Advanced Algorithms and Models for Computational Biology

-- a machine learning approach

Computational Genomics I: Sequence Alignment

Eric Xing

Lecture 3, January 25, 2005

Reading: Chap. 2,6 DEKM book

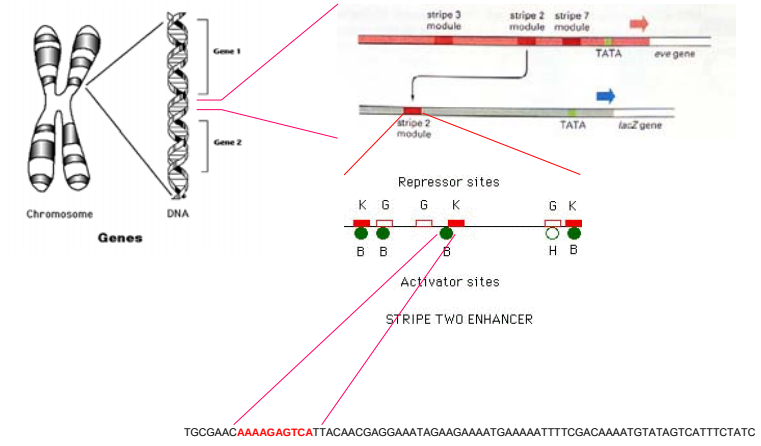
Modeling biological sequences

- Kinds of questions we want to ask
  - How to align two sequence to reveal conserved regions?
  - Is this sequence a motif (e.g., binding site, splice site)?
  - is this sequence part of the coding region of a gene?
  - Are these two sequences evolutionarily related?
  - ...
- What we will not address (covered last semester)
  - How multiple sequences can be optimally aligned
  - how sequencing results of a clone library can be assembled
  - What is the most parsimonious phylogeny of a set of sequences
- **Machine learning** : extracting useful information from a corpus of data *D* by building good (predictive, evaluative or decision) models

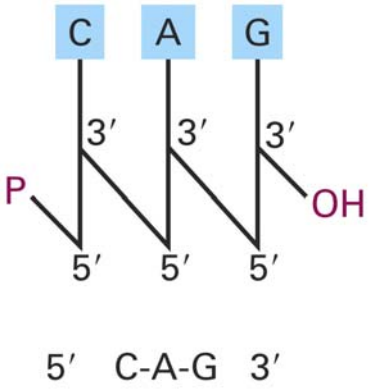
Modeling biological sequences, ctd

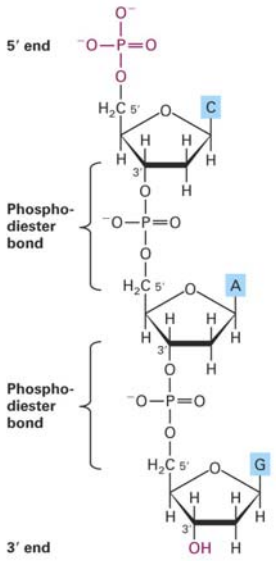
- We will use *probabilistic models* of sequences -- not the only approach, but usually the most powerful, because
  - sequences are the product of an evolutionary process which is stochastic in nature,
  - want to detect biological "signal" against "random noise" of background mutations,
  - data may be *missing* due to experimental reasons or intrinsically *unobservable*, and
  - we want to integrate multiple (heterogeneous) data and incorporate prior knowledge in a flexible and principled way,
  - ....
- Computational analysis only generate *hypothesis*, which must be tested by experiments
  - Site-directed mutagenesis (to alter the sequence content)
  - Knockouts/insertions of genes/sites (deletion/addition of elements)
  - Functional perturbations (pathway inhibitors, drugs, ...)
- From one-way learning to close-loop learning:
  - Active learning: can a machine design smart experiments?

Hierarchical structure of the genome



The DNA strand has a chemical polarity





Writing DNA sequence

- One strand is written by listing its bases in 5' to 3' order

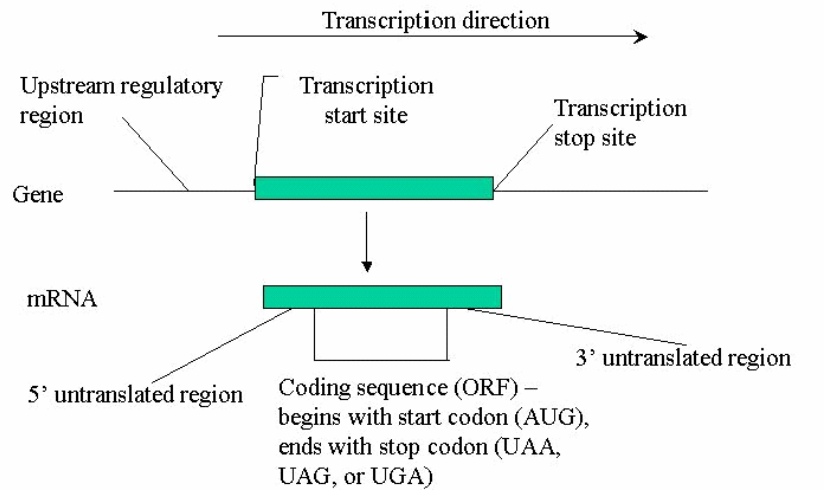
  5' ACCGTTACT 3'

