Pattern Discovery in Biosequences

SDM 2005 tutorial (Appendix)

Stefano Lonardi

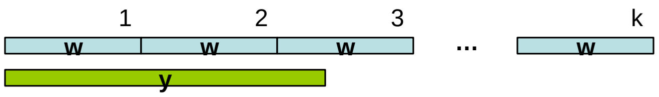
University of California, Riverside

Index

Periodicity of Strings

Periods of a string

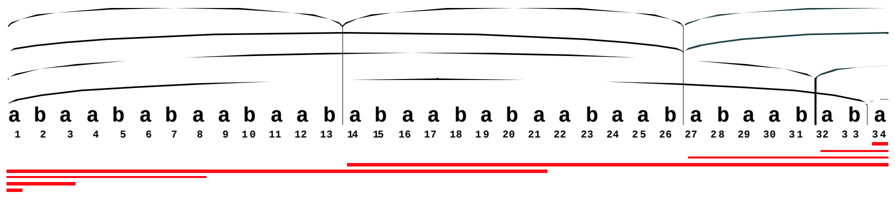
- Definition: a string *y* has *period w*, if *y* is a non-empty prefix of w^k for some integer k=1



- Definition: The period *y* of *y* is called *trivial* period
- Definition: the set of period lengths of *y* is

$$P(y)=\{\ |w|: w \text{ is a period of } y,\ w\neq y\}$$

Example



P(y) = {13,26,31,33}

Borders of a string

- Definition: a *border w* of *y* is any nonempty string which is both a prefix and a suffix of *y*
- Definition: the border *y* of *y* is called the *trivial* border
- Fact: a string *y* has a period of length $d<m$ iff it has a non-trivial border of length m-d

Finding Borders/Periods

- Borders can be found using the *failure function* of the string as done, e.g., in the preprocessing step of the classical linear time string search algorithms (Knuth, Morris, Pratt)

- Borders can be computed in $O(|y|)$, and so do periods

Profiles for sequence classification

Profiles as classifiers

- Profiles can be used directly to implement very simple classifiers

- Suppose we have a sample *S+* of known sites, and a sample *S-* of non-sites
- Given a new sequence *x*, how do we classify *x* in *S+* or in *S-?*

Example: CRP binding sites

- S+=

```
{TTGTGGC,  ACGTGAT,    CTGTGAC,
  TTTTGAT,  ATGTGAG,    ATGAGAC,
  AAGTGTC,  TTGTGAG,    TTGTGAG}
  ATTTGCA,  CTGTAAC,
  CTGTGCG,  CTGTAAC,
  ATGCAAA,  TTGTGAC,
  GTGTTAA,  GCCTGAC,
  ATTTGAA,  TTGTGAT,
  TTGTGAT,  TTGTGAT,
  ATTTATT,  GTGTGAA,
```

Cyclic AMP receptor protein TFs in E.coli [Stormo & Hartzell, 89]

Training (CRP sites)

- Assume a Bernoulli model for each position

A	0.350	0.043	0.000	0.043	0.130	0.830	0.260
C	0.170	0.087	0.043	0.043	0.000	0.043	0.300
T	0.130	0.000	0.780	0.000	0.830	0.043	0.170
G	0.350	0.870	0.170	0.910	0.043	0.087	0.260

- Assume the uniform Bernoulli model for the non-sites *S*-, that is $p_A=0.25$, $p_C=0.25$, $p_T=0.25$, $p_G=0.25$ for all the positions

Testing

- Suppose you get *x* = **`GGTGTAC`**
  Is *x* more likely to belong to *S*+ or to *S-?*
  In other words, it is more likely to be generated from the Bernoulli model for *S*+ or from the uniform Bernoulli model (for *S*-)?
- Let's compute the probability

$$P(x = GGTGTAC \mid S+) = .35{*}.87{*}.78{*}.91{*}.83{*}.83{*}.3 = 0.045$$
$$P(x = GGTGTAC \mid S-) = (.25)^7 = 0.0000061$$
$$LR(x) = P(x \mid S+) / P(x \mid S+)$$

Sequence alignment

Global alignment

- Clearly there are many other possible alignments
- Which one is *better?*
- We assign a *score* to each
  - *match (e.g., 2)*
  - *insertion/deletion (e.g., -1)*
  - *substitution (e.g., -2)*
- Both previous alignments scored
  *4*2+3*(-1)+1*(-2)=3   4*2+1*(-1)+2*(-2)=3*

Scoring function

- Given two symbols *a, b* in $\Sigma \cup \{-\}$ we define $\sigma(a,b)$ the score of aligning *a* and *b*, where *a* and *b* can <u>not</u> be both "**-**"

