The Abdus Salam
International Centre
for Theoretical Physics
50th Anniversary 1964-2014

2584-3

Spring College on the Physics of Complex Systems

26 May - 20 June, 2014

Sequence Alignments

Martin Weigt
Université Pierre et Marie Curie
Paris
France

Sequence alignments

Aim: Compare two AA sequences

$(a_1, \ldots, a_{L_a})$

$(b_1, \ldots, b_{L_b})$

If sufficiently similar:

- common evolutionary origin
- closely related 3D protein structure and function.

What do we need:

a) a similarity matrix for amino acids

→ based on physico-chemical properties

or

→ based on pre-existing curated alignments

and/or codon accessibility

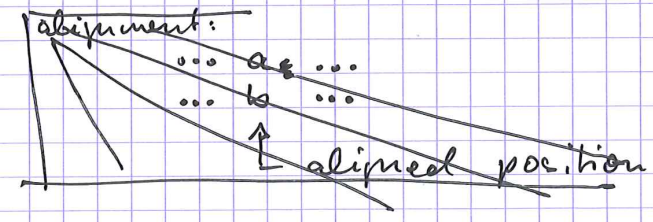
b) a penalty for introducing a score

Why: a → substitutions: one AA exchanged by another

b → inserts / deletions of AAs into sequence.

⇒ reflects evolutionary processes.

a) Scoring amino-acid matches



Ros 2.2

Consider ungapped alignment

$\underline{a} = (a_1, \dots, a_L)$
$\underline{b} = (b_1, \dots, b_L)$ } $a_i, b_i \in \underbrace{\mathcal{A} \setminus \{-\}}_{\text{only AAs}}$

Like in last lecture, we assume positions to be independent. Also like in last lecture, we compare a random null model (R) with a "good" model, here called match model (M)

$$P(\underline{a}, \underline{b} \mid R) = \prod_{i=1}^{L} \underbrace{q(a) \cdot q(b)}_{\text{basically our null model.}}$$

$$P(\underline{a}, \underline{b} \mid M) = \prod_{i=1}^{L} \underbrace{p(a,b)}_{\text{joint appearance of } a,b \text{ in matched alignment positions.}}$$

Score = log-odds ration

$$S = \log \frac{P(\underline{a}, \underline{b} \mid M)}{P(\underline{a}, \underline{b} \mid R)}$$

$$= \sum_{i=1}^{L} s(a_i, b_i)$$

with

$$s(a,b) = \log \frac{p(a,b)}{q(a) \cdot q(b)}$$ } score matrix / substitution matrix

⟶ { PAM matrix
BLOSUM matrices } ⇒ assume for a moment to be given.

6) Gap penalties

.. a_i a_{i+1} .. a_{i+k} a_{i+k+1} ...

.. b_i $-$ $-$ ... $-$ b_{i+k+1}

gap of length k

→ two penalties
- gap-open penalty d
- gap-extension penalty e

→ contribution to score is negative

$$\gamma(k) = -d - (k-1)e$$

In general: $0 < e \leq d$

easier to extend gap than to open a new one

→ penalizes less extended gap regions.

Total score = Sum of substitution scores for aligned AAs, and gap scores for gaps.

Assumption for following part:
- $s(a,b)$ known
- $d = e$ known (generalization easy)

Aliging two sequences

Needleman-Wunsch algorithm → global alignment

$(a_1, \dots, a_{L_a})$, $(b_1 \dots b_{L_b})$

AIM: Find alignment (insert gaps) such that Score $F(L_a, L_b)$ is maximized.

IDEA: Dynamic programming → iterative construction of $F(L_a, L_b)$ starting from $F(0,0)$

Assume, we have the best alignment for $(a_1, \dots, a_i)$, $(b_1, \dots, b_j)$ $\quad 0 \le i \le L_a$, $0 \le j \le L_b$

$\Rightarrow$ score $F(i,j)$

a
b

→ extension by one position

(a) a_{i+1}, b_{j+1} $\quad (i+1, j+1)$ → $F(i,j) + S(a_{i+1}, b_{j+1})$

a_{i+1}, – $\quad (i+1, j)$ → $F(i,j) - d$

–, b_{j+1} $\quad (i, j+1)$ → $F(i,j) - d$

$\Rightarrow$ to arrive at (i,j) we could start at

$(i-1, j-1)$ → add AA match (a_i, b_j)

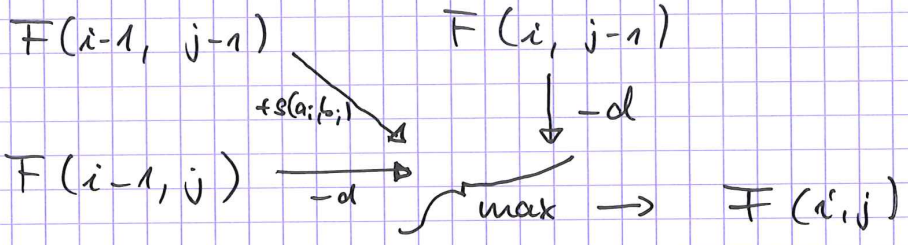
$(i-1, j)$ → add $(a_i, -)$

$(i, j-1)$ → add $(-, b_j)$

$\Rightarrow$ select the one of highest score

Ros 2.5

Schematically



$$\Rightarrow \quad F(i,j) = \max \begin{cases} F(i-1,j-1) + s(a_i, b_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Initial condition: $F(0,0) = 0$.

