

EXAM NOTE FOR Biological Sequence Analysis (2020)

MAO, Xiqing
2020.10.29
2021.02.21

Pairwise alignment

Scoring

Substitution matrices, BLOSUM62

For a protein family, cluster proteins into many groups based on the $n\%$ identity between the sequences (e.g., $\geq 62\%$).



$$p_{ab} = \frac{1}{L \times \binom{C}{2}} \sum_{i=1}^C \sum_{j>i}^C \sum_{l=1}^L \left(\frac{N_l(C_i, a)}{k_i} \times \frac{N_l(C_j, b)}{k_j} \right),$$

$$q_a = \frac{1}{LC} \sum_{i=1}^C \sum_{l=1}^L \frac{N_l(C_i, a)}{k_i},$$

where C is the numbers of non-overlapping clusters, $N_l(C_i, x)$ is the number of times that residue a appears in the l^{th} column of cluster C_i , k_i is the is the numbers of sequence of cluster C_i ($\sum k_i = k$), L is the length of sequence.

$$S(a, b) = 2 \log_2 \left(\frac{p_{ab}}{q_a q_b} \right)$$

PAM (Point Accepted Mutation) See BSA book.

Difference between PAM10 and BLOSUM62

- 1 Best score decreases, more gaps, and length increases.
- 2 Allows almost only perfect matches.
- 3 Much more negative off the diagonal than BLOSUM62

4 Gaps are 'cheaper' than mis-matches.

Gap penalties

1 Linear score

$$\gamma(g) = -gd$$

2 Affine score

$$\gamma(g) = -d - (g - 1)e,$$

where d is gap open, e is gap extension, g is gap length.

Dynamic programming

Global: Needleman-Wunsch

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{cases}$$

Local: Smith-Waterman

$$F(i, j) = \max \begin{cases} \boxed{0} \\ F(i-1, j-1) + s(x_i, y_i), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{cases}$$

Traceback It works by building the alignment in reverse, starting from the final cell, and following the pointers that we stored when building the matrix.

Difference The min score in Smith-Waterman is 0. The final cell can be anywhere instead of right corner.

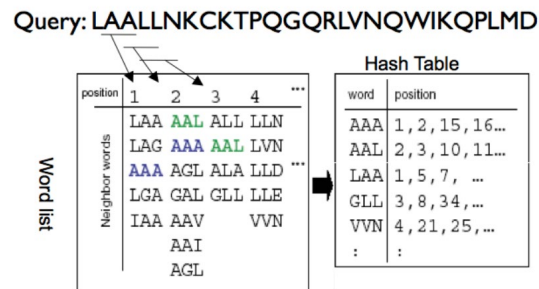
Affine Gap penalties

$$M(i, j) = \max \begin{cases} M(i-1, j-1) + s(x_i, y_i), \\ I_x(i-1, j-1) + s(x_i, y_i), \\ I_y(i-1, j-1) + s(x_i, y_i). \end{cases}$$
$$I_x(i, j) = \max \begin{cases} M(i-1, j) - d, \\ I_x(i-1, j) - e. \end{cases}$$
$$I_y(i, j) = \max \begin{cases} M(i-1, j) - d, \\ I_y(i-1, j) - e. \end{cases}$$

Heuristic alignment algorithms

BLAST

- 1 Prepare query sequence
 - 1.1 Remove low-complexity region or sequence repeats
 - 1.2 Make sequence to **overlap** k-mers, and construct a table of all possible words from each k-mer.
 - 1.3 Aligned words with k-mer, keep high-scoring ones.
 - 1.4 Index, make a hash table, mark high-scoring words in the table with location of k-mer



- 2 Scan the database for exact matches with high-scoring words.
- 3 Extension: use the location of matched words to find k-mers as seeds, using Smith-Waterman to extension until reach scoring threshold.

Why faster than S-W?

- 1 Employ a hash table to index the query sequence, so time proportional to size of database, space requirement proportional to the query size.
- 2 Only search interested k-mers.

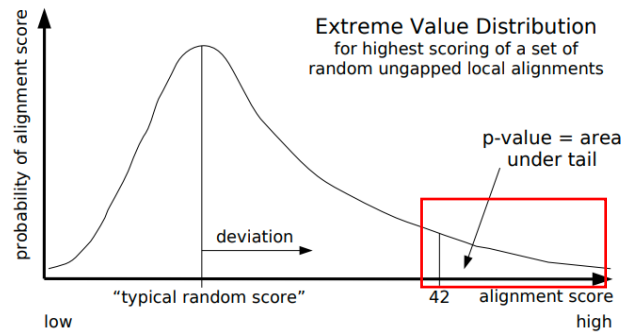
FASTA

- 1 Find k-word matches diagonally.
- 2 Hunting the best matches (scoring, substitution matrix). This step is similar to the extension process in the Blast algorithm, that is, find the maximum score and no gap area.
- 3 Joint and extension, remove some unlikely fragments.
- 4 Using 'band' S-W to search in a restricted area in order to find better alignment.

Significance of a search result

Let **ABCDE** be the query sequence. The length of seq is 5, so there are 20^5 random sequences, and we can use 20^5 random sequences to against database. (each score count should be divided by 20^5).

The distribution of such probability looks like



$$y = e^{-e^{-\lambda(x-\mu)}},$$

where $\mu = \frac{1}{\lambda} \ln Kmn$, K is a constant, m and n are the length of database and query seq respectively, λ is a positive root of $\sum q_a q_b e^{\lambda S(a,b)}$. For example, if $S(a,b) = \log(p_{ab}/p_a p_b)$, then $\lambda = 1$ (which means $e^{\lambda S(a,b)} = p_{ab}/q_a q_b$).

For BLOSUM62:

$$S(a,b) = 2 \log_2 \frac{p_{ab}}{p_a p_b},$$

since $\log_2 x = \log x / \log 2$, we have

$$\log \frac{p_{ab}}{p_a p_b} = \left(\frac{\log(2)}{2} \right) (2 \log_2 \frac{p_{ab}}{p_a p_b}),$$

so $\lambda = \log(2) / 2$.

The best score with S-W algorithm should in one of the ends of the distribution (as close as possible to the right side).

P-value

P-value is the probability to obtain the same score or higher if we searched a database of random sequences with the same size as the actual database with a random query sequence of the same length as the actual query.

$$\begin{aligned} P(S \geq x) &= 1 - e^{-e^{-\lambda(S-\mu)}} \\ &= 1 - e^{-e^{-\lambda\left(S - \frac{1}{\lambda} \ln Kmn\right)}} \\ &= 1 - e^{-e^{\ln Kmn - \lambda S}} \\ &= 1 - e^{-Kmn e^{-\lambda S}} \end{aligned}$$

E-value $E(S)$

E-value is the expected number (NOT probability) of matches with the same score or higher if we searched a database of random sequences with the same size as the actual database with a random query sequence of the same length as the

actual query.

$$E(S) = K m n e^{-\lambda S}.$$

or

$$E(s) = pD$$

$E(S)$ should be significantly less than 10^{-5} . If database doubled, E-value doubled.

Searching database

Hash table

Words $\xrightarrow{\text{hash function}}$ Hash value (Key) \rightarrow Sorted Key \rightarrow Hash Table

Collision When a bucket has more than one Key. A good hash functions give minimum number of collisions.

A perfect Hash Functions, enumeration of words

$$x_k + x_{k-1}M + x_{k-2}M^2 + \cdots + x_1M^{k-1},$$

where k is the position of words, and different letters are numbered from 0 to $M - 1$.

Example, ATCAGAA:

A	C	G	T
0	1	2	3

	NUMBER	K	VALUE
A	0	1	$0 \times 4^0 = 0$
A	0	2	$0 \times 4^1 = 0$
G	2	3	$2 \times 4^2 = 256$
A	0	4	$0 \times 4^3 = 0$
C	1	5	$1 \times 4^4 = 256$
T	3	6	$3 \times 4^5=3072$
A	0	7	$0 \times 4^6 = 0$
SUM			3854

Other sample hash function ASCII character encoding. However, any

permutation of letters in the key will give the same value.

BLAT, index database

- 1 Prepare database
 - 1.1 Index database by k-mers (non-overlap)
 - 1.2 Generate a hash table for all k-mers
 - 1.3 Generate binary tree
- 2 Search matched k-mers: between k-mers from Query sequence and k-mers from database, calculate Key.
- 3 Extension: if diagonal numbers similar, then analyzed further to see if it can be extended beyond the two matching words

PSI-BLAST

A Position Specific Iterative BLAST

- 1 Blast a single seq against database with BLOSUM62, keep significant matches.
- Repeat {
 - 2 Build a PSSM (profile) by matches, using pseudo count.
 - 3 use PSSM find new matches, calculate E-value.}

Suffix tree

Use the public preset of strings to reduce unnecessary string comparisons to achieve the purpose of improving query efficiency.

Terminology Root, node, parent, children, leaf, edge, terminal char \$

Prefix A, AB, ABC, ABCD, ABCDE

Suffix E, DE, CDE, BCDE, ABCDE

Prefix tree (trie) $O(n)$ Use prefix to build a tree, each edge represents a letter, each path represents a prefix.

