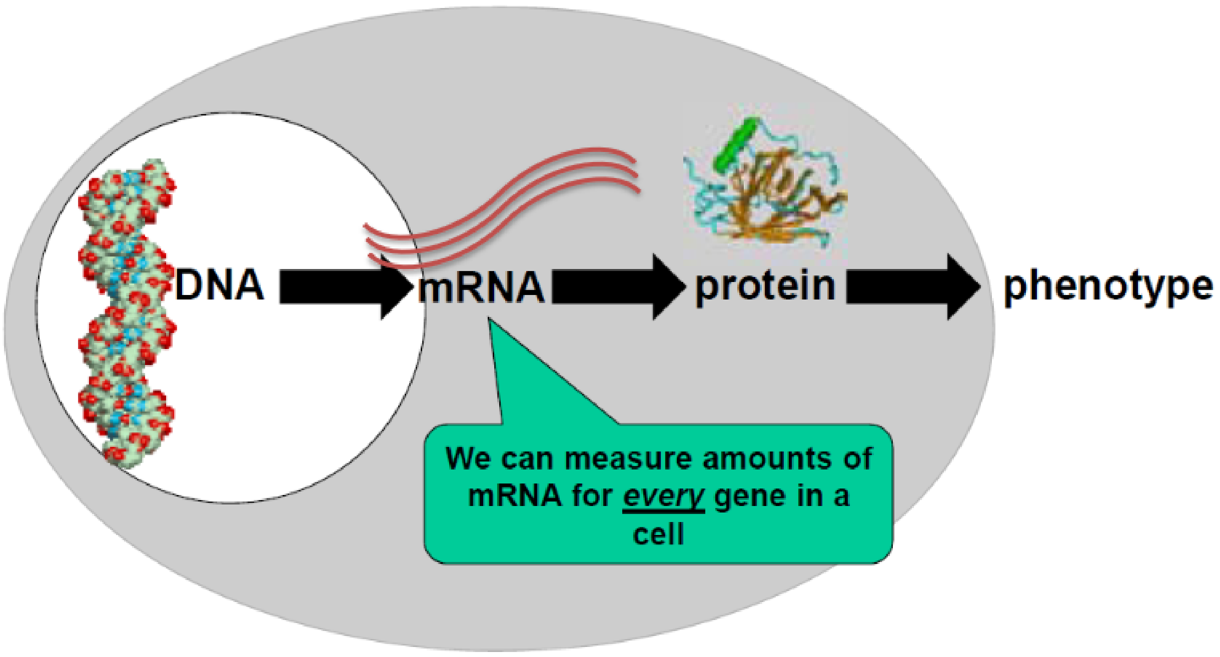
COM SCI 122 Week 6

Aidan Jan

February 14, 2025

Clustering

Central Dogma

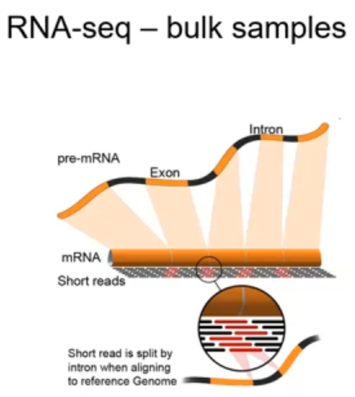


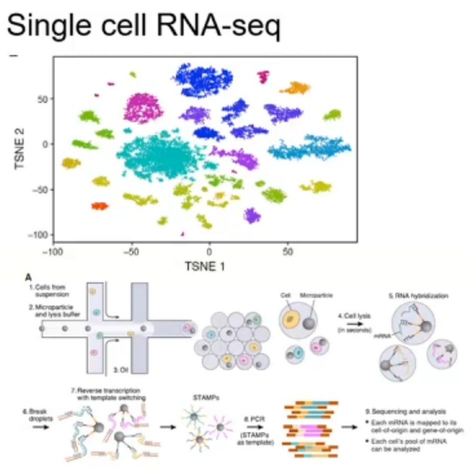
- DNA is transcribed to RNA, which is translated to protein. Proteins determine phenotype.
- Phenotype is a lot easier to measure (e.g., gene expression) rather than genes themselves

Measuring Gene Expression

- Organisms typically have on the order of thousands or tens of thousands of protein coding genes (e.g., 20000 genes in human)
- Gene expression profiling used to be limited to a single gene at a time. This changed with the publication of the first microarray in 1995.
- By 1997, scaled up to measure expression of all yeast genes.

