Next Generation Bioinformatics Tools

Fall 2012

Day 2 – Sequence Analysis and Phylogeny

Hesham H. Ali
UNO Bioinformatics Research Group
College of Information Science and Technology

UNIVERSITY OF NEBRASKA AT OMAHA

Sequence Comparison

- Biology has a long tradition of comparative analysis leading to discovery.
- The number of sequences available for comparison has been growing explosively.
- Efficient algorithms already exist for solving many sequence comparison related problems.
- Sequences may be incomplete and/or having different sizes

...Its Not Straightforward

- Compare the two DNA sequences:
  GACGATTAG and **GATCGAATAG**

GACGATTAG
GATCGAATAG

- 20% similar...but add room for gaps....

GA-CGATTAG
GATCGAATAG

- 80% similar

Why Compare Sequences?

- Identification of organisms using obtained sequences
- Classification of organisms- evolutionary analysis
- Function prediction
- Structure prediction
- Motif Finding
- Such a tool would have to utilize biological knowledge and databases to identify sequences
- Clustering and Phlogenetic Analysis

Sequence Comparison Solutions

- Sequence alignment
- Sequence comparison using data compression
- Sequence comparison using landmarks/motifs
- Sequence comparison using Longest Common Subsequence
- Sequence Comparison using composition based methods and various statistical properties

Each method works well for different types of Bioinformatics problems

String Matching

- Exact Matching: What is the problem?
- Importance of the exact matching problem
- The Naïve string matching method
- String matching with finite automaton
- Classical Comparison–Based Methods
- Boyer Moore Algorithm
- Knuth-Morris-Pratt Algorithm (KMP)
- Applications of exact matching

Exact Matching: What's the problem?

- Given a string P called a *pattern* and a longer string T called the *text*, the exact matching problem is to find all occurrences, if any, of pattern P in text T.
  - If P = "aba" and T = "bbabaxababay" then P occurs in T starting at locations 3, 7, and 9. Note that two occurrences of P may overlap (as at locations 7 and 9).

Importance of Exact Matching

- Obvious applications such as Word-processing, in utilities such as *grep* on UNIX.
- In textual information retrieval systems such as Medline, Lexis, etc
- Library catalog searching programs.
- Directories, Dictionary, etc.
- In Molecular Biology there are several hundred specialized databases holding raw DNA, RNA and Amino acid strings, or processed patterns (called Motifs) derived from raw string data.

Classical Comparison Based Methods

- Naïve string matching
- Finite Automaton Algorithms
- KMP Algorithm
- Boyer Moore Algorithm

The Naïve Method

- Finds all valid shifts using a loop that checks the condition

  $P[1...m] = T[s+1..s+m]$

  for each of the $(n - m + 1)$ possible values of s.

The Naïve Method

- Can be interpreted graphically as sliding a "template" containing the pattern over the text.
- The For loop considers each possible shift explicitly.
- The Test condition determines whether the current shift is valid or not.
- The worst case running time is

  $O((n - m + 1)m)$

String matching with Finite State Automata

- Many string-matching algorithms build a finite automaton that scans the text string T for all occurrences of the pattern P.
- These string-matching automaton are very efficient
  - They examine each text character *exactly once*, taking constant time per text character.
  - The time taken after the automaton is built – O(n).

String matching with Finite Automata

- Finite automata: A *finite automaton* M is a 5-tuple (Q, q0, A, S, d), where
  - Q is a finite set of *states*.
  - q0 belongs to set Q is the *start state*.
  - A subset of Q is a distinguished set of *accepting states* or *final states*.
  - S is a finite *input alphabet*.
  - d is a function from QxS into Q, called the *transition function* of M.

Boyer-Moore Algorithm

- If the pattern P is relatively long and the alphabet Sigma is reasonably large, then the Boyer-Moore algorithm is likely to be the most efficient string-matching algorithm.
- Contains 3 clever ideas
  - The right-to-left scan.
  - The bad character shift rule
  - And the good suffix shift rule

Knuth-Morris-Pratt Algorithm (KMP)

- It is a linear time string matching algorithm.
- Achieves a O(n+m) running time by avoiding the computation of the transition function d altogether.
- It does the pattern matching using an auxiliary function Phi[1..m], which is pre-computed from the pattern in time O(m).

But we are more interested in Non-Exact matching

- Sequencing errors
- Repeats, translocation, insertions/deletions (indels)
- Evolutionary aspects of biological sequences
- Evolutionary aspects of systems and organisms

Sequence Alignment

Sequence Alignment

- Goal: To enable researchers to determine whether two sequences display sufficient similarity to justify the inference of homology.
- Definition: Given two sequences of sizes m and n, an alignment is the insertion of spaces in arbitrary locations along the sequences so that they end up with the same size. Possible restriction: No space in one sequence is aligned with a space in the other.

Sequence Alignment

- Types:
  - Pairwise vs Multiple
    - Pairwise: Comparing 2 sequences at a time
    - Multiple: Comparing 2+ sequences (pairwise can be multiple)
  - Local vs. Global
    - Local: Comparing short sequences/regions of sequence (e.g. CDS regions)
    - Global: Comparing entire sequences/regions (e.g. genomes, chromosomes)

Global Alignment

- Alignment is defined as the insertion of spaces in arbitrary locations along the sequences so that they end up with the same size.
- No space in one sequence be aligned with a space in the other.

Example:

- Compare the two DNA sequences:
  GACGATTAG and **GATCGAATAG**

GACGATTAG
GATCGAATAG

- 20% similar…but add room for gaps….

GA-CGATTAG
GATCGAATAG

- 80% similar

Score of Alignment

- GA-CGATTAG
  GATCGAATAG

- 80% similarity
- Scoring:
  - Match = +1
  - Mismatch = -1
  - Space = -2

- G A - C G A T T A G
  G A T C G A A T A G
  1 1 -2 1 1 1 -1 1 1 1 Sum = 5

The Main Equation

Sim(S1[1…i],S2[1…j]) = max {
Sim(S1[1…i-1],S2[1…j-1]) + p(i,j)
Sim(S1[1…i],S2[1…j-1]) δ(-,j),
Sim(S1[1…i-1],S2[1…j]) δ(i,-) }

Where p(i,j) = either +1 (match) or –1 (mismatch)
and δ(-,j) = δ(i,-) = -2

Global Alignment

- Consider:
  - GA-CGATTAG
  - GATCGAATAG
- Scoring an alignment
- Total score = (8*1) + (1*-1) + (1*-2) = 5
- Observations
  - For (i, j) we need only (i-1, j), (i-1, j-1), (i, j-1)

  - A[i,j] = max {a[i-1, j-1] + p(i,j),
    a[i, j-1] – d(-,j),
    a[i-1, j] – d(i,-)}

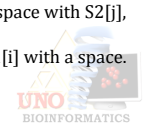
Complexity (Cost) of the Alignment Problem

- Generating all possible alignments
  (exponential)
- Smart algorithm using dynamic programming
  (efficient optimal algorithm)

  Modification of the DP solution to the longest common subsequence problem

Optimal Alignment: DP Approach

- Key observation: the optimal alignment for the two sequences S1[1…m] and S2[1…n] can be evaluated iteratively by finding the optimal alignment for S1[1…i] and S2[1…j] for all i<m and j<n.
- To find the alignment for S1[1…i] and S2[1…j], we have three choices only:
  - Align S1[1…i-1] with S2[1…j-1] and match S1[i] with S2[j], or
  - Align S1[1…i] with S2[1…j-1] and match a space with S2[j], or
  - Align S1[1…i-1] with S2[1…j] and match S1[i] with a space.



Global Comparison: The Basic Algorithm (Needleman-Wunsch)

```
Algorithm Similarity
Input: sequences S1 and S2
Output: similarity Score between S1 and S2
m = |S1|
n  = |S2|
for i=0 to m  do d[i,0] = i*-2
for j=0 to n   do d[0,j] = j*-2
for i=1 to m do
      for j=1 to n  do
        d[i,j] = max(d[i-1,j-1]+p(i,j), d(i-1,j)-2, d[i,j-1]-2)
return d[m,n]
```

Gapped Alignments

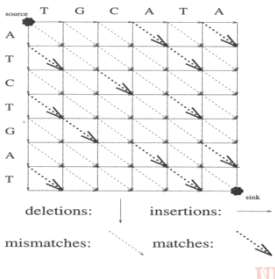
- Biological sequences
  - Different lengths
  - Regions of insertions and deletions
- Notion of gaps (denoted by '-')

```
A L I G N M E N T
  | | |   | | | |
- L I G A M E N T
```


Possible Residue Alignments

- Match
- Mismatch (substitution or mutation)
- Insertion/Deletion (INDELS – gaps)

Edit Graph



Different Alignments

- Which alignment is best?

```
A - C - G G - A C T
|   |   |       | |
A T C G G A T _ C T

A T C G G A T C T
|   | | |     | |
A - C G G - A C T
```


Alignment Scoring Scheme

- Possible scoring scheme:
  match: +2
  mismatch: -1
  indel –2

- Alignment 1: $5 * 2 - 1(1) - 4(2) = 10 - 1 - 8 = 1$
- Alignment 2: $6 * 2 - 1(1) - 2\,(2) = 12 - 1 - 4 = 7$

Dynamic Programming

- Solve optimization problems by dividing the problem into subproblems
  - Overlapping subproblems
  - Optimal substructure
  - Useful tabulation
- Sequence alignment has optimal substructure property
  - Subproblem: alignment of prefixes of two sequences
  - Each subproblem is computed once and stored in a matrix

DP Example

Sequence #1: GAATTCAGTTA; M = 11
Sequence #2: GGATCGA; N = 7

- $s(a_i b_j) = +5$ if $a_i = b_j$ (match score)
- $s(a_i b_j) = -3$ if $a_i \neq b_j$ (mismatch score)
- $w = -4$ (gap penalty)

View of the DP Matrix

- M+1 rows, N+1 columns

	-	G	A	A	T	T	C	A	G	T	T	A
-												
G												
G												
A												
T												
C												
G												
A												

Global Alignment (Needleman-Wunsch)

- Attempts to align all residues of two sequences

- *INITIALIZATION*: First row and first column set
- $S_{i,0} = w * i$
- $S_{0,j} = w * j$

Initialized Matrix (Needleman-Wunsch)

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44
G	-4											
G	-8											
A	-12											
T	-16											
C	-20											
G	-24											
A	-28											

Matrix Fill (Global Alignment)

$S_{i,j}$ = MAXIMUM{

$S_{i-1, j-1} + s(a_i, b_j)$ (match/mismatch in the diagonal),

$S_{i,j-1} + w$ (gap in sequence #1),

$S_{i-1,j} + w$ (gap in sequence #2)

