Hierarchical clustering implementation
• **Single linkage (nearest neighbor):** In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.

• **Complete linkage (furthest neighbor):** In this method, the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors").

• **Group average linkage:** In this method, the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.
### Single-Link Hierarchical Clustering

**Iteration.**
- Closest pair of clusters \((i, j)\) is one with the smallest \(dist\) value.
- Replace row \(i\) by min of row \(i\) and row \(j\).
- Infinity out row \(j\) and column \(j\).
- Update \(d_{min}[i]\) and change \(d_{min}[i']\) to \(i\) if previously \(d_{min}[i'] = j\).

<table>
<thead>
<tr>
<th>dmin</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gene0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>5.5</td>
<td>7.3</td>
<td>8.9</td>
<td>5.8</td>
</tr>
<tr>
<td>1</td>
<td>5.5</td>
<td>-</td>
<td>6.1</td>
<td>2.14</td>
<td>5.6</td>
</tr>
<tr>
<td>2</td>
<td>7.3</td>
<td>6.1</td>
<td>-</td>
<td>7.8</td>
<td>5.6</td>
</tr>
<tr>
<td>3</td>
<td>8.9</td>
<td>2.14</td>
<td>7.8</td>
<td>-</td>
<td>5.5</td>
</tr>
<tr>
<td>4</td>
<td>5.8</td>
<td>5.6</td>
<td>5.6</td>
<td>5.5</td>
<td>-</td>
</tr>
</tbody>
</table>

Closest pair: \((0, 1)\), \(dist = 1.55\)

Gene1 closest to gene3, \(dist = 2.14\)

New min dist: \((1, 3)\), \(dist = 2.14\)
Single-Link Clustering: Java Implementation

Single-link clustering.
• Read in the data.

```java
public static void main(String[] args) {
    int M = StdIn.readInt();
    int N = StdIn.readInt();

    // read in N vectors of dimension M
    Vector[] vectors = new Vector[N];
    String[] names = new String[N];
    for (int i = 0; i < N; i++) {
        names[i] = StdIn.readString();
        double[] d = new double[M];
        for (int j = 0; j < M; j++)
            d[j] = StdIn.readDouble();
        vectors[i] = new Vector(d);
    }
```
Single-Link Clustering: Java Implementation

Single-link clustering.
- Read in the data.
- Precompute $d[i][j] = \text{distance between cluster } i \text{ and } j$.
- For each cluster $i$, maintain index $d_{\text{min}}[i]$ of closest cluster.

```java
double INFINITY = Double.POSITIVE_INFINITY;
double[][] d = new double[N][N];
int[] dmin = new int[N];
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        if (i == j) d[i][j] = INFINITY;
        else d[i][j] = vectors[i].distanceTo(vectors[j]);
        if (d[i][j] < d[i][dmin[i]]) dmin[i] = j;
    }
}
```
Single-Link Clustering: Main Loop

```cpp
for (int s = 0; s < N-1; s++) {
  // find closest pair of clusters (i1, i2)
  int i1 = 0;
  for (int i = 0; i < N; i++)
    if (d[i][dmin[i]] < d[i1][dmin[i1]]) i1 = i;
  int i2 = dmin[i1];

  // overwrite row i1 with minimum of entries in row i1 and i2
  for (int j = 0; j < N; j++)
    if (d[i2][j] < d[i1][j]) d[i1][j] = d[j][i1] = d[i2][j];
  d[i1][i1] = INFINITY;

  // infinity-out old row i2 and column i2
  for (int i = 0; i < N; i++)
    d[i2][i] = d[i][i2] = INFINITY;

  // update dmin and replace ones that previous pointed to
  // i2 to point to i1
  for (int j = 0; j < N; j++) {
    if (dmin[j] == i2) dmin[j] = i1;
    if (d[i1][j] < d[i1][dmin[i1]]) dmin[i1] = j;
  }
}
```
Store Centroids in Each Internal Node

Cluster analysis. Centroids distance / similarity.

Easy modification to TreeNode data structure.