Suffix tree, a zipped suffix trie.

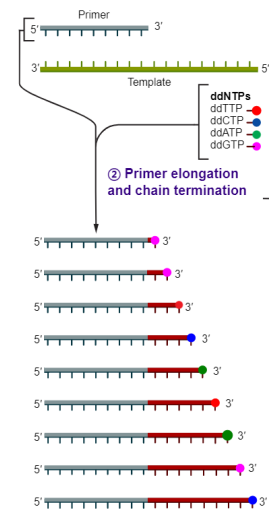
The label of each edge is a substring, rather than a single letter. Search for exact matching sequences can be done in time proportional to the length of the query.

Direct observation of the object of study (DNA/genes), transforming biology from a largely descriptive science into a quantifiable science, one of the reasons

bioinformatics as a field even exists.

The first in history - Sanger sequencing.

- 1 Synthesis
- 2 Modified dideoxynucleotides as terminators, which stop the replication.
- 3 Partial DNA copies sorted by size using electrophoresis.
- 4 Read from small to large one.



Next generation sequencing

Shotgun sequencing: Illumina

- 1 Preparation: add adapters
- 2 Cluster generation
 - 2.1 cluster amplification, Bridge PCR (repeat)
 - 2.1.1 Polymerase creates complimentary fragment
 - 2.1.2 Wash origin temple
 - 2.1.3 Bridge, replicate double strands, un-bridge and linearized
 - 2.2 Wash reverse strand
- 3 Sequencing
 - 3.1 With lighted complimentary nucleotides: first read
 - 3.2 Index read 1
 - 3.3 Bridge over again, form double strands, linearized
 - 3.4 Generate second reverse read and index read 2
- 4 Separate reads by index, locally clustered the same base call, pair reads

Compare

- 1 low labor required
- 2 better cost-efficiency, higher output volume and faster run time
- 3 much higher sequencing depth
- 4 short read length

Reads

Quality

$$P(\text{wrong base}) = 10^{-\frac{Q}{10}}$$

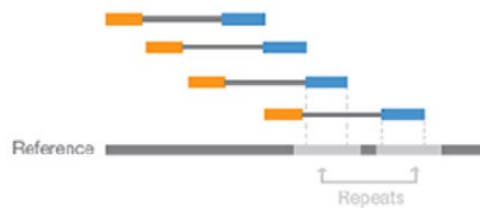
Sanger encoded $Q + 33$ (Sanger offset) \rightarrow ASCII. Generally, $Q > 30$ means good quality. $Q = 0$ means non-uniquely mapped.

```

Header  → @ILLUMINA-C90280_0030_FC:5:1:2675:1090#NNNNNN/1
Sequence → ATTCCTGGCCTTTTCCAGGCCTGCCTGCTCGAGC
          +
Qualities → BAAAGECEE<EEDFEDF3DBDBB=A+=>9>>88?
          (prob. that base call is wrong)

```

Paired reads: 150bp for each end, distance, help assemble.



Mapping

Problems:

- 1 M(B)illions of reads and large reference.
- 2 Repeat in reference.
- 3 Mutation, sequencing errors

Solution:

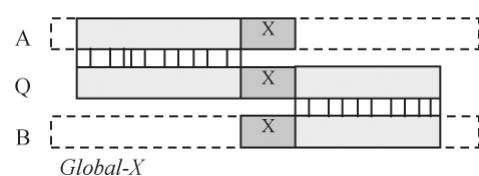
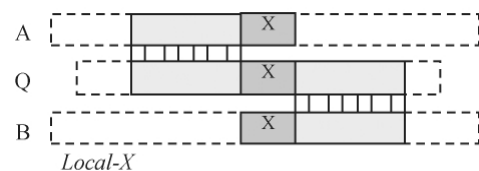
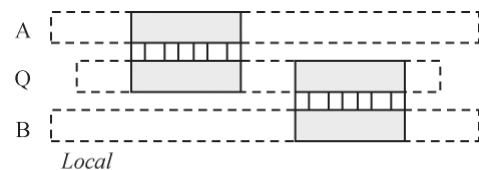
Burrows-Wheeler transform (BWT)

Chimeric alignments:

`gcctaAGCTAA ttagctTAGGC`. One of the linear alignments in a chimeric alignment is considered the “**representative**” alignment, and the others are called “**supplementary**” and are distinguished by the supplementary alignment flag.

It can be caused by real biological variation, esp. structural variation, or specific types of experiments.

Singletons: mate did not map. It can be caused by large insertions, or the organism in the sample is actually not the one we’re mapping to, they merely share a common region.

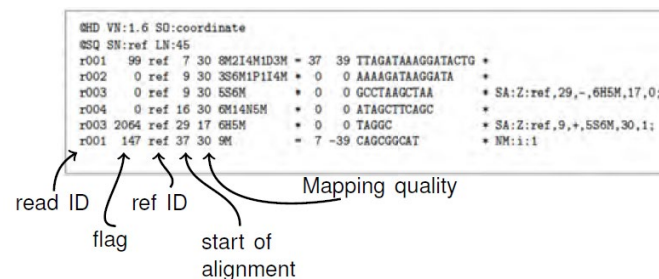


Multimapping: repeats in the reference, which will have several alignments one of which will be **primary** and the others **secondary**.

Multimapping reads are not used for SNP calling, since their placement is unsure, unless there is a best-scoring, unique, non-overlapping primary alignment.

SAM Format

- 1 Read ID
- 2 Flag: 0 mapped single segment
- 3 Reference ID
- 4 Start of alignment: count when aligned.
- 5 Mapping quality



@HD VN:1.6 SO:coordinate											
@SQ SN:ref LN:45											
r001	99	ref	7	30	8M2I4M1D3M	=	37	39	TTAGATAAAGGATACTG	*	
r002	0	ref	9	30	3S6M1P1I4M	*	0	0	AAAAGATAAGGATA	*	
r003	0	ref	9	30	5S6M	*	0	0	GCCTAAGCTAA	*	SA:Z:ref,29,-,6H5M,17,0;
r004	0	ref	16	30	6M14NSM	*	0	0	ATAGCTTCAGC	*	
r003	2064	ref	29	17	6H5M	*	0	0	TAGGC	*	SA:Z:ref,9,+,5S6M,30,1;
r001	147	ref	37	30	9M	=	7	-39	CAGCGGCAT	*	NM:i:1

Whole genome sequencing

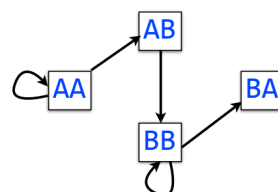
Why? To find something NEW: genome, compounds, pathways, difference in species

Challenges: same as in mapping, and: Non-uniform depth (how overlap), target genome unknown, huge amount of data.

WGS typically involve assembly.

de novo assembly: when no reference, no matched reference

deBruijn graphs: k-mers, **overlap**, Eulerian path.



Variant calling

Genetic variant: single nucleotide variant (SNV), structural variants, insertions, deletions

On NGS, an extension of mapping, vcf format. We know the correct location of a read in a reference, and, are there any differences between the read and the reference? Mutations? Sequencing errors?



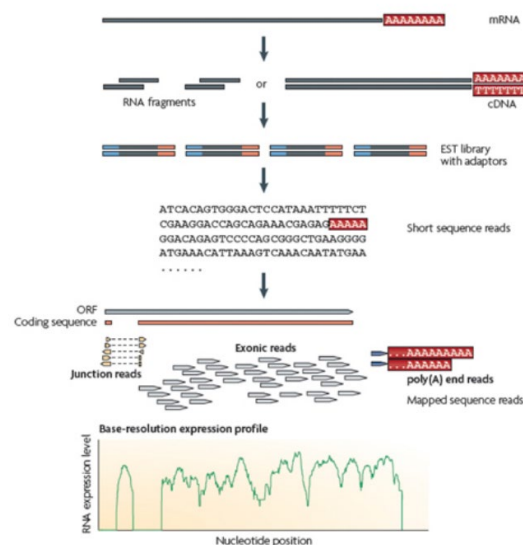
RNA Seq and Transcriptomics

每个染色体由一堆基因组成，不是所有的基因都是活跃的，只有一部分基因是可以表达。而表达的中间过程要经历 mRNA 转录本。通过高通量测序，可知哪些基因是活跃可以表达的，并且产生了多少转录本。

RNA seq: translated to DNA, add adaptors, which read is actual gene?

Problem: introns and exons

Solution: some program, or just map into coding region.



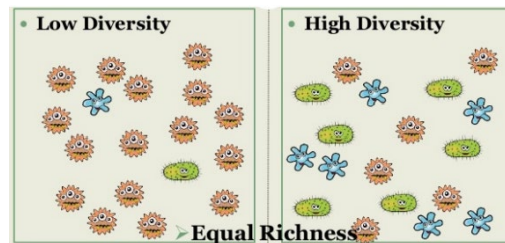
Metagenomics

Metagenomics, study microbial communities by **direct sequencing environment DNA**. It allows us to study complex environments and microbes that **cannot be cultured in the lab**.

Abundance: the number of individuals of each species in an area

Richness: the number of species in an area

Diversity: the richness and abundance and the distribution of these species in that ecosystem.