There are two additional major waves of technologies for measuring gene expression:

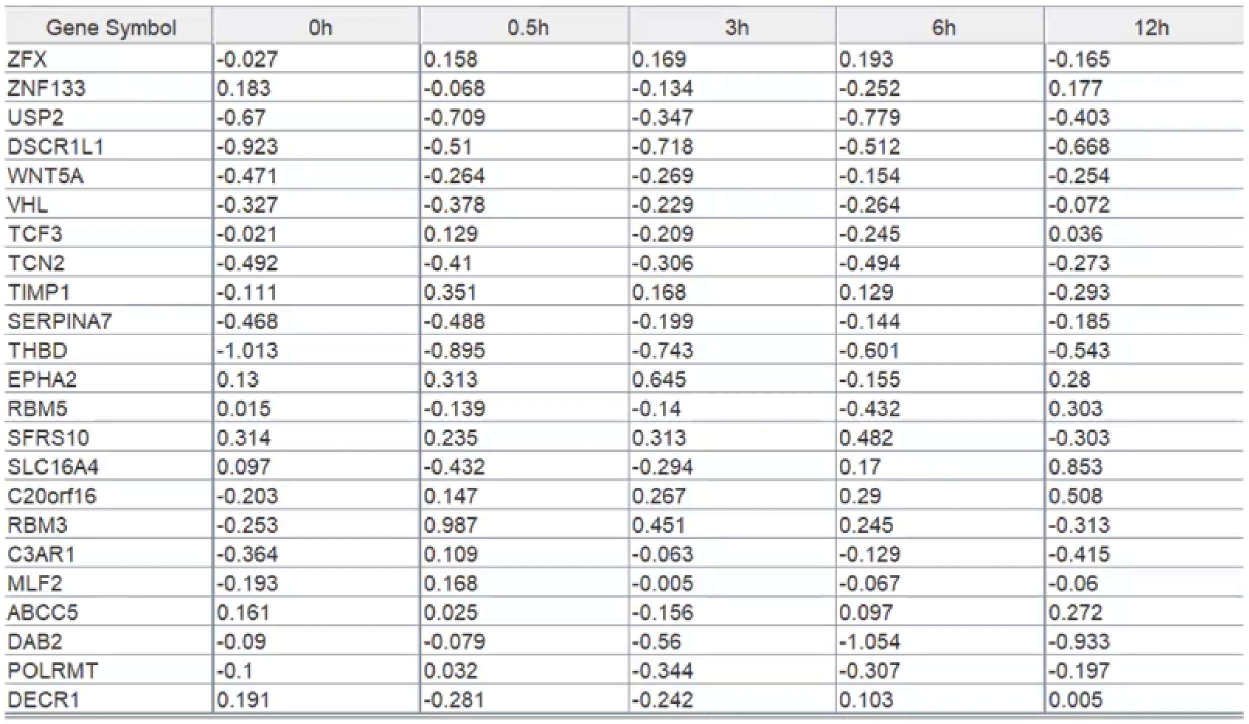

RNA-seq – bulk samples
pre-mRNA
Exon
Intron
mRNA
Short reads
Short read is split by intron when aligning to reference Genome



Macosko et al, *Cell* 2015

Computational problem of clustering genes remains common accross technologies.