- In the previous example
  $\sigma(a,b)=+2$ *if* $a=b$
  $\sigma(a,b)=-2$ *if* $a\neq b$
  $\sigma(-,a)=-1$
  $\sigma(a,-)=-1$

Global alignment

- <u>Definition</u>: Given two strings *w* and *y*, an *alignment* is a mapping of *w,y* into strings *w',y'* that may contain spaces, where $|w'|=|y'|$ and the removal of the spaces from *w'* and *y'* returns *w* and *y*
- <u>Definition</u>: The *value* of an alignment *(w',y')* is

$$\sum_{i=1}^{|w'|} s\left(w'_{[i]}, y'_{[i]}\right)$$

Global alignment

- Definition: A *optimal global alignment* of *w* and *y* is one that achieves maximum score
- How to find it?
- How about checking all possible alignments?

Checking all alignments

```
$|w| = |y| = m$
for all $i$, $0 \le i \le m$ do
  for all subsequences $A$ of $w$ with $|A| = i$ do
    for all subsequences $B$ of $y$ with $|B| = i$ do
      form an alignment that matches $A_{[j]}$ with $B_{[j]}$
      $\forall\ 1 \le j \le i$, and matches all others with spaces
```

Example

- Given w=**ATCTG**
  y=**CATGA** *(m=5)*
- Suppose *i=2*
- Suppose we choose A=**CG** B=**CT**
- We are considering the score of the following alignment (length is *2m-i=8*)

```
ATCT-G--
--C-ATGA
```

Time complexity

A string of length m has $\binom{m}{i}$ subsequences of length i.

Thus, there are $\binom{m}{i}^2$ pairs of subsequences, each of length i. The length of each alignment is $2m$-1.

The total number of operations is at least

$$\sum_{i=0}^{m}\binom{m}{i}^2(2m-1) \geq m\sum_{i=0}^{m}\binom{m}{i}^2 = m\binom{2m}{i} > 2^{2m}$$

Checking all alignments

- The previous algorithms runs in $O(2^{2m})$ time
- How bad is it?
- Choose $m=1,000$ and try to wait your computer to run $2^{2,000}$ operations!

Needleman & Wunsch, 70

- The first algorithm to solve efficiently the alignment of two sequences
- Based on *dynamic programming*
- Runs in $O(m^2)$ time
- Uses $O(m^2)$ space

Alignment by dyn. programming

- Let *w* and *y* be two strings, $|w|=n$, $|y|=m$
- Define $V(i,j)$ as the value of the alignment of the strings $w_{[1..i]}$ with $y_{[1..j]}$
- The idea is to compute $V(i,j)$ for all values of $0 \leqslant i \leqslant n$ and $0 \leqslant j \leqslant m$
- In order to do that, we establish a recurrence relation between $V(i,j)$ and $V(i-1,j)$, $V(i,j-1)$, $V(i-1,j-1)$

Alignment by dyn. programming

$$V(i,j) = \max \begin{Bmatrix} V(i-1, j-1) + s\,(w_{[i]}, y_{[j]}) \\ V(i-1, j) + s\,(w_{[i]}, "-") \\ V(i, j-1) + s\,("-", y_{[j]}) \end{Bmatrix}$$

$$V(0,0) = 0$$

$$V(i,0) = V(i-1,0) + s\,(w_{[i]}, "-")$$

$$V(0,j) = V(0,j-1) + s\,("-", y_{[j]})$$

Example

$s(a,a) = +1$	[match]
$s(a,b) = -1, \text{ if } a \neq b$	[substitution]
$s(a,\text{"}-\text{"}) = -2$	[deletion]
$s(\text{"}-\text{"},a) = -2$	[insertion]

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + s(w_{[i]}, y_{[j]}) \\ V(i-1, j) + s(w_{[i]}, \text{"}-\text{"}) \\ V(i, j-1) + s(\text{"}-\text{"}, y_{[j]}) \end{cases}$$

