a prime (the size of the hash table)

Assessing the significance of an optimal local alignment

- Running SW on a pair of sequences we find that the optimal local alignment score is s : is it significant, or expected by chance?
- How do we assign significance to this score? Or to a blast search?

Assessing the significance of an optimal local alignment

- Running SW on a pair of sequences we find that the optimal local alignment score is s : is it significant, or expected by chance?
- How do we assign significance to this score? Or to a blast search?
- We first need to decide how random alignments look like
 - Null model: $X_{1:m}$ and $Y_{1:n}$ are *independently* generated according to an iid process
- The p -value of s is the (null) probability that the optimal local alignment score of $X_{1:m}$ and $Y_{1:n}$ is $\geq s$

Assessing the significance of an optimal local alignment

- Running SW on a pair of sequences we find that the optimal local alignment score is s : is it significant, or expected by chance?
- How do we assign significance to this score? Or to a blast search?
- We first need to decide how random alignments look like
 - Null model: $X_{1:m}$ and $Y_{1:n}$ are *independently* generated according to an iid process
- The p -value of s is the (null) probability that the optimal local alignment score of $X_{1:m}$ and $Y_{1:n}$ is $\geq s$
 - A minute p -value indicates that either we are very (un)lucky or that the null hypothesis is false
 - While a p -value greater than the traditional 0.05 cutoff indicates a possibly spurious alignment

Computing p -values – an example

- We observe a die landing on 6 in 300 out of 1000 rolls
- Assuming a fair die, what is the p -value of this observation?
 - The p -value is $P(X \geq 300)$ where X is a binomial $(1000, \frac{1}{6})$ random variable
- How can we compute this p -value?

Computing p -values – an example

- We observe a die landing on 6 in 300 out of 1000 rolls
- Assuming a fair die, what is the p -value of this observation?
 - The p -value is $P(X \geq 300)$ where X is a binomial $(1000, \frac{1}{6})$ random variable
- How can we compute this p -value?
 - Using the exact formula:
 - ▶ $\sum_{k=300}^{1000} \binom{1000}{k} \left(\frac{1}{6}\right)^k \left(\frac{5}{6}\right)^{(1000-k)}$
 - Through simulations (Monte Carlo):
 - ▶ repeatedly roll a die 1000 times and keep tally of the number of times it lands on 6
 - Using an asymptotic result:
 - ▶ the normal approximation to the binomial distribution

Computing the p -value of an optimal alignment score

- The exact test is prohibitively expensive even for ungapped alignments
- A Monte Carlo estimation can be derived by repeatedly:
 - drawing a random pair of independent sequences $X_{1:m}, Y_{1:n}$
 - applying Smith-Waterman to find the optimal score
- This MC approach is impractical for database searches

Computing the p -value of an optimal alignment score

- The exact test is prohibitively expensive even for ungapped alignments
- A Monte Carlo estimation can be derived by repeatedly:
 - drawing a random pair of independent sequences $X_{1:m}, Y_{1:n}$
 - applying Smith-Waterman to find the optimal score
- This MC approach is impractical for database searches
- Instead, BLAST estimates the p -value of the reported alignments based on asymptotic analysis augmented by several ad-hoc tricks
- Specifically, Dembo et al. (1994) showed that the limiting distribution of the score of optimal *ungapped* alignment is a Gumbel distribution (as $m, n \rightarrow \infty$)
- The adjustments to finite m, n as well as to *gapped* alignments have been worked by others (mostly Altschul)

More on Limit theorems

- Limit theorems deal with universal large scale limits
 - limits that do not depend on the particular initial distribution

More on Limit theorems

- Limit theorems deal with universal large scale limits
 - limits that do not depend on the particular initial distribution
- The two fundamental limit theorems that facilitate the analysis of a sum of iid random variables X_i are
 - The LLN (Law of Large Numbers): if the mean $\mu := E(X_i)$ exists then $\frac{1}{n} \sum_{i=1}^n X_i \longrightarrow \mu$
 - The CLT (Central Limit Theorem): if σ^2 , the variance of X_i , exists then the distribution of $\frac{(\sum_{i=1}^n X_i - \mu)}{\sqrt{n}\sigma}$ converges to the standard normal distribution
- The limits only depend on μ and σ^2

