Introduction to Bioinformatics

Lecture 3:
Sequence alignment

Sequence alignment

- *The biological problem*
- Global alignment
- Local alignment
- Multiple alignment

Background: comparative genomics

- Basic question in biology: *what properties are shared among organisms?*
- Genome sequencing allows comparison of organisms at DNA and protein levels
- Comparisons can be used to
  - Find evolutionary relationships between organisms
  - Identify functionally conserved sequences
  - Identify corresponding genes in human and model organisms: develop models for human diseases

Homologs

- Two genes (sequences in general) g_B and g_C evolved from the same ancestor gene g_A are called *homologs*
- Homologs usually exhibit conserved functions
- Close evolutionary relationship => expect a high number of homologs

g_A = agtgtccgttaagtgcgttc

g_B = agtgccgttaaagttgtacgtc

g_C = ctgactgtttgtggttc

Sequence similarity

- We expect homologs to be "similar" to each other
- Intuitively, similarity of two sequences refers to the degree of match between corresponding positions in sequence

agtgccgttaaagttgtacgtc

ctgactgtttgtggttc

- What about sequences that differ in length?

Similarity vs homology

- Sequence similarity is not sequence homology
  - If the two sequences g_B and g_C have accumulated enough mutations, the similarity between them is likely to be low

#mutations		#mutations	
0	agtgtccgttaagtgcgttc	64	acagtccgttcgggctattg
1	agtgtccgttatagtgcgttc	128	cagagcactaccgc
2	agtgtccgcttatagtgcgttc	256	cacgagtaagatatagct
4	agtgtccgcttaagggcgttc	512	taatcgtgata
8	agtgtccgcttcaaggggcgt	1024	acccttatctacttcctggagtt
16	gggccgttcatgggggt	2048	agcgacctgcccaa
32	gcagggcgtcactgagggct	4096	caaac

Homology is more difficult to detect over greater evolutionary distances.

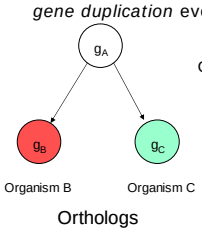
Similarity vs homology (2)

- Sequence similarity can occur by chance
  - *Similarity does not imply homology*
- Consider comparing two short sequences against each other

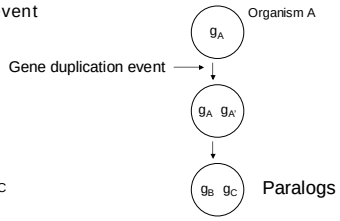
168

Orthologs and paralogs

- We distinguish between two types of homology
  - Orthologs: homologs from two different species, separated by a *speciation* event
  - Paralogs: homologs within a species, separated by a *gene duplication* event



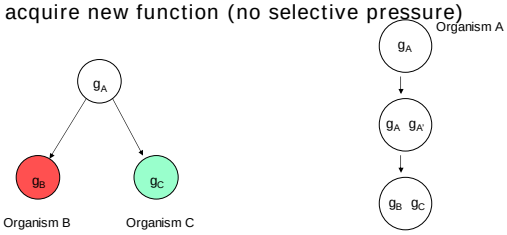
Orthologs



169

Orthologs and paralogs (2)

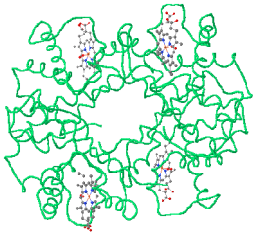
- Orthologs typically retain the original function
- In paralogs, one copy is free to mutate and acquire new function (no selective pressure)



170

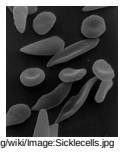
Paralogy example: hemoglobin

- Hemoglobin is a protein complex which transports oxygen
- In humans, hemoglobin consists of four protein subunits and four non-protein heme groups



Hemoglobin A, www.rcsb.org/pdb/explore.do?structureId=1GZX

Sickle cell diseases are caused by mutations in hemoglobin genes

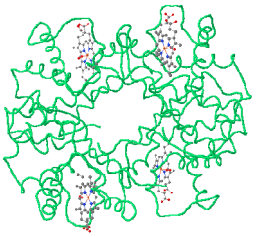


http://en.wikipedia.org/wiki/Image:Sicklecells.jpg

171

Paralogy example: hemoglobin

- In adults, three types are normally present
  - Hemoglobin A: 2 alpha and 2 beta subunits
  - Hemoglobin A2: 2 alpha and 2 delta subunits
  - Hemoglobin F: 2 alpha and 2 gamma subunits
- Each type of subunit (alpha, beta, gamma, delta) is encoded by a separate gene