}

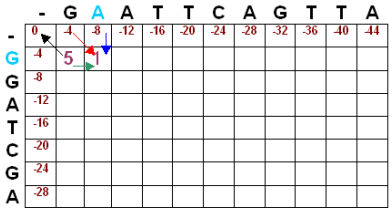
Matrix Fill (Global Alignment)

$S_{1,1} = MAX[S_{0,0} + 5, S_{1,0} - 4, S_{0,1} - 4] = MAX[5, -8, -8]$

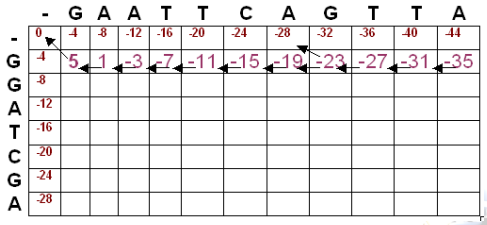
	-	G	A	A	T	T	C	A	G	T	T	A
-	0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44
G	-4	5										
G	-8											
A	-12											
T	-16											
C	-20											
G	-24											
A	-28											

Matrix Fill (Global Alignment)

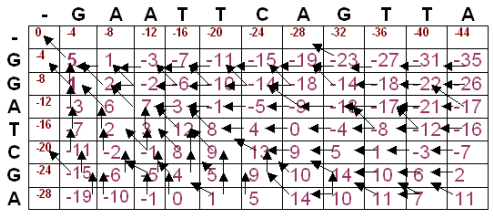
$S_{1,2} = MAX[S_{0,1} - 3, S_{1,1} - 4, S_{0,2} - 4] = MAX[-4 - 3, 5 - 4, -8 - 4] = MAX[-7, 1, -12] = 1$



Matrix Fill (Global Alignment)



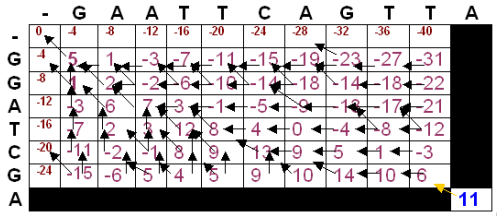
Filled Matrix (Global Alignment)



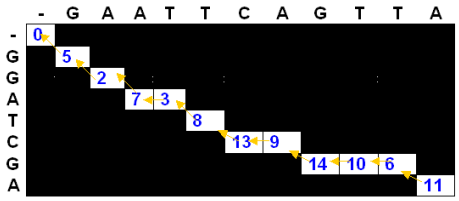
Trace Back (Global Alignment)

- Maximum global alignment score = 11 (value in the lower right hand cell).
- Traceback begins in position $S_{M,N}$; i.e. the position where both sequences are globally aligned.
- At each cell, we look to see where we move next according to the pointers.

Trace Back (Global Alignment)



Global Trace Back




```
G A A T T C A G T T A
|   |   | |   |     |
G G A - T C - G - - A
```

Checking Alignment Score

```
G A A T T C A G T T A
|   |   | |   |     |
G G A – T C – G – – A

+ – + – + + – + – – +
5 3 5 4 5 5 4 5 4 4 5
```

5 – 3 + 5 – 4 + 5 + 5 – 4 + 5 – 4 – 4 + 5 = 11 ✓

Variations of the Alignment Problem

- Local Comparison: Alignment between sub-strings of two sequences.
- Semi-global comparison: Ignoring some of the end spaces in the sequences

  CAGCA--CTTGGATTCTCGG
  ------CAGCGTGG ---------- -19 or 3
  CAGC----------G----T-------GG -12
- Comparing multiple sequences:

  MQPILLL
  MLR-LL-
  MK- ILLL
  MPPVLIL

Local Alignment

- Given two Strings S1 and S2, find sub-strings A and B of S1 and S2, respectively. Whose similarity (optimal global alignment value) is maximum over all pairs of sub-strings from S1 and S2.
- Why? – Global alignment of protein is often meaningful when the 2 strings are members of the same protein family.
- Local alignment is critical because proteins from very different families are often made up of the same structural and functional subunits.

Local Alignment Algorithm (Smith-Waterman)

```
Algorithm Similarity
Input: sequences S1 and S2
Output: similarity Score between S1 and S2
m = |S1|
n  = |S2|
for i=0 to m  do d[i,0] = ?
for j=0 to n   do d[0,j] = ?
for i=1 to m do
        for j=1 to n  do
          d[i,j] = max{d[i-1,j-1]+p(i,j), d(i-1,j)-2, d[i,j-1]-2, ?)
return d[m,n]
```

Local Alignment

- Smith-Waterman: obtain highest scoring local match between two sequences
- Requires 2 modifications:
  - Initial scores are zeros
  - When a value in the score matrix becomes negative, reset it to zero (begin of new alignment)

  Make sure mismatches have negative scores

Local Alignment Initialization

Values in row 0 and column 0 set to 0.

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Matrix Fill (Local Alignment)

$S_{i,j}$ = MAXIMUM{
  $S_{i-1, j-1}$ + s(a_i, b_j) (match/mismatch in the diagonal),
  $S_{i,j-1}$ + w (gap in sequence #1),
  $S_{i-1,j}$ + w (gap in sequence #2),
  0
}

Matrix Fill (Local Alignment)

$S_{1,1}$ = MAX[$S_{0,0}$ + 5, $S_{1,0}$ - 4, $S_{0,1}$ – 4,0] = MAX[5, -4, -4, 0] = 5

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5										
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Matrix Fill (Local Alignment)

$S_{1,2}$ = MAX[$S_{0,1}$ -3, $S_{1,1}$ - 4, $S_{0,2}$ – 4, 0] = MAX[0 - 3, 5 – 4, 0 – 4, 0] = MAX[-3, 1, -4, 0] = 1

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	1									
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Matrix Fill (Local Alignment)

$S_{1,3} = MAX[S_{0,2} - 3, S_{1,2} - 4, S_{0,3} - 4, 0] = MAX[0 - 3, 1 - 4, 0 - 4, 0] = MAX[-3, -3, -4, 0] = 0$

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	1	0								
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Filled Matrix (Local Alignment)

	-	G	A	A	T	T	C	A	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	5	1	0	0	0	0	0	5	1	0	0
G	0	5	2	0	0	0	0	0	5	2	0	0
A	0	1	10	7	3	0	0	5	1	2	0	5
T	0	0	6	7	13	8	4	1	2	6	7	3
C	0	0	2	3	8	9	13	9	5	2	3	4
G	0	5	1	0	4	5	9	10	14	10	6	2
A	0	1	10	6	2	1	4	14	10	11	7	11

Maximum Local Alignment

```
G A A T T C - A     G A A T T C - A
|   | |  |  |     |   |  | |  |
G G A T - C G A     G G A - T C G A

+ - + + - + - +     + - + - + + - +
5 3 5 5 4 5 4 5     5 3 5 4 5 5 4 5
```

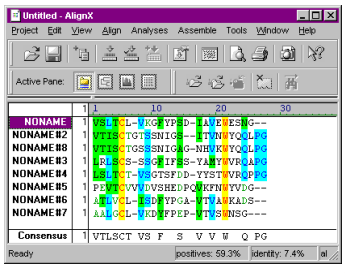
Multiple Sequence Alignment

- Similar regions conserved across organisms
  - Same or similar function
  - Same or similar structure
- Simultaneous alignment of similar regions yields:
  - regions subject to mutation
  - regions of conservation
  - mutations or rearrangements causing change in conformation or function

Multiple Sequence Alignment

- New sequence can be aligned with known sequences
  - Yields insight into structure and function
- Multiple alignment can detect important features or motifs
- GOAL: Take 3 or more sequences, align so greatest number of characters are in the same column
- Difficulty: introduction of multiple sequences increases combination of matches, mismatches, gaps

Example Multiple Alignment



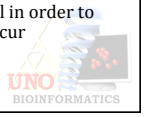
- Example alignment of 8 IG sequences.

Approaches to Multiple Alignment

- Dynamic Programming
- Progressive Alignment
- Iterative Alignment
- Statistical Modeling

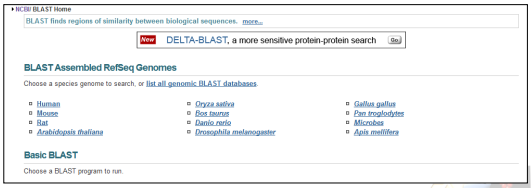
CLUSTALW

- Perform pairwise alignments of all sequences
- Use alignment scores to produce phylogenetic tree
- Align sequences sequentially, guided by the tree
  - Enhanced Dynamic Programming used to align sequences
  - Genetic distance determined by number of mismatches divided by number of matches
  - Gaps are added to an existing profile in progressive methods
  - CLUSTALW incorporates a statistical model in order to place gaps where they are most likely to occur



BLAST

- **B**asic **L**ocal **A**lignment **S**earch **T**ool
- Identifies similar sequences based on a query



BLAST

- Basic Local Alignment Search Tool
  - Altschul et. al., 1990 J. Mol. Biology
  - One of the most widely used tool
    - 41812 citations until August 2012
  - Much faster than Smith-Waterman algorithm because it uses heuristics
    - may not guarantee optimal alignments as Smith-Waterman does

BLAST

- For fast results,
  - Minimizes the search space
  - May result in loss in sensitivity
- 3 heuristic layers are used
  - Seeding
  - Extension
  - Evaluation

Program	Database	Query
BLASTN	Nucleotide	Nucleotide
BLASTP	Protein	Protein
BLASTX	Protein	Nucleotide translated into protein
TBLASTN	Nucleotide translated into protein	Protein
TBLASTX	Nucleotide translated into protein	Nucleotide translated into protein

BLAST

Types of BLAST algorithms:

- Megablast:
  - Contiguous: Nearly identical sequences
  - Discontiguous: Cross-species comparison
- Position specific:
  - PSI-BLAST: Automatically generates a position specific score matrix (PSSM)
  - RPS-BLAST: Searches a database of PSI-BLAST PSSMs

Specialized BLAST options

Specialized BLAST

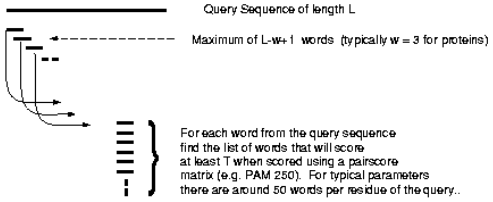
Choose a type of specialized search (or database name in parentheses.)

