

Multiple Sequence Alignment Sum-of-Pairs and ClustalW

Ulf Leser

## This Lecture

- Multiple Sequence Alignment
- The problem
- Theoretical approach: Sum-of-Pairs scores
- Practical approach: ClustalW


## Multiple Sequence Alignment

- We now align multiple ( $k>2$ ) sequences
- Note: Also BLAST aligns only two sequences
- Why?
- Imagine $k$ sequences of the promoter region of genes, all regulated by the same transcription factor $f$. Which subsequence within the $k$ sequences is recognized by $f$ ?
- Imagine $k$ sequences of proteins that bind to DNA. Which subsequence of the $k$ sequences code for the part of the proteins that performs the binding?
- General
- We want to know the common part(s) in $k$ sequences
- "common" does not mean identical
- This part can be anywhere within the sequences


## Definition

- Definition
- A multiple sequence alignment (MSA) von $k$ Strings $s_{i j} 1 \leq i \leq k$, is a table of $k$ rows and I columns ( $\mid \geq \max \left(\left|s_{k}\right|\right)$ ), such that
- Row i contains the sequence of $s_{i j}$ with an arbitrary number of blanks being inserted at arbitrary positions
- Every symbol of every $s_{i}$ stands in exactly one column
- No column contains only blanks

```
AACGTGATTGAC
TCGAGTGCTTTACAGT
GCCGTGCTAGTCG
TTCAGTGGACGTGGTA
GGTGCAGACC
```



## Good MSA

- We are searching for good (optimal) MSAs
- Defining „optimal" here is not as simple as in the $k=2$ case
- Intuition
- All sequences had a common ancestor and evolved by evolution
- We want to assume as few evolutionary events as possible
- Thus, we want few columns ( $\sim$ few INSDELs)
- Thus, we want homogeneous columns ( $\sim$ few replacements)



## This Lecture

- Multiple Sequence Alignment
- The problem
- Theoretical approach: Sum-of-Pairs scores
- Practical approach: ClustalW


## What Should we Count?

- For two sequences
- We scored each column using a scoring matrix
- Find the alignment such that the total score is maximal
- But - how do we score a column with $5^{*}$ T, $3^{*} \mathrm{~A}, 1^{*}$ _?
- We would need an exponentially big scoring matrix
- Alternative: Sum-of-Pairs Score
- We score an entire MSA
- We score the alignment of each pair of sequences in the usual way
- We aggregate over all pairs to score the MSA
- We need a clever algorithm to find the MSA with the best score


## Formally

- Definition
- Let $M$ be a MSA for the set $S$ of $k$ sequences $S=\left\{s_{1}, \ldots, s_{k}\right\}$
- The alignment of $s_{i}$ with $s_{j}$ induced by $M$ is generated as follows
- Remove from $M$ all rows except $i$ and $j$
- Remove all columns that contain only blanks
- The sum-of-pairs score (sop) of $M$ is the sum of all pair-wise induced alignment scores
- The optimal MSA for $S$ wrt. to sop is the MSA with the highest sopscore over all possible MSA for $S$


## Example



- Given a MSA over k sequences of length I - how complex is it to compute its sop-score?
- How do we find the best MSA?


## Analogy

- Think of the $\mathrm{k}=2$ case

|  |  | 0 | 1 | 2 | 3 | 4 | 5 | - | 7 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | w | r | i | † | e | r | S |
| 0 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | v | 1. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | i | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 7 |
| 3 | n | 3 | 3 | 3 | 3. | 3 | 4 | 5 | 6 |
| 4 | † | 4 | 4 | 4 | 4 | 3 | 4 | 5 | 6 |
| 5 | n | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 6 |
| ${ }^{6}$ | e | 6 | 6 | 6 | 6 | 5 | 4 | 5 | 6 |
| 7 | r | 7 | 7 | 6 | 7 | 6 | 5 | 4 | 5 |

- Every alignment is a path through the matrix
- The three possible directions (down, right, down-right) conform to the three possible constellations in a column (XX, X_, _X)
- With growing paths, we align growing prefixes of both sequences


## Analogy



## Dynamic Programming in two Dimensions

- We compute the best possible alignment $d(i, j)$ for every pair of prefixes (lengths $i, j$ ) using the following formula

$$
d(i, j)=\min \left\{\begin{array}{c}
d(i, j-1)+1 \\
d(i-1, j)+1 \\
d(i-1, j-1)+t(i, j)
\end{array}\right\} \begin{aligned}
& \text { Insertion in one seq } \\
& \text { Insertion in the other seq } \\
& \text { Match or mismatch }
\end{aligned}
$$