- Store Vector in each node.
  - leaf nodes: directly corresponds to a gene
  - internal nodes: centroid = average of all leaf nodes beneath it
- Maintain count field in each TreeNode, which equals the number of leaf nodes beneath it.
- When setting z to be parent of x and y,
  - set z.count = x.count + y.count
  - set z.vector = αp + (1−α)q, where p = x.vector and q = y.vector, and α = x.count / z.count
Analysis and Micro-Optimizations

Running time. Proportional to $MN^2$ (N genes, M arrays)
Memory. Proportional to $N^2$.

Ex. $[M = 50, N = 6,000]$ Takes 280MB, 48 sec on fast PC.

Some optimizations.
- Use float instead of double
- Store only lower triangular part of distance matrix
- Use squares of distances instead of distances.

How much do you think would this help?
Sequence!

Some slides from Mona Singh, Serafim Batzoglou, Olga Troyanskaya
Bio-Sequences

Complete genomes of >1000 organisms


> 100 billion bases in Genbank (ncbi)

>509,000 proteins in SWISSPROT (hand curated); >9,300,000 proteins in TREMBL (computer annotated).

us.expasy.org/sprot
Next Gen Sequencers

Illumina/Solexa High Throughput Sequencing Machine

>20 billion bases per run

Illumina’s Spring 2009 charge for sequencing your genome:
  $48,000 - 30 fold coverage
Biomolecules as Strings

Macromolecules are the chemical building blocks of cells

- **Proteins**
  - 20 amino acids

- **Nucleic acids**
  - 4 nucleotides \{A, C, G, T\}
Role of Evolution

Molecular structures and mechanisms are reused and changed during evolution.

Often mechanisms that are conserved can be detected based on sequence similarity.

Powerful tool for annotation.
Ex: Protein Sequences

Horse vs Human Myoglobin  *(Global alignment of sequences)*

GLSDGEWQQVLNVWGKVEADIAGHGQEVILIRLFTGHPETLEKFKDFKHLKTEAEMKASED
GLSDGEWQLVLNVWGKVEADIPGHGQEVILIRLFKGHPETLEKFKDFKHLKSEDEMKASED

LKKHGTVVLTALGGILKKGKGHHEAEELKPLAQSHATKHKIPIKYLEFISDAIIVHLHSKHP
LKKHGATVLTALGGILKKGKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVQLQSKHP

GDFGADAQGAMTKALELFRNDIAAKYKELGFQG
GDFGADAQGAMNKALELFRKDMASNYKELGFQG

Same protein in two different organisms, can ID based on sequence similarity - 88% identical

Myoglobin - intracellular storage of oxygen
Global alignment: Issues with transferring annotations

Horse Myoglobin vs Human Hemoglobin Alpha

MGLSDGEWQQVLNVWGKVEADIAHGQEVLIRLFTGHPETLEKFDKFKHLKTEAEMKASEDL
MVLSPADKTNVAWGKVGAHAGEYGAEALEERMFLSFTPTKTYFPHTFDLSHGSAQVKG----

KKHGTVVLTALGGILKKKGHEAEKLPLAQSHATKHKIPIKYLEFISDAIITHVLSHKSHPG
--HGKKVADALTNAVAHVDDMPNALSALESDDLHAKLRVDPVNFKLLSHCLLVTLAAHLPA

DFGADAQGAMTKALELFNRNDIAAKYKEGLGFQG
EFTPAVHASLDKFASVSTVLTSKYR------

~25% identical; other “similar” amino acids
Myoglobin - intracellular storage of oxygen
Hemoglobin - transports oxygen
Basic Tool to Detect Sequence Similarity: Alignments

Given:
• a pair (or more) of sequences (DNA or protein)
• a method for scoring the similarity of a pair of characters (=bases or amino acids)

Determine: correspondences between characters in the sequences such that the similarity score is maximized
Pairwise global alignment

Given two sequences, a scoring scheme with a gap function, line up the sequences (with insertion of gaps) to maximize the score.

E.g., match = 1
mismatch = -1
gap = -2

E.g., say your two sequences are
AACAGTTACC, TAAGGTCA

AACAGTTACC
TA-AGGT-CA

Score = ?
Naïve way to find optimal alignments

1. Enumerate all possible alignments
2. Score all possible alignments
3. Take best scoring alignment
4. Problem: There are too many possible alignments between 2 sequences !!
5. Solution: dynamic programming
   • RECALL: homework assignment from last term!
Pairwise Alignment

Needleman & Wunsch, *Journal of Molecular Biology*, 1970

Dynamic programming (DP): general technique to solve an instance of a problem by taking advantage of computed solutions for smaller subparts of the problem.