Extreme value distribution (EVD)

- A somewhat less sweeping limit theorem captures the limiting behavior of a maximum of iid random variables
- For example, let X_i be independent exponential random variables
- Let $M_n := \max_{i \leq n} X_i$
- It is not hard to show that for any $x \in \mathbb{R}$,

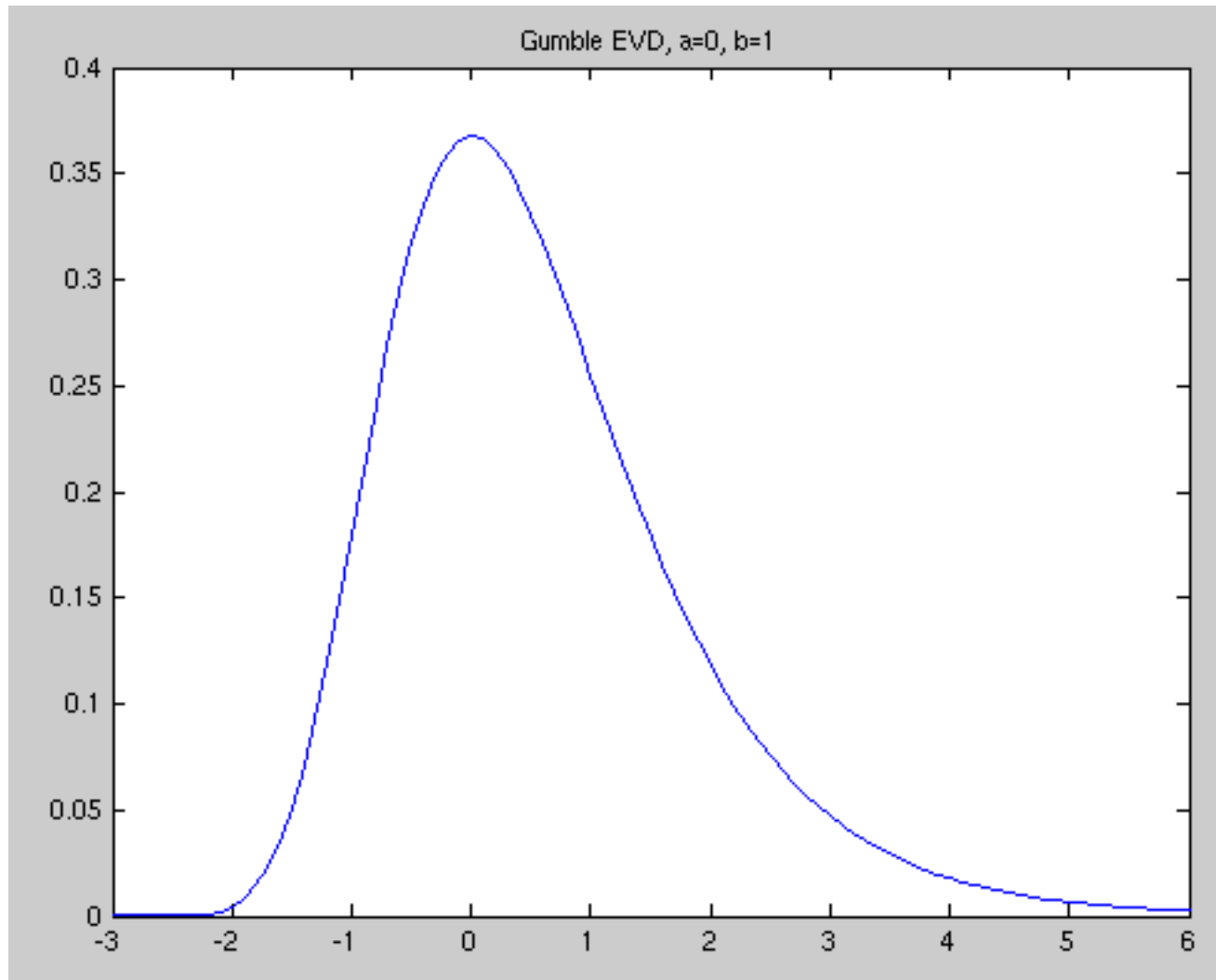
$$P(M_n - \log n \leq x) \longrightarrow e^{-e^{-x}}$$