- Each strand uniquely determines the complementary strand, which runs in the opposite direction:

  5' ACCGTTACT 3'
  3' TGGCAATGA 5'

- So the reverse complement of ACCGTTACT is written TGGCAATGA
- In general people write one strand and in 5' to 3' order
  - This is the ordering that a polymerase or a ribosome scan the sequence
  - Establishes a common standard for genome nomenclatures

Gene structure in prokaryotes

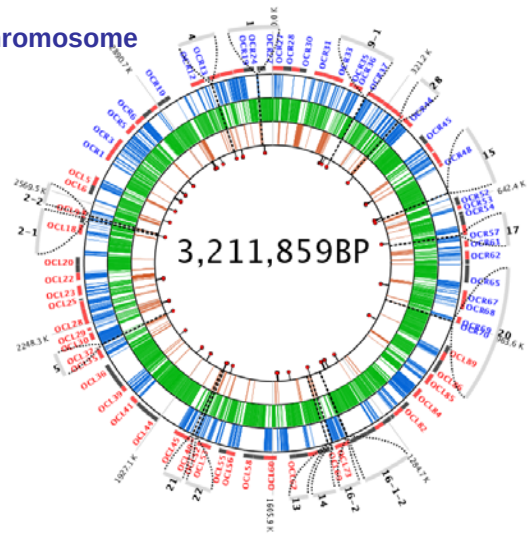


Gene structure in prokaryotes

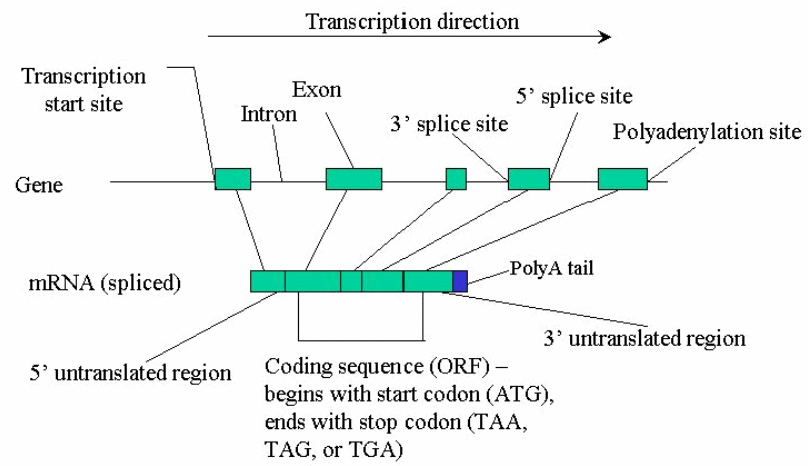
- **A protein-coding gene consists of the following, in 5' to 3' order**
  - An ***upstream regulatory region***, generally < 50 bp, which turns transcription on and off.
  - A ***transcription start site*** where RNA polymerase incorporates 1st nucleotide into nascent mRNA.
  - A 5' *untranslated region*, generally < 30bp, that is transcribed into mRNA but not translated.
  - The **translation start site** marking the start of the coding region. Consists of a **start codon**, which causes the start of translation
  - The ***coding region*** of the gene (typically=1000bp), consisting of a sequence of codons.
  - The ***translation stop site*** marking the end of coding region. Consists of a **stop codon**, which causes the release of the polypeptide at conclusion of translation.
  - A 3' *untranslated region*, transcribed into RNA but not translated.
  - The ***transcription stop site*** marking where the RNA polymerase concludes transcription.