$$V(0,0) = 0$$

$$V(i,0) = V(i-1,0) + s(w_{[i]}, \text{"}-\text{"})$$

$$V(0,j) = V(0, j-1) + s(\text{"}-\text{"}, y_{[j]})$$

		A	G	C
	0	-2	-4	-6
A	-2	1	-1	-3
A	-4	-1	0	-2
A	-6	-3	-2	-1
C	-8	-5	-4	-1

Example

AG-C
AAAC

		A	G	C
	0	-2	-4	-6
A	-2	1	-1	-3
A	-4	-1	0	-2
A	-6	-3	-2	-1
C	-8	-5	-4	-1

Example

A-GC
AAAC

		A	G	C
	0	-2	-4	-6
A	-2	1	-1	-3
A	-4	-1	0	-2
A	-6	-3	-2	-1
C	-8	-5	-4	-1

Example

-AGC
AAAC

		A	G	C
	0	-2	-4	-6
A	-2	1	-1	-3
A	-4	-1	0	-2
A	-6	-3	-2	-1
C	-8	-5	-4	-1

Variations

- Local alignment [Smith, Waterman 81]
- Multiple sequence alignment (local or global)
- Theorem [Wang, Jiang 94]: the optimal sum-of-pairs alignment problem is *NP*-complete

Expectation Maximization

Expectation maximization

The goal of EM is to find the model that maximizes the (log) likelihood

$$L(q) = \log P(x \mid q) = \log \sum_y P(x, y \mid q).$$

Suppose our current estimated of the parameters is q^t.
We want to know what happens to L when we move to q.

$$L(q) - L(q^t) = \log \frac{\sum_y P(x, y \mid q)}{\sum_y P(x, y \mid q^t)} = \log \frac{\sum_y P(x \mid y, q) P(y \mid q)}{\sum_y P(x \mid y, q^t) P(y \mid q^t)}$$

Expectation maximization

After some (complex) algebraic manipulations one finally gets

$$L(q) - L(q^t) = Q(q \mid q^t) - Q(q^t \mid q^t) + \sum_y P(y \mid x, q^t) \log \frac{P(y \mid x, q^t)}{P(y \mid x, q)}$$

where $Q(q \mid q^t) \equiv \sum_y P(y \mid x, q^t) \log P(x, y \mid q)$.

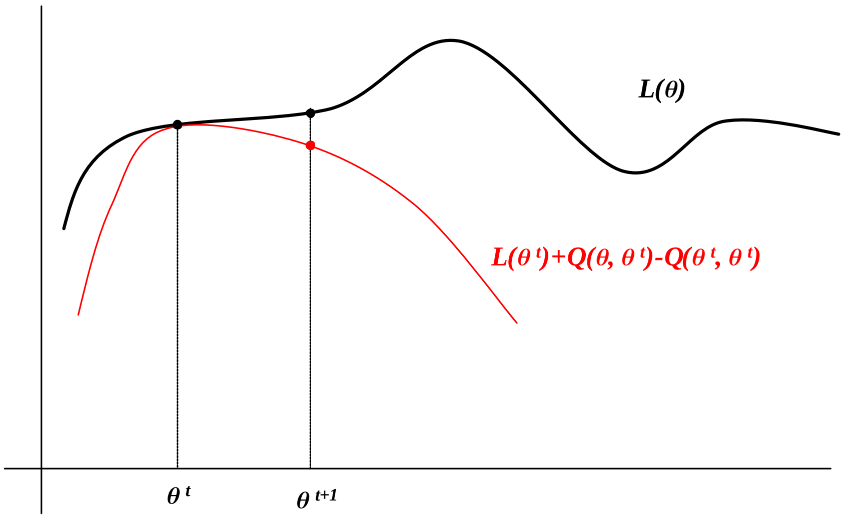
Convergence

The last term is $H\left(P(y|x,q^t)||P(y|x,q)\right)$ which is always non-negative, and therefore

$$L(q)-L(q^t)\geq Q(q\,|q^t)-Q(q^t\,|q^t)$$

with equality iff $P(y|x,q^t)=P(y|x,q^{t+1})$.
Choosing $q^{t+1}=\arg\max_q Q(q\,|q^t)$ will always make the difference positive and thus the likelihood of the new model q^{t+1} larger than the likelihood of q^t.

A step of EM



Expectation maximization

EM iterates 1), 2) until convergence

1) <u>E-Step</u>: compute the $Q(\theta \mid \theta^{t})$ function with respect to the current parameters θ^{t}

2) <u>M-Step</u>: choose $\theta^{t+1} = \mathrm{argmax}_{\theta}\, Q(\theta \mid \theta^{t})$

Expectation maximization

- The likelihood increases at each step, so the procedure will always reach a maximum asymptotically
- It has been proved that the number of iterations to convergence is *linear* in the input size
- Each step, however, require quadratic time in the size of the input

Expectation maximization

- More importantly, EM can get stuck (easily) in local maxima

- Standard techniques in combinatorial optimization can be used to alleviate this problem

EM for pattern discovery

- The first attempt to use EM for pattern discovery has been proposed by Lawrence and Reilly [Proteins, 1990]