- The RHS is an example of the *Gumbel* distribution function given by a CDF

$$F_G(x) = e^{-e^{-(x-a)/b}}$$

Gumbel distribution

- Below is the graph of the Gumbel distribution with $(a = 0, b = 1)$



BLAST's E -value

- In addition to the p -value of an observed score, s , BLAST prints its E -value:
 - The *expected* number alignments that score $\geq s$ assuming the database and query are independent of one another
- It is easy to compute this E -value given the p -value
- BLAST's E -value is the canonical benchmark for assessing the significance of a match against the query

Sequence motifs

- By a sequence motif we generally mean a collection of sites sharing similarities due to common functionality
 - For example, binding sites of a certain transcription factor

Sequence motifs

- By a sequence motif we generally mean a collection of sites sharing similarities due to common functionality
 - For example, binding sites of a certain transcription factor
- We often identify the motif with its mathematical model, the most common of which is the PWM or PFM (Position Weight/Frequency Matrix)

- For example:

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

- Under the PWM model each column describes a multinomial distribution over the residues at that position and the positions are assumed to be independently sampled

Estimation

- Deducing the entries of the PWM from an aligned list of sites is an estimation problem
- A common such technique is maximum likelihood estimation (MLE)
 - Find the likelihood function: the probability of the data given the value of the missing parameters
 - ▶ $L(\Theta; \mathbf{n}) = \prod_{i=1}^l \prod_{k=1}^4 \theta_{ik}^{n_{ik}}$, where n_{ik} is the observed number of occurrence of the letter k at position i and θ_{ik} denotes the missing parameter of the corresponding probability

Estimation

- Deducing the entries of the PWM from an aligned list of sites is an estimation problem
- A common such technique is maximum likelihood estimation (MLE)
 - Find the likelihood function: the probability of the data given the value of the missing parameters
 - ▶ $L(\Theta; \mathbf{n}) = \prod_{i=1}^l \prod_{k=1}^4 \theta_{ik}^{n_{ik}}$, where n_{ik} is the observed number of occurrence of the letter k at position i and θ_{ik} denotes the missing parameter of the corresponding probability
 - Find the value of the parameters that maximizes the likelihood
 - ▶ $\hat{\theta}_{ik} = n_{ik}/n$, where $n = \sum_{k=1}^4 n_{ik} =$ number of aligned sites

Estimation

- Deducing the entries of the PWM from an aligned list of sites is an estimation problem
- A common such technique is maximum likelihood estimation (MLE)
 - Find the likelihood function: the probability of the data given the value of the missing parameters
 - ▶ $L(\Theta; \mathbf{n}) = \prod_{i=1}^l \prod_{k=1}^4 \theta_{ik}^{n_{ik}}$, where n_{ik} is the observed number of occurrence of the letter k at position i and θ_{ik} denotes the missing parameter of the corresponding probability
 - Find the value of the parameters that maximizes the likelihood
 - ▶ $\hat{\theta}_{ik} = n_{ik}/n$, where $n = \sum_{k=1}^4 n_{ik} =$ number of aligned sites
- For our specific estimation problem we often add pseudo counts to avoid over-fitting so $\hat{\theta}_{ik} = (n_{ik} + 1)/(n + 4)$

Example

sites

TACGAT
 GATACT
 TATATT
 TATGAT
 TATAAT
 TATAAT

estimated PWM

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

Scanning for sites

- Goal: scan the input sequence x for putative binding sites modeled by M , a known $l \times 4$ PWM
- For each word w of length l in x , say $w = x_{j+1:j+l}$, we ask whether it was generated by the PWM or by our background model

Scanning for sites

- Goal: scan the input sequence x for putative binding sites modeled by M , a known $l \times 4$ PWM
- For each word w of length l in x , say $w = x_{j+1:j+l}$, we ask whether it was generated by the PWM or by our background model
- Recall that the Neyman-Pearson Lemma suggests deciding according to the likelihood ratio: $\frac{P_M(w)}{P_B(w)}$
 - where P_M, P_B are the probabilities of the word under the PWM and the background model

