Multiple Sequence Alignment

From Chapter 6 of the Sung book

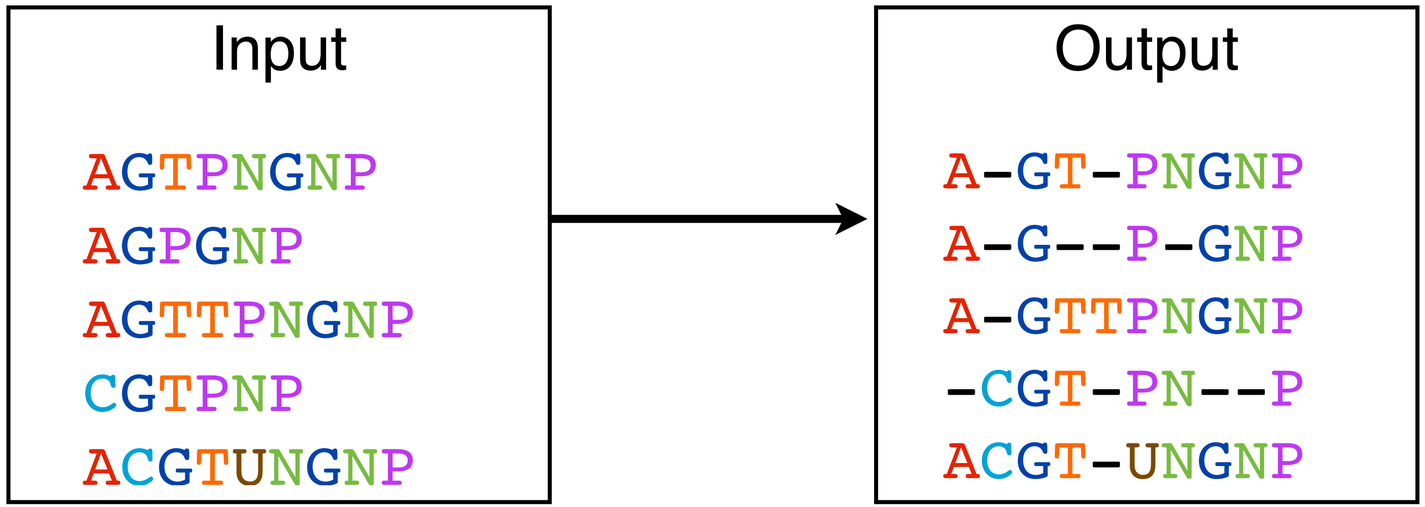
Multiple Sequence Alignment Problem

Given

- A set of sequences $s_1, s_2, \ldots, s_k$ (of length n)
- An objective function

Find:

- an ℓ by k matrix ($\ell \geq n$)
- where row i contains the characters from sequence s_i in order with inserted gap characters
- that is optimal under the objective function.



Multiple Sequence Alignment

Whats the objective function:

- most popular -- **Sum-of-Pairs *Objective*:**
  - given some scoring function for a pairwise alignment *PairScore(s_1',s_2')* the score of the multiple alignment is:

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \leq i < j \leq k} PairScore(s'_i, s'_j)$$

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

```
s'1  A-GT-PNGNP
s'2  A-G--P-GNP
s'3  A-GTTPNCNP
s'4  -CGT-PN--P
```

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

40

s'_1 A GT PNGNP

s'_2 A G- P-GNP

$PairScore(s'_1, s'_2) = 10 \times$

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

40 **62**

s'_1 A GT-PNGNP

$$PairScore(s'_1, s'_3) = 10 \times$$

s'_3 A GTTPNCNP

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

40 **62** **34**

s'_1 A-GT PNGNP

$$PairScore(s'_1, s'_4) = 10 \times$$

s'_4 -CGT PN--P

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

40 62 34 36

s'2 A G--P-GNP

s'3 A GTTPNCNP

$PairScore(s'_2, s'_3) = 10 \times$

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

40 62 34 36 4

s'_2 A–G– P–GNP

s'_4 –CGT PN––P

$PairScore(s'_2, s'_4) = 10 \times$

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4)$$

40 **62** **34** **36** **4** **29**

$$PairScore(s'_3, s'_4) = 10 \times$$

```
s'3  A-GTTPNCNP
s'4  -CGT-PN--P
```

$$SPScore(\{s'_1, s'_2, \ldots, s'_k\}) := \sum_{1 \le i < j \le k} PairScore(s'_i, s'_j)$$

$$PairScore(s'_i, s'_j) := 10\mathbf{mt} - 3\mathbf{ms} - 1\mathbf{id} - 4\mathbf{gp}$$

$$PairScore(s'_1, s'_2) + PairScore(s'_1, s'_3) + PairScore(s'_1, s'_4) + PairScore(s'_2, s'_3) + PairScore(s'_2, s'_4) + PairScore(s'_3, s'_4) \quad = 205$$

40 62 34 36 4 29

```
s'1  A-GT-PNGNP
s'2  A-G--P-GNP
s'3  A-GTTPNCNP
s'4  -CGT-PN--P
```

Finding an optimal MSA

Can we find an optimal multiple sequence alignment?