- Make specific primers with Primer-BLAST
- Search trace archives
- Find conserved domains in your sequence (cds)
- Find sequences with similar conserved domain architecture (cdart)
- Search sequences that have gene expression profiles (GEO)
- Search immunoglobulins (IgBLAST)
- Search using SNP flanks
- Screen sequence for vector contamination (vecscreen)
- Align two (or more) sequences using BLAST (bl2seq)
- Search protein or nucleotide targets in PubChem BioAssay
- Search SRA transcript and genomic libraries
- Constraint Based Protein Multiple Alignment Tool
- Needleman-Wunsch Global Sequence Alignment Tool
- Search RefSeqGene
- Search WGS sequences grouped by organism

BLAST Algorithm

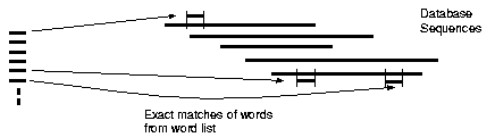
BLAST Algorithm

(1) For the query find the list of high scoring words of length w.



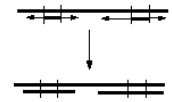
BLAST Algorithm

(2) Compare the word list to the database and identify exact matches.



BLAST Algorithm

(3) For each word match, extend alignment in both directions to find alignments that score greater than score threshold S.



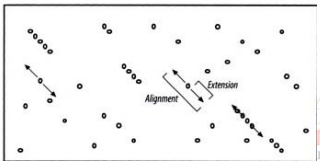
Maximal Segment Pairs (MSPs)

BLAST

- Seeding
  - BLAST assumes significant alignments have common words
  - BLAST first finds common words also called word hits
  - In NCBI BLAST, blastn has minimum word size = 7

BLAST

- Extension
  - The word hits are seeds and alignments can be generated from the individual seeds
  - From seeds, the alignment is extended on either sides until the score drops to a threshold value

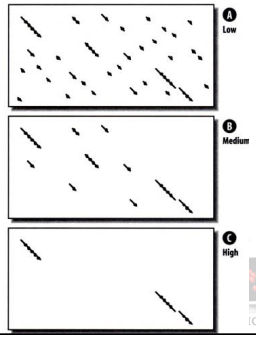


BLAST

- Evaluation
  - After the seeds are extended to create alignments
  - The alignments are evaluated to determine their statistical significance
    - Significant alignments are called HSPs (High Scoring Pairs)
    - An alignment threshold effectively removes random, low scoring alignments
    - Normalized score for all HSPs is calculated
    - E-Value
      - How many alignment with a given score are expected by chance

BLAST

- A = Low alignment thresholds
  - Shows all low scoring alignments
- B = Medium alignment thresholds
- C = High alignment thresholds



BLAST output report

- 4 major parts
  - Header
  - One line summaries
  - Alignments
  - Footer

BLAST output 1 (Header)

- It contains name of the program (blastn, blastp, blastx, tblastn, tblastx)
- Reference to scientific literature
- Query Sequence
- Database Information

```
BLASTP 2.2.5 [Nov-16-2002]

Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs", Nucleic Acids Res. 25:3389-3402.

Query= 002567 MYOGLOBIN.
         (145 letters)

Database: nr
           1,230,998 sequences; 391,609,117 total letters

Searching.....................................................done
```


BLAST output 2 (One-line summary)

- Each line has:
  - name of the sequence
  - highest scoring alignment and
  - E-value for the HSP
  - Usually limited to 500 hits

```
                                                                  Score    E
Sequences producing significant alignments:                      (bits) Value

gi|7428631|pir||GGGAA globin [validated] - slug sea hare          222   6e-58
gi|2133520|pir||S64703 myoglobin - slug sea hare (fragment)       222   7e-58
gi|70584|pir||GGGA2A globin - Kuroda's sea hare (tentative seque...  217   2e-56
gi|9257039|pdb|1DM1|A Chain A, 2.0 A Crystal Structure Of The Do...  215   7e-56
gi|230148|pdb|1MBA|  Myoglobin (Met) (pH 7.0) >gnl|BL_ORD_ID|302...  214   2e-55
gi|121267|sp|P29287|GLB_BURLE Globin (Myoglobin) >gnl|BL_ORD_ID|...  212   8e-55
```


BLAST output 3 (Alignments)

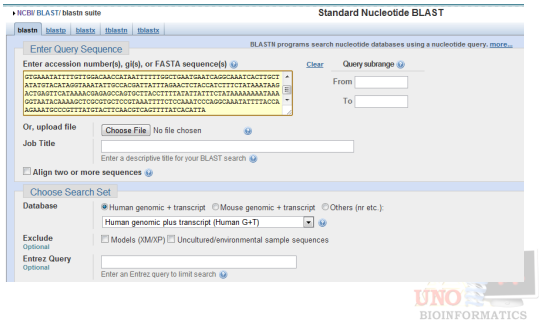
- Major Bulk of the report
- Shows the match, gap, mismatch information for each HSP
  - Score, E-value also reported

```
>gi|7428631|pir||GGGAA globin [validated] - slug sea hare
          Length = 146

 Score =  222 bits (566), Expect = 6e-58
 Identities = 112/146 (76%), Positives = 127/146 (86%), Gaps = 2/146 (1%)

Query: 2   ALSAADAGLLAQSWAPVFANSAANGDSFLVALFTQFPESANFFNDFKGKSLADIQASPKL 61
           +LSAA+A L  +SWAPVFAN  ANGD+FLVALF +FP+SANFF DFKGKS+ADI+ASPKL
Sbjct: 1   SLSAAEADLAGKSWAPVFANKDANGDAFLVALFEKFPDSANFFADFKGKSVADIKASPKL 60

Query: 62  RDVSSRIFARLNEFVSNAADAGKMCSMLQQFATEHAGFGVGSAQFQNVRSMFPGFVASLS 121
           RDVSSRIF RLNEFV+NAADAGKM +ML QFA EH GFGVGSAQF+NVRSMFPGFVAS++
Sbjct: 61  RDVSSRIFTRLNEFVNNAADAGKMSAMLSQFAKEHVGFGVGSAQFENVRSMFPGFVASVA 120

Query: 122 AP--AGDAAWNSLFGLIISALQSAGK 145
           AP   DAAW  LFGLII AL++AGK
Sbjct: 121 APPAGADAAWTKLFGLIIDALKAAGK 146
```

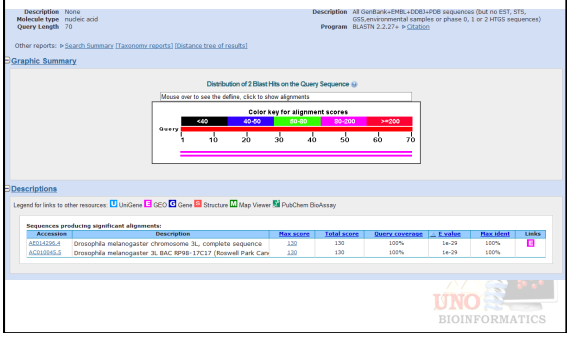
BLAST output 4 (Footer)

- It reports various parameters
  - Scoring scheme (Matrix)
  - Neighborhood word threshold score (T)
  - Word size (A for Amino acid)

BLAST Example



BLAST Example



Further Reading

- Altschul, et al., J. Mol Biology, 1990
- Korf, et al., BLAST, O'Reilly Publication, 2003

Basic Local Alignment Search Tool

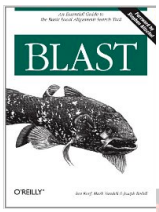
Stephen F. Altschul[1], Warren Gish[1], Webb Miller[2] Eugene W. Myers[3] and David J. Lipman[1]

[1]*National Center for Biotechnology Information National Library of Medicine, National Institutes of Health Bethesda, MD 20894, U.S.A.*

[2]*Department of Computer Science The Pennsylvania State University, University Park, PA 16802, U.S.A.*

[3]*Department of Computer Science University of Arizona, Tucson, AZ 85721, U.S.A.*

(Received 26 February 1990; accepted 15 May 1990)



UNO BIOINFORMATICS

Non-Alignment Approaches

UNO BIOINFORMATICS

One Method is not Enough

- Would different objectives of sequence comparison demand different comparison approaches
- Recently, alignment free methods have been approached:
  - Data Compression based approaches
  - Motifs based approaches
  - Composition based approaches and other Statistics based approaches

UNO BIOINFORMATICS

Problems with Alignment

- Optimal alignment is expensive, $O(n^2)$
- Alignment based approach use a very fine grain perspective which may not be suitable for all applications
- Fails to compare long sequences, easy to fool by repetitions and translocations
- It is independent of the input domain
- Inaccurate with incomplete genomes

Alignment-free methods?

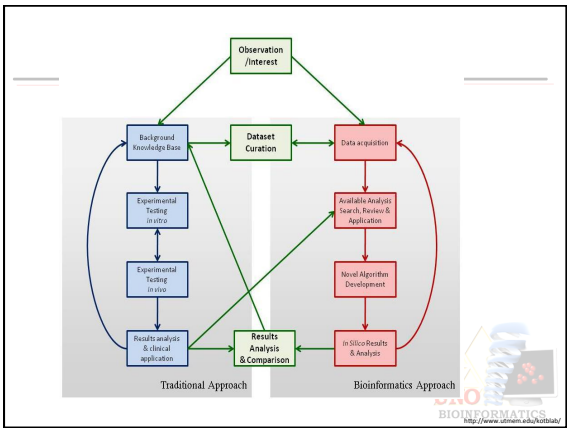
Problems with Multiple Sequence Alignment?

- Multiple sequence alignment of the whole genomes is not feasible
  - Genomic rearrangements, insertion and deletions
  - High resource requirements
  - Long running time
- To compare genomes, specific genes are selected manually and compared across organisms
  - But the process could be manual

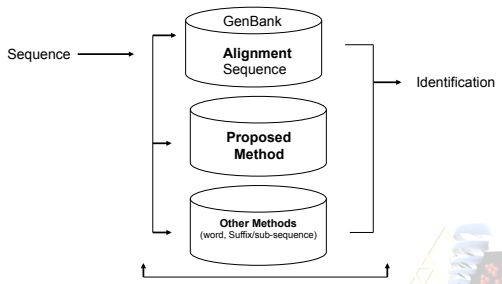
Next Generation Tools

- Dynamic: Custom built and domain dependent
- Collaborative: Incorporate biological knowledge and expertise
- Intelligent: based on a learning model that gets better with additional data/information

Intelligent Collaborative Dynamic (ICD) Tools



Proposed Vision (Integrated Advance Identification System)



Feature Frequency Profile[1]

- The frequency of all possible words (l-mers)
- A sliding window of length 'l' is run
  - All 'l-mers' are counted and tabulated in a vector
    - Feature vector $C_l = < c_{l,1}, \ldots\ldots\ldots\ldots c_{l,k} >$ where k = number of possible features
    - For DNA, alphabet size is 4
      - number of possible features of length $l = 4^l$

[1]Sims, G.E. et al., Alignment-free genome comparison with frequency feature profiles (FFP) and optimal resolutions, PNAS February 2009.