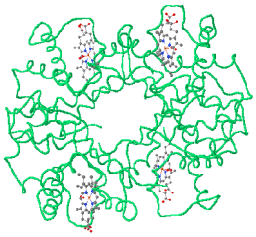


Hemoglobin A, www.rcsb.org/pdb/explore.do?structureId=1GZX

172

Paralogy example: hemoglobin

- The subunit genes are paralogs of each other, i.e., they have a common ancestor gene
- Exercise: hemoglobin human paralogs in NCBI sequence databases
  http://www.ncbi.nlm.nih.gov/sites/entrez?db=Nucleotide
  - Find human hemoglobin alpha, beta, gamma and delta
  - Compare sequences



Hemoglobin A, www.rcsb.org/pdb/explore.do?structureId=1GZX

173

Orthology example: insulin

- The genes coding for insulin in human (*Homo sapiens*) and mouse (*Mus musculus*) are orthologs:
  - They have a common ancestor gene in the ancestor species of human and mouse
  - Exercise: find insulin orthologs from human and mouse in NCBI sequence databases

174

Sequence alignment

- Alignment specifies which positions in two sequences match

```
acgtctag   acgtctag    acgtctag
||           |||||     || |||||
actctag-   -actctag    ac-tctag
```

2 matches	5 matches	7 matches
5 mismatches	2 mismatches	0 mismatches
1 not aligned	1 not aligned	1 not aligned

175

Sequence alignment

- Maximum alignment length is the total length of the two sequences

```
acgtctag-------   -------acgtctag

--------actctag   actctag--------
```

0 matches	0 matches
0 mismatches	0 mismatches
15 not aligned	15 not aligned

176

Mutations: Insertions, deletions and substitutions

Indel: insertion or deletion of a base with respect to the ancestor sequence

```
acgtctag
 |  |||||
-actctag
```

Mismatch: substitution (point mutation) of a single base

- Insertions and/or deletions are called *indels*
  - *We can't tell whether the ancestor sequence had a base or not at indel position!*

177

Problems

- What sorts of alignments should be considered?
- How to score alignments?
- How to find optimal or good scoring alignments?
- How to evaluate the statistical significance of scores?

In this course, we discuss each of these problems briefly.

178

Sequence Alignment (chapter 6)

- The biological problem
- *Global alignment*
- Local alignment
- Multiple alignment

179

Global alignment

- Problem: find optimal scoring alignment between two sequences (Needleman & Wunsch 1970)
- Every position in both sequences is included in the alignment
- We give score for each position in alignment
  - Identity (match) +1
  - Substitution (mismatch) -μ
  - Indel -δ
- Total score: sum of position scores

180

Scoring: Toy example

- Consider two sequences with characters drawn from the English language alphabet: WHAT, WHY

```
WHAT
||
WH-Y
```

S(WHAT/WH-Y) = 1 + 1 – δ – μ

```
WHAT
-WHY
```

S(WHAT/-WHY) = – δ – μ – μ – μ

181

Dynamic programming

- How to find the optimal alignment?
- We use previous solutions for optimal alignments of smaller subsequences
- This general approach is known as dynamic programming

182

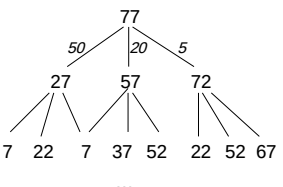
Introduction to dynamic programming: the money change problem

- Suppose you buy a pen for 4.23€ and pay for with a 5€ note
- You get 77 cents in change – what coins is the cashier going to give you if he or she tries to minimise the number of coins?
- The usual algorithm: start with largest coin (denominator), proceed to smaller coins until no change is left:
  - 50, 20, 5 and 2 cents
- This greedy algorithm is *incorrect*, in the sense that it does not always give you the correct answer

183

The money change problem

- How else to compute the change?
- We could consider all possible ways to reduce the amount of change
- Suppose we have 77 cents change, and the following coins: 50, 20, 5 cents
- We can compute the change with *recursion*
  - *C(n) = min { C(n – 50) + 1, C(n – 20) + 1, C(n – 5) + 1 }*
- Figure shows the recursion tree for the example



- Many values are computed more than once!
- This leads to a correct but very inefficient algorithm

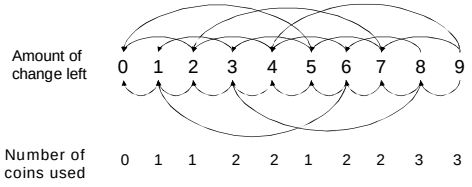
184

The money change problem

- We can speed the computation up by solving the change problem for all i ≤ n
  - Example: solve the problem for 9 cents with available coins being 1, 2 and 5 cents
- Solve the problem in steps, first for 1 cent, then 2 cents, and so on
- In each step, utilise the solutions from the previous steps

185

The money change problem



- Algorithm runs in time proportional to Md, where M is the amount of change and d is the number of coin types
- The same technique of storing solutions of subproblems can be utilised in aligning sequences

186

Representing alignments and scores

Alignments can be represented in the following tabular form.