- yes! we can use the same dynamic programming methods we had for pairwise alignment
- assume there are only 3 sequences, then the recursion is the following:

$$V[i,j,k] = \max \begin{cases} V[i-1,j-1,k-1] & +\delta(s_1[i], s_2[j]) + \delta(s_2[j], s_3[k]) + \delta(s_1[i], s_3[k]) \\ V[i-1,j-1,k] & +\delta(s_1[i], s_2[j]) + \delta(s_2[j],'-') + \delta(s_1[i],'-') \\ V[i-1,j,k-1] & +\delta(s_1[i],'-') + \delta(s_2[j], s_3[k]) + \\ V[i,j-1,k-1] & +\delta('-', s_2[j]) + \delta(s_2[j], s_3[k]) + \\ V[i-1,j,k] & +2\delta(s_1[i],'-') \\ V[i,j-1,k] & +2\delta(s_2[j],'-') \\ V[i,j,k-1] & +2\delta(s_3[k],'-') \end{cases}$$

O(k²2ᵏnᵏ)-time!!

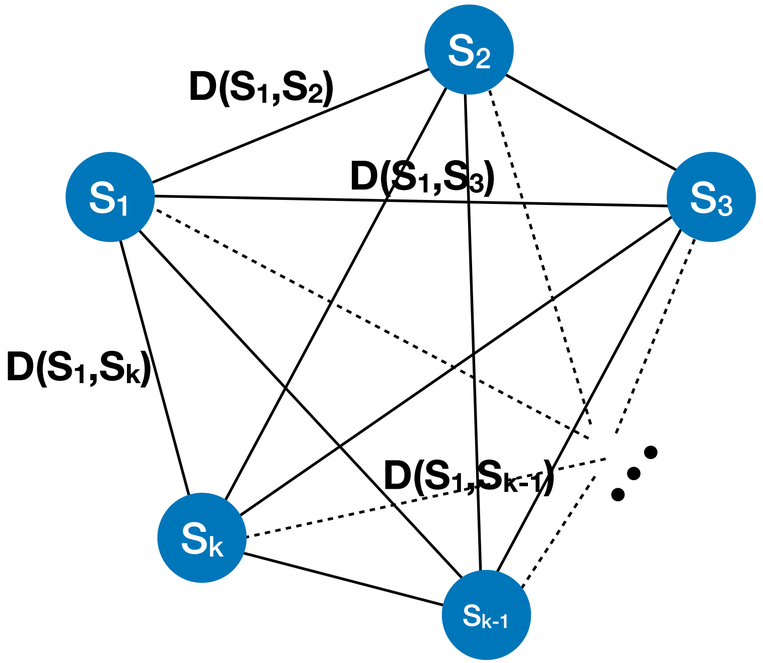
What happens with 4 sequences? How many clauses are in the max? How big is V

Finding an optimal MSA

Kececioglu (1993)* showed that the problem of finding multiple sequence alignments under any standard formulation is **NP-Hard**!

*This was somewhat concurrent with Wang and Jiang (1994) both for constant gap costs, but later work by Kececioglu and Sarrett showed that this is also true for linear (affine) gap costs.

The Center Star Method



$$S_c = \textbf{arg} \min_{1 \le i \le k} \left\{ \sum_{1 \le j \le k} D(S_i, S_j) \right\}$$

The final step is to build an alignment so that all of the alignments between S_c and S_i are satisfied.

The Center Star Method

S_1: CCTGCTGCAG
S_2: GATGTGCCG
S_3: GATGTGCAG
S_4: CCGCTAGCAG
S_5: CCTGTAGG

mismatch and indel penalties = 1

The Center Star Method

Assume all sequences are length n

- computing all of the pairwise scores is $O(k^2n^2)$-time ($O(n^2)$ each)
- finding S_c takes $O(k^2)$-time
- finding the pairwise alignments takes $O(kn^2)$-time
- inserting additional gaps for each sequence takes $O(k^2n)$-time.

The Center Star Method

$$\sum_{1 \le i < j \le k} d_{M^*}(i,j) \ge \sum_{1 \le i < j \le k} D(S_i, S_j)$$

How much worse than the optimal alignment is the multiple alignment produced using the center star method?