HMMs

Weight matrix (PSSM)

Let generate some sequences:

$$X^1 = x_1^1, x_2^1, \dots, x_l^1,$$

$$X^2 = x_1^2, x_2^2, \dots, x_l^2,$$

...

then the probability of the sequence is

$$P(X^i) = p_1(x_1^i)p_2(x_2^i) \dots p_l(x_l^i),$$

the log-odds of it will be

$$\log \frac{P(X^i)}{Q(X)} = \log \frac{p_1(x_1^i)}{q(x_1)} + \dots + \log \frac{p_l(x_l^i)}{q(x_l)},$$

where $q(x)$ is the probability of letter x in all position (background).

Sometimes, it is good to take the background distribution into account, rather than use uniform distribution.

Information in a site

Shannon entropy

$$H(x) = - \sum_i p(x_i) \log p(x_i)$$

The relative entropy

$$H(p||q) = \sum_a p(a) \log \frac{p(a)}{q(a)}$$

If $q(a)$ is a uniform distribution, then the relative entropy is equivalent to

Shannon entropy.

Average score of sequence

$$\sum_a p_1(a) \log \frac{p_1(a)}{q(a)} + \sum_a p_2(a) \log \frac{p_2(a)}{q(a)} + \dots + \sum_a p_l(a) \log \frac{p_l(a)}{q(a)}$$

Mutual information

$$M(X; Y) = \sum_{i,j} p(x_i, y_i) \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}$$

LOGO

Let I be the information content of position i of the alignment:

$$I(i) = \sum_{b \in A} I(b, i) = \sum_{b \in A} p_i(b) \log \frac{p_i(b)}{q(b)} \text{ (bits per position),}$$

where b is one of the bases $\in \{A, T, C, G\}$, $p_i(b)$ is the probability of base b at position i , $q(b)$ is the priori distribution of the bases for that genome (background).

Let set $q(b) = 0.25$. If only 1 base appears in the alignment, such as an A, then $I(A, i) = 2 \text{ bit}$, while the other bases are 0 (we define $0 \log 0 = 0$), meaning that $I(i) = 2 + 3 \times 0 = 2 \text{ bit}$.

If two bases appeared with equal frequency (as in $p_i(A) = p_i(T) = 0.5, p_i(C) = p_i(G) = 0$), $I(i)$ would be $0.5 + 0.5 + 2 \times 0 = 1 \text{ bit}$

If all 4 bases appeared with equal frequencies, then $I(i) = 0$.

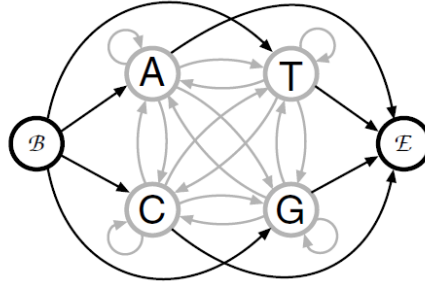
Then, the height of base b at position i :

$$p_i(b) \times I(i)$$

Donor site



Markov chain



Let sequence $X = x_1, x_2, \dots, x_l$.

Transition probability ($s \rightarrow t, t': \text{all letters}$)

$$a_{st} = P(x_i = t | x_{i-1} = s) = \frac{c_{st}}{\sum_{t'} c_{st'}}$$

Probability of sequence X

$$P(X) = P(x_l, \dots, x_2, x_1) = P(x_l | x_{l-1}, \dots, x_1) P(x_{l-1} | x_{l-2}, \dots, x_1) \dots P(x_1).$$

Key property: the probability of each symbol x_i depends only on the value of the preceding symbol x_{i-1} , not on the entire previous sequence. Thus

$$P(X) = p(x_1) \prod_{i=2}^l a_{x_{i-1} x_i}.$$

Beginning and ending of sequences

Define $x_0 = \mathcal{B}$, so for instance the probability of the first letter in the sequence is

$$P(x_1 = s) = a_{\mathcal{B}s},$$

and the probability of ending with residue t is

$$P(\mathcal{E} | x_l = t) = a_{t\mathcal{E}}.$$

HMM

Transition probability

$$a_{kl} = P(\pi_i = l | \pi_{i-1} = k).$$

Emission probability

$$e_k(b) = P(x_i = b | \pi_i = k).$$

Probability of sequence X over path π

$$P(X, \pi) = a_{0\pi_1} \prod_{i=1}^l e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}$$

HMM Decoding: The Viterbi algorithm

	0	x_1	x_2	x_3	...	x_{i-1}	x_i	...
$\mathcal{B}(0)$	1	0	0	0	0	0	0	0
1	0					$v_1(i-1)$		
2	0					$v_2(i-1)$		
3	0					$v_3(i-1)$		
...	0					...		
k	0					$v_k(i-1)$		
...	0					...		
l	0					...	$v_l(i) = \dots \times a_{kl} \times e_l(x_i)$	
...	0					...		
...								
\mathcal{E}	0					...		End

The Viterbi can find the most probable state path (decoding).

Algorithm

Initialization ($i = 0$):

$$v_0(0) = 1, v_k(0) = 0 \text{ for } k > 0.$$

Recursion ($i = 1 \dots L$):

$$v_l(i) = e_l(x_i) \max_k (v_k(i-1) a_{kl})$$

$$ptr_i(l) = \operatorname{argmax}_k (v_k(i-1) a_{kl})$$

Termination:

$$P(x, \pi^*) = \max_k (v_k(L) a_{k0})$$

$$\pi_L^* = \operatorname{argmax}_k (v_k(L) a_{k0}).$$

Traceback ($i = 1 \dots L$):

$$\pi_L^* = ptr_i(\pi_i^*).$$

Evaluation: The forward and backward algorithm

The forward

The forward algorithm can calculate all probability of all possible path:

$$P(X) = \sum_{\pi} P(x, \pi).$$

The quantity corresponding to the Viterbi variable $v_k(i)$ in the forward algorithm is

$$f_l(i) = P(x_1 \dots x_i, \pi_i = l),$$

which is the probability of the observed sequence up to and including x_i , requiring that $\pi_i = l$.

The forward Algorithm

Initialization ($i = 0$):

$$f_0(0) = 1, f_k(0) = 0 \text{ for } k > 0.$$

Recursion ($i = 1 \dots L$):

$$f_l(i) = e_l(x_i) \sum_k (f_k(i-1) a_{kl})$$

Termination:

$$P(X) = \sum_k f_k(L) a_{k0}.$$

The backward

The probability of producing the entire observed sequence X with the i th symbol being produced by state k :

$$\begin{aligned} P(X, \pi_i = k) &= P(x_1 \dots x_i, \pi_i = k) P(x_{i+1} \dots x_L | x_1 \dots x_i, \pi_i = k) \\ &= P(x_1 \dots x_i, \pi_i = k) \mathbf{P}(\mathbf{x}_{i+1} \dots \mathbf{x}_L | \boldsymbol{\pi}_i = \mathbf{k}) \\ &= f_k(i) \mathbf{b}_k(i) \end{aligned}$$

The backward algorithm

Initialization ($i = L$):

$$b_k(L) = a_{k0} \text{ for all } k.$$

Recursion ($i = L-1, \dots, 1$):

$$b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$$

Termination:

$$P(X) = \sum_l a_{0l} e_l(x_1) b_l(1)$$

Posterior probabilities The probability that observation x_i came from state k given the observed sequence:

$$\begin{aligned} P(\pi_i = k | X) &= \frac{P(X, \pi_i = k)}{P(X)} \\ &= \frac{f_k(i) b_k(i)}{P(X)}. \end{aligned}$$

We can now define $\hat{\pi} = \operatorname{argmax}_k P(\pi_i = k | X)$, which represents the most likely state at position i of sequence X . Thus, with the posterior decoding we can compute the **most likely state at each position**. This in general is more helpful than the Viterbi path π^* .

Note also that the posterior decoding at position i , $\hat{\pi}_i$ may not coincide with π_i^* . Furthermore, the posterior decoding may give an invalid sequence of states,

unlike the Viterbi decoding, since there may be zero probability from transition from $\hat{\pi}_i$ to $\hat{\pi}_{i+1}$.

In other words, the Viterbi algorithm gives the most likely valid sequence of states that generated the sequence X , while the posterior decoding gives the most likely state at each position, and the resulting path **may not be a valid** sequence of states due to **zero transition probability between states of two consecutive positions**.

Parameter estimation

When the state sequence is known (remember to add pseudo counts):

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}, e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}.$$

When the state sequence is unknown: B-W.

- 1 The expected times of using a_{kl} is by summing overall positions of overall training sequences.

$$A_{kl} = \frac{\sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1)}{\sum_j P(X^j)}.$$

- 2 The expected number of times that letter b appears in state k :

$$E_k(b) = \frac{\sum_{\{i|x_i^j=b\}} f_k^j(i) b_k^j(i)}{\sum_j P(X^j)}$$

HMM learning: B-W

Algorithm: Baum-Welch

Initialization: Pick **arbitrary** model parameters.

Recurrence:

Set all the A and E variables to their pseudo count values r (or to zero).

For each seq j in $1 \dots n$:

Calculate $f_k(i)$ for sequence j using the forward algorithm.