- <u>Input</u>: multisequence $\{x_1,x_2,\ldots,x_k\}$
  pattern length m

- <u>Output</u>: a matrix profile $q_{i,b}$, b Î Σ, $1 \leqslant i \leqslant m$, and positions s_j, $1 \leqslant j \leqslant k$, of the profile

EM for pattern discovery

- Assumption: there is exactly one occurrence of the profile in each sequence

- The missing information in this case are the positions s_j of the motif in $\{x_1,x_2,\ldots,x_k\}$ (in fact, *if* we knew the positions, the problem of finding the profile would be trivial)

Lawrence-Reilly EM

The objective is to maximize the following log likelihood

$$L(q) = k\sum_{i=1}^{m}\sum_{b\in\Sigma} f^{i}(b)\log(q_{b,i})$$
$$+\ k(n-m)\sum_{b\in\Sigma} f^{0}(b)\log(q_{b,0})$$

where $q_{b,0}$ is the unknown distribution outside the site,

$q_{b,i}$ is the unknown distribution inside the site (profile),

$f^{0}(b)$ is the observed count of b outside the site,

$f^{i}(b)$ is the observed count of b in the site at position i

Lawrence-Reilly EM

The value of q that maximizes the log likelihood L is

$$q_{i,b} = f^i(b)/k$$

$$q_{0,b} = f^0(b)/(k(n-m))$$

which corresponds to idea of computing the profile by counting the symbols column-by-column

Lawrence-Reilly EM

- E-step: use the current parameters $q^{(t)}$ to compute

  $$P(\text{observing } x_i | \text{profile starts at position } s \text{ in } x_i)$$

  for all $1 \le i \le k,\ 1 \le s \le |x_i| - m + 1$, and then

  $$r_{i,s} = P(\text{profile starts at position } s \text{ in } x_i) \text{ using Bayes}$$

  for all $1 \le i \le k,\ 1 \le s \le |x_i| - m + 1$.

  Align the profile at each position (i, s) and for each column $1 \le j \le m$, accumulate in the $\hat{q}_{x_{i,[s+j-1]},j}$ the contributions of $r_{i,s+j-1}$. At the end, $\hat{q}$ contains the expected count of each symbol in each position of the profile.

Lawrence-Reilly EM

- M-step: use the expected count $\hat{q}$ of each symbol in each position to compute the ML (re)estimate of the parameters

$$q_{b,i}^{(t+1)} = \frac{\hat{q}_{b,i}}{k}, \quad b \in \Sigma,\ 1 \le i \le m$$

$$q_{b,0}^{(t+1)} = \frac{\hat{q}_{b,0}}{k(n-m)}, \quad b \in \Sigma$$

- Termination: when $\left\| q^{(t+1)} - q^{(t)} \right\| \le e$ or max iterations reached

Lawrence-Reilly EM

- Constrains in the structure of the profile can be easily incorporated (e.g., being palindrome)

- Variable length gaps within the profile can be handled by adding new variables to the model (that increase the complexity of the model, however)

Discovering Profiles

Discovering Profiles

- If one assumes the unknown profile to have been generated by a sequence of independent r.v.s then the observed frequency of letters in the columns of the profile are the ML estimates of the distributions of the r.v.s

- Unfortunately we do not know the positions of the profile in the multisequence

Gibbs sampler

Gibbs sampling

- Proposed by Lawrence, *et al.*, [Science, 1993]
- Web servers at
  http://bayesweb.wadsworth.org/gibbs/gibbs.html and
  http://argon.cshl.org/ioschikz/gibbsDNA/
- Input: multisequence $\{x_1,x_2,\ldots,x_k\}$
  pattern length m
- Output: a matrix profile $q_{i,b}$, b Î Σ, $1 \leqslant i \leqslant m$, and positions s_j, $1 \leqslant j \leqslant k$, of the profile in the k sequences

Gibbs sampling

- The algorithm maintains the background distribution $p_{\mathbf{A}},\ldots,p_{\mathbf{T}}$ of the symbols not described by the profiles

- *P(y)* is the probability of *y* based on the background distribution p_b, *b* Î Σ

- *Q(y)* is the probability of *y* based on the profile $q_{i,b}$, $1 \leqslant i \leqslant m$, *b* Î Σ

Gibbs sampling

- <u>Idea</u>: the profile is obtained by locating the positions which maximizes *Q(y)/P(y);* once the positions are obtained a new, more accurate, version of the profile can be obtained