## Dynamic Programming in three Dimensions

- We compute the best possible alignment $d(i, j, k)$ for every triple of prefixes (lengths $i, j, k$ ) using the following formula



## All Possible Steps

- $d(i-1, j-1, k-1)$
- $d(i, j-1, k-1)$
- $d(i, j, k-1)$
- $d(i, j-1, k)$
- $d(i-1, j, k)$
- $d(i-1, j-1, k)$
- $d(i-1, j, k-1)$


| 1 | $\mathbf{G}$ |
| :--- | :--- |
| $\mathbf{2}$ | - |
| 3 | - |



| 1 | $G$ |
| :--- | :--- |
| 2 | - |
| 3 | $C$ |



## Concrete Examples



- Best sop-score for $\mathrm{d}(\mathrm{i}, \mathrm{j}-1, \mathrm{k})$ is known
- We want to compute $\mathrm{d}(\mathrm{i}, \mathrm{j}, \mathrm{k})$
- This requires to align one symbol with two blanks (blank/blank does not count)
- $d(i, j, k)=d(i, j-1, k)+2$

$$
d(i-1, j, k-1)
$$



- Best sop-score for $\mathrm{d}(\mathrm{i}-1, \mathrm{j}, \mathrm{k}-1)$ is known
- We want to compute $\mathrm{d}(\mathrm{i}, \mathrm{j}, \mathrm{k})$
- This requires aligning a blank with $\mathrm{s}_{1}[\mathrm{i}-1]$ and with $\mathrm{s}_{3}[\mathrm{k}-1]$ and to align $\mathrm{s}_{1}[\mathrm{i}-1]$ and $\mathrm{s}_{3}[\mathrm{k}-1]$
- $d(i, j, k)=d(i-1, j, k-1)+2+c_{i k}$


## Initialization

- We need to start somewhere
- Of course, we have $\mathrm{d}(0,0,0)=0$
- Aligning in one dimension: $\mathrm{d}(\mathrm{i}, 0,0)=2 * \mathrm{i}$
- Dito for $\mathrm{d}(0, \mathrm{j}, 0), \mathrm{d}(0,0, k)$
- Aligning in two dimensions: $\mathrm{d}(\mathrm{i}, \mathrm{j}, 0)=\ldots$
- Let $d_{a, b}(i, j)$ be the alignment score for $S_{a}[1 . . i]$ with $S_{b}[1 . . j]$
$-\mathrm{d}(\mathrm{i}, \mathrm{j}, 0)=\mathrm{d}_{1,2}(\mathrm{i}, \mathrm{j})+(\mathrm{i}+\mathrm{j})$
$-d(i, 0, k)=d_{1,3}(i, k)+(i+k)$
$-d(0, j, k)=d_{2,3}(j, k)+(j+k)$


## Algorithm

```
initialize matrix d;
for i := 1 to | | | |
    for j := 1 to |S |
        for k := 1 to |S |
```



```
            if (S'(i) = S S (k)) then coik := 0; else cik := 1;
            if (S}\mp@subsup{S}{2}{(j) = S (k)) then cojk := 0; else cojk := 1;
            d
            d
            d
            d
            d5 := d[i - 1,j,k) + 2;
            d
            d
            d[i,j,k] := min(di, d
        end for;
    end for;
end for;
```


## Bad News - Complexity

- For 3 sequences of length $n$, we have to perform
- There are $n^{3}$ cells in the cube
- For each cell (top-left-front corner), we need to look at 7 corners
- Together: O(7*n³) computations
- For $k$ sequences of length $n$
- There are $\mathrm{n}^{k}$ cell corners in the cube
- For each corner, we need to look at $2^{\mathrm{k}}-1$ other corners
- Together: $\mathrm{O}\left(2^{\mathrm{k}} * \mathrm{n}^{\mathrm{k}}\right)$ computations
- Actually, the problem is NP-complete


## This Lecture

- Multiple Sequence Alignment
- The problem
- Theoretical approach: Sum-of-Pairs scores
- Practical approach: ClustalW


## Scoring a MSA

- Let's take one step back
- What happened during evolution?