FFP contd..

- Feature Frequency Profile is the Probability distribution vector of feature vector

$$FFP_{i,genome1} = \frac{C_i}{\sum_i C_{i,i}}$$

- FFPs from two genomes are compared using Jensen-Shannon Divergence
  - this metric is used as distance between the genomes

Exercise

- Later in the exercise, we have 6 different bacterial genomes to compare
  - 2 Streptococcus genomes
  - 3 Staphylococcus genomes
  - 1 Lactobacillus genome
- We use FFP to find distance between each of these genomes and draw a phylogeny tree

Compression Based Techniques

- Each sequence is scanned and linearly independent strings are obtained and form a dictionary
- Differences among dictionaries reflects dissimilarity among input sequences
- Repetitions and translocation don't impact the dictionaries as compared to alignment
- For any two sequences x and y, we need
  - $C(x)$, $C(y)$, $C(xy)$ and $C(yx)$.

Example

S = AACGTTACCATTG R = CTAGGGACTTAT
Q = ACGGTCACCAA

$H_E(S)$ = *A/AC/G/T/ACC/AT/TG*
$H_E(R)$ = *C/T/A/G/GGA/CTT/AT*
$H_E(Q)$ = *A/C/G/GT/CA/CC/AA*

- $H_E(SQ)$ = *A/AC/G/T/ACC/AT/TG/ACGG/TC/ACCAA*
- $H_E(RQ)$ = *C/T/A/G/GGA/CTT/AT/ACG/GT/CA/CC/AA*

c(SQ) –c(S)= 3 *c(RQ) –c(R)= 5*

➔ *Q is "closer" to S than R*

Distance (S, Q) = c(SQ)-c(S)+c(QS)-c(Q)

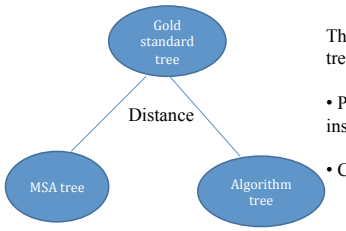
Compression-Based Methods

- The choices
  - Kolomogrov complexity (3 distance measures)
  - Lempel-Ziv complexity (4 distance measures)
  - Clustering
    - UPGMA
    - NJ
  - Gold standard tree
  - Path-length difference

How to evaluate compression Methods?

- Find the distances between each pair of sequences
- Build the evolutionary tree. (compression-based tree)
- Build the evolutionary tree of sequence alignment (alignment-based tree)
- Measure the distance between the two trees to the reference tree
- The closer tree to the reference tree; is a result for an algorithm with higher accuracy

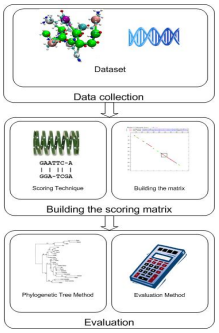
Evaluation of Comparison Approaches



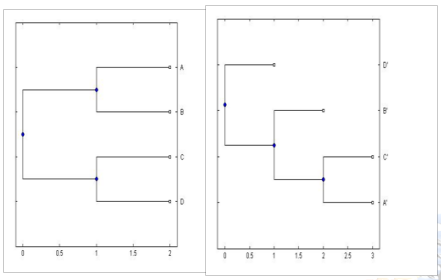
The distance between trees

- Problems with visual inspection!
- Computational ways?
  - Accurate
  - Fast

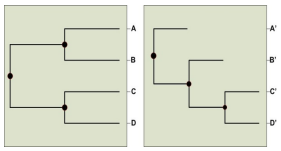
Evaluation, Evaluation



Path-Difference Length

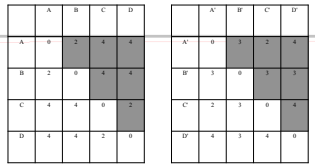


Distance Between Trees



	A	B	C	D
A	0	2	4	4
B	2	0	4	4
C	4	4	0	2
D	4	4	2	0

	A'	B'	C'	D'
A'	0	3	4	4
B'	3	0	3	3
C'	2	3	0	2
D'	4	3	4	0



$(AB – A'B')^2 + (AC - A'C')^2 + (AD - A'D')^2 + (BC - B'C')^2 + (BD - B'D')^2 + (CD – C'D')^2 = (2 - 3)^2 + (4 - 2)^2 + (4 - 4)^2 + (4 - 3)^2 + (4 - 3)^2 + (2 - 4)^2 = 11$

$(AB + AC + AD + BC + BD + CD) = 20.$

the distance between the two trees would be $\sqrt{11 / 20} = 0.1658 = 16.58\%$

Conducted Experiments

- First Experiment:
  - Deal with biological sequences in general, DNA and protein sequences.
  - measure the compression methods against alignment methods
- Second Experiment:
  - Deal with DNA sequences that have specific properties
  - Use sequences with high degree of repetitions
- Third Experiment:
  - Incomplete sequences and several fragments from the sequences
  - Several fragments from the sequences not in order

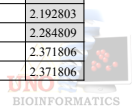
Input Datasets

- The first datasets:
  - CK-36-PDB: Chew-Kedem dataset of 36 protein domains, amino acid sequences in FASTA format.
  - AA-15-DNA: Apostolico dataset of 15 species, mitochondrial DNA complete genomes.
- The second datasets (the mitochondrial DNA complete genomes), as the genomes fragments were extracted using this dataset.

Results

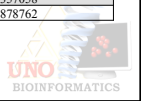
Comparisons of the compression algorithms and multiple sequence alignment for the protein dataset CK-36-PDB

		Protein dataset CK-36-PDB	
Test Algorithm	Variant	Neighbor-Joining	UPGMA
Kolmogorov using Huffman coding	CD	2.395244	3.169468
	NCD	2.328382	2.264505
	UCD	2.328382	2.264505
Kolmogorov using LZW compression	CD	2.176959	2.165911
	NCD	2.210704	2.215544
	UCD	2.305268	2.238781
Lempel - Ziv complexity	Distance 1	2.337454	2.26598
	Distance 2	2.248862	2.192803
	Distance 3	2.244591	2.284809
	Distance 4	2.222918	2.371806
Multiple Sequence Alignment		2.182934	2.371806



Comparisons of the compression algorithms and multiple sequence alignment for the Mitochondrial genome dataset in experiment 1

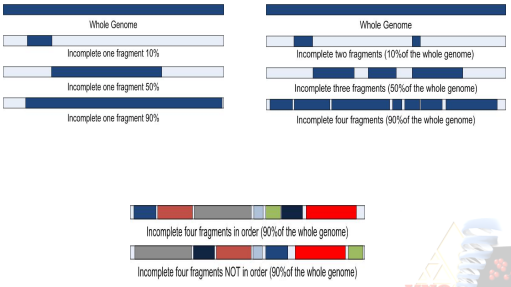
		Mitochondrial Genome dataset AA-15-DNA	
Test Algorithm	Variant	Neighbor-Joining	UPGMA
Kolmogorov using Huffman coding	CD	7.871585	7.871585
	NCD	7.871582	7.871582
	UCD	7.871582	7.871582
Kolmogorov using LZW compression	CD	3.034474	3.034474
	NCD	2.797647	2.797647
	UCD	2.878755	2.878755
Lempel Ziv complexity	Distance 1	1.357058	1.357058
	Distance 2	1.357058	1.357058
	Distance 3	1.357058	1.357058
	Distance4	1.357058	1.357058
Multiple Sequence Alignment		1.5547053	1.878762



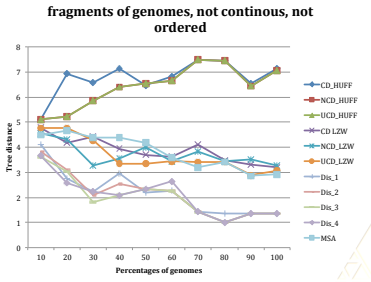
Comparisons of the compression algorithms and multiple sequence alignment for the Mitochondrial genomes

		Mitochondrial Genome dataset of sequecnes with high repetitions of subsequences.	
Test Algorithm	Variant	Neighbor-Joining	UPGMA
Kolmogorov using Huffman coding	CD	12.3431994	17.8482358
	NCD	12.3431994	17.8482337
	UCD	12.3431994	17.8482337
Kolmogorov using LZW compression	CD	10.5262895	10.5263158
	NCD	6.4460256	10.5263158
	UCD	9.8464668	10.5263158
Lempel Ziv complexity	Distance 1	0.0000000	0.0000000
	Distance 2	0.0000000	0.0000000
	Distance 3	0.0000000	0.0000000
	Distance4	0.0000000	0.0000000
Multiple Sequence Alignment		6.4460256	0.0000000

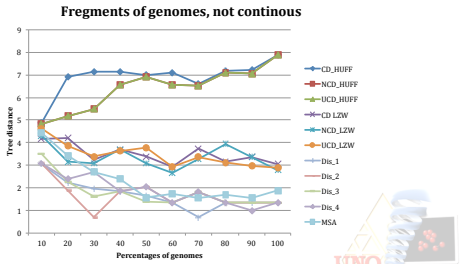
Using Compression to Compare Incomplete Fragments of Genomes



Results of Third Experiment



Using Fragmented Genomes



Analysis of Results

- Compression techniques are more like to cluster genomes with errors, as compression look at these data in a linear fashion rather than in a parallel fashion
- Multiple sequence alignment does not consider the input domain in obtaining similarity measures which limits its use for a diverse input set
- Compression methods identify important signals/ motifs in the input sequences and use them in the process

Summary

- Next Generation Bioinformatics Tools need to be Intelligent, Collaborative, and Dynamic
- Biomedical scientists and Bioinformatics researchers need to work together to best utilize the combination of tools development and domain expertise
- The outcome of such collaboration has the potential of achieving explosive results with significant impact on human health and overall understanding of biological mysteries

Using Restriction Endonuclease Cut Order for Classification and Identification of Fungal Sequences

Sequence Identification/Classification

Current Approaches

1. Computational approach – Pairwise local and Multiple Sequence Alignment
2. Laboratory Method – RFLP, Southern Blotting

Existing Methods - Limitations

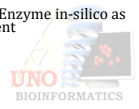
Pairwise or Multiple Alignment

1. Alignment is 'fine-grained' approach
2. More computation intensive and so NP hard for large dataset
3. Introduces gaps - gaps are interpreted as evolutionary events in molecular phylogeny, misaligned sequences have no useful biological information
4. Heuristics like BLAST is employed

Laboratory Methods (RFLP)

1. Only feasible for few sequences
2. Human and procedural error
3. In-silico RFLP methods (TRFLP program) requires Alignment as the second step for sequence identification

Utilize 'coarse-grain-features' of RFLP/Restriction Enzyme in-silico as opposed to the 'fine-grain-features' of alignment computationally.