Each alignment corresponds to a path through the table.

```
WHAT
||
WH-Y
```

	-	W	H	A	T
-					
W					
H					
Y					

187

Representing alignments and scores

```
WH-AT
||
WHY--

WHAT---

----WHY
```

	-	W	H	A	T
-					
W					
H					
Y					

188

Representing alignments and scores

```
WHAT
||
WH-Y
```

Global alignment score $S_{3,4} = 2\text{-}\delta\text{-}\mu$

	-	W	H	A	T
-	0				
W		1			
H			2	2-δ	
Y					2-δ-μ

189

Filling the alignment matrix

	-	W	H	A	T
-					
W		Case 1		Case 2	
H		Case 3			
Y					

Consider the alignment process at shaded square.

Case 1. Align H against H (match)

Case 2. Align H in WHY against – (indel) in WHAT

Case 3. Align H in WHAT against – (indel) in WHY

190

Filling the alignment matrix (2)

	-	W	H	A	T
-					
W		Case 1		Case 2	
H		Case 3			
Y					

Scoring the alternatives.

Case 1. $S_{2,2} = S_{1,1} + s(2, 2)$

Case 2. $S_{2,2} = S_{1,2} - \delta$

Case 3. $S_{2,2} = S_{2,1} - \delta$

$s(i, j) = 1$ for matching positions,

$s(i, j) = -\mu$ for substitutions.

Choose the case (path) that yields the maximum score.

Keep track of path choices.

191

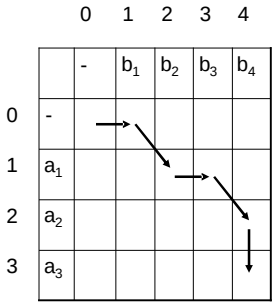
Global alignment: formal development

$A = a_1a_2a_3\ldots a_n$,
$B = b_1b_2b_3\ldots b_m$

b_1	b_2	b_3	b_4	-
-	a_1	-	a_2	a_3

- Any alignment can be written as a unique path through the matrix
- Score for aligning A and B up to positions i and j:

$S_{i,j} = S(a_1a_2a_3\ldots a_i, b_1b_2b_3\ldots b_j)$



Scoring partial alignments

- Alignment of $A = a_1a_2a_3\ldots a_i$ with $B = b_1b_2b_3\ldots b_j$ can be end in three possible ways
  - Case 1: $(a_1a_2\ldots a_{i-1})\ a_i$ / $(b_1b_2\ldots b_{j-1})\ b_j$
  - Case 2: $(a_1a_2\ldots a_{i-1})\ a_i$ / $(b_1b_2\ldots b_j)$ -
  - Case 3: $(a_1a_2\ldots a_i)$ – / $(b_1b_2\ldots b_{j-1})\ b_j$

Scoring alignments

- Scores for each case:
  - Case 1: $(a_1a_2\ldots a_{i-1})\ a_i$ / $(b_1b_2\ldots b_{j-1})\ b_j$
  - Case 2: $(a_1a_2\ldots a_{i-1})\ a_i$ / $(b_1b_2\ldots b_j)$ –
  - Case 3: $(a_1a_2\ldots a_i)$ – / $(b_1b_2\ldots b_{j-1})\ b_j$

$$s(a_i, b_j) = \begin{cases} +1 \text{ if } a_i = b_j \\ -\mu \text{ otherwise} \end{cases}$$

$s(a_i, -) = s(-, b_j) = -\delta$

Scoring alignments (2)

- First row and first column correspond to initial alignment against indels:
  - $S(i, 0) = -i\ \delta$
  - $S(0, j) = -j\ \delta$
- Optimal global alignment score $S(A, B) = S_{n,m}$

		0	1	2	3	4
		-	b_1	b_2	b_3	b_4
0	-	0	-δ	-2δ	-3δ	-4δ
1	a_1	-δ				
2	a_2	-2δ				
3	a_3	-3δ				

Algorithm for global alignment

```
Input sequences A, B, n = |A|, m = |B|
Set S_{i,0} := -δi for all i
Set S_{0,j} := -δj for all j
for i := 1 to n
  for j := 1 to m
    S_{i,j} := max{S_{i-1,j} – δ, S_{i-1,j-1} + s(a_i,b_j), S_{i,j-1} – δ}
  end
end
```

Algorithm takes O(nm) time

Global alignment: example

μ = 1
δ = 2

		-	T	G	G	T	G
	-	0	-2	-4	-6	-8	-10
	A	-2					
	T	-4					
	C	-6					
	G	-8					
	T	-10					?

Global alignment: example

μ = 1

δ = 2

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3			
T	-4					
C	-6					
G	-8					
T	-10					?

198

Global alignment: example (2)

μ = 1

δ = 2

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3	-5	-7	-9
T	-4	-1	-2	-4	-4	-6
C	-6	-3	-2	-3	-5	-5
G	-8	-5	-2	-1	-3	-4
T	-10	-7	-4	-3	0	-2

```
ATCGT-
|  ||
-TGGTG
```

199