Calculate $b_k(i)$ for sequence j using the backward algorithm.

Add the contribution of sequence j to A and E .

Calculate the new model parameters a and e .

Calculate the new log likelihood of the model:

$$l(X^1, \dots, X^n | \theta) = \log P(X^1, \dots, X^n | \theta) = \sum_{j=1}^n \log P(X^j | \theta).$$

Termination:

Stop if the change in log likelihood is less than some predefined threshold or the maximum number of iterations is exceeded.

Alternatively: Viterbi training

Initialization: start with an **arbitrary** set of parameters θ .

Recurrence:

Perform Viterbi, to find π^*

Calculate $A_{kl}, E_k(b)$ according to π^* and pseudocounts

Calculate the new parameters a, e .

Termination: when convergence

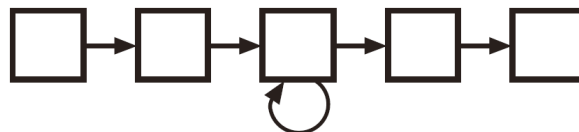
Unlike Baum-Welch, we use both θ and π^* to estimate the number of transitions and emissions $A_{kl}, E_k(b)$, which we then use to find the new set of parameters.

Viterbi training is not usually recommended in general because the performance is worse than Baum-Welch in general. However, if you are planning to use your model for Viterbi, it may be faster.

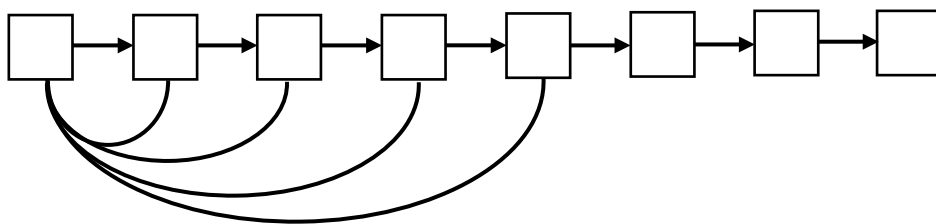
HMM Length modeling

Length distribution: when emission probability does not change

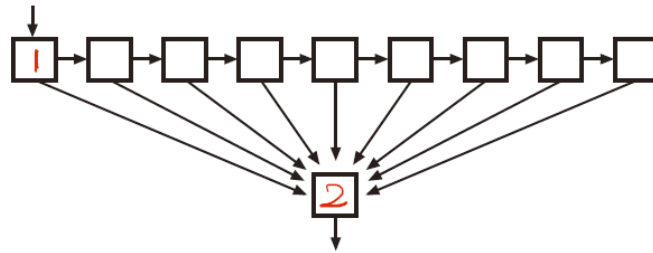
- 1 5 or more



- 2 5~8, can set different probability for different length. Usually, the emission probability would be same (tied).

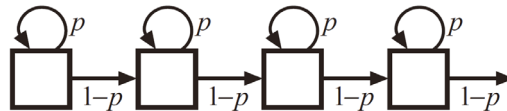


- 3 2~10



- 4 Min 4. Can calculate the probability of certain length.

$$P(l) = \binom{l-1}{n-1} p^{l-n} (1-p)^n$$



Silent states ○

Profile HMMs

Sequence evolution

Proteins are more conserved than folds, folds are more conserved than sequence. It is better to do protein search (conserved and more information).

Sequence profiles

Profile is a representation of a multiple alignment, summarizing the frequency of letters at each position – pretty like PSSM, except that **it can treat gaps**.

We can do a standard alignment by dynamic programming with a profile replacing one of the two sequences: define a score between a position in the profile and a single letter.

Scoring

- 1 Using substitution matrix, take an average score:

HBA_HUMAN	..VCA--HAGEY...
HBB_HUMAN	...V----NVDEV...
MYG_PHYCA	...VEA--DVAGH...
GLB3_CHITP	...VKG-----D...
GLB5_PETMA	...VYS--TYETS...
LGB2_LUPLU	...FNA--NIPKH...
GLB1_GLYDI	...IAGADNGAGV...
	*** *****

One amino acid **a** against a profile

$$\text{Score}(\text{col 1}, a) = \frac{5S(V, a) + S(F, a) + S(I, a)}{7},$$

No bio foundation in such scoring system. For example, column 1 is much more strongly conserved than column 2, but the information in column 1 may be smeared out just as much by the substitution matrix as that in column 2. And if seven letters in a column are same, say 7 V, the score will be the same as against only 1 V.

- 2 Another non-probability scoring system (see MSA).
- 3 Weight matrix scoring (PSSM), ignore gaps.

let L be the number of columns, $e_i(a)$ be the frequency of letter a in column i (regularized by pseudo counts):

$$P(X|M) = \prod_{i=1}^L e_i(x_i).$$

Log-odds score:

$$S = \sum_{i=1}^L \log\left(\frac{e_i(x_i)}{q_{x_i}}\right).$$

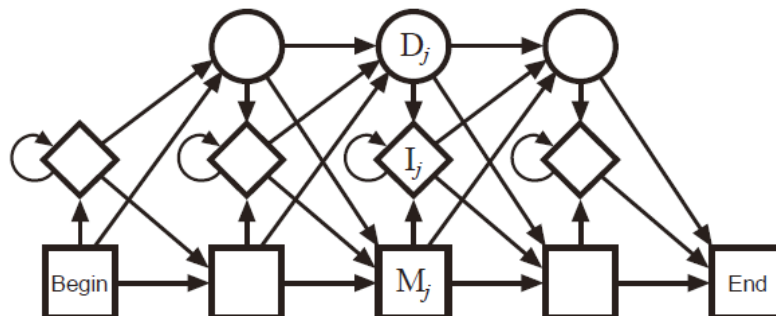
Profile HMMs

PSSM as Profile HMM

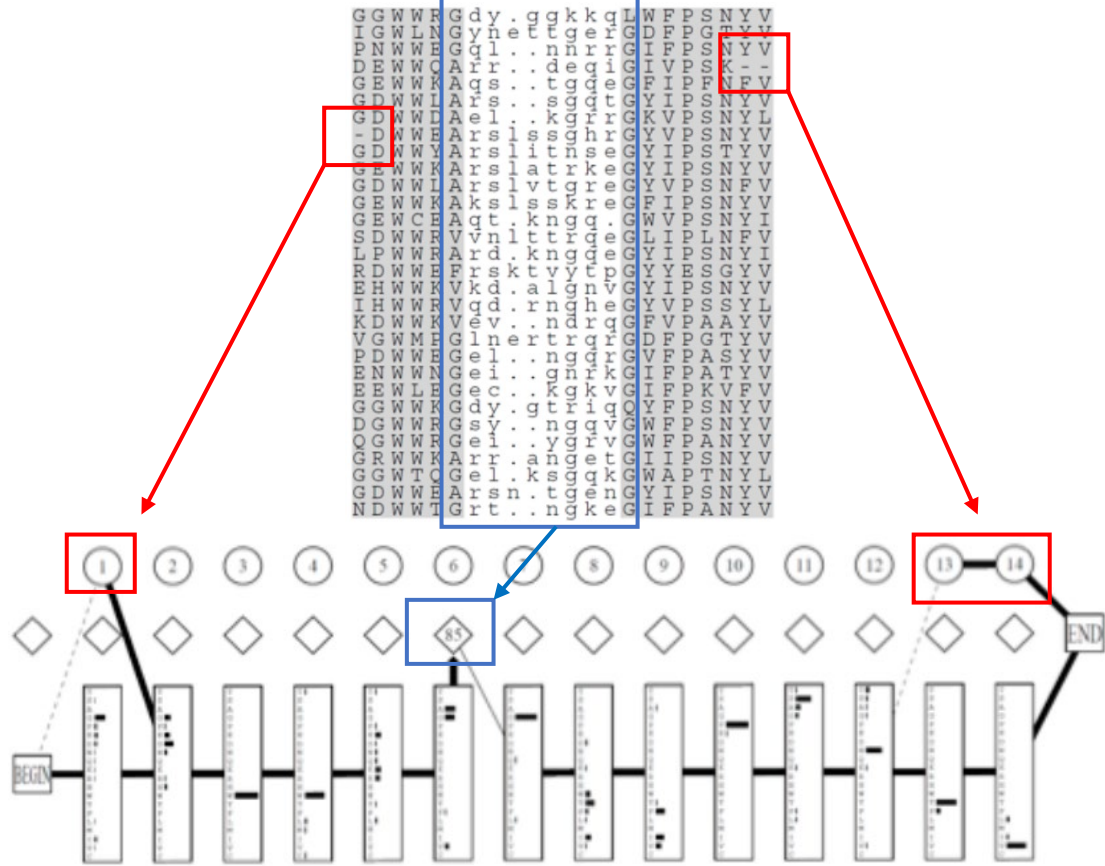
- 1 Regard as state $M(\text{atch})$.
- 2 Position in PSSM = states in M in HMM
- 3 Frequency at each position = emission probability
- 4 Transition probability = 1
- 5 Treat gaps and insertions: D and I

New states: $D(\text{eletion})$ and $I(\text{nsertion})$

The D has no emission probability, and the same as pairwise alignment, the score of $M \rightarrow D$ could be different from $D \rightarrow D$. We assume that $e_{I_i}(a) = q_a$, so there is no log-odds contribution from the emission of insertion.