Restriction Endonuclease

- Proteins that recognize a specific nucleotide subsequence
- Generally 4 to 8 bases long
- Cuts double stranded DNA molecule
- Example:

Hae III recognizes GG ↓ CC site and cuts the product into two fragments as shown - adjacent G and C.

TTTTACGCCCCCTCGAGGCCACCCCCGAAA……GAG (size= 600bp)

TTTTACGCCCCCTCGAGG

Fragment 1
Size = 18bp

CCACCCCCGAAA……GAG

Fragment 2
Size = 600-18=582bp

RFLP

- RFLP = Restriction Fragment Length Polymorphism
  - Widely used laboratory method in molecular identification and Phylogenetic studies.
  - Requires the sequences to be cut into smaller fragments with the help of restriction endonucleases.
  - Variation in the position of these sites along the DNA leads to fragments of varying lengths.
  - High-resolution gel electrophoresis of the digested product, allows visualization of fragment-patterns , which is used to determine the similarity between the sequences.

RFLP

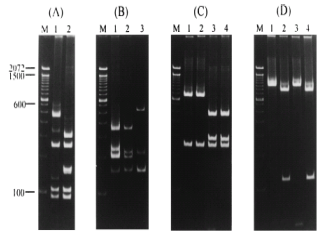


FIG. 2. Restriction digestion patterns of the universal PCR products. (A) *Msl*I digestion patterns of the universal PCR products from *S. aureus* (lane 1) and *S. epidermidis* (lane 2). (B) *Alu*I digestion patterns of the universal PCR products from *S. pneumoniae* (lane 1), *E. faecium* (lane 2), and *E. faecalis* (lane 3). (C) *Dde*I digestion patterns of the universal PCR products from *E. coli* (lane 1), *K. pneumoniae* (lane 2), *S. marcescens* (lane 3), and *E. cloacae* (lane 4). (D) *Rsa*I digestion patterns of the universal PCR products from *E. coli* (lane 1), *K. pneumoniae* (lane 2), *S. marcescens* (lane 3), and *E. cloacae* (lane 4). Lanes M are the same as in Fig. 1.

Enzyme Cut Order (ECO)

- ECO of a DNA sequence S for a particular set of restriction enzymes {Ez} is an ordered sequence of enzymes in the order each enzyme (ez ∈ Ez) cuts the sequence
  - ECO may also include position of nucleotide from the start of sequence where the cut occur.
- ECO is an array of pairs consisting of enzyme id and cut position
- Example:

A part of the sequence is
S = TTTTACGC?GCCCTCGAGG?CCACCCTGG?CCA… …GAG
The Restriction Enzymes are

ENZ ID	ENZ NAME	CUT SITE	CUT POS
1	HaeIII	GGCC	2 (GG | CC)
2	AccII	CGCG	3 (CGC|G)

(Cut sites in blue and red is shown by ?)

The Enzyme cut order = { 2,1,2..}
The Enzyme cut order with position = { (2,8),(1,18),(1,27)..}

Enzyme Cut Order

Acc. Id	Organism	Enzyme Cut Order
OPL416069	*Oligoporus placentas*	33,108,108,209,165,209, 165,165,33,33
OPL249267	*Oligoporus placentas*	33,108,108,209,165,209, 165,165,33
AY310442	*Nectria haematococca*	108,108,209,165,209,165,209, 108,109,108,33,33,108,209,165
AY188918	*Nectria haematococca*	108,108,209,165,209,165,209, 108,109,108,33,33,108,209,165
AY138847	*Nectria mauritiicola*	165,109,108,109,109,209,165,209, 165,209,108,109,108,109,108, 209,109,108,209,165

Observation:
Closely related organisms have similar Enzyme Cut Order

ECO Similarity Score

- The similarity score between two ECO should reflect:
  - Number of similar enzymes and
  - Order in which these enzyme cut the sequence
- Similarity score will be higher if we find larger number of similar enzymes appearing in the same order among two Enzyme Cut Orders.
- Similarity score can be obtained from the Longest Common Subsequence (LCS) among two strings, where the strings are the ECO
  - The length of Longest Common Subsequence (LCS) between two ECO (E1 and E2) of two corresponding sequences (S1 and S2) are considered as the *Enzyme Cut Order Similarity Score* between E1 and E2.

ECO Properties

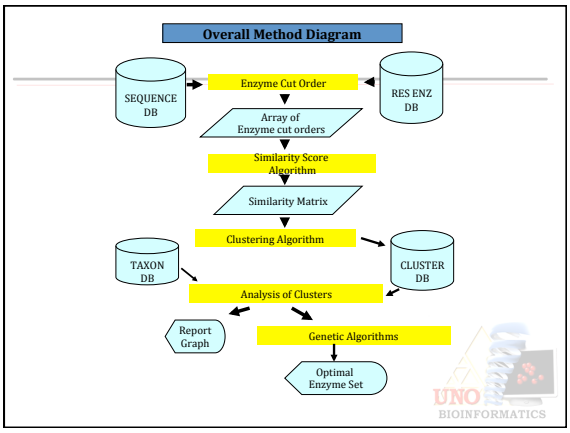
- Enzyme Cut Order is a distinguishing characteristic of DNA sequences
- The similarity between two sequences can be defined by Enzyme Cut Order Similarity Score
- ECO-similarity score can be measured as the length of LCS among the corresponding Enzyme Cut Orders of the DNA sequences of the organisms

Main Hypothesis

- *Organisms closer to each other in the Phylogenetic tree have highly similar Enzyme Cut Order*
- Longest Common Subsequence (LCS) among the corresponding Enzyme Cut Orders is a quantitative measure to determine similarity
- Number of Enzymes used in the analysis determines the granularity of classification

ECO-Based Classification

- *Step 1:* Data Collection
- *Step 2:* Identify Enzyme Set
- *Step 3:* Obtain ECO
- *Step 4:* Calculated ECO Similarity Matrix
- *Step 5:* Clustering
- *Step 6:* Build Phylogenetic Tree



Step 1: Sequence Data Collection and Curation

- Created a local database of GenBank sequences obtained in FASTA or XML format
- Reference these sequences against taxon database
- Create a curated taxonomy database for these sequences using user-defined taxonomical rules
- Fungi ITS Sequences from Genbank
  - Organism description" of the genbank entries (or OrgName_Lineage in XML format)
  - Classification categories included Kingdom, Division, Class, Order, Family, Genus, Species
  - Use simple suffix rule and the position to decide

Suffix	Position	Tax Category
-cota	2nd and After Fungi	Division
-etes	Between Division and Order	Class
-ales	Between Class and Family	Order
-ceae	Before the known Genus	Family

Step 2: Enzyme Data Collection

- Create a database of restriction enzymes obtained from REBASE
- Add more relevant information about these restriction enzymes (Isoschizomers, Commercial availability, Reverse Cutsite) for later potential use
- Appropriate recognition sequence containing bases other than A, T, G and C were interpreted as per IUB ambiguity code (Eur. J. Biochem. 150: 1-5, 1985).

Enz ID	Enzyme Name	Recognition Sequence	Recognition Sequence with IUB conversion	Cut Position (From start)
1	AarI	CACCTGCNNNN^	CACCTGC[ACGT][ACGT][ACGT][ACGT]	12
3	AatI	AGG^CCT	AGGCCT	4
5	AccI, FblI, XmiI	GTMKAC	GT[AC][GT]AC	6
6	AccII, Bsh1236I, MvnI, BstFNI, BstUI, FmuDII	CGCG^	CGCG	4

Step 3: Build Enzyme Cut Order DB

- Obtained Enzyme cut order using user defined set of restriction enzymes {Ez}.
- The Enzyme cutorder is obtained for every test sequences and every enzyme in {Ez}
- Evaluate the effect of the size and type of restriction endonuclease.
  Different sets of (Ez) were chosen with the following properties.
  1. Enzymes that cut at least one of the sequences from the given sequence data
  2. Enzymes that cut 50% of the sequences of the given sequence data
  3. Enzymes that cut all the sequences at least once
  4. Random enzyme set (consisting a mixture from the sets listed previously)
  5. Commonly used restriction enzymes in a biology laboratory working with the RFLP of fungi.

Step 4: Similarity Matrix Score

- Create a similarity matrix for a complete weighted graph for each Enzyme Set {Ez}
  - each node represents one of the input sequences and the weight between two nodes is similarity score SS using the LCS between two corresponding enzyme cut-order
  - G_{Ez} = (V,E) where each node v ϵ V represents a sequence and the weight on each edge $e_{v1,v2}$ ϵ E is the similarity score between the two sequences represented by v1 and v2

Step 5: Clustering

The Similarity matrix is clustered and the cluster is analyzed for its phylogenetic accuracy

Clustering algorithms employed:
- Maximum gap based exclusive clustering
- Hierarchical clustering
- Similarity Clustering
  - Newly developed clustering algorithm
  - Based on node merging and clique cover in the constructed graph

Experiments

- Collect Sequence Data
  - Target: Internal Transcribed Spacer (ITS) region of rRNA
  - Source: GenBank
- Extract taxonomic information form GenBank sequences
- Collect Restriction Endonuclease (RE) information
- Store Sequence and RE information in a local database
- Three sets of data
  1. AspCan: Sequences form the genus *Aspergillus and Candida*
  2. All9Genus (Randomly chosen 9 genera from AllFungi)
  3. AllFungi

Sequence Database

Division	Class	Order	Family	Genus	Species
Basidio mycota	3	22	47	142	551
Asco mycota	11	20	49	137	443
Glomero mycota	1	2	2	2	10
Zygo mycota	1	2	3	4	21
Chytridio mycota	1	1	1	1	1

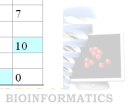
Impact of Using Different Enzyme Sets

- Sequence (Set-1)
  - Type = Internal Transcribed Spacer
  - Size (N) = 7
  - Taxonomy
    - Ascomycota = 5
      - Nectria sp. = 3
      - Lirula sp. = 2
    - Bacidiomycota = 2
      - Oligoporus sp.= 2
- Enzyme (Set1)
  - Example = TaqI , HaeIII, HinfI, AluI, RsaI, MspI)
  - Size (N) = 6
  - Property = Frequent cutter
- Enzyme (Set2)
  - Size (N) = 57

Results using Enzyme Set 1

- All sequences are perfectly clustered
- Similarity Gap is close and reflected on highlighted samples

SqI	Organism	1	2	3	4	5	6	7
1	*Nectria mauritiicola*	0	12	12	11	11	6	6
2	*Nectria haematococca*	12	0	15	9	9	8	8
3	*Nectria haematococca*	12	15	0	9	9	8	8
4	*Lirula macrospora*	11	9	9	0	18	7	7
5	*Lirula macrospora*	11	9	9	18	0	7	7
6	*Oligoporus placentas*	6	8	8	7	7	0	10
7	*Oligoporus placentas*	6	8	8	7	7	10	0



Results using Enzyme Set 2

1. Obtained better Similarity Matrix (Higher Similarity Gap)
2. Larger Enzyme set may have better clustering result
3. All Sequences are perfectly clustered

Acc. ID	Genus	Species							
AY138847	Nectria	mauritiicola	0	103	104	75	73	57	57
AY188918	Nectria	haematococca	103	0	128	72	70	62	62
AY310442	Nectria	haematococca	104	128	0	72	70	60	60
AF462441	Lirula	macrospora	75	72	72	0	123	64	64
AF462440	Lirula	macrospora	73	70	70	123	0	64	64
OPL249267	Oligoporus	placentus	57	62	60	64	64	0	118
OPL416069	Oligoporus	placentus	57	62	60	64	64	118	0

How to Find an Optimal Enzyme Set?