- Real number of events: 8
- sop-score: 2+3+6+6+2+...
- Everything is counted multiple times

CT_TGC_A
GT_TGACA
GT_TGTTA
GTATTTCT
GTATTTGA

## Different Scoring Function

- If we knew the phylogenetic tree of the $k$ sequences
- Align every parent with all its children
- Aggregate all alignment scores
- This gives the "real" number of evolutionary operations
- But: Finding the true phylogenetic tree requires a MSA
- Not be covered here
- Use a heuristic: ClustalW
- Higgins, D. G. and Sharp, P. M. (1988). "CLUSTAL: a package for performing multiple sequence alignment on a microcomputer." Gene 73(1): 237-44.
- Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994). "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." Nucleic Acids Res 22(22): 4673-80.


## ClustalW

- Main idea
- Compute a "good enough" phylogeny - the guide tree
- Use the guide tree to iteratively align small MSA to larger MSA
- Starting from single sequences
- To escape the curse of dimensionality, compute MSA iteratively
- "Progressive" MSA
- Does not necessarily find the best solution
- Needs a fast method to align two MSAs
- Works quite well in practice
- Many other, newer (better) proposals
- DAlign, T-Coffee, HMMT, PRRT, MULTALIGN, ...


## Step 1: Compute the Guide Tree

- Compute all pair-wise alignments and store in similarity matrix M
$-M[i, j]=\operatorname{sim}\left(s_{i}, s_{j}\right)$
- Compute the guide tree using hierarchical clustering
- Choose the smallest M[i,j]
- Let $s_{i}$ and $s_{j}$ form a new (next) branch of the tree
- Compute the distance from the ancestor of $s_{i}$ and $s_{j}$ to all other sequences as the average of the distances to $s_{i}$ and $s_{j}$
- Set $M^{\prime}=\mathrm{M}$
- Delete rows and columns i and j
- Add a new column and row (ij)
- For all $k \neq i j: M^{\prime}[i j, k]=(M[i, k]+M[j, k]) / 2$
- Iterate until $\mathrm{M}^{\prime}$ has only one column / row


## Sketch



## Example

|  | A | B | C | D | E |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A |  | 17 | 59 | 59 | 77 |
| B |  |  | 37 | 61 | 53 |
| C |  |  |  | 13 | 41 |
| D |  |  |  |  | 21 |



|  | A | B | E | CD |
| :--- | :--- | :--- | :--- | :--- |
| A |  | 17 | 77 | 59 |
| B |  |  | 53 | 49 |
| E |  |  |  | 31 |



|  | $E$ | $C D$ | $A B$ |
| :--- | :--- | :--- | :--- |
| $\mathbf{E}$ |  | 31 | 65 |
| $\mathbf{C D}$ |  |  | 54 |



## Example



## Step 2: Progressive MSA

- Pair-wise alignment in the order given by the guide tree
- Aligning a MSA $M_{1}$ with a MSA $M_{2}$
- Use the usual (global) alignment algorithm
- To score a column, compute the average score over all pairs of symbols in this columns
- Example

| A | $\ldots . \mathrm{P} . .$. |  |
| :--- | :--- | :---: |
| B | $\ldots . .$. | Score of this column |
| C | $\ldots . . .$. | $\left(2^{*} s(P, A)+s(P, Y)+\right.$ |
|  |  | $2 * s(G, A)+s(G, Y)+$ |
| D | $\ldots . . .$. | $2 * s(P, A)+s(P, Y)) / 9$ |
| E | $\ldots . . .$. |  |
| F | $\ldots . . .$. |  |

## Issues

- There is a lot to say about whether hierarchical clustering actually computes the "correct" tree
- See phylogenetic algorithms, „Algorithms in Bioinformatics"
- Clustal-W actually uses a different, more accurate algorithm called "neighbor-joining"
- Clustal-W is fast $\left(\mathrm{O}\left(\mathrm{k}^{2} * \log (\mathrm{k})\right)\right.$
- For $k$ sequences; plus cost for computing alignments, especially $M$
- Idea behind progressive alignment
- Find strong signals (highly conserved blocks) first
- Outliers are added last
- Increases the chances that conserved blocks survive
- Several improvements to this scheme are known


## Problems with progressive MSA



## Further Reading

- Merkl \& Waack, chapter 13
- Böckenhauer \& Bongartz, chapter 5.3