The bacterial genome

The E. coli chromosome



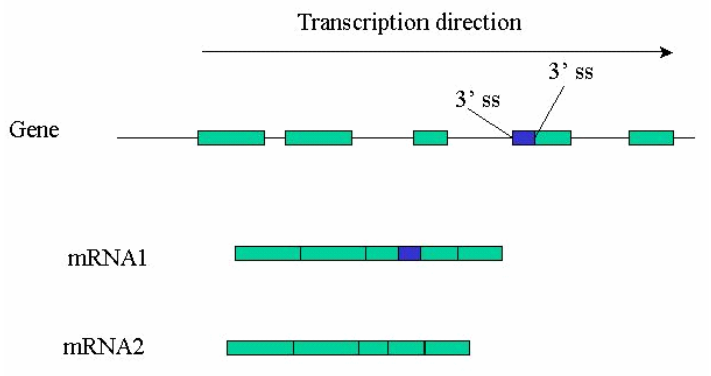
Gene structure in eukaryotes



Gene structure in eukaryotes

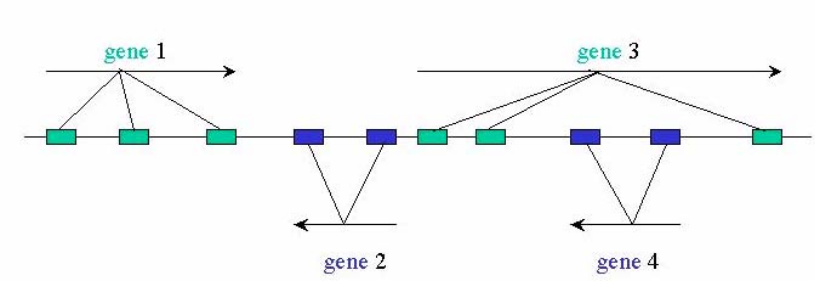
- **A typical gene consist of the following, in 5' to 3' order**
  - An ***upstream regulatory region***, often larger and more complex than in prokaryotes, parts of which may be several thousand bases or more upstream of transcription start site.
  - A ***transcription start site***.
  - A **5' *untranslated region***, often larger than in prokaryotes, and which may include sequences playing a role in translation regulation.
  - The ***coding sequence***, which unlike the case with prokaryotes, may be interrupted by non—coding regions called introns. These are spliced out of the transcript to form the mature mRNA (and sometimes the splicing can occur in more than one way).
  - The ***translation stop site***.
  - A **3' *untranslated region***, which may contain sequences involved in translational regulation.
  - **A *polyadenylation (playA) signal***, which indicates to the cell's RNA processing machinery that the RNA transcript is to be cleaved and a poly-adenine sequence (AAAAAA…) tail appended to it
  - The ***transcription stop site***.

Alternative splicing

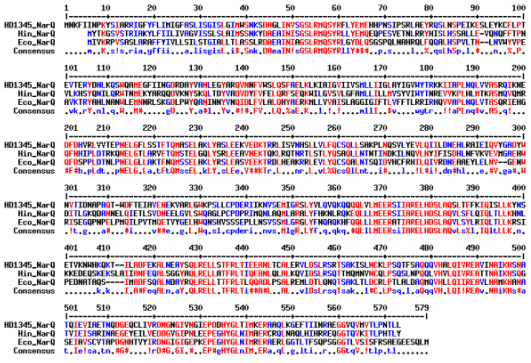


Eukaryotic genome structure

- Genes may be transcribed in either direction, and can overlap



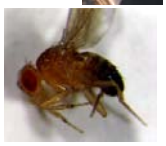
Sequence alignment:



Complete DNA Sequences





nearly 200 complete genomes have been sequenced






Evolution



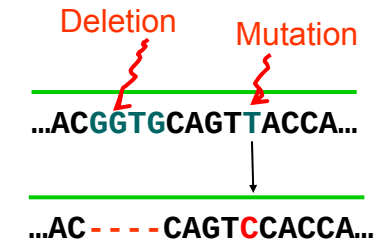






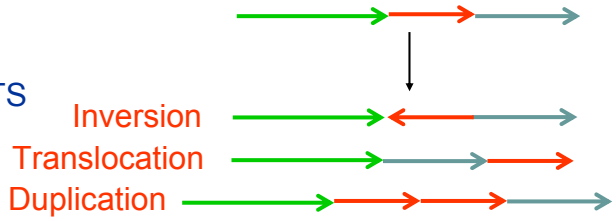



Evolution at the DNA level

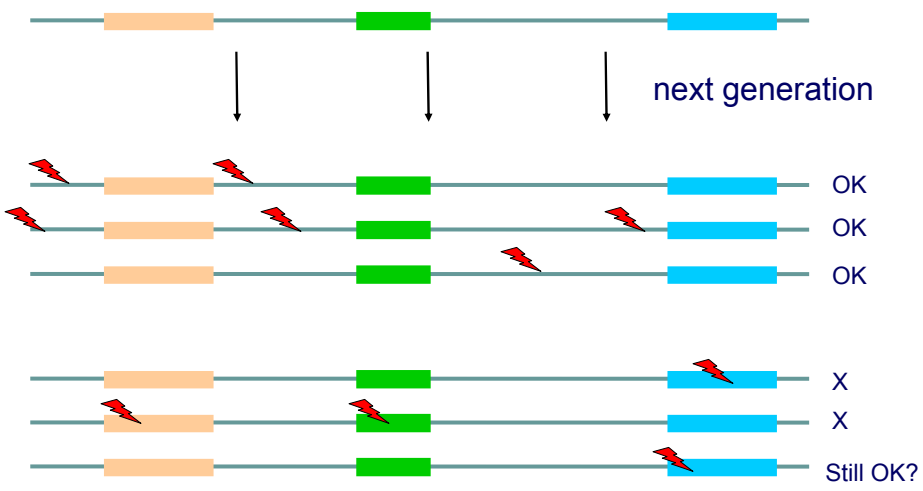


SEQUENCE EDITS

REARRANGEMENTS



Evolutionary outcome



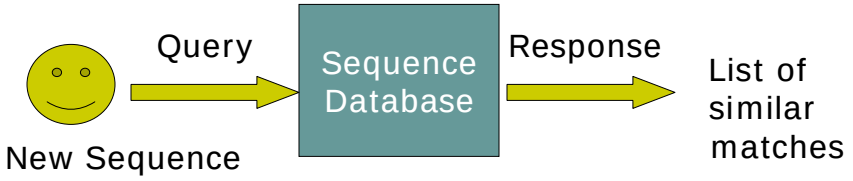
Sequence conservation implies functional conservation

Alignment is the key to

- Finding important regions
- Determining function
- Uncovering the evolutionary forces

Sequence-based functional prediction

- Sequence similarity is useful in *predicting the function of a new sequence*…
- … assuming that sequence similarity implies structural and functional similarity.