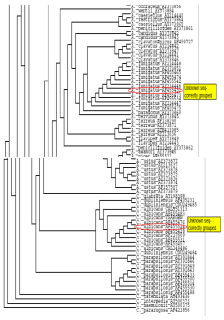
- Optimal enzyme set is defined as the minimal size enzyme set that shows highest phylogenetic resolution
- Find optimal enzyme set for a particular dataset using genetic algorithms
- The Fitness Function is based on the expected and actual count of an organism in the cluster. The score is quantitatively determined in terms of Sensitivity and Positive Predictive Value
- The Selection is either Roulette-wheel selection, tournament selection or random selection
- Uniform, Single-Point or Two-Point crossover is used along with a user specified crossover rate

GA Results

DB (No. Species)	Species Clustered Perfectly	Enzyme Set (Number Of enzymes)
AspCan (43)	31	3,4,6,10,13,15,19,23,24,26,42,48,62,72,76,82,97,99,100,104,109,120,122,149,152,153,160,164,172,177,182,188,197,208,209 (35)
	30	3,4,6,10,24,26,48,72,97,109,122,152,160,164,172,182,197,209 (18)
	29	3,48,72,109,160,172,209 (7) (AatI,Hin6I,BfaI,HpaII,FseI,CviRI,TaqI)
	29	3,48,72,109,172,209 (6) (AatI,Hin6I,BfaI,HpaII,CviRI,TaqI)
	28	3,48,109,172 (4) (AatI,Hin6I,HpaII,CviRI)
All9Genus (26)	21	3,4,6,13,19,22,24,28,33,39,41,47,50,57,58,64,71,75,82,93,97,100,108,109,122,139,142,143,147,149,164,177,183,194,206,209 (36)
	19	3,6,24,33,39,47,64,82,97,100,108,109,122,143,183,206 (16)
	19	3,6,24,33,39,47,64,97,108,109,143,183,206 (13)
	19	6,33,39,47,97,108,109,183,206 (9)
	18	6,47,108,109,183 (5) (AccII, AspCNI, HaeIII,HpaII, MseI)

BIOINFORMATICS

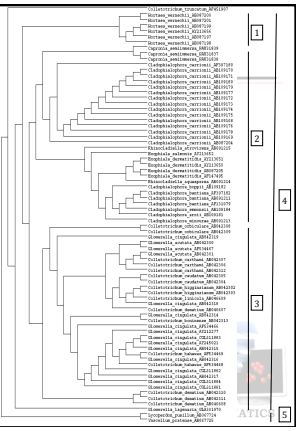
Identification of Unknown Sequences



UNO BIOINFORMATICS

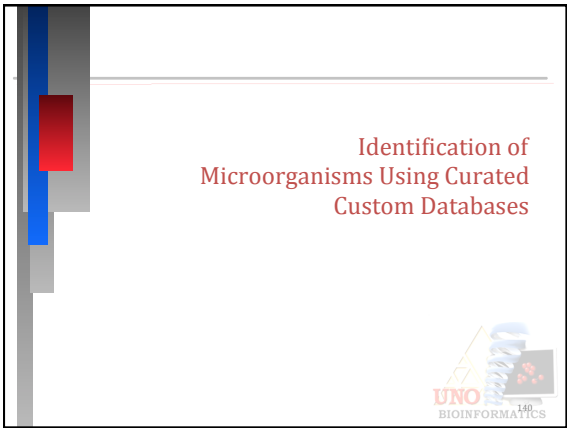
Clustering of Sequences from All9Genus DB

Division-level taxonomy
4= Ascomycota
5= Basidiomycota
Class-level taxonomy
1= Dothideomycetes
2= Chaetothynomycetes
3= Sordaromycetes



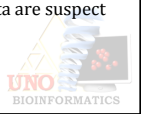
Summary

- The inherent biological property of Restriction enzymes to recognize specific DNA a sequence is a valuable parameter in the analysis of DNA sequence
- The order in which multiple RE cut multiple sequences can be modeled to classify set of DNA sequences at varying degree of detail.
- The length of LCS is a quantitative measure to determine similarity between sequences
- The proposed alignment-free approach provides an alternative method for sequence identification and classification



The Sequence Identification Problem

- Identification of organisms using obtained sequences is a very important problem
- Relying on wet lab methods only is not enough
- Employing identification algorithms using signature motifs to complement the experimental approaches
- Currently, no robust software tool is available for aiding researchers and clinicians in the identification process
- Such a tool would have to utilize biological knowledge and databases to identify sequences
- Issues related to size of data and quality of data are suspect and would need to be dealt with



Problem Definition and Motivation

- Identification of organisms using obtained sequences is a very important problem
- Relying on wet lab methods only is not enough
- Employing identification algorithms using signature motifs complement the experimental approaches
- Currently, no robust software tool is available for aiding researchers and clinicians in the identification process
- Such tool would have to utilize Biological knowledge and databases to identify sequences
- Issues related to size of data and quality of data is suspect would need to be dealt with

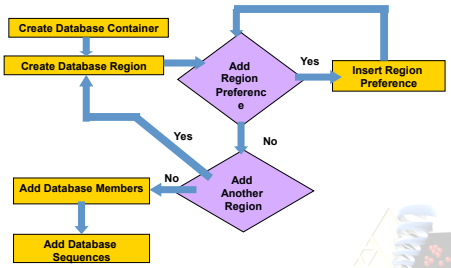
The Computational Approach

- Sequence similarity and graph clustering are employed to identify unknown sequences
- Earlier results were not conclusive
- Local similarity in specific regions rather than global similarity is used, in particular, test validity of identifying *Mycobacterium* based on ITS region and 16S region
- Graph Clustering based on region similarity produced very good results, particularly when using ITS region
- Grammar based description of selected regions is used for identification

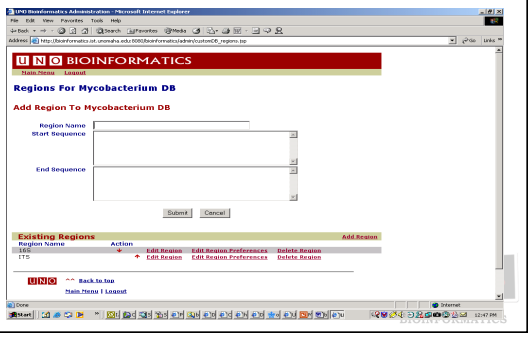
Custom Databases

- Allowing researchers to create custom sets of genetic data suited to their specific needs.
- Allowing researchers to control the quality of genetic data in their custom data sets through fine-tuning parameters.
- Searching data using optimal alignment algorithms, rather than using heuristic methods.
- Giving researchers/clinicians the ability to formulate sequence identification concepts and test their ideas against a validated database
- Incorporating information from GenBank if needed

Creating a Custom Database



Create Custom Database Regions



How to Define Region Preferences

- Simple Definition
  - Letters (ACGT)
  - Wild Card (N)
  - Limits (wild cards, mismatches, Region Size)
- Grammar Based Definition
  - Employs regular expression for flexible region definitions
  - Powerful and Robust but a bit more complex

Regular Expressions

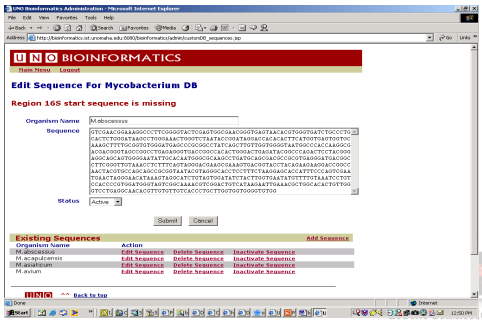
Expression	Expression Name	Function
Metacharacter Matches		
.	*wildcard match*	Match any one character
[...]	*character class*	Match any character inside braces
[^...]	*negated character class*	Match any character not listed
\char	*escaped character*	Matches to a terminal (e.g. \! becomes !)
Counting Modulators		
?	*question*	One subsequence allowed, optional
+	*plus*	One required, more optional
*	*star*	Any number allowed, but optional
{min, max}	*specified range*	min required, max allowed
Position Matchers		
^	*caret*	Matches position at start of word
$	*dollar*	Matches position at end of word
\<	*word boundary*	Matches position at beginning of word
\>	*word boundary*	Matches position at end of word
Clarification and Flexibility Operators: Topology Line Operators		
|	*alternation*	Matches subsequence before or after pipe
(,)	*parentheses*	Logically groups subsequences
\1,\2,...	*back-reference*	Matched pattern must occur again

Table 1: RegEx functionality

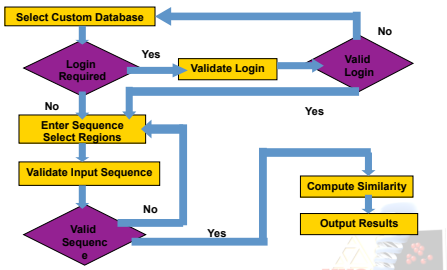
BioRegEx

Expression	Expression Name	Function
Metacharacter Matches		
.	*wildcard match*	Match any one character
[...]	*character class*	Match any character inside braces
[.\%.\%.\%]	*percentage character class*	Replace ρ with user defined match costs
[^...]	*negated character class*	Match any character not listed
[^.\%.\%.\%]	*negated percentage character class*	Replace ρ with user defined Match costs. Characters listed match at 1 - *percantageCost*
Counting Modulators		
?	*question*	One subsequence allowed, optional
+	*plus*	One required, more optional
*	*star*	Any number allowed, but optional
{min, max}	*specified range*	min required, max allowed
Position Matchers		
^	*caret*	Matches position at start of line
$	*dollar*	Matches position at end of line