- Initialize the initial positions s_j randomly

Gibbs sampling

Gibbs sampler iterates 1), 2) until convergence

1) Predictive update step: randomly choose one of the *k* sequences, say *r*. The matrix profile $q_{i,b}$ and the background frequencies p_b are recomputed from the current positions s_j in all sequences excluding *r*
2) Sampling step: assign a weight $z(y)=Q(y)/P(y)$ to each substring *y* of length *m*. Select randomly a substring *y* with probability $z(y)/\Sigma_y z(y)$, and then update s_j

Gibbs sampling

- The more accurate the pattern description in step 1), the more accurate the determination of its position in step 2), and vice versa
- Once some correct positions have been selected by chance, $q_{i,b}$ begins to reflect, albeit imperfectly, the unknown pattern
- This process tends to recruit further correct positions which in turn improve the discriminating power of the evolving pattern

Gibbs sampling

- How to update the matrix profile $q_{i,b}$ and the background frequencies p_b?
- We set $q_{i,b}=(f^i(b)+d_b)/(k-1+\Sigma_c d_c)$ where $f^i(b)$ is the number of times we observe symbol b in the position i of the profile (currently placed at position s_j), except for sequence r (d_b are pseudo-counts)
- We set the background probabilities $p_b=f(b)/\Sigma_c f(c)$ for all symbols in positions not covered by the profile

Phase shift problem

- Suppose that the "strongest" pattern begin, for example, at position *7, 19, 8, 23, …*
- If Gibbs happens to choose $s_1=9$, $s_2=21$ it will most likely choose $s_3=10$ and $s_4=25$
- The algorithm can get stuck in local maxima, which are the shifted form of the optimal pattern

Phase shift problem

- The problem can be alleviated by adding a step in which the current set of positions are compared with sets of shifted left and right positions, up to a certain number of symbols

- Probability ratios may be calculated for all positions, and a random selection is made with respect to the appropriate weight

Gibbs sampling

- It can be generalized to:

- Find also the length of pattern *m*

- Find a set of matrix profiles, instead of one

Gibbs sampling

- Since Gibbs sampler is an heuristic rather than a rigorous optimization procedure, one cannot guarantee the optimality of the result

- It is a good practice to run the algorithm several times from different random initial positions

Gibbs sampling vs. EM

- Although EM and Gibbs are built on common statistical foundation, the authors claim that Gibbs outperforms EM both in term of time complexity and performance

- *"EM is deterministic and tends to get trapped by local optima which are avoided by Gibbs … HMMs permit arbitrary gaps … have greater flexibility, but suffer the same penalties …"*

Expectation Maximization and MEME

Expectation maximization

- EM was designed by Dempster, Laird, Rubin [1977]
- EM is a family of algorithms for maximum likelihood estimation of parameters with “missing data”

EM, when?

- When we want to find the maximum likelihood estimate of the parameters of a model and
  - data is incomplete, or
  - the optimization of the maximum likelihood function is analytically intractable but the likelihood function can be simplified by assuming the existence of additional, *missing,* parameters value

Expectation maximization

- EM approaches the problem of missing information by iteratively solving a sequence of problems in which <u>expected information is substituted for missing information</u>

Expectation maximization

- All EM algorithms consists of two steps:

1) the expectation step (E-step)

2) the maximization step (M-step)

- The expectation step is with respect to the unknown underlying variables, using the current estimate of the parameters and conditioned upon the observation

Expectation maximization

- The maximization step provides a new estimate of the parameters

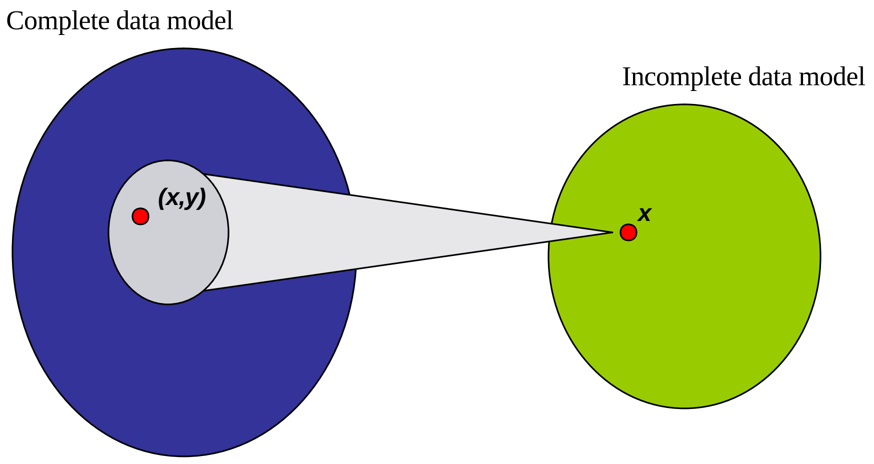
- $\theta^1 \blacktriangleright \theta^2 \blacktriangleright \theta^3 \blacktriangleright \ldots \blacktriangleright \theta^t \blacktriangleright \theta^{t+1} \blacktriangleright \ldots$