Example: build a profile HMM (not for alignment)



- 1 Emission probability: calculate frequency of each letter in each column.
- 2 $a_{0D_1} = a_{M_{12}D_{13}} = \frac{1}{30}, a_{D_{13}D_{14}} = 1.$
- 3 $a_{M_6I_6} = 1, a_{I_6I_6} = \frac{\text{looping } I \rightarrow I}{30 \times 7} = 0.85, a_{I_6M_7} = 1 - 0.85 = 0.15$. Note that $e_{I_6}(a) = q(a).$
- 4 What if something in expectation? It is best to regularized by pseudo counts.

Search with Profile HMMs: give a score to measure similarity between interesting sequence and given profile.

- 1 Viterbi

$$V_j^M(i) = \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j}, \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j}, \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j}; \end{cases}$$

$$V_j^I(i) = \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_j^M(i-1) + \log a_{M_jI_j}, \\ V_j^I(i-1) + \log a_{I_jI_j}, \\ V_j^D(i-1) + \log a_{D_jI_j}; \end{cases}$$

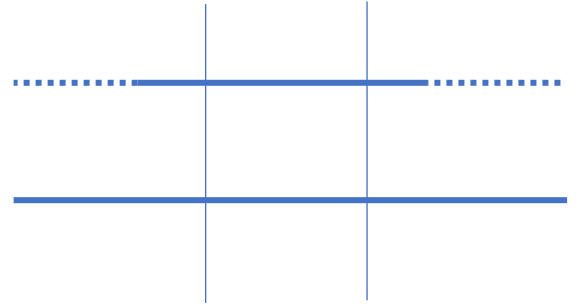
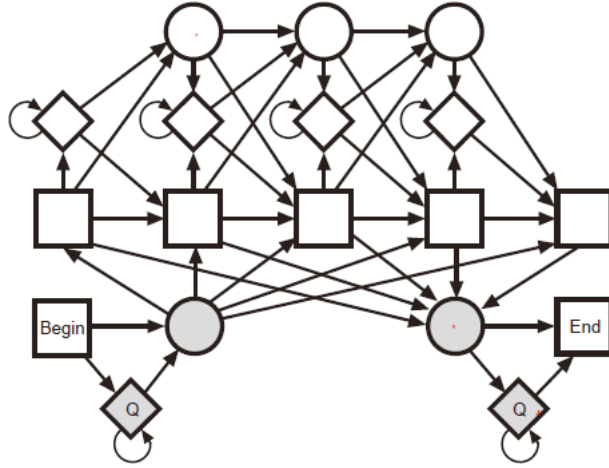
$$V_j^D(i) = \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1}D_j}, \\ V_{j-1}^I(i) + \log a_{I_{j-1}D_j}, \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j}. \end{cases}$$

2 Forward:

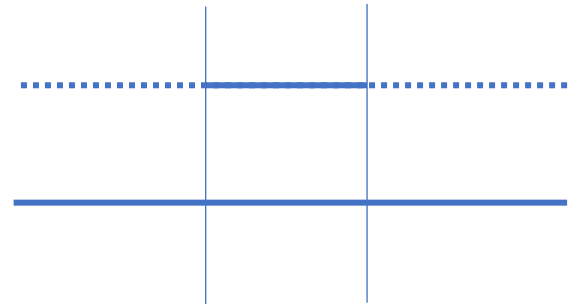
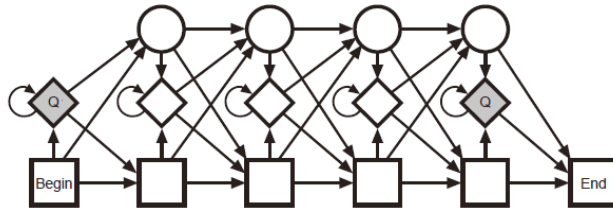
$$\begin{aligned}
 F_j^M(i) &= \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \log [a_{M_{j-1}M_j} \exp(F_{j-1}^M(i-1)) \\
 &\quad + a_{I_{j-1}M_j} \exp(F_{j-1}^I(i-1)) + a_{D_{j-1}M_j} \exp(F_{j-1}^D(i-1))] ; \\
 F_j^I(i) &= \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \log [a_{M_jI_j} \exp(F_j^M(i-1)) \\
 &\quad + a_{I_jI_j} \exp(F_j^I(i-1)) + a_{D_jI_j} \exp(F_j^D(i-1))] ; \\
 F_j^D(i) &= \log [a_{M_{j-1}D_j} \exp(F_{j-1}^M(i)) + a_{I_{j-1}D_j} \exp(F_{j-1}^I(i)) \\
 &\quad + a_{D_{j-1}D_j} \exp(F_{j-1}^D(i))] .
 \end{aligned}$$

More search types

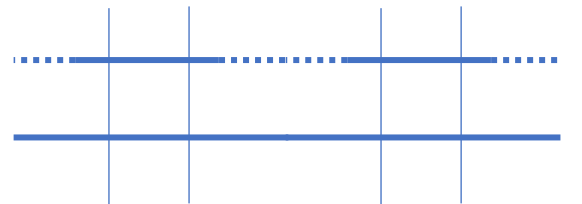
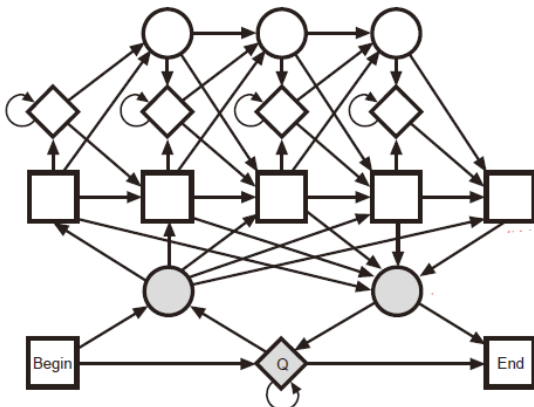
Local — Smith-Waterman type



Global in model, local in sequence



Several local matches



Pseudo counts

Laplace rule: add '1'

Example: for previous a_{0D_1} , it would be

$$\frac{1 + 1}{30 + 20} = \frac{2}{50}.$$

From background distribution

$$e_{M_j} = \frac{c_{ja} + A \times q_a}{\sum_{a'} c_{ja'} + A}$$

where A is some constant like 20 for amino acids, q_a is background distribution.

From substitution matrices

Letter Frequencies for column j and letter a :

$$f_{ja} = \frac{c_{ja}}{\sum_{a'} c_{ja'}},$$

where a' is the total count of all amino acids in the column.

Rewrite definition of substitution matrix:

$$S(a, b) = \log \left(\frac{P(a, b)}{q_a q_b} \right) = \log \left(\frac{P(a|b)P(b)}{q_a q_b} \right) = \log \left(\frac{P(a|b)}{q_a} \right)$$

since $P(b) = q_b$. By taking the exponential function on both sides and isolating $P(a|b)$, we get the substitution probabilities

$$P(a|b) = q_a e^{S(a, b)}.$$

This is the probability of observing letter a given that the letter in the profile is b . Since there are (usually) several different letters in column j of the profile, we will mix these probabilities according to the letter frequencies. Therefore, we set the pseudo count to

$$\alpha_{ja} = \sum_{b \in \text{col } j} P(a|b) f_{jb} = \sum_b q_a e^{S(a, b)} f_{jb}.$$

So instead of using $e_{M_j}(a) = f_{ja}$, we add these pseudo counts:

$$e_{M_j} = \frac{c_{ja} + A \times \alpha_{ja}}{\sum_{a'} c_{ja'} + A}.$$

MSA

Definition: Given N sequences X_1, X_2, \dots, X_n , insert gaps (-) in each sequence x_i , such that all sequences have the same length L , and score of the

global map is **maximum**.

Why MSA? Because it can identify/build on highly conserved positions.

Objectives: reflect evolution and structure.

How to make an MSA?

- 1 Find conserved residues, e.g., hydrophobic, disulfide bond
- 2 Align the key residues, core structural elements.
- 3 Handle gaps: gaps should be aligned, and should disrupt the conserved columns as little as possible because number of conserved columns should be high
- 4 Scoring, it more important than algorithm.
- 5 Find highest scoring.

Applications: build a profile HMM to find remote homologs, classification of proteins, evolutionary analyses.

Scoring

$$S(m) = G + \sum_i S(m_i),$$

where m_i is the multiple alignment m of column i , $S(m_i)$ is the score for column i , and G is a function for scoring the gaps that occur in the alignment. The question is how to score each column i and how to treat gaps.

Minimum Entropy

			<i>Col_i</i>	
<i>Seq_j</i>			m_i^j	

m is an MSA. Let m_i^j be the symbol in column i for sequence j . Let c_{ia} be the observed counts for residue a in column i ; $c_{ia} = \sum_j \delta(m_i^j = a)$.

Let m_i be the column m_i^1, \dots, m_i^N of aligned symbols in column i , and let c_i be the count vector c_i^1, \dots, c_i^K of observed symbols in column i for an alphabet of K different residues.