Scanning for sites

- Goal: scan the input sequence x for putative binding sites modeled by M , a known $l \times 4$ PWM
- For each word w of length l in x , say $w = x_{j+1:j+l}$, we ask whether it was generated by the PWM or by our background model
- Recall that the Neyman-Pearson Lemma suggests deciding according to the likelihood ratio: $\frac{P_M(w)}{P_B(w)}$
 - where P_M, P_B are the probabilities of the word under the PWM and the background model
- The background model is typically a Markov chain of order ≥ 3
- In a Markov chain of order m the distribution of the next state (letter) depends only on the last m states (letters)
 - It is specified in terms of a transition probability $p(x_i | x_{i-1}, x_{i-2}, x_{i-3})$ ($m = 3$)

De novo motif finding

- Simultaneously looking for a motif model and sites that will optimize a scoring function is significantly more difficult than scanning for TFBSs
- Below is an overly simplified example of an OOPS (one occurrence per sequence) motif finding problem
- tagcttcatcgttgacttctgcagaaagcaagctcctgagtagctggccaagcgagc
 tgcttgtgcccggctgcggcggttgtatcctgaatacgccatgcgccctgcagctgc
 tagaccctgcagccagctgcgcctgatgaaggcgcaacacgaaggaaagacgggacc
 agggcgacgtcctattaaagataatccccgaacttcatagtgtaatctgcagctg
 ctcccctacaggtgcaggcacttttcggatgctgcagcggccgtccgggggtcagttg
 cagcagtgttacgcgagggttctgcagtgctggctagctcgaccggattttgacgga
 ctgcagccgattgatggaccattctattcgtgacaccgacgagaggcgtcccccg
 gcaccaggccgttcctgcaggggcccaccctttgagttaggtgacatcattcctatgt
 acatgcctcaaagagatctagtctaaatactacctgcagaacttatggatctgaggg
 agaggggtactctgaaaagcgggaacctcgtgtttatctgcagtggtccaaatcctat

De novo motif finding

- Simultaneously looking for a motif model and sites that will optimize a scoring function is significantly more difficult than scanning for TFBSs
- Below is an overly simplified example of an OOPS (one occurrence per sequence) motif finding problem
- tagcttcatcgttgacttctgcagaaagcaagctcctgagtagctggccaagcgagc
 tgcttgtgccggctgcggcggttgtatcctgaatacgccatgcgccctgcagctgc
 tagaccctgcagccagctgcgctgatgaaggcgcaacacgaaggaaagacgggacc
 agggcgacgtcctattaaagataatccccgaacttcatagtgtaatctgcagctg
 ctcccctacaggtgcaggcacttttcggatgctgcagcggccgtccgggggtcagttg
 cagcagtgttacgcgagggtctgcagtgctggctagctcgaccggattttgacgga
 ctgcagccgattgatggaccattctattcgtgacaccgacgagaggcgtcccccg
 gcaccaggccggttcctgcaggggccaccctttgagttaggtgacatcattcctatgt
 acatgcctcaaagagatctagtctaaatactacctgcagaacttatggatctgaggg
 agaggggtactctgaaaagcgggaacctcgtgtttatctgcagtggtccaaatcctat

MEME (Bailey & Elkan '94)

- One of the most popular motif finders
- Recall that given the motif model we can easily scan the sequences for sites
- Conversely, given the sites deducing the PWM is trivial
- MEME makes use of this by starting from a heuristically chosen initial PWM and then alternating between the two tasks

MEME (Bailey & Elkan '94)

- One of the most popular motif finders
- Recall that given the motif model we can easily scan the sequences for sites
- Conversely, given the sites deducing the PWM is trivial
- MEME makes use of this by starting from a heuristically chosen initial PWM and then alternating between the two tasks
- Specifically, it iterates the following two steps until convergence
 - assign each input word w a weight according to how well it fits the current PWM: the LR $L_w = \frac{P_M(w)}{P_B(w)}$
 - update the PWM by taking a weighted average of all the words

MEME (Bailey & Elkan '94)