- The two steps are iterated until convergence

General framework for EM

- Suppose we want to find the parameters θ of a model (*training*)
- We observe x (*training set*)
- The probability of x under θ is also determined by the missing data y

Incomplete data model



An occurrence of (x,y) implies an occurrence of x, however only x can be observed. This observation reveals the subset $\{(x,y), \textit{for all } y\}$

Expectation maximization

- Example: For HMMs, *x* is the sequence we want to learn from, θ is the transition and emission probabilities, *y* is the path through the model
- Example: In the case of Random Projections, *x* are the subsequences corresponding to a cell with count higher than the threshold, θ are the parameters of a representation of the *(m,d)* pattern, *y* are all the missing positions

MEME

- Proposed by Bailey and Elkan [Machine Learning J., 1995]
- "Multiple EM for Motif Elicitation" (MEME) is an improved version of the expectation maximization approach by Lawrence and Reilly [Proteins, 1990] (see appendix)
- Designed to discover profiles (no gaps)
- Server at http://meme.sdsc.edu/meme/

MEME

- There are three main differences w.r.t. Lawrence *et al.:*

1) the initial profiles are not chosen randomly, but they are substrings which actually occur in the sequences
2) the assumption that there is only one occurrence of the motif is dropped
3) once a profile has been found, it is reported, and the iterative process continues

Using substring as starting points

- <u>Idea</u>: substrings actually occurring in sequence are better starting points than random choices
- Each substring is converted into a profile
- Assigning 1.0 to the occurring symbol and 0.0 to the others is a bad choice, because EM cannot move from this
- The authors arbitrarily assign probability 0.5 to the symbol and 0.5/3 for the other three

Using substring as starting points

- It would be too expensive to run EM until convergence from each substring
- It turns out that this is not necessary
- EM converges very quickly from profiles obtained from substrings, and the best starting point can be found running only <u>one</u> iteration

MEME algorithm

- Repeat
  - For each substring *y* in $\{x_1, x_2, \ldots, x_k\}$ do
    - Run <u>one</u> EM iteration with profile computed from *y*
    - Choose the profile *q* with highest likelihood
    - Run EM until convergence starting from *q*
    - Report the profile *q*
    - Erase the occurrences of *q* from dataset
- Until max number of iterations is reached

Dealing with multiple occurrences

- MEME allows to drop the “one-per-sequence” assumption
- The basic idea is to require the user to supply an estimated number of occurrences of the unknown profile and use that to normalize the estimation process of the EM algorithm
- The authors claim that the exact value of the number of occurrences is not critical

Finding multiple profiles

- MEME does not stop after finding the most likely profile
- Once a profile is found and reported, it is “probabilistically erased” by changing some position-dependent weight
- The process continues until a number of predetermined motifs have been found
- (see appendix for mega-prior heuristic)

Megaprior heuristics for MEME

Convex combination problem

- Bailey and Gribskov [ISMB, 1996] describe a problem common to all statistical methods (HMMs, Gibbs, MEME) which discover profiles in <u>protein sequences</u>

- These algorithms are prone to produce profiles that are incorrect because two or more distinct patterns can be incorrectly combined