We assume that sequences have all been generated independently, the probability of a column m_i is

$$P(m_i) = \prod_a P_{ia}^{c_{ia}},$$

where P_{ia} is the probability of residue a in column i , it can be estimated from counts c_{ia} :

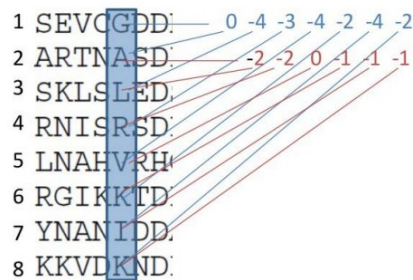
$$P_{ia} = \frac{c_{ia}}{\sum_{a'} c_{ia'}}.$$

We can define a column score as the negative logarithm of this probability:

$$S(m_i) = - \sum_a c_{ia} \log P_{ia},$$

so a completely conserved column would score 0.

Sum-of-Pairs score



$$S(m_i) = \sum_{k < l} s(m_i^k, m_i^l),$$

where the $s(m_i^k, m_i^l)$ is the substitution score for replacing residue m_k with residue m_l .

Progressive alignment method

Feng-Doolittle

Algorithm:

- 1 Calculate a diagonal matrix of $N(N-1)/2$ scores between all pairs of N sequences by standard pairwise alignment, and converting raw alignment scores to approximate pairwise 'distance' D :

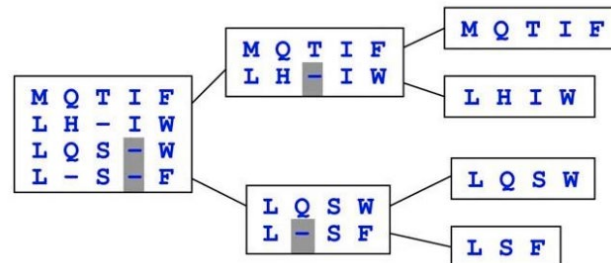
$$D = -\log S_{eff} = -\log \frac{S_{obs} - S_{rand}}{S_{max} - S_{rand}},$$

where S_{obs} is the observed pairwise alignment score; S_{max} is the maximum score, which is the average of the score of aligning either sequence to itself; and S_{rand} is the expected score for aligning two random sequences of the same length and residue composition.

- 2 Construct a guide tree from the distance matrix.
- 3 Starting from the first node added to the tree, align the child nodes (which may be two sequences, a sequence and an alignment, or two alignments).

- 4 Repeat for all other nodes in the order that they were added to the tree (i.e. from most similar pairs to least similar pairs) until all sequences have been aligned.

Profile-profile alignment



Assume we have two multiple alignments (or ‘profiles’), one containing sequence $1 \sim n$, and the other containing sequence $n + 1 \sim N$. An alignment of these two profiles means that gaps are inserted in whole columns, so the alignment within one of the profiles is not changed. The score of the global alignment is then

$$\begin{aligned} \sum_i S(m_i) &= \sum_i \sum_{k < l \leq N} S(m_i^k, m_i^l) \\ &= \sum_i \sum_{k < l \leq n} S(m_i^k, m_i^l) + \sum_i \sum_{n < k < l \leq N} S(m_i^k, m_i^l) + \sum_i \sum_{k \leq n, n < l \leq N} S(m_i^k, m_i^l), \end{aligned}$$

which is the score of profile 1 plus the score of profile 2 plus the score of **profile 1 and profile 2 (cross term)**, and let $s(-, a) = s(a, -) = -g$ and $s(-, -) = 0$.

ClustalW (newer: Clustal Ω)

Iterative refinement

T-Coffee

...

Build a guide tree

Many MSA algorithms require a guide tree to align. Guide tree is a binary tree whose leaves represent sequences and whose interior nodes represent alignments. The root node represents a complete multiple alignment. The nodes furthest from the root represent the most similar pairs.

An example to build a guide tree: **UPGMA**

Besides the way to calculate distance in Feng-Doolittle, we can also calculate in following way:

We define the distance d_{ij} between two clusters C_i and C_j :

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq},$$

where $|C_i|$ and $|C_j|$ denote the number of sequences in clusters i and j , respectively. Let $C_k = C_i \cup C_j$, and C_l is any other cluster, then:

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}.$$

Algorithm: UPGMA

Initialization:

Assign each sequence i to its own cluster C_i ,

Define one leaf of tree T for each sequence, and place at height zero.

Iteration:

Find minimal distance d_{ij} for the two clusters i, j . (If there are several equidistant minimal pairs, pick one randomly.)

Define a new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l .

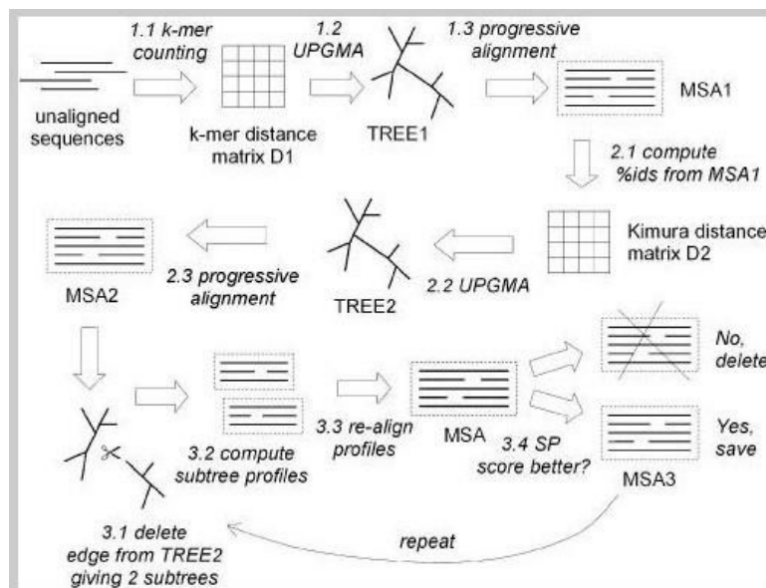
Define a node k with daughter nodes i and j , and place it at height $d_{ij}/2$.

Add k to the current clusters and remove i and j .

Termination:

When only two clusters i, j remain, place the root at height $d_{ij}/2$.

Based on k-mer distance: MSUAL



MSA from profile HMMs

Algorithm: Multiple alignment using profile HMMs

Initialization:

Choose the length of the profile HMM and initialize parameters.

Training:

Estimate the model using the Baum–Welch algorithm or the Viterbi alternatively. It is usually necessary to use a heuristic method for avoiding local optima.

Multiple alignment:

Align all sequences to the final model using the Viterbi algorithm and build a multiple alignment.

Coevolution

Example

Coevolution can be found in intramolecular, intermolecular, and even between species.

- 1 Disulfide bond
- 2 Compensatory mutations are conserved, while single “mismatched” changes are not (due to evolution pressure).
- 3 Viral coat proteins and RNA coevolve to retain key interactions
- 4 Shapes of flowering plants match those of bird beaks

Motivation

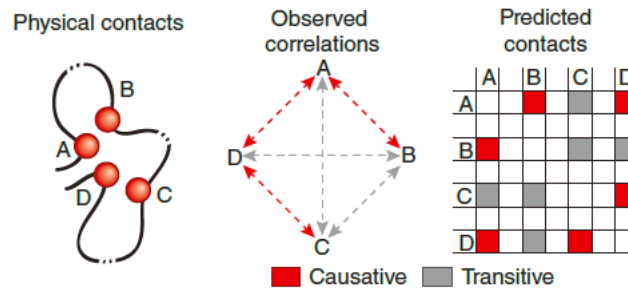
Create a model that captures & reproduces sequences from a family. Can HMM do this task? No, the letter in HMM only depends in previous one (not enough remote).

Major challenge

In the 18-year history of contact-prediction methods using correlated mutations, all methods used local mutual information or other local statistical models:

$$M(X, Y) = \sum_{i,j} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)},$$

and it means how much can we learn about position x by observing position y , which assume **that pairs of residue positions (or ‘columns i and j ’) are statistically independent of other pairs of residues**. In real proteins, however, residues can contact many other residues, and local models can not deal with such ‘indirect contact’:



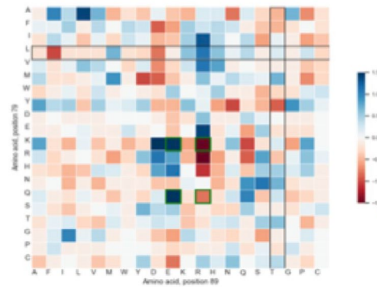
A global statistical model: Maximum entropy

Transitive correlations can be removed by global statistical approaches. A ‘global’ modeling approach treats correlated pairs of residues as dependent on each other, rather than as statistically independent, thereby minimizing the effects of transitivity and spurious noise.