Gene Expression Data



Gene Symbol	0h	0.5h	3h	6h	12h
ZFX	-0.027	0.158	0.169	0.193	-0.165
ZNF133	0.183	-0.068	-0.134	-0.252	0.177
USP2	-0.67	-0.709	-0.347	-0.779	-0.403
DSCR1L1	-0.923	-0.51	-0.718	-0.512	-0.668
WNT5A	-0.471	-0.264	-0.269	-0.154	-0.254
VHL	-0.327	-0.378	-0.229	-0.264	-0.072
TCF3	-0.021	0.129	-0.209	-0.245	0.036
TCN2	-0.492	-0.41	-0.306	-0.494	-0.273
TIMP1	-0.111	0.351	0.168	0.129	-0.293
SERPINA7	-0.468	-0.488	-0.199	-0.144	-0.185
THBD	-1.013	-0.895	-0.743	-0.601	-0.543
EPHA2	0.13	0.313	0.645	-0.155	0.28
RBM5	0.015	-0.139	-0.14	-0.432	0.303
SFRS10	0.314	0.235	0.313	0.482	-0.303
SLC16A4	0.097	-0.432	-0.294	0.17	0.853
C20orf16	-0.203	0.147	0.267	0.29	0.508
RBM3	-0.253	0.987	0.451	0.245	-0.313
C3AR1	-0.364	0.109	-0.063	-0.129	-0.415
MLF2	-0.193	0.168	-0.005	-0.067	-0.06
ABCC5	0.161	0.025	-0.156	0.097	0.272
DAB2	-0.09	-0.079	-0.56	-1.054	-0.933
POLRMT	-0.1	0.032	-0.344	-0.307	-0.197
DECR1	0.191	-0.281	-0.242	0.103	0.005

- Each row is a gene.
- Each column is a different experimental condition

Heatmap Representation of Gene Expression Data

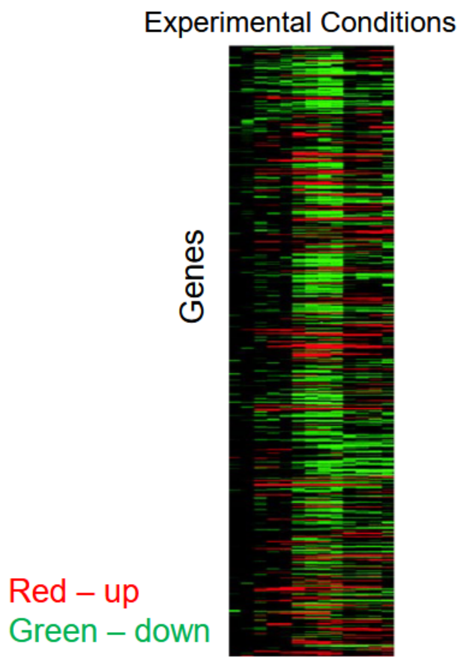
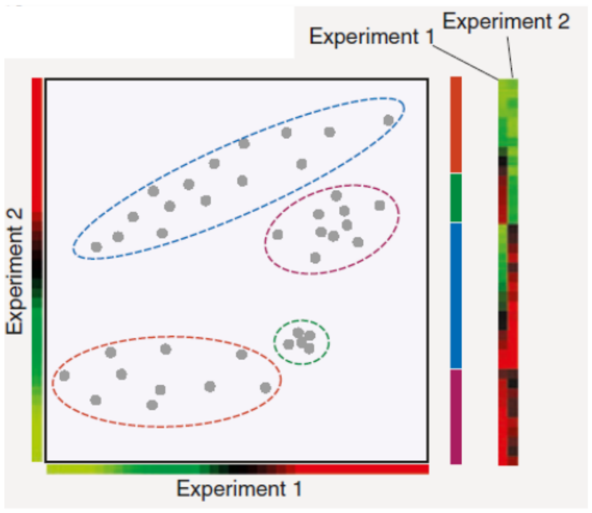


Image from *Eisen et al*, 1998

Gene Expression Clustering Problem

- Group unlabeled data points
- Informally want:
  - Data points "near" each other in the same clusters
  - Data points "far" from each other in different clusters



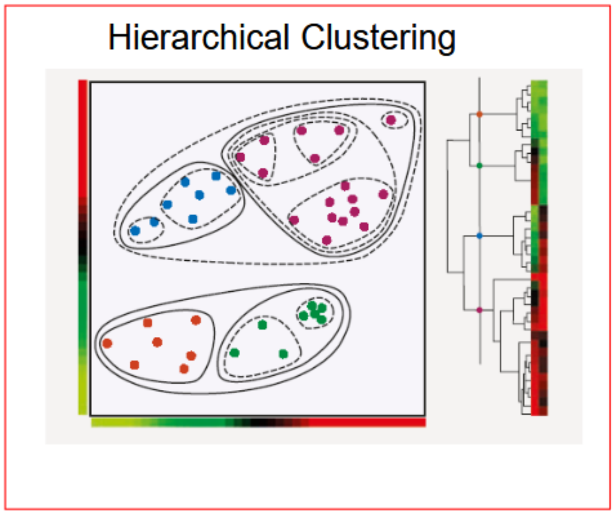
Why do we cluster genes by expression?