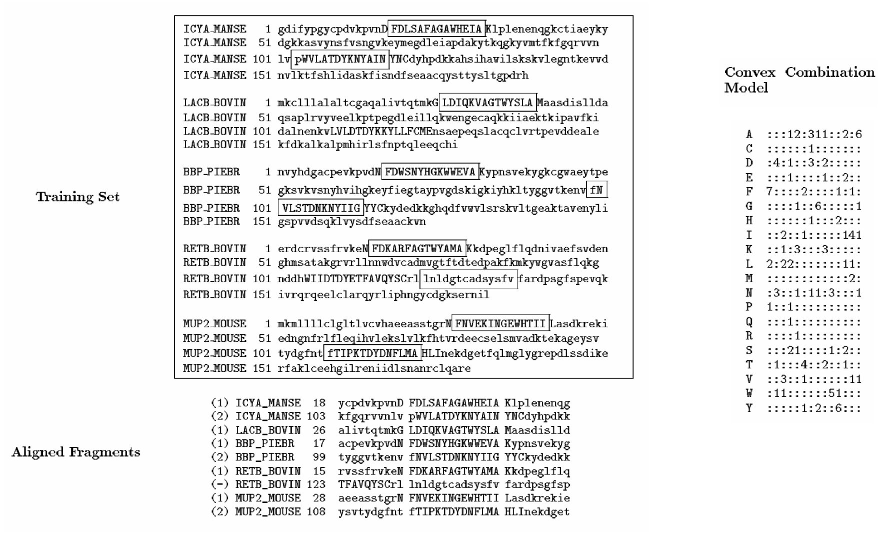
Convex combination problem

- MEME is likely to produce these profile if the estimated number of occurrences is inaccurate or missing
- MEME tends to select a profile that is a combination of two or more patterns because the convex combination can maximize the objective function by explaining more of the data using fewer free parameters

Convex combination problem

- The authors call this profile *convex combination*, because the parameters of the profile that erroneously combines distinct patterns are a weighted average of the parameters of the correct profiles, where the weights are positive and sum up to one – i.e., a convex combination

Example of convex combination



Training Set

```
ICYA_MANSE   1 gdifypgycpdvkpvnD FDLSAFAGAWHEIA Klplenenqgkctiaeyky
ICYA_MANSE  51 dgkkasvynsfvsngvkeymegdleiapdakytkqgkyvmtfkfgqrvvn
ICYA_MANSE 101 lv pWVLATDYKNYAIN YNCdyhpdkkahsihawilskskvlegntkevvd
ICYA_MANSE 151 nvlktfshlidaskfisndfseaacqysttysltgpdrh

LACB_BOVIN   1 mkclllalaltcgaqalivtqtmkG LDIQKVAGTWYSLA Maasdisllda
LACB_BOVIN  51 qsaplrvyveelkptpegdleillqkwengecaqkkiiaektkipavfki
LACB_BOVIN 101 dalnenkvLVLDTDYKKYLLFCMEnsaepeqslacqclvrtpevddeale
LACB_BOVIN 151 kfdkalkalpmhirlsfnptqleeqchi

BBP_PIEBR    1 nvyhdgacpevkpvdN FDWSNYHGKWWEVA Kypnsvekygkcgwaeytpe
BBP_PIEBR   51 gksvkvsnyhvihgkeyfiegtaypvgdskigkiyhkltyggvtkenv fN
BBP_PIEBR  101  VLSTDNKNYIIG YYCkydedkkghqdfvwvlsrskvltgeaktavenyli
BBP_PIEBR  151 gspvvdsqklvysdfseaackvn

RETB_BOVIN   1 erdcrvssfrvkeN FDKARFAGTWYAMA Kkdpeglflqdnivaefsvden
RETB_BOVIN  51 ghmsatakgrvrllnnwdvcadmvgtftdtedpakfkmkywgvasflqkg
RETB_BOVIN 101 nddhWIIDTDYETFAVQYSCrl lnldgtcadsysfv fardpsgfspevqk
RETB_BOVIN 151 ivrqrqeelclarqyrliphngycdgksernil

MUP2_MOUSE   1 mkmllllclgltlvcvhaeeasstgrN FNVEKINGEWHTII Lasdkreki
MUP2_MOUSE  51 edngnfrlfleqihvlekslvlkfhtvrdeecselsmvadktekageysv
MUP2_MOUSE 101 tydgfnt fTIPKTDYDNFLMA HLInekdgetfqlmglygrepdlssdike
MUP2_MOUSE 151 rfaklceehgilreniidlsnanrclqare
```

Aligned Fragments

```
(1) ICYA_MANSE  18  ycpdvkpvnD FDLSAFAGAWHEIA Klplenenqg
(2) ICYA_MANSE 103  kfgqrvvnlv pWVLATDYKNYAIN YNCdyhpdkk
(1) LACB_BOVIN  26  alivtqtmkG LDIQKVAGTWYSLA Maasdislld
(1) BBP_PIEBR   17  acpevkpvdN FDWSNYHGKWWEVA Kypnsvekyg
(2) BBP_PIEBR   99  tyggvtkenv fNVLSTDNKNYIIG YYCkydedkk
(1) RETB_BOVIN  15  rvssfrvkeN FDKARFAGTWYAMA Kkdpeglflq
(-) RETB_BOVIN 123  TFAVQYSCrl lnldgtcadsysfv fardpsgfsp
(1) MUP2_MOUSE  28  aeeasstgrN FNVEKINGEWHTII Lasdkrekie
(2) MUP2_MOUSE 108  ysvtydgfnt fTIPKTDYDNFLMA HLInekdget
```