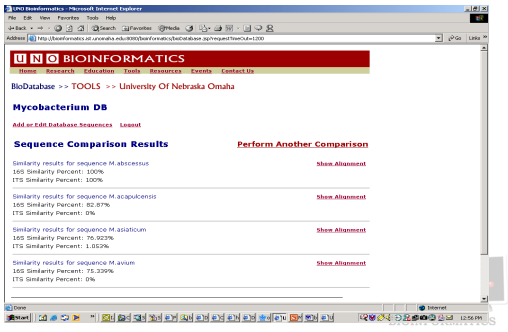
Sequence Validation



Searching in a Custom Database



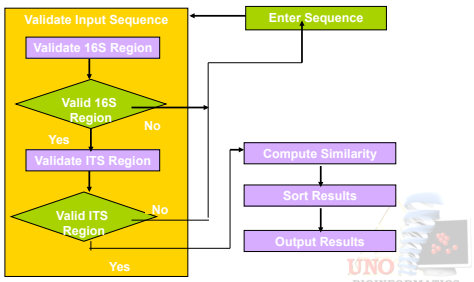
Interpreting Search Results



The Mycobacterium Case Study

- 30 species associated with variety of human and animal diseases such as tuberculosis
- Certain pathogenic species specific to humans. Some only affect animals
- Certain pathogenic species are drug-resistant
- Laboratory identification slow, tedious, and error-prone
- Sequencing provides an alternative to laboratory methods
- Researchers wanted to test validity of identifying *Mycobacterium* based on ITS region and 16S region

Case Study: *Mycobacterium*



Results of Case Study

- Identified *Mycobacterium* to species & strain level
- 72 of 78 previously identified isolates identified
- 6 remaining may be new strains
- MAC-A correctly identified in BioDatabase.
- MAC-A mistakenly first identified as *M.malmoense* using NCBI BLAST against GenBank
- Highlights problems with GenBank data and BLAST heuristics

Nebraska gets its very own organism

- While trying to pinpoint the cause of a lung infection in local cancer patients, they discovered a previously unknown micro-organism. And they've named it "mycobacterium nebraskense," after the Cornhusker state.
- It was discovered few weeks ago using Mycoalign: A Bioinformatics program developed at PKI

Source: Omaha World Herald,

Clustering and Phylogeny

Clustering, Arrays and Trees

- Clustering is a key step in several Bioinformatics problems, primarily:
  - Analysis of Microarray data; and
  - Building evolutionary trees
- Many versions of the clustering problem are NP-hard, near-optimal solutions can be obtained using smart heuristics.
- A good example is Hierarchical clustering that assumes a constant molecular clock (rate of evolution) along all branches of the tree. Two closest sequences are clustered first, then next two closest, etc. A rooted tree is produced.

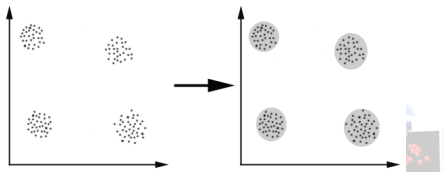
Data Analysis Cycle

- Data Generation and Collection
- Data Access, Storage and Retrieval
- Data Integration
- Data Visualization
- Analysis and Data Mining
- Decision Support
- Validation and Discovery

What Is Clustering?

A *cluster* is a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

It is about hiding information?



Clustering Problem Formulation

- Let $O = \{O_1, O_2, \ldots, O_n\}$ be a set of n objects, and let $C = \{C_1, C_2, \ldots, C_k\}$ be a partition of O into k subsets such that $\cup_{i=1}^{k} Ci = O$ and $Ci \cap Cj = \phi$
- C_i is called a cluster, and C is a clustering solution.
- The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. The data should be homogeneous and well separated.

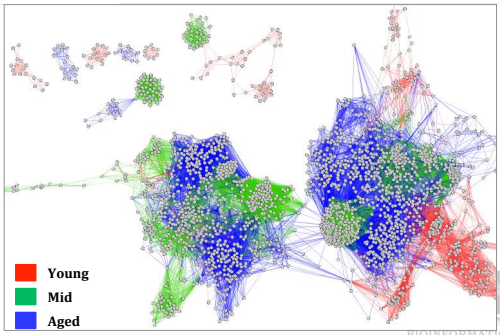
How to decide what constitutes a good clustering?

- No absolute "best" criterion which would be independent of the final aim of the clustering.
- Consequently, the user must supply this criterion, in such a way that the result of the clustering will suit their needs.

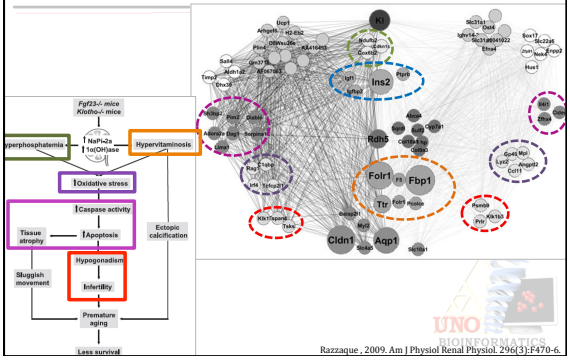
Applications of Clustering

- Viewing and analyzing vast amounts of biological data as a whole set can be perplexing
- It is easier to interpret the data if they are partitioned into clusters combining similar data points
- Importance of clustering is analyzing microarray data, regulation data, and network analysis data

Network Analysis in Aging

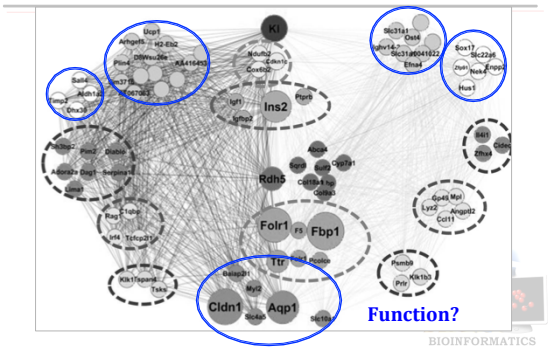


Validation



Razzaque , 2009. Am J Physiol Renal Physiol. 296(3):F470-6.

Discovery



Inferring Gene Functionality

- Researchers want to know the functions of newly sequenced genes
- Simply comparing the new gene sequences to known DNA sequences often does not give away the function of gene
- For 40% of sequenced genes, functionality cannot be ascertained by only comparing to sequences of other known genes
- Microarrays allow biologists to infer gene function even when sequence similarity alone is insufficient to infer function.
- Genome wide variants is quickly emerging as a key alternative to obtain correlation relationships

Design Microarray Experiments

- We can conclude the basic steps for designing a microarray experiment:

1. Prepare DNA chip using your chosen target DNAs.
2. Generate a hybridization solution containing a mixture of fluorescently labeled cDNAs.
3. Incubate hybridization mixture containing fluorescently labeled cDNAs with your DNA chip.
4. Detect bound cDNA using laser technology and store data in a computer.
5. Analyze data using computational methods.

Using Microarrays

- **Green**: expressed only from control
- **Red**: expressed only from experimental cell
- **Yellow**: equally expressed ir both samples
- **Black**: NOT expressed in either control or experimental cells

Microarray Data

- Microarray data are usually transformed into an intensity matrix
- The intensity matrix allows biologists to make correlations between different genes (even if they are dissimilar) and to understand how genes functions might be related

Intensity (expression level) of gene at measured time

Time:	Time X	Time Y	Time Z
Gene 1	10	8	10
Gene 2	10	0	9
Gene 3	4	8.6	3
Gene 4	7	8	3
Gene 5	1	2	3

Clustering of Microarray Data

- Plot each datum as a point in N-dimensional space
- Make a distance matrix for the distance between every two gene points in the N-dimensional space
- Genes with a small distance share the same expression characteristics and might be functionally related or similar.
- Clustering reveal groups of functionally related genes

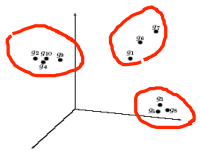
Clustering of Microarray Data

Time	1 hr	2 hr	3 hr
g_1	10.0	8.0	10.0
g_2	10.0	0.0	9.0
g_3	4.0	8.5	3.0
g_4	9.5	9.5	8.5
g_5	4.5	8.5	2.5
g_6	10.5	9.0	12.0
g_7	5.0	8.5	11.0
g_8	2.7	8.7	2.0
g_9	9.7	2.0	9.0
g_{10}	10.2	1.0	9.2

(a) Intensity matrix, I

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

(b) Distance matrix, d

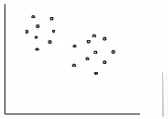


(c) Expression patterns as points in three-dimensional space.

Homogeneity and Separation Principles

- **Homogeneity:** Elements within a cluster are close to each other
- **Separation:** Elements in different clusters are further apart from each other
- …clustering is not an easy task!

Given these points a clustering algorithm might make two distinct clusters as follows



Bad Clustering

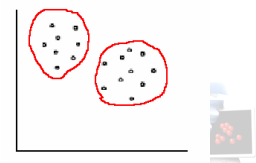
This clustering violates both Homogeneity and Separation principles

Close distances from points in separate clusters

Far distances from points in the same cluster

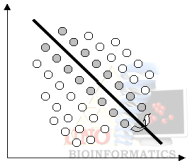
Good Clustering

This clustering satisfies both Homogeneity and Separation principles



Hard and Fuzzy Clustering

- Hard clustering (exclusive) -- each object is assigned to one and only one cluster.
  - The separation of points is achieved by a straight line
  - Much more popular than fuzzy clustering
- Fussy clustering (overlapping) -- each object can belong to more than one clusters with different degrees of membership .
  - Add membership function; data will be associated to an appropriate membership value



Distance (Similarity) Measure

- Clustering algorithms are distance (similarity) based.
- Correlation coefficient: a similarity measure.