Let us try a simple example

Task: align (A, T) and (A)

with scores $s(a,b) = \begin{cases} +1 & \text{if } a = b \\ -1 & \text{if } a \neq b \end{cases}$

$d = 2$

	·	A	T
·	0	→ -2	→ -2
	↓	↘	
A	-2	+1	← -1

$$F(1,1) = \max \begin{pmatrix} 0+1 \\ -2-2 \\ -2-2 \end{pmatrix} = +1$$

$$F(2,1) = \max \begin{Bmatrix} -2-1 \\ -2-2 \\ +1-2 \end{Bmatrix} = -1$$

⇒ We may fill a table of size $(L_a+1)\times(L_b+1)$

⇒ $F(L_a, L_b)$ is the final score:

But how to construct the corresponding alignment?

⇒ traceback the path which actually brought you to (L_a, L_b)

⇒
A T
A –
✓

⇒ Need not only to remember the $F(i,j)$, but also which extension realized the max score until $F(i,j)$.

Local alignment: Smith-Waterman algorithm

⇒ allows for ending alignment, starting new one, if scores become negative.

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1, j-1) + s(a_i, b_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

!

⇒ start traceback at largest entry of F table.
traceback until you hit $F=0$.

Deriving alignment scores

Scores are crucial for the performance of the alignment algorithm.

Naive guess: Take large number of aligned sequences
→ count, how frequently you find AA pairs aligned, [~~numbers of gaps~~] in aligned ungapped sequences.
→ p_{ab}, p_a, q_b as normalized frequencies

$$S(a,b) = \ln \frac{p(a,b)}{q(a) \cdot q(b)}$$

BUT:
- NOT a good sample:
  - Sequences come in families
    → dependencies
  - Sequences have variable divergence times in evolution.
    → imagine two closely related sequences
    → most $a = b$
    → $S(a,b) < 0$ for all $a \neq b$.
    just due to short divergence time.

BLOSUM matrices (Henikoff & Henikoff '92)

- Start from number of sequence alignments.
- Cluster sequences at L% of identical amino-acids ($L = 50, 62$ used in practice)

→ clusters $\mathcal{C}_i$, $i = 1 \dots n_{clusters}$

$$A(a,b) = \underbrace{\sum_{\mathcal{C}_i \neq \mathcal{C}_j}}_{\text{all cluster pairs}} \frac{N_{ij}(a,b)}{\|\mathcal{C}_i\| \cdot \|\mathcal{C}_j\|}$$

$N_{ij}(a,b)$ = number of aligned pairs (a,b), with first sequence in cluster $\mathcal{C}_i$, second in cluster $\mathcal{C}_j$

$\|\mathcal{C}_i\|$ = number of seqs in $\mathcal{C}_i$

Two advantages:

- do not ~~compose~~ count too similar sequences → sufficiently divergent seqs.
- reweighting with cluster number → avoid oversampling of regions with many sequences.

$$\Rightarrow \boxed{p(a,b) = \frac{A(a,b)}{\sum_{c,d} A(c,d)} \qquad q(a) = \sum_b p(a,b) \qquad S(a,b) = \log \frac{p(a,b)}{q(a)\,q(b)}}$$

BLAST Basic local alignment search tool.

⇒ heuristic for searching in large databases

Idea LCTTFK } ∘ cut sequence in words of length k

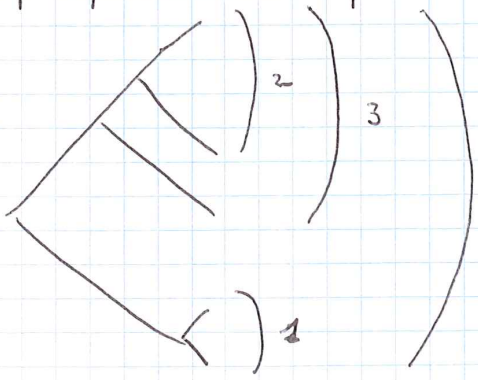
(k=3 proteins,
k=11 DNA, RNA)

- construct list of words of length k with high alignment scores (ex. ICT and LCT for LCT ...)
- search matches in database
- use matches as seeds for search with Smith-Waterman

Multiple-sequence alignment (MSA)

Progressive methods: (Clustal W) M seqs.

- calculate $\binom{M}{2}$ pairwise alignments
- organize Seqs in tree
- progressive alignment of subtrees



Problem: relative alignment of early aligned sequences never corrected

Iterative refinement (MAFFT, Muscle)

- take out one sequence, realign against profil
- iterate.

→ local max. of alignment score