Sequence Alignment

AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC

-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC

Definition

Given two strings $x = x_1x_2...x_M$, $y = y_1y_2...y_N$,

An alignment of two sequences ***x*** and ***y*** is an arrangement of ***x*** and ***y*** by position, where ***a*** and ***b*** can be padded with gap symbols to achieve the same length.

Editing Distance

- Sequence edits: **AGGCCTC**
  - Mutations **AGGACTC**
  - Insertions **AGGGCCTC**
  - Deletions **AGG_CTC**
- We can turn the edit protocol into a measure of distance by assigning a "cost" or "weight" ***S*** to each operation.
  - For example, for arbitrary characters u,v from set A we may define
    $S(u,u) = 0$; $S(u,v) = 1$ for $u \neq v$; $S(u,-) = S(-,v) = 1$. (*Unit Cost*)
  - This scheme is known as the **Levenshtein distance**, also called unit cost model. Its predominant virtue is its simplicity.
- In general, more sophisticated cost models must be used.
  - For example, replacing an amino acid by a biochemically similar one should weight less than a replacement by an amino acid with totally different properties.

Scoring Function

- **Scoring Function:**

  Match: +m

  Mismatch: −s (a more sophisticated score matrix can be used for proteins)

  Gap: −d

  Score F = (# matches) × m − (# mismatches) × s − (#gaps) × d

- The ***Alignment Score*** of ***x*** and ***y*** is the score of an optimal alignment of ***x*** and ***y*** under a score function *S*. We denote it by *F*(***x***,***y***).
  - For example, using the score function corresponding to the unit cost model in our previous example, we obtain the following score:

    a: AGCACAC-A or AG-CACACA

    b: A-CACACTA ACACACT-A

    cost: -2 cost: -4
    - Here it is easily seen that the left-hand assignment is optimal under the unit cost model, and hence the alignment score *F*(***a***,***b***) = -2.

Scoring Matrices

- Physical/Chemical similarities
  - comparing two sequences according to the properties of their residues may highlight regions of structural similarity
- The matrix that performs best will be the one that best reflects the evolutionary separation of the sequences being aligned
  - The most commonly used mutation matrices: **PAM or BLOSUM**

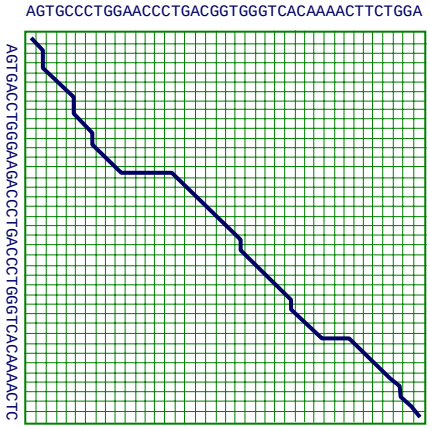
Below diagonal: BLOSUM62 substitution matrix
Above diagonal: Difference matrix obtained by subtracting the PAM 160 matrix entrywise.

(Henikoff & Henikoff 1992)

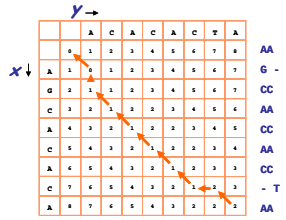
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
	0	-1	1	0	2	1	1	2	1	2	0	0	2	4	1	5	1	2	-2	5	C
		2	0	-2	0	-1	0	0	0	1	0	0	0	1	0	1	-1	1	1	-1	S
C	9		2	-1	-1	-1	0	0	0	0	0	0	-1	0	-1	1	0	1	1	3	T
S	-1	4		2	-2	-1	-1	0	0	-1	-1	-1	1	1	0	-1	0	0	2	1	P
T	-1	1	5		2	-1	-2	-2	-1	0	0	1	1	0	0	1	0	1	1	2	A
P	-3	-1	-1	7		2	0	-1	-2	0	1	1	0	0	-1	0	-1	1	2	4	G
A	0	1	0	-1	4		3	-1	-1	0	0	1	-1	0	-1	0	-1	0	0	0	N
G	-3	0	-2	-2	0	6		2	-1	-1	-1	0	-1	0	0	0	0	2	1	3	D
N	-3	1	0	-2	-2	0	6		1	0	0	2	2	1	-1	0	0	2	2	4	E
D	-3	0	-1	-1	-2	-1	1	6		0	-2	0	1	1	-1	0	0	1	3	3	Q
E	-4	0	-1	-1	-1	-2	0	2	5		2	-1	0	1	0	-1	0	1	2	2	H
Q	-3	0	-1	-1	-1	-2	0	0	2	5		-1	-1	0	-1	1	0	1	3	-4	R
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8		1	-2	-1	1	1	2	3	1	K
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5		-2	-1	-1	0	1	2	4	M
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5		-1	1	0	0	1	3	I
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5		-1	0	-1	1	2	L
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4		0	1	2	4	V
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4		-1	-2	1	F
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4		-1	2	Y
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6		-1	W
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

How do we compute the best alignment?