- One of the most popular motif finders
- Recall that given the motif model we can easily scan the sequences for sites
- Conversely, given the sites deducing the PWM is trivial
- MEME makes use of this by starting from a heuristically chosen initial PWM and then alternating between the two tasks
- Specifically, it iterates the following two steps until convergence
 - assign each input word w a weight according to how well it fits the current PWM: the LR $L_w = \frac{P_M(w)}{P_B(w)}$
 - update the PWM by taking a weighted average of all the words
 - This is the case for the OOPS (one occurrence per sequence) model, other models such as the ZOOPS (zero or one occurrence) are slightly more involved

Expectation Maximization or EM (Dempster et al. 1977)

- As suggested by its name, MEME implements EM
- EM is a deterministic algorithm for maximizing the likelihood of the missing parameters Θ given the observed data x
 - Θ is the PWM and x is the input sequences

Expectation Maximization or EM (Dempster et al. 1977)

- As suggested by its name, MEME implements EM
- EM is a deterministic algorithm for maximizing the likelihood of the missing parameters Θ given the observed data x
 - Θ is the PWM and x is the input sequences
- EM is applicable in cases where this maximization is straightforward *had* we known some missing data y
 - y specifies the locations of the sites

Expectation Maximization or EM (Dempster et al. 1977)

- As suggested by its name, MEME implements EM
- EM is a deterministic algorithm for maximizing the likelihood of the missing parameters Θ given the observed data x
 - Θ is the PWM and x is the input sequences
- EM is applicable in cases where this maximization is straightforward *had* we known some missing data y
 - y specifies the locations of the sites
- Solution: guess y then maximize Θ and use the new model to update our guess of y
- More precisely the updated guess is distributional: all possible y are considered, each weighted according to your belief in it, based on the current value of Θ

Gibbs Sampler (Lawrence et al. '93)

- A probabilistic algorithm that, like MEME, alternates between selecting sites and updating the PWM
- As in MEME, using the current PWM, M , every word w in the sample is given a weight $L_w = \frac{P_M(w)}{P_B(w)}$

Gibbs Sampler (Lawrence et al. '93)

- A probabilistic algorithm that, like MEME, alternates between selecting sites and updating the PWM
- As in MEME, using the current PWM, M , every word w in the sample is given a weight $L_w = \frac{P_M(w)}{P_B(w)}$
- In MEME we use a “soft assignment” of words to the list of selected sites
 - all words are considered, each weighted according to its LR score
- Here we randomly choose a site in a sequence, one sequence at a time, with probability proportional to its LR score

Gibbs Sampler (Lawrence et al. '93)

- A probabilistic algorithm that, like MEME, alternates between selecting sites and updating the PWM
- As in MEME, using the current PWM, M , every word w in the sample is given a weight $L_w = \frac{P_M(w)}{P_B(w)}$
- In MEME we use a “soft assignment” of words to the list of selected sites
 - all words are considered, each weighted according to its LR score
- Here we randomly choose a site in a sequence, one sequence at a time, with probability proportional to its LR score
- After updating the selected site in the current sequence and before moving to the next sequence we reestimate the PWM
- The last two steps are repeated through many iterations

- There is no convergence in the naive sense
 - The program terminates when the last K iterations gave no improvement to the maximal observed likelihood (plateau period)
- EM is prone to get stuck in a local maximum of the likelihood function
- Gibbs sampler is less prone to get stuck however it tends to be significantly slower

Positional priors in MEME and the Gibbs Sampler

- The probabilistic model we implicitly introduced for both programs assumes that each site in a sequence is a priori equally likely
- However, we might have information to the contrary, for example,
 - nucleosome occupancy data (Narlikar et al. 2007)
 - sequence conservation data from closely related species (Gordan et al. 2008)

Positional priors in MEME and the Gibbs Sampler

- The probabilistic model we implicitly introduced for both programs assumes that each site in a sequence is a priori equally likely
- However, we might have information to the contrary, for example,
 - nucleosome occupancy data (Narlikar et al. 2007)
 - sequence conservation data from closely related species (Gordan et al. 2008)
- It is easy to introduce positional priors into both algorithms
 - multiply the LR weight of each word by its prior