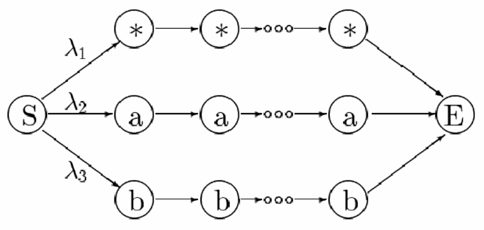
Convex Combination Model

```
A  :::12:311::2:6
C  ::::::1:::::::
D  :4:1::3:2:::::
E  :::1::::1::2::
F  7::::2::::1:1:
G  ::::1::6:::::1
H  ::::::1:::2:::
I  ::2::1:::::141
K  ::1:3:::3:::::
L  2:22:::::::11:
M  ::::::::::::2:
N  :3::1:11:3:::1
P  1::1::::::::::
Q  :::1::::::::::
R  ::::1:::::::::
S  :::21::::1:2::
T  :1:::4::2::1::
V  ::3::1:::::::11
W  :11::::::51:::
Y  :::::1:2::6:::
```

From Bailey and Gribskov [ISMB, 1996]

Example

- Suppose we generate a random sequence, with a symmetric Bernoulli source and we inject two substrings of size *m*, **aa...aaa**, and **bb...bbb**
- The following HMM would explain the sequence

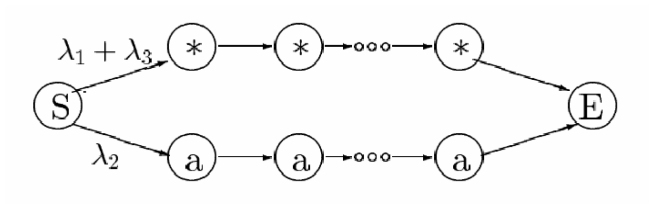


$$\begin{tikzcd}
 & * \arrow[r] & * \arrow[r] & \circ\circ\circ \arrow[r] & * \arrow[dr] & \\
S \arrow[ur, "\lambda_1"] \arrow[r, "\lambda_2"] \arrow[dr, "\lambda_3"'] & a \arrow[r] & a \arrow[r] & \circ\circ\circ \arrow[r] & a \arrow[r] & E \\
 & b \arrow[r] & b \arrow[r] & \circ\circ\circ \arrow[r] & b \arrow[ur] &
\end{tikzcd}$$

From Bailey and Gribskov [ISMB, 1996]

Example

- One would expect that MEME finds

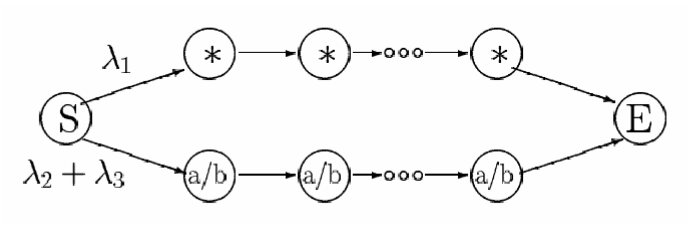


or the one modeling the all “**b**” component

From Bailey and Gribskov [ISMB, 1996]

Example

- Unfortunately, the following convex combination has sometimes higher likelihood



From Bailey and Gribskov [ISMB, 1996]

Convex combination problem

- Convex combinations are undesirable because the make unrelated sequence region to appear to be related
- The problem becomes worse and worse as the size of the alphabet, the length of the profile, or the size of the dataset increases
- In fact, convex combinations are less of a problem with DNA sequences

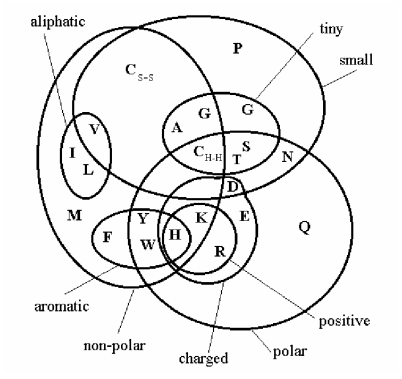
Convex combination problem

- Bailey and Gribskov propose a heuristic solution based on the use of prior distributions, called *megaprior heuristic*

- Megaprior heuristic is now part of MEME

Megaprior heuristic

- The idea is to use our prior knowledge about the similarities about the amino acids



Megaprior heuristic

- The heuristic is based on the biological knowledge about what constitute a "reasonable" column in a profile
- The prior distribution favor amino-acids in the same class to be in the same column
- Although it does not forbid two amino acid, say one hydrophobic and hydrophilic, to be in the same column, it makes it less likely to happen