- A alignment corresponds to a path in the alignment matrix



Example:



Too many possible alignments:

$O(2^{M+N})$

Dynamic Programming

- The optimum alignment is obtained by tracing the highest scoring path from the top left-hand corner to the bottom right-hand corner of the matrix (or the lowest editing-distance path from bottom right-hand corner to top left-hand corner)
- When the alignment steps away from the diagonal this implies an insertion or deletion event, the impact of which can be assessed by the application of a gap penalty
- Dynamic Programming: recursively solve nested problems each of a manageable size

Dynamic Programming

- Three possible cases:

1. x_i aligns to y_j

   $x_1 \ldots\ldots x_{i-1} \quad x_i$

   $y_1 \ldots\ldots y_{j-1} \quad y_j$

   $$F(i,j) = F(i-1, j-1) + \begin{cases} m, \text{ if } x_i = y_j \\ -s, \text{ if not} \end{cases}$$

2. x_i aligns to a gap

   $x_1 \ldots\ldots x_{i-1} \quad x_i$

   $y_1 \ldots\ldots y_j \quad -$

   $$F(i,j) = F(i-1, j) - d$$

3. y_j aligns to a gap

   $x_1 \ldots\ldots x_i \quad -$

   $y_1 \ldots\ldots y_{j-1} \quad y_j$

   $$F(i,j) = F(i, j-1) - d$$

Dynamic Programming (cont'd)

- How do we know which case is correct?

Inductive assumption:

$F(i, j-1), F(i-1, j), F(i-1, j-1)$ are optimal

Then,

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

Where $s(x_i, y_j) = m$, if $x_i = y_j$; $-s$, if not

Example

x = AGTA
y = ATA

m = 1
s = -1
d = -1

$F(i,j)$ $i = 0$ 1 2 3 4

j			A	G	T	A
0						
1	A					
2	T					
3	A					

Optimal Alignment:

F(4,3) = 2

AGTA
A - TA

Alignment is additive

- Observation:
  The score of aligning $x_1 \ldots x_M$ $y_1 \ldots y_N$
  is additive

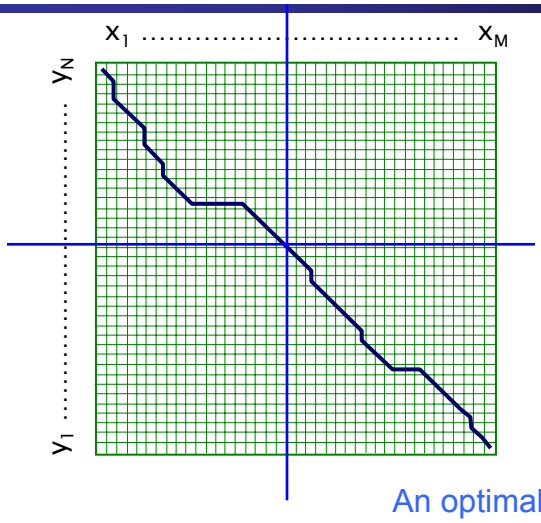
  Say that $x_1 \ldots x_i$ $x_{i+1} \ldots x_M$
  aligns to $y_1 \ldots y_j$ $y_{j+1} \ldots y_N$

  The two scores add up:

  $F(x[1:M], y[1:N]) = F(x[1:i], y[1:j]) + F(x[i+1:M], y[j+1:N])$

  $F^*(x[1:M], y[1:N]) = \text{Max}_{ij} \{F^*(x[1:i], y[1:j]) + F^*(x[i+1:M], y[j+1:N])\}$

The Needleman-Wunsch Matrix



Every nondecreasing path

from (0,0) to (M, N)

corresponds to an alignment of the two sequences

An optimal alignment is composed of optimal subalignments

The Needleman-Wunsch Algorithm

1. Initialization.
   a. $F(0, 0) = 0$
   b. $F(0, j) = -j \times d$
   c. $F(i, 0) = -i \times d$

2. Main Iteration. Filling-in partial alignments
   a. For each $i = 1 \ldots\ldots M$
      For each $j = 1 \ldots\ldots N$

$$F(i, j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & \text{[case 1]} \\ F(i-1, j) - d & \text{[case 2]} \\ F(i, j-1) - d & \text{[case 3]} \end{cases}$$

$$Ptr(i,j) = \begin{cases} \text{DIAG, if [case 1]} \\ \text{LEFT, if [case 2]} \\ \text{UP, if [case 3]} \end{cases}$$