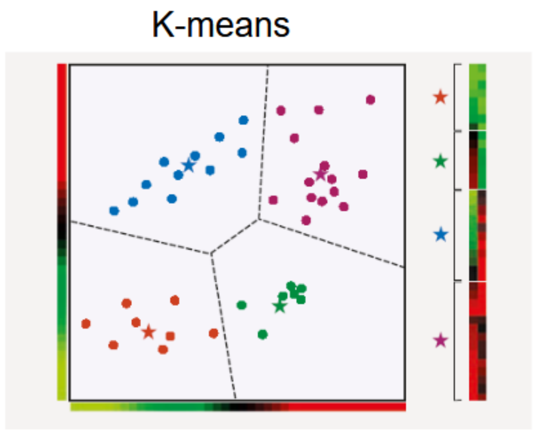
- Genes with similar gene expression patterns across experimental conditions are often involved in the same biological process or co-regulated (e.g., regulated by the same transcription factor)

- By identifying sets of genes with similar expression patterns can lead to insight into biological processes associated with the conditions, gene regulatory mechanism, and roles of genes with unknown function
  - Example: identify sequence patterns around transcription start sites of genes with similar expression patterns (ch. 2)

Clustering Algorithms

- Many clustering algorithms exist. E.g., Hierarchical clustering, K-means clustering





Hierarchical Clustering

1. Initially each point is its own cluster.
2. Find the pair of clusters with the smallest distance between them or equivalently are the most similar.
3. Merge into parent cluster.
4. Repeat steps 2 and 3 until the desired number of clusters remain.

How do we measure distance?