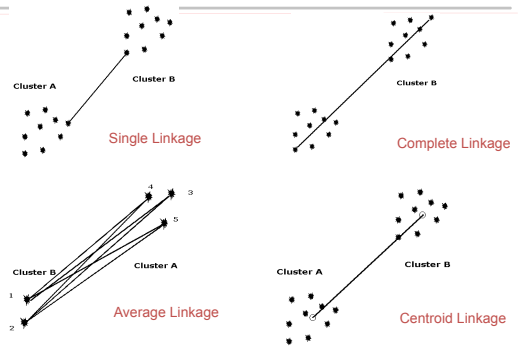
$$S(X,Y) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{Xi - \overline{X}}{\Phi_X}\right)\left(\frac{Yi - \overline{Y}}{\Phi_Y}\right) \qquad \Phi_X = \sqrt{\sum_{i=1}^{m}\frac{(Xi-\overline{X})^2}{m}}$$

$$\overline{X} = \sum_{i=1}^{m} Xi / m$$

- Euclidean distance: a dissimilarity measure.

$$D(X,Y) = \sqrt{\sum_{i=1}^{m}(Xi - Yi)^2}$$

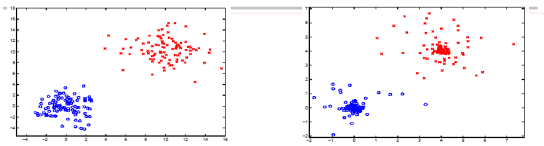
Similarity Criteria between Clusters



Problems of Distance Measure

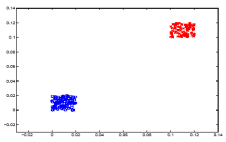
- Each distance measure has the advantage and the disadvantage.
- Different formulas lead to different clustering results.
- Domain knowledge must be used to guide the formulation of a suitable distance measure for each particular application.

Examples



Well separated but not particularly homogeneous clusters

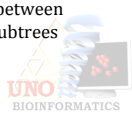
Homogeneous but not well separated clusters



The ideal situation

Clustering Techniques

- Agglomerative: Start with every element in its own cluster, and iteratively join clusters together
- Divisive: Start with one cluster and iteratively divide it into smaller clusters
- Hierarchical: Organize elements into a tree, leaves represent genes and the length of the paths between leaves represents the distances between genes. Similar genes lie within the same subtrees



Clustering Methods

- Hierarchical methods (Hierarchical clustering, Super-paramagnetic clustering, Message Passing Clustering)
- Partition methods (K-means)
- Graph theoretic methods (CLICK, CAST)
- Neural network approaches (SOM)
- Various bi-clustering algorithms
- Other approaches:
  - Bayesian clustering, Matrix tree incision, Spectral clustering,…
- Clustering results can be different for different methods and distance metrics

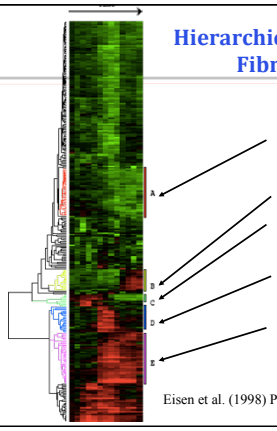
Agglomerative hierarchical clustering algorithm

1. Start by assigning each item to a cluster, so that if you have *N* items, you now have *N* clusters, each containing just one item.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size *N*.

Note: * There is no point in having all the *N* items grouped in a single cluster but, once you have got the complete hierarchical tree, if you want *k* clusters you just have to cut the *k*-1 longest links (branches).

* Different distance measures (single, complete, average or centroid linkage) may result in different clustering dendrogram.

Hierarchical Clustering Example: Fibroblast clustering

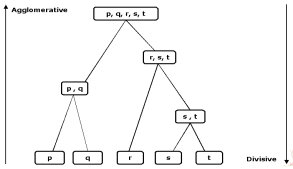


- Cholesterol biosynthesis
- Cell cycle
- Immediate-early response
- Signaling and angiogenesis
- Wound healing and tissue remodeling

Eisen et al. (1998) PNAS 95:14863.

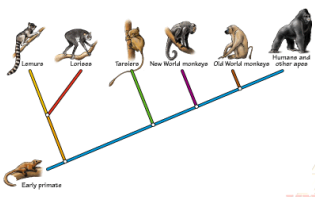
Hierarchical Clustering

- Merging (agglomerative): start with every instance as a separate cluster then combine until only one cluster remains.
- Splitting (Divisive): make one large cluster, then split up into smaller pieces until each single instance is in a cluster.



Hierarchical Clustering

- Hierarchical Clustering is often used to reveal evolutionary history



Hierarchical Clustering Algorithm

1. Hierarchical Clustering (***d*** , *n*)
2. Form *n* clusters each with one element
3. Construct a graph ***T*** by assigning one vertex to each cluster
4. **while** there is more than one cluster
5. Find the two closest clusters C_1 and C_2
6. Merge C_1 and C_2 into new cluster C with $|C_1| + |C_2|$ elements
7. **Compute distance from *C* to all other clusters**
8. Add a new vertex ***C*** to ***T*** and connect to vertices C_1 and C_2
9. Remove rows and columns of ***d*** corresponding to C_1 and C_2
10. Add a row and column to ***d*** corrsponding to the new cluster ***C***
11. return ***T***

The algorithm takes a *n*x*n* distance matrix ***d*** of pairwise distances between points as an input.

Different ways to define distances between clusters may lead to different clustering

K-Means Clustering

- **Input**: A set, ***V***, consisting of *n* points and a parameter *k*
- **Output**: A set ***X*** consisting of *k* points (*cluster centers*) that minimizes the squared error distortion *d(**V**,**X**)* over all possible choices of ***X***

K-Means

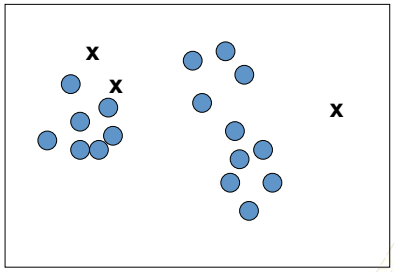
- Iteratively solve for optimal cluster centers and partitions
- Need to know the number of clusters ahead of time
- Function optimization method that iterates so that objects within clusters are most similar
- Usually uses Euclidean distance
- Solution is not unique, clustering can depend on your starting point

K-Means Algorithm

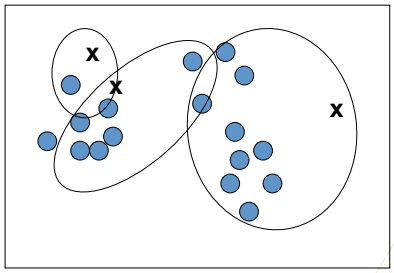
1. Choose *K* initial group (cluster) centers at random.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the *K* centroids for *K* clusters.
4. Repeat Steps 2 and 3 until the centroids no longer move.

➢ This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

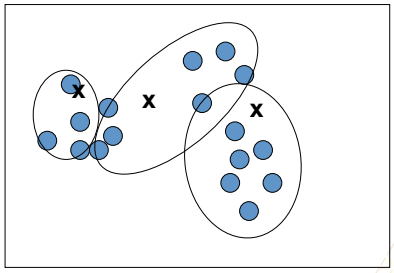
Example (k=3) Initialization



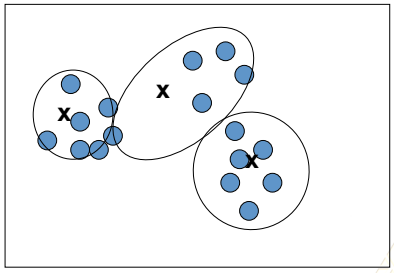
Iteration 1



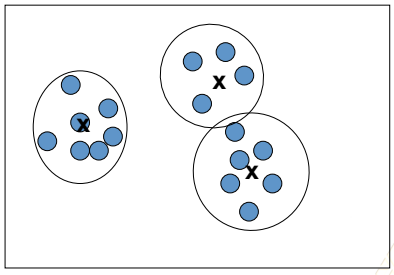
Iteration 2



Iteration 3



Iteration 4



Comparison of clustering algorithms

- Hierarchical clustering
  - + Widely used.
  - + Easy to understand.
  - + Does not require the number of clusters *a priori.*
  - – Difficult to implement well.
  - – Requires post-processing.
  - – Unstable.
  - – Greediness can lock in early mistakes.
  - – There is no reason to think that expression data is organized hierarchically.

Comparison of clustering algorithms

- k-means
  - – Less widely used.
  - + Easy to understand.
  - – Requires the number of clusters *a priori.*
  - + Easy to implement.
  - + Scales well.
  - + Stable.
  - – Creates unorganized cluster that are hard to interpret.

Bi-Clustering

- The term "Biclustering" was first used by Cheng and Church in gene expression data analysis.
- Gene expression data or expression data
- Data Matrix
  - – Each gene – One row
  - – Each condition – One column
  - – Each element – expression level of a gene under specific condition

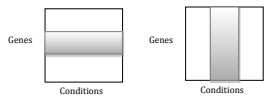
Biclustering

- The term "Biclustering" was first used by Cheng and Church in gene expression data analysis [Year 2000]
- Clusters do not need to include all parameters (genes in Bioinformatics) for all conditions
- Data Matrix
  - Each gene – One row
  - Each condition – One column
  - Each element – expression level of a gene under specific condition

Clustering Versus Biclustering

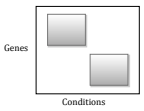
- Clustering –
  - Applied to either rows or columns of the data matrix separately
  - Each gene is defined using all the conditions
  - Each condition is characterized by the activity of all the genes that belong to it
- Biclustering –
  - performs clustering in these two dimensions simultaneously
  - Each gene is selected using only a subset of the conditions
  - Each condition is selected using only a subset of the genes

Traditional Clustering



- Applied to either rows or columns of the data matrix separately
- Each gene is defined using all the conditions
- Each condition is characterized by the activity of all the genes that belong to it

Biclustering (Cont.)

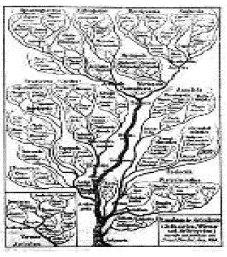


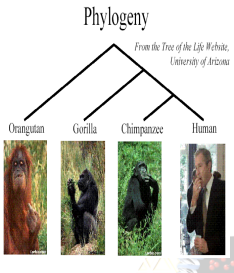
- Performs clustering in these two dimensions simultaneously
- Each gene is selected using only a subset of the conditions
- Each condition is selected using only a subset of the genes

Phylogeny

- Science of estimating the evolutionary past
- Phylogenetic analysis is the means used to estimate evolutionary relationships based on observable evidence
- Evidence can include morphology, physiology, and other properties of organisms. Paleontological and geological evidence is also used.
- Linnaeus's system of grouping and naming organisms to reflect evolutionary relationship- Phenotype Phylogenetic

Phylogenetic Trees





Phylogenetic Analysis

Relationship among species

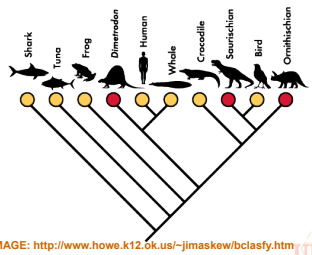


IMAGE: http://www.howe.k12.ok.us/~jimaskew/bclasfy.htm

Exercise Day 2

- Sequence Comparison using alignment and non-alignment approaches
- Use the comparison scores to cluster a set of organisms and build a phylogenetic Trees