Assume the distances (alignment scores) satisfy the triangle inequality:

- given an MSA *M*, let $d_M(X,Y)$ be the alignment score for the induced pairwise alignment between *X* and *Y* in *M*.
- the triangle inequality says: $d_M(X,Y) \le d_M(X,Z) + d_M(Z,Y)$

$$\frac{\sum_{1 \le i < j \le k} d_M(i,j)}{\sum_{1 \le i < j \le k} d_{M^*}(i,j)} \le 2$$

$$\sum_{1 \le i < j \le k} d_M(i,j) = \frac{1}{2} \sum_{1 \le i \le k} \sum_{1 \le j \le k} d_M(i,j)$$

Let *M* be the alignment computed by the center star method and *M** be the optimal alignment.

Progressive Alignment

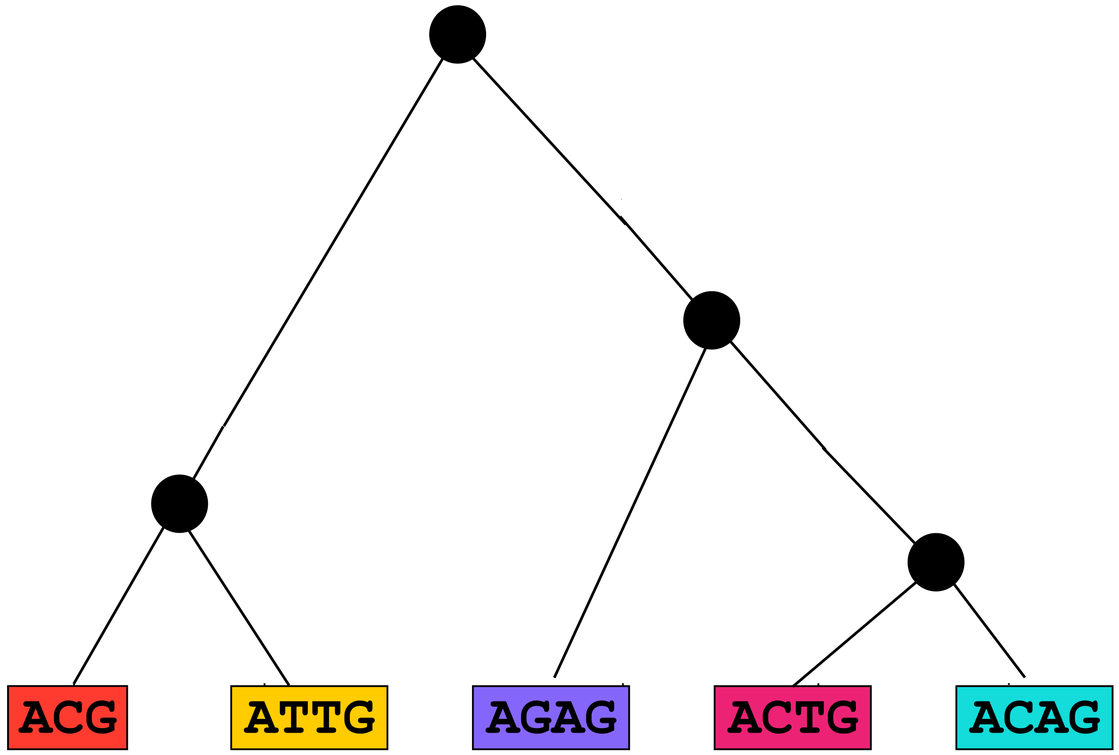
Similar to center star in that we use pairwise alignments to help build multiple alignments.

Introduced by Feng and Doolittle in 1987.

Basic idea:

- compute pairwise alignment scores for each pair of sequences
- generate a **guide tree** which ensures similar sequences are near to each other
- align sequences (or groups) one-by-one from the leaves of the tree

Progressive Alignment



ATTG	0.2			
AGAG	0.3	0.3		
ACTG	0.5	0.3	0.2	
ACAG	0.5	0.5	0.2	0.1
	ACG	ATTG	AGAG	ACTG

"Progressive alignment from consensus sequences"

[from Balcan, *et al.* 2019, arXiv:1908.02894]

ClustalW

Most used multiple sequence alignment tool, available in many web-based repositories.

Originally published by Higgins and Sharp (1987) and made widely available by Thompson, Higgins, and Gibson (1994).

ClustalW

Algorithm

- Calculate the $\binom{n}{2}$ pairwise alignments.
- Compute the pairwise distance between sequences as $1 - \frac{x}{y}$ where *x* is the number of gap characters, and *y* is the number of matches.
- Use the neighbor-joining method to create the guide tree (we will talk about the details of this later).
- From the leaves compute the alignment at each internal node
  - each alignment will be between either: (i) two sequences, (ii) two partial alignment, or (iii) a sequence and a partial alignment.