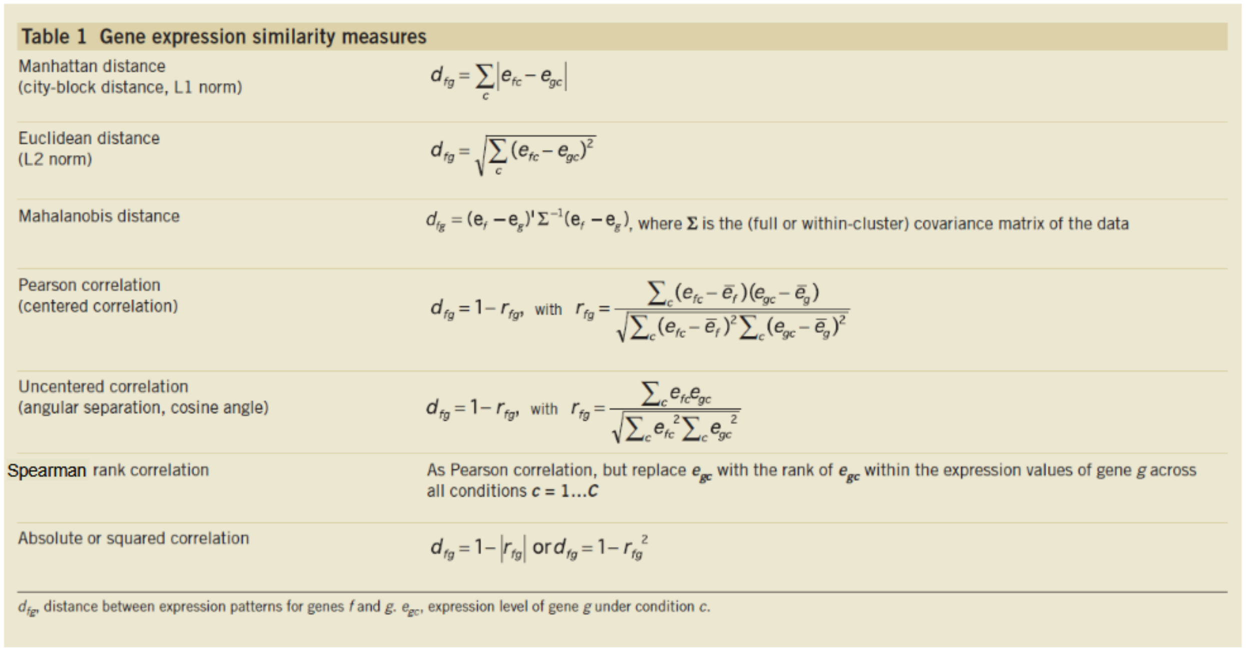


Table 1 Gene expression similarity measures

Manhattan distance (city-block distance, L1 norm)	$d_{fg} = \sum_c \lvert e_{fc} - e_{gc} \rvert$
Euclidean distance (L2 norm)	$d_{fg} = \sqrt{\sum_c (e_{fc} - e_{gc})^2}$
Mahalanobis distance	$d_{fg} = (\mathbf{e}_f - \mathbf{e}_g)' \Sigma^{-1} (\mathbf{e}_f - \mathbf{e}_g)$, where Σ is the (full or within-cluster) covariance matrix of the data
Pearson correlation (centered correlation)	$d_{fg} = 1 - r_{fg}$, with $r_{fg} = \frac{\sum_c (e_{fc} - \bar{e}_f)(e_{gc} - \bar{e}_g)}{\sqrt{\sum_c (e_{fc} - \bar{e}_f)^2 \sum_c (e_{gc} - \bar{e}_g)^2}}$
Uncentered correlation (angular separation, cosine angle)	$d_{fg} = 1 - r_{fg}$, with $r_{fg} = \frac{\sum_c e_{fc} e_{gc}}{\sqrt{\sum_c e_{fc}^2 \sum_c e_{gc}^2}}$
Spearman rank correlation	As Pearson correlation, but replace e_{gc} with the rank of e_{gc} within the expression values of gene g across all conditions $c = 1 \ldots C$
Absolute or squared correlation	$d_{fg} = 1 - \lvert r_{fg} \rvert$ or $d_{fg} = 1 - r_{fg}^2$

d_{fg}, distance between expression patterns for genes f and g. e_{gc}, expression level of gene g under condition c.

D'haeseleer, Nature Biotech 2005

- Pearson Correlation is popular since it clusters based on shape and relative changes which can often be more informative than absolute expression levels.
  - However, the Pearson correlation can fail to provide output if the variance is 0. In this case, it is undefined.
  - Typically, filter genes do not change expression before clustering.

Measuring Distance between Clusters

- Single Linkage Clustering:

$$D(X,Y) = \min_{x \in X, y \in Y} d(x,y)$$

- Complete Linkage Clustering:

$$D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$$

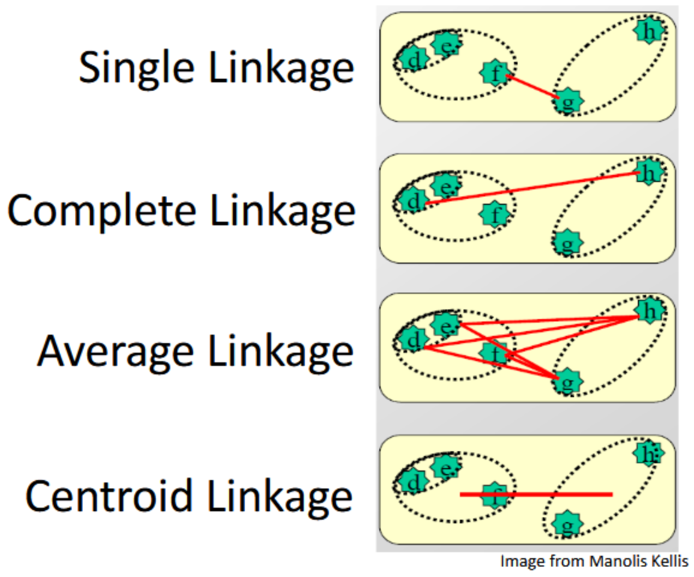
- Average Linkage Clustering:

$$D(X,Y) = \frac{1}{|X| \cdot |Y|} \sum_{x \in X} \sum_{y \in Y} d(x,y)$$

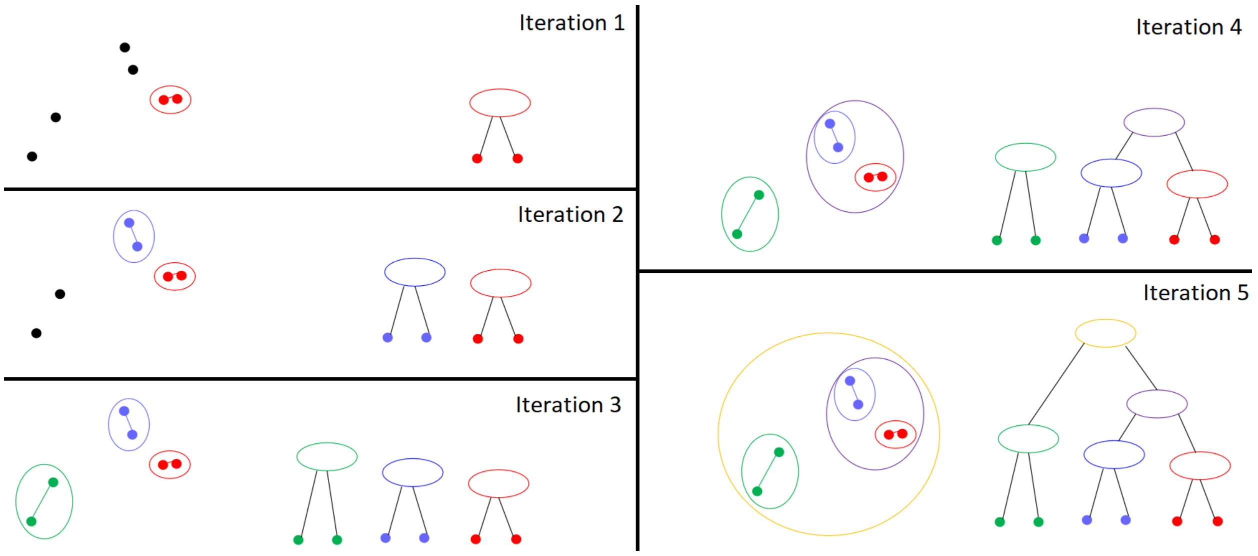
- Centroid Linkage Clustering:

$$D(X,Y) = \|c_X - c_Y\|$$

  - where c_X and c_Y are the mean of X and Y and data assumed to be in $\mathbb{R}^d$