3. Termination. F(M, N) is the optimal score, and from Ptr(M, N) can trace back optimal alignment

Performance

- Time:

  O(NM)

- Space:

  O(NM)

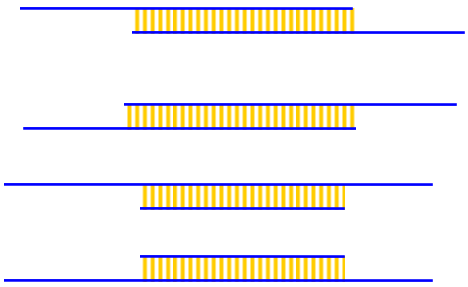
- Later we will cover more efficient methods

A variant of the basic algorithm:

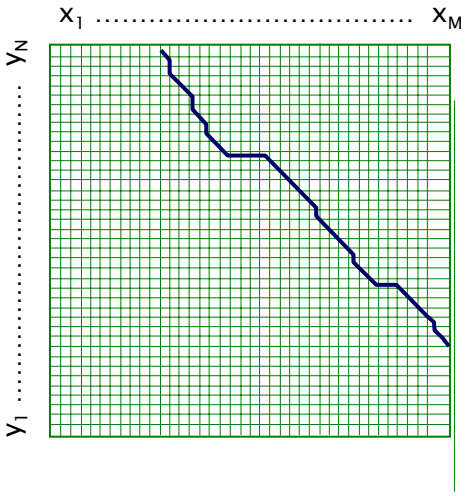
- Maybe it is OK to have an unlimited # of gaps in the beginning and end:

```
----------CTATCACCTGACCTCCAGGCCGATGCCCCTTCCGGC
GCGAGTTCATCTATCAC--GACCGC--GGTCG--------------
```

  - Then, we don't want to penalize gaps in the ends
- Different types of overlaps



The Overlap Detection variant



Changes:

1. Initialization

   For all i, j,

   $F(i, 0) = 0$

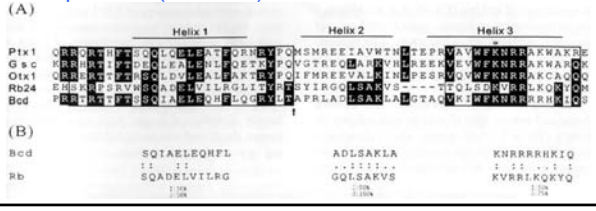
   $F(0, j) = 0$

2. Termination

$$F_{OPT} = \max \begin{cases} \max_i F(i, N) \\ \max_j F(M, j) \end{cases}$$

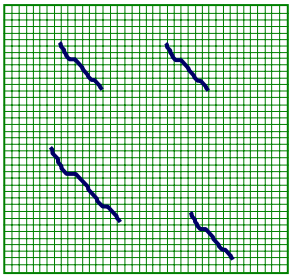
The local alignment problem

- The problem:
  - Given two strings $x = x_1 \ldots\ldots x_M$,
    $y = y_1 \ldots\ldots y_N$
  - Find substrings x', y' whose similarity (optimal global alignment value) is maximum
  - e.g. x = aaaaccccgggg
    y = cccgggaaccaacc
- Why
  - Genes are shuffled between genomes
  - Portions of proteins (domains) are often conserved



The Smith-Waterman algorithm

Idea: Ignore badly aligning regions



Modifications to Needleman-Wunsch:

Initialization: $F(0, j) = F(i, 0) = 0$

Iteration: $F(i, j) = \max \begin{cases} 0 \\ F(i-1, j) - d \\ F(i, j-1) - d \\ F(i-1, j-1) + s(x_i, y_j) \end{cases}$

The Smith-Waterman algorithm

Termination:

1. If we want the best local alignment…

$$F_{OPT} = \max_{i,j} F(i, j)$$

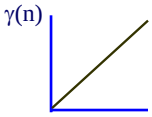
2. If we want all local alignments scoring > t

?? For all i, j find F(i, j) > t, and trace back

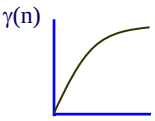
Complicated by overlapping local alignments

Scoring the gaps more accurately

- Current model:
  - Gap of length n
  - incurs penalty n×d



- However, gaps usually occur in bunches
  - Convex (saturating) gap penalty function:



γ(n):
for all n, $\gamma(n+1) - \gamma(n) \leq \gamma(n) - \gamma(n-1)$

Convex gap dynamic programming

Initialization: same

Iteration:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ \max_{k=0\ldots i-1} F(k,j) - \gamma(i-k) \\ \max_{k=0\ldots j-1} F(i,k) - \gamma(j-k) \end{cases}$$

Termination: same

Running Time: $O(N^2M)$ (assume N>M)