Here, determine alignment of two sequences by determining alignment of all suffixes of the sequences.

- (suffixes are subparts we'll save solutions for... )
Dynamic Programming Idea

Say aligning **AAAC** with **AGC**

Consider what happens in the first column

Three possible options; each corresponds to different alignment of first column, choose each one and add this to best alignment of suffixes

Score of aligning these characters

Consider best Alignment of these suffixes
Dynamic Programming Idea

If we knew answers to these *three* subproblems, then we'd know the best alignment score between AAAC and AGC.

Consider minimum of these *three* cases.
Dynamic Programming Idea

Given an $m$-character sequence $s$, and an $n$-character sequence $t$ construct an $(m+1) \times (n+1)$ matrix $\text{sim}$ where we’ll store answers to subproblems

$$\text{sim}[i, j] = \text{score of the best alignment of the suffix } i...m \text{ of } s \text{ with the suffix } j...n \text{ of } t.$$
Aligning \textit{AAAC} with \textit{AGC}

Best alignment score of \textit{AAAG} with \textit{C}

Best alignment score of \textit{AC} with \textit{GC}
Dynamic Programming Rule

\[
sim[i, j] \rightarrow \sim[i, j+1] \\
\sim[i+1, j] \rightarrow \sim[i+1, j+1]
\]

\[(\text{gap cost}) + g + \text{sc}(s[i], t[j])\]

(similarity score between \(s[i]\) and \(t[j]\))
How long does DP take?

Target sequence of length m

Dynamic programming matrix
How long does DP take?

There are $nm$ entries in the matrix.

Target sequence of length $m$

Dynamic programming matrix

Query sequence of length

Each entry requires a constant number $c$ of operations.

The total number of required operations is approximate $nmc$. We say that the algorithm is “order $nm$” or “$O(nm)$.”
Local Alignment

Just described global alignment, where we are looking for best match between sequences from one end to the other.

Often (and more commonly), we will want a local alignment, the best match between subsequences of $s$ and $t$. 
Local Alignment DP Algorithm


Interpretation of array values is different from global sequence alignment

\[ \text{sim}[i, j] = \text{score of the best alignment of a prefix of the } i..m \text{ suffix of } s \text{ and a prefix of the } j..n \text{ suffix of } t \]

Algorithm is simple modification of DP just described - whenever score goes below 0, start from scratch!

I.e., consider four cases and take max
Database search

Given a sequence of interest, can you find other similar sequences (to get a hint about structure/function)?

• E.g, NCBI BLAST site
  • Input sequence, gives back all significant sequence matches
  • Performs local alignments
Heuristic Methods for Sequence Database Searching

Quadratic algorithm too slow for large databases with high query traffic heuristic methods do fast approximation to dynamic programming

• FASTA [Pearson & Lipman (1988) PNAS 85, p2444]
  • http://www2.ebi.ac.uk/fasta3

• BLAST [Altschul et al. (1990) JMB 215, p403]
  • http://www.ncbi.nlm.nih.gov/BLAST
Speeding up searches

Give up optimality, use heuristics

For a query sequence, require its matches to share a k-mer exactly (e.g., k=11)

Fundamental innovation: use hashing (or other search data structures) to find (quickly) places in database where each k-mer in the query sequence occurs
BLAST algorithm

• Remove low-complexity regions.
• Make a list of all words of length 3 amino acids or 11 nucleotides.
• Augment the list to include similar words.
• Scan the database for occurrences of the words
• Connect nearby occurrences.
• Extend the matches.
• Prune the list of matches using a score threshold.
• Evaluate the significance of each remaining match.
  • Very important!
• Perform Smith-Waterman to get an alignment.
BLAST Notes

May fail to find all high-scoring segment pairs
- Heuristic approach

Empirically, more than an order of magnitude faster than Smith-Waterman

Large impact:
• NCBI’s BLAST server handles thousands of queries a day
• Most used (and cited) bioinformatics program