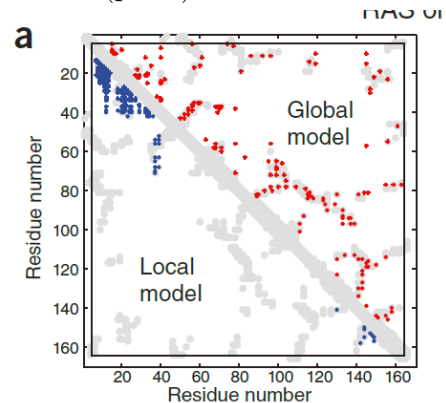
One global statistical approach is known as entropy maximization under data constraints. In contrast to simple mutual information, the conditional mutual information can be thought of as the degree of covariation between residues at positions a and b that is due solely to direct effects of a on b , factoring out contributions to the correlation that are caused by interaction of both a and b with the rest of the network of residues.

Steps

- 1 Create a multiple sequence alignment between many members of an evolutionarily related protein family.
- 2 Calculate the covariance matrix of dimension $(20L)^2$, where L is the length of the protein sequence: by counting how often a given pair of the 20 amino acids, say alanine and lysine, occurs in a particular pair of positions, say position 15 and 67, in any one sequence.



- 3 Summing over all sequences in the multiple sequence alignment. This large matrix contains the raw data capturing all residue pair relationships across evolution up to second order (pairs).



- 4 One can then compute a measure of causative correlations (the conditional mutual information) in the global statistical approaches by taking the inverse of the covariance matrix, and resulting explicit probability model for a sequence in the particular protein family resulting from inversion of the covariation matrix contains numerical estimates of direct pair interactions:

$$P(A|J, h) = \frac{1}{Z} \exp \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^N J_{ij}(a_i, a_j) + \sum_{i=1}^N h_i(a_i) \right),$$

where $J_{ij}(a_i, a_j)$ and $h_i(a_i)$ constrain the agreement of the probability model with pair and single residue occurrences, respectively.

Application

- 1 Use covarying residues to constrain structural modelling
- 2 Coevolution constraints lead to accurate protein structure models (AlphaFold)
- 3 Design of new sequences in the same family

Motif discovery

Binding sites and other sequence patterns are often called *motifs*. Motifs are

sequence elements that are approximately **conserved** – short fragments that align well.

Word-based methods

Word Statistics using a Markov Background

We want to calculate the probability that the word $(a_1 \dots a_n)$ appears in a sequence that has been randomly generated by an m th order Markov model with $m < n$. Let us define $f(a_1 \dots a_m)$ as the frequency of the word $(a_1 \dots a_m)$ in the sequence. We can then estimate an m th order Markov model with transition probabilities $P_m(a_{m+1}|a_1 \dots a_m) = f(a_1 \dots a_m, a_{m+1}) / \sum_c f(a_1 \dots a_m, c)$. Then the probability of the word $a_1 \dots a_n$ would be

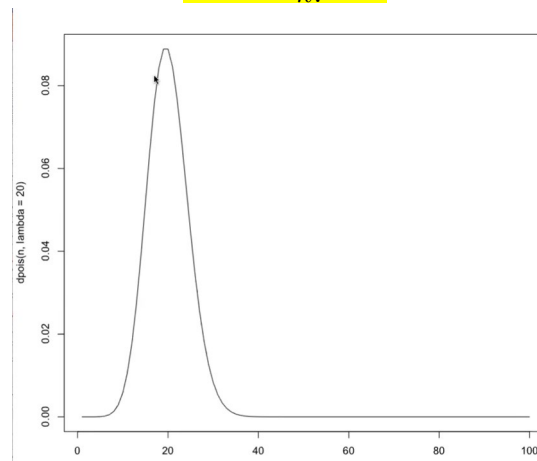
$$P(a_1 \dots a_n) = f(a_1 \dots a_m) \prod_{i=m+1}^n P_m(a_i | a_{i-1} \dots a_{i-m}).$$

If l is the length of sequence, then the **expected** numbers of such word occurrences are around

$$\lambda = pl.$$

Then the probability of observing k words is

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}.$$



Now we need to compare two sequence: if a word in our interesting sequence occurs z times, a P-value is obtained by summing terms with $k \geq z$ for under-represented, or $z \geq k$ for over-represented.

Word Statistics with a background probability

However, when it comes to low complexity/repeated words, like **GGGGG**, Markov chain will have a bad performance. And biological sequences often do not behave like Markov chains.

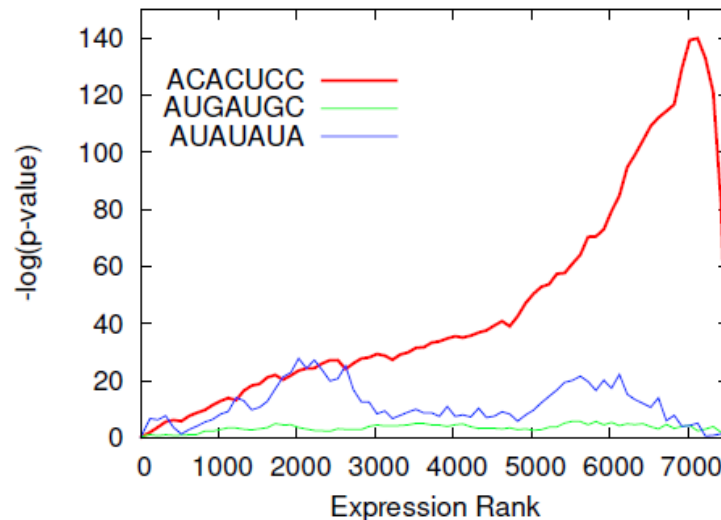
One solution is to use a real background, like all 3' UTR from human genome – just any set that we expect different, instead of Markov background. Then compare occurrence in two sets and calculate a p-value.

One way to calculate occurrence is to do a proper statistical test: let us assume that the background is one long sequence of length L and the word under study occurs K times in the background sequence, for foreground it is l and k . You ask how likely it is to select k words in the foreground set (l possible position where you can put length n) and K in the background set out of the total number of ways you can select $k + K$ words out of $L + l$:

$$\frac{\binom{l}{k} \binom{L}{K}}{\binom{l+L}{k+K}}.$$

With this probability, we can calculate final p-value as above: by summing these probabilities for all cases more extreme than the observed one, i.e., the cases with k or more words in the foreground set if you are testing for over-representation.

So how to choose cut-off of foreground and background?



Instead of selecting a positive set (and a negative set) arbitrarily, we can look for words that correlate with expression.

In the Mir-122 transfection experiment, we

- 1 Sort gene in order of overexpression
- 2 Calculate the Fisher p-value for any possible cut-off (or at least a large number of cut-offs).
- 3 Above the cut-off are used as the foreground set and those below the cut-off are used as background.

- For each word, plot the p-value (or its negative log). We can find interesting word and which genes are most strongly affected (for this case that would be ≤ 7000).

Weight matrix based methods

Gibbs Sampling

Steps:

- Select a random position as the start of motif at each sequence, say a .
- Single out a random sequence $X^r = x_1, \dots, x_L$, at position a^r we assume have a motif
- We calculate a PSSM as usual from all sequences *except* X^r :

Let p_i be the frequency of letters in position i of the **motifs** and q is the frequency of the letters **outside the motifs**.

$$p_i(b) = \frac{c_i(b) + \alpha(b)}{\sum_{b'} [c_i(b') + \alpha(b')]}, \quad q(b) = \frac{C(b) + \alpha(b)}{\sum_{b'} [C(b') + \alpha(b')]},$$

where $c_i(b)$ and $C(b)$ are the count of letter b at position i in the motif and outside motifs, respectively.

We define the PSSM W :

$$W(i, b) = \log \frac{p_i(b)}{q(b)}.$$

and the score of the motif:

$$S_W(X, a) = \log \frac{p_1(x_a) \dots p_n(x_{a+n-1})}{q(x_a) \dots q(x_{a+n-1})}.$$

So the probability of a sequence X have a **motif** is:

$$\begin{aligned} P(X|W, a) &= q(x_1) \dots q(x_{a-1}) p_1(x_a) \dots p_n(x_{a+n-1}) q(x_{a+n}) \dots q(x_L) \\ &= q(x_1) \dots q(x_{a-1}) q(x_{a+n}) \dots q(x_L) \times \frac{p_1(x_a) \dots p_n(x_{a+n-1})}{q(x_a) \dots q(x_{a+n-1})} \\ &\quad \times q(x_a) \dots q(x_{a+n-1}) \\ &= P(X) \times \frac{p_1(x_a) \dots p_n(x_{a+n-1})}{q(x_a) \dots q(x_{a+n-1})} \\ &= P(X) \exp(S_W(X, a)). \end{aligned}$$

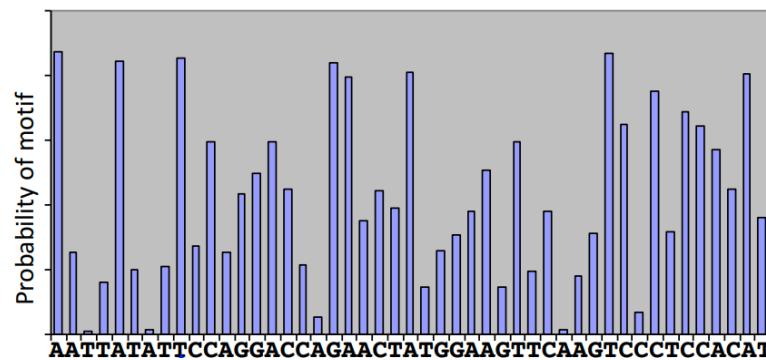
Now the probability that the motif occurs at position a in the sequence X is:

$$P(a|W, X) = \frac{P(X|W, a) \times P(a)}{P(W, X)} = \frac{P(X|W, a) \times P(a)}{\sum_i P(X|W, i) P(i)} = \frac{\exp(S_W(X, a))}{\sum_i \exp(S_W(X, i))},$$

we assume the probability of start a motif in position a is uniform (the same as $P(i)$), so it can be eliminated.