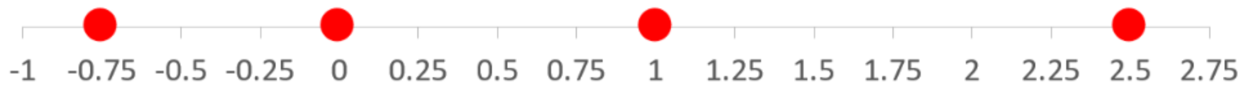


Hierarchical Clustering Example 1

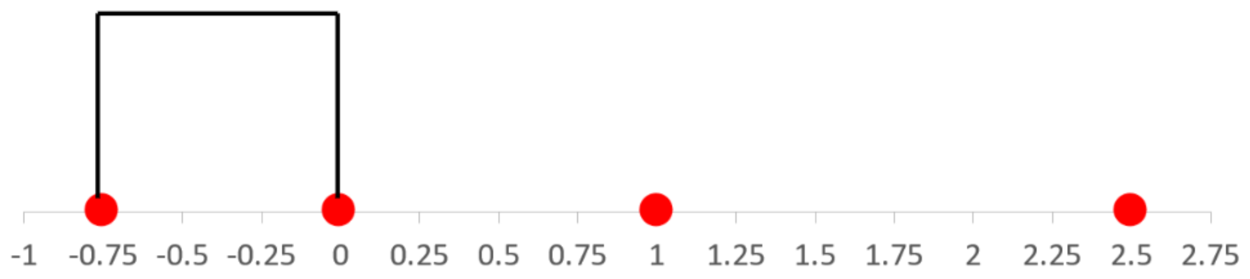


Hierarchical Clustering Example 2

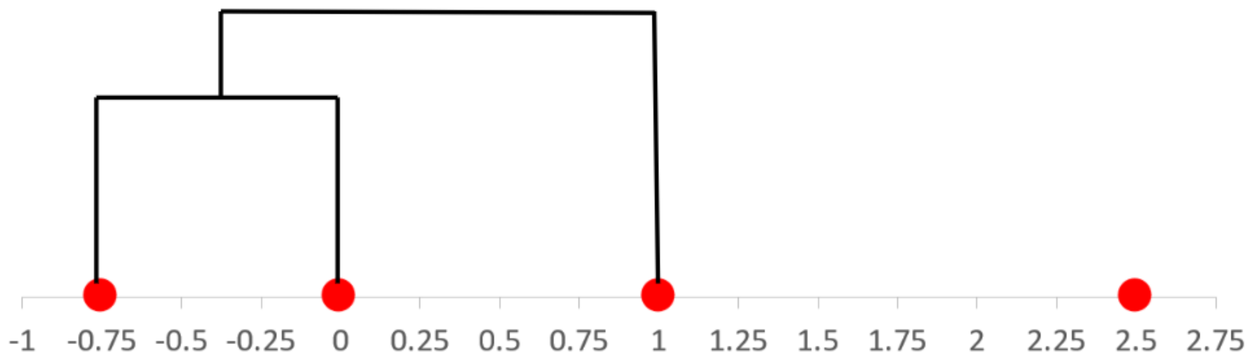
Suppose we want to perform hierarchical clustering with Euclidean distance using single linkage clustering to the four data points below.



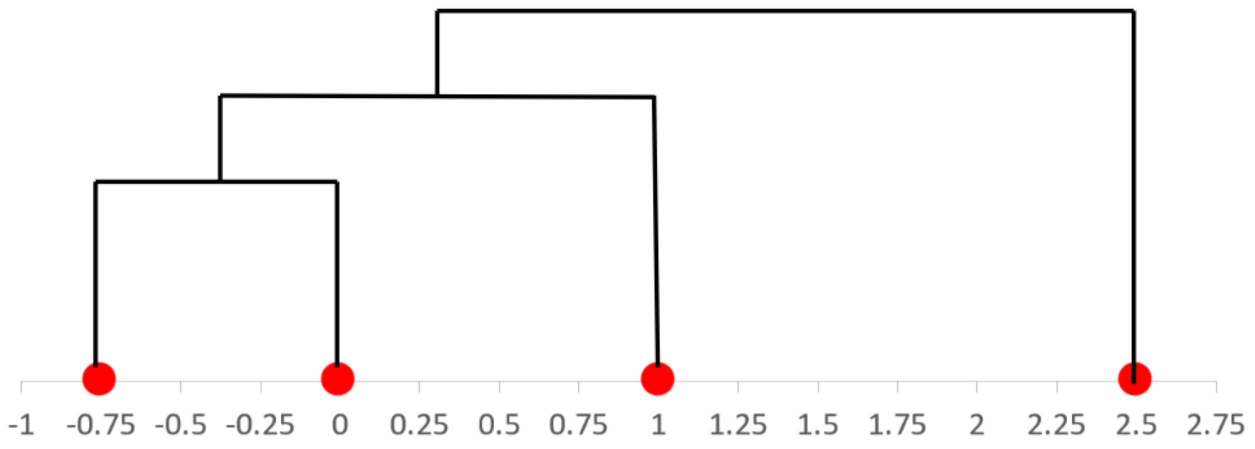
Finding the distance between each pair of points, the left two points are the closest together. Therefore, we merge them into one group.



The next closest group would be merging the (1) into the group we just created, since the distance between the (1) point is 1, and the distance between (1) and (2.5) is 1.5.