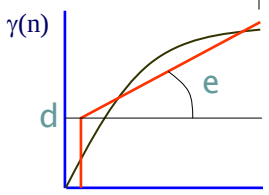
Space: $O(NM)$

Compromise: affine gaps

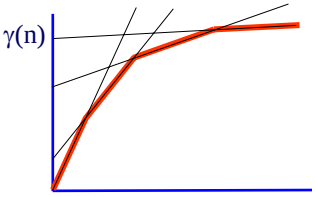
- Simple piece-wise linear gap penalty

$\gamma(n) = d + (n - 1) \times e$

gap open	gap extend



- Fancier Piece-wise linear gap penalty



- Think of how you would compute optimal alignment with this gap function in O(MN)

Bounded Dynamic Programming

- Assume we know that x and y are very similar

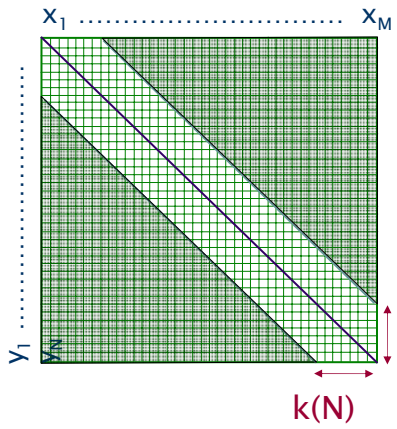
Assumption: # gaps(x, y) < k(N) (say N>M)

Then, x_i | y_j implies $|i - j| < k(N)$

We can align x and y more efficiently:

Time, Space: $O(N \times k(N)) << O(N^2)$

Bounded Dynamic Programming



Initialization:

F(i,0), F(0,j) undefined for i, j > k

Iteration:

For i = 1…M

For j = max(1, i – k)…min(N, i+k)

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i, j-1) - d, \text{ if } j > i - k(N) \\ F(i-1, j) - d, \text{ if } j < i + k(N) \end{cases}$$

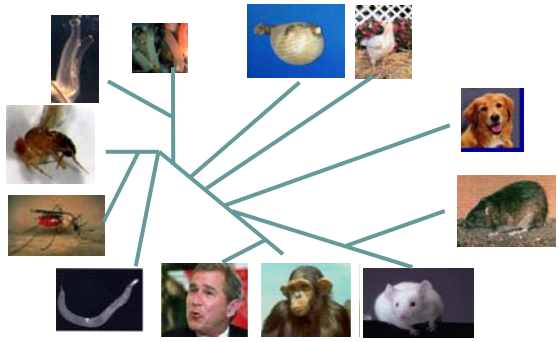
Termination: same

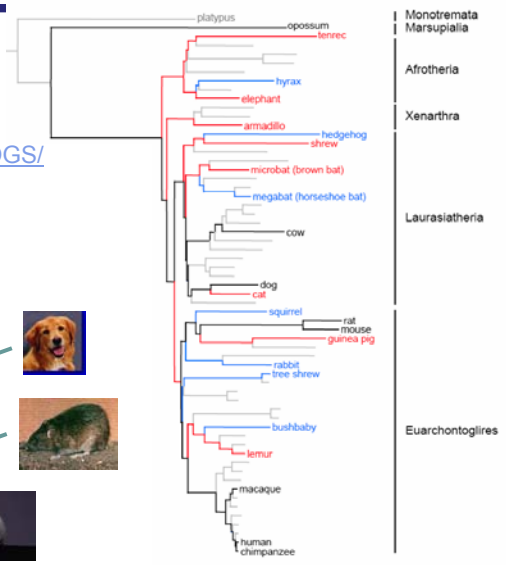
Easy to extend to the affine gap case

State of biological databases

http://www.genome.gov/10005141

http://www.cbs.dtu.dk/databases/DOGS/



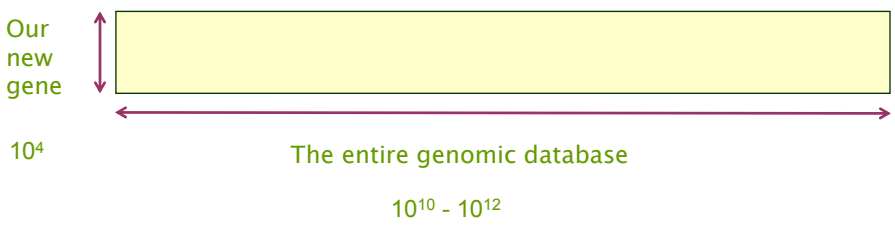


State of biological databases

- Number of genes in these genomes:
  - Mammals: ~25,000
  - Insects: ~14,000
  - Worms: ~17,000
  - Fungi: ~6,000-10,000
  - Small organisms: 100s-1,000s
- Each known or predicted gene has one or more associated protein sequences
- >1,000,000 known / predicted protein sequences