CGGTTT
AGACAG
AAGGTG
GTCATC
ATTTCG
(seq. r left out)
CTGCCA
CCGGGT
CGTGGA
AAATTG
GAATTT
CTACAT
TAGCCC
TCCTCT
TTAAGA
AGATTT
CTATTA
CACCAA

- Now we can calculate the probability of each possible position having motif in test sequence X^r .



- Sample a new position a^r according to probability above in X^r .
- Stop when 'converged'.

Problem:

- Stuck in shifted patterns. Example: true motif **ACGATG**, stuck in **ATGXXX**.
Solution: check if shift (move) to right or left some position has a higher probability.
- Find the right width of PSSM.

Profile HMM

Stuck in a local maximum of the likelihood.

MEME

Similar to HMM, but

- Seeds training from all possible words
- Allows for more than one site per sequence
- Looks for more than one pattern
- Probably the most used pattern discovery tool

Applications on HMMs: Gene finding

Prokaryotes

ORF is a region that starts with a start codon and ends with a stop codon.

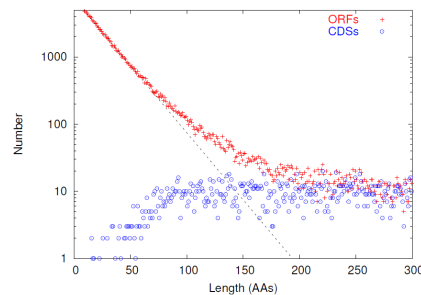
How do we discriminate real coding ORFs from those ORFs that just appear randomly anywhere in the DNA sequence?

- Non-coding ORF is short. Therefore, very long ORFs are usually genes.
- Random ORFs are different from true gene: Codon usage statistics (e.g.

log-odds)

Codon	Amino Acid	Frequency (%)	Expectation (%)	Log odds (bits)
AAA	K (Lys)	0.78	1.06	-0.4497
AAG	K (Lys)	3.14	1.36	1.2106
AAC	N (Asn)	1.63	1.36	0.2637
AAT	N (Asn)	0.39	1.06	-1.4375
AGA	R (Arg)	1.08	1.36	-0.3296
AGG	R (Arg)	5.05	1.72	1.5499

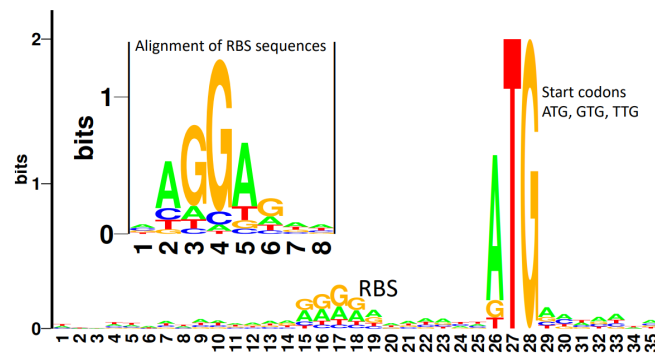
- Length distribution is different:



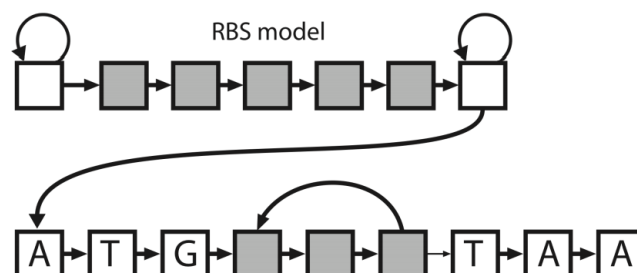
Ribosome binding site scoring

The ribosome binds to the ribosome binding site (RBS) just upstream of the beginning of the coding region on the mRNA.

In picture below, the RBS in a region of low conservation about 10 bp upstream of the start codon. The sequences were aligned at the start codon (as you can see in the logo), and because the distance between the RBS and the start codon vary, the RBS's are **not correctly aligned**. After alignment of the RBSs, the logo looks like the middle logo in the figure.



Combine in HMM



Order 1 HMM for probability of a sequence, capture dinucleotide preferences.

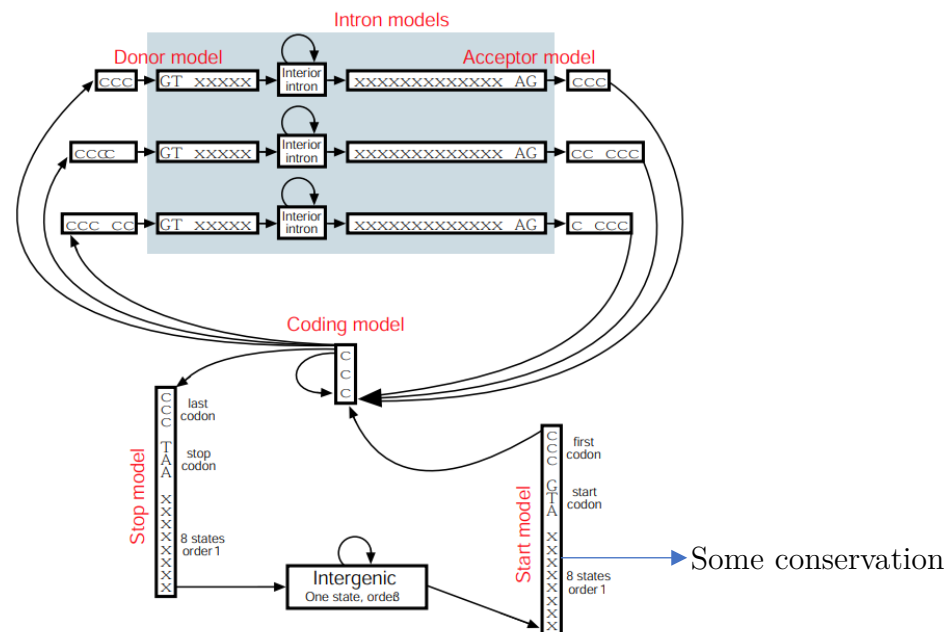
$$P(ACGTGTC \dots) = p_1(A)p_2(C|A)p_3(T|C)p_4(G|T)p_5(T|G)p_6(C|T) \dots$$

Order 2 HMM for codon structure:

$$p(*|CA) = \frac{c(CA*)}{c(CAA) + c(CAT) + c(CAC) + c(CAG)}$$

Collect sets of coding gene, train HMM, viterbi.

Eukaryotes



Other examples

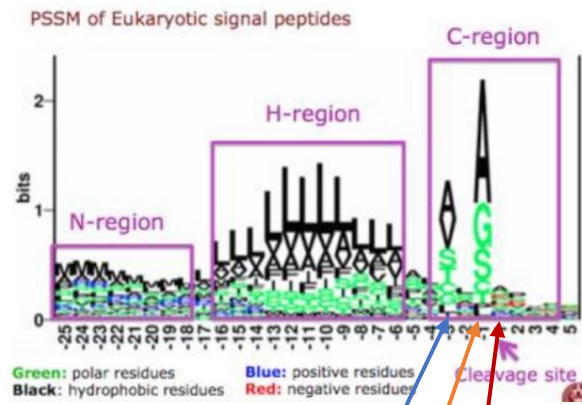
Signal peptides

A signal peptide tells the cell where the protein should go. It has a tripartite structure:

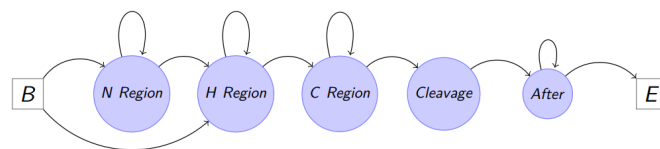
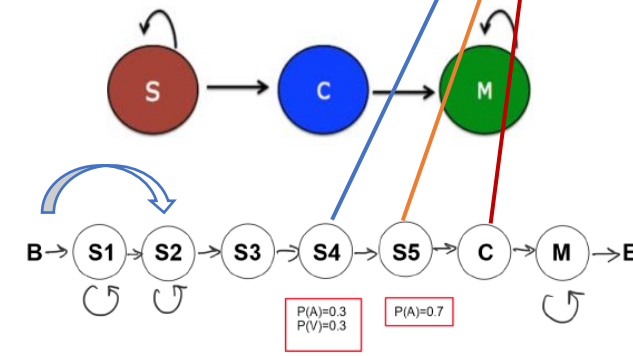
- Charged N-region: short stretch of positively charged aminoacids (not always present)
- Hydrophobic H-region: long hydrophobic core region
- C-region: contains the signal peptidase consensus cleavage site.



Visualization:



Improvement:



Membrane proteins