How do you align two alignments?

ClustalW

Profile-to-profile alignment:

- for an individual column define

$$PSP\left(A_1[i], A_2[j]\right) =: \sum_{x,y\in\Sigma} g_x^i g_y^j \delta(x,y)$$

- here g^i_x is the count of xs in column i of A_1 (similarly for g^j_y)
- then we can find the best (non-affine gap) global alignment using the following recurrence:

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + PSP(A_1[i], A_2[j]) \\ V(i-1,j) + PSP(A_1[i], -) \\ V(i,j-1) + PSP(-, A_2[j]) \end{cases}$$

Computing the g values can be done in a total of $O(k_1n_1)$ for A_1, and $O(k_2n_2)$ for A_2.

- Then computing table V takes $O(n_1n_2)$-time to compute.
- The total running time is then $O(k_1n_1+k_2n_2+n_1n_2) \in O(kn+n^2)$.

ClustalW

Align

```
PPGVKSEDCAS
PATGVKEDCAS
PPDGKSED--S


GATGKDCCS
GATGKDCAS
```

ClustalW

			G	A	T	G	K	D	C	C	S
			G	A	T	G	K	D	C	A	S
P	P	P									
P	A	P									
G	T	D									
V	G	G									
K	V	K									
S	K	S									
E	E	E									
D	D	D									
C	C	-									
A	A	-									
S	S	S									

ClustalW

Algorithm

- $O(k^2n^2)$ Calculate the $\binom{n}{2}$ pairwise alignments.
- $O(k^2)$ Compute the pairwise distance between sequences as $1 - \frac{x}{y}$ where *x* is the number of gap characters, and *y* is the number of matches.
- $O(k^3)$ Use the neighbor-joining method to create the guide tree (we will talk about the details of this later).
- From the leaves compute the alignment at each internal node $O(k^2n+kn^2)$
  - each alignment will be between either: (i) two sequences, (ii) two partial alignment, or (iii) a sequence and a partial alignment. $O(kn+n^2)$

$O(k^2n^2 + k^2 + k^3 + k^2n + kn^2) \in O(k^2n^2 + k^3)$

Issues with progressive alignment

"Once a gap, always a gap"

- Progressive alignment does not realign sequences, so once a gap is introduced at a certain position, it cannot be reconsidered.
- This means the order things are aligned can influence the alignment.

Iterative Methods

To overcome the issues with progressive alignment, some methods first create an initial multiple alignment, then continually break apart then re-align subsets of sequences to improve the alignment.

Popular examples are MAFFT (Katoh, 2002) and MUSCLE (Edgar, 2004).

MUSCLE
(MUltiple Sequence Comparison by Log-Expectation)

The Log-Expectation (LE) score replaces the PSP score in ClustalW.

Does not ignore spaces in the two profiles when aligning.

$$LE(A_1[i], A_2[j]) = (1 - f_G^i)(1 - f_G^j)\log \sum_{x,y \in \Sigma} f_x^i f_y^j$$

- here f^i_G is the frequency of gaps in column i of A_1 (similarly for f^j_G).
- f^i_x and f^j_y are the *normalized* frequency of other characters:

$$f_x^i =: \frac{g_x^i}{\left(\sum_{x' \in \Sigma} g_{x'}^j \right)}$$

MUSCLE
(MUltiple Sequence Comparison by Log-Expectation)

Algorithm:

1. **draft progressive alignment** -- similar to ClustalW but with
   - LE score for aligning profiles,
   - a more efficient tree building algorithm, and
   - a more efficient pairwise comparison (using *k*-mer counting).
2. **improved progressive alignment** -- using the alignment from (1)
   - redefine the pairwise distances using the Kimura distance $-\ln\left(1 - D - \frac{D^2}{5}\right)$
   - *D* is the fraction of matches.
   - re-align.
3. **refinement** -- deleting an edge in the guide tree creates two sub-groups of sequences with induced sub-alignments.
   - Extract those two sub-alignments and realign them.
   - Only keep the new alignment if the *SP* score is increased.
   - Stop when SP has not improved: in a predefined number of iterations or when all edges are visited.

So many aligners!!

- ClustalW
- ClustalΩ
- MAFFT
- MUSCLE
- PREFAB
- Opal
- Kalign
- T-Coffee
- MSA
- Dialign
- MACAW
- MUMMALS
- Prank
- Probalign
- Probcons
- POA
- SATé
- PASTA
- MSAProbs
- M-Coffee (meta-alignment)
- FastR (RNA)
- PMFastR (RNA)
- ...