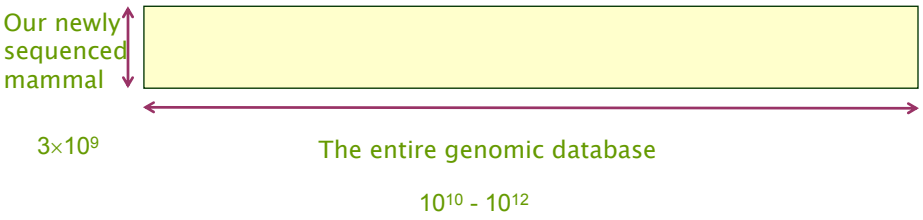
Some useful applications of alignments

- Given a newly discovered gene,
  - Does it occur in other species?
  - How fast does it evolve?
- Assume we try Smith-Waterman:



Some useful applications of alignments

- Given a newly sequenced organism,
- Which subregions align with other organisms?
  - Potential genes
  - Other biological characteristics
- Assume we try Smith-Waterman:



Indexing-based local alignment

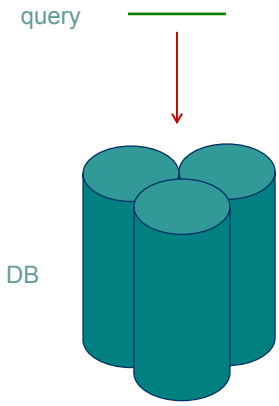
- BLAST- **B**asic **L**ocal **A**lignment **S**earch **T**ool

Main idea:

1. Construct a dictionary of all the words in the query
2. Initiate a local alignment for each word match between query and DB

Running Time: O(MN)

However, orders of magnitude faster than Smith-Waterman



Multiple alignment

- *The simultaneous alignment of a number of DNA or protein sequences is one of the commonest tasks in bioinformatics.*
- Useful for:
  - phylogenetic analysis (inferring a tree, estimating rates of substitution, etc.)
  - detection of homology between a newly sequenced gene and an existing gene family
  - prediction of protein structure
  - demonstration of homology in multigene families
  - determination of a consensus sequence (*e.g.,* in assembly)
- Can we naively use DP?
  - need to deal with k-dimensional table for k sequences …

Extending the pairwise alignment algorithms

- Generally not feasible for more than a small number of sequences (~5), as the necessary computer time and space quickly becomes prohibitive.
  - Computational time grows as N^m, where m = number of sequences.
  - For example, for 100 residues from 5 species, 1005 = 10,000,000,000 (*i.e.*, the equivalent of two sequences each 100,000 residues in length.)
- Nor is it wholly desirable to reduce multiple alignment to a similar mathematical problem to that tackled by pairwise alignment algorithms.
- Two issues which are important in discussions of multiple alignment are:
  - the treatment of gaps: position-specific and/or residue-specific gap penalties are both desirable and feasible, and
  - the phylogenetic relationship between the sequences (which must exist if they are alignable): it should be exploited.

Progressive alignment

- Up until about 1987, multiple alignments would typically be constructed manually, although a few computer methods did exist.
- Around that time, algorithms based on the idea of **progressive alignment** appeared.
  - In this approach, a pairwise alignment algorithm is used iteratively,
    - first to align the most closely related pair of sequences,
    - then the next most similar one to that pair, and so on.
  - The rule "once a gap, always a gap" was implemented, on the grounds that the positions and lengths of gaps introduced between more similar pairs of sequences should not be affected by more distantly related ones.
- The most widely used progressive alignment algorithm is currently **CLUSTAL W**.
  - Other methods include the profile HMM-based methods

CLUSTAL W

- The three basic steps in the **CLUSTAL W** approach are shared by all progressive alignment algorithms:

  A. Calculate a matrix of **pairwise distances** based on pairwise alignments between the sequences

  B. Use the result of A to build a **guide tree**, which is an inferred phylogeny for the sequences

  C. Use the tree from B to guide the **progressive alignment** of the sequences

- We will omit details

Web-based multiple sequence alignment

- ClustalW
  - www2.ebi.ac.uk/clustalw/
  - dot.imgen.bcm.tmc.edu:9331/multi-align/Options/clustalw.html
  - www.clustalw.genome.ad.jp/
  - bioweb.pasteur.fr/intro-uk.html
  - pbil.ibcp.fr
  - transfac.gbf.de/programs.html
  - www.bionavigator.com
- PileUp
  - helix.nih.gov/newhelix
  - www.hgmp.mrc.ac.uk/
  - bcf.arl.arizona.edu/gcg.html
  - www.bionavigator.com
- Dialign
  - genomatix.gsf.de/
  - bibiserv.techfak.uni-bielefeld.de/
  - bioweb.pasteur.fr/intro-uk.html
  - www.hgmp.mrc.ac.uk/
- Match-box
  - www.fundp.ac.be/sciences/biologie/bms/matchbox_submit.html
- For reviews: G. J. Gaskell, BioTechniques 2000, 29:60, and
  - www.techfak.uni-bielefeld.de/bcd/Curric/MulAli/welcome.html

Acknowledgments

- **Serafim Batzoglou**: for many of the slides adapted or modified from his lecture slides at Stanford University
- **Terry Speed**: for some of the slides modified from his lectures at UC Berkeley
- **Phil Green** and **Joe Felsenstein**: for some of the slides modified from his lectures at Univ. of Washington