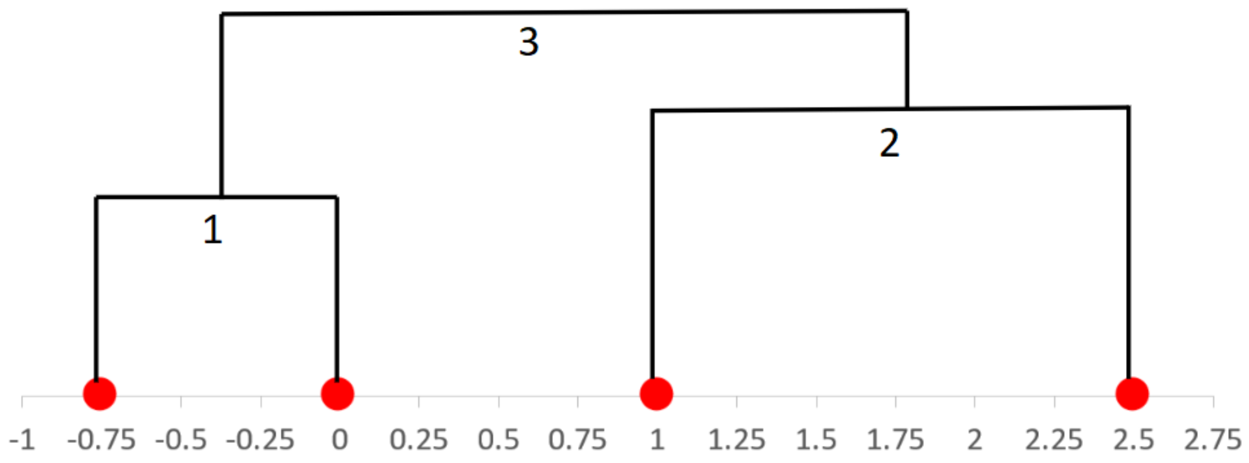


Finally, we merge the 2.5 point.



Hierarchical Clustering Example 3

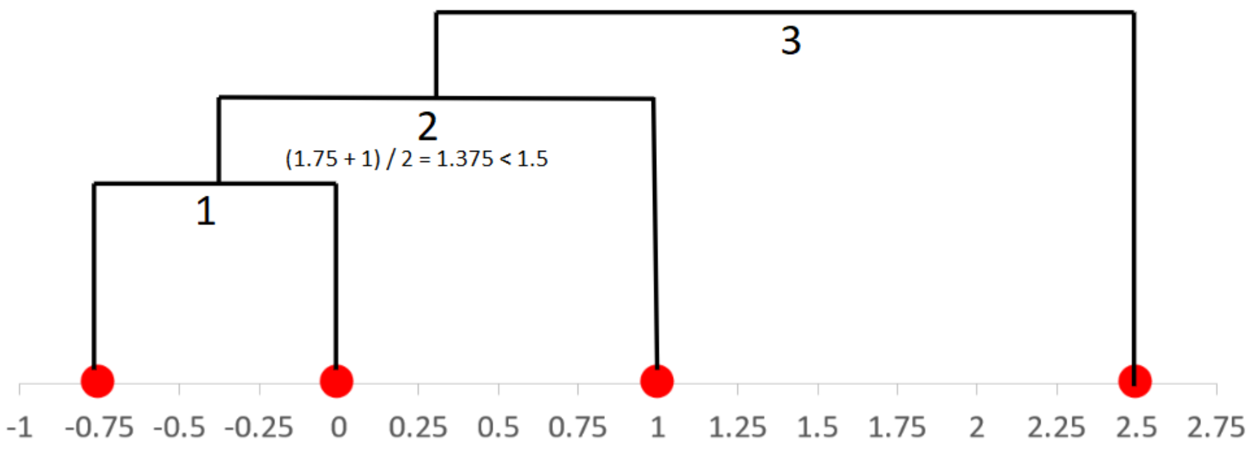
Suppose we want to perform hierarchical clustering with the same points as above, but this time, we use **complete linkage clustering**.



- Notice that complete linkage clustering led to a different dendogram than single linkage.

Hierarchical Clustering Example 4

Same points, but this time with average linkage clustering.

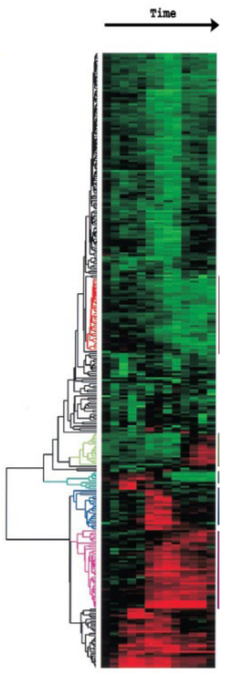


Runtime Complexity of Hierarchical Clustering

- Let n be the number of data points
- $O(n^2)$ time to compute all pairwise distances
- $O(n)$ iterations
- $O(n^3)$ if all pairwise distances recomputed and/or iterated over each iteration
- Depending on details of linkage method and implementation, this can run in $O(n^2)$ or $O(n^2 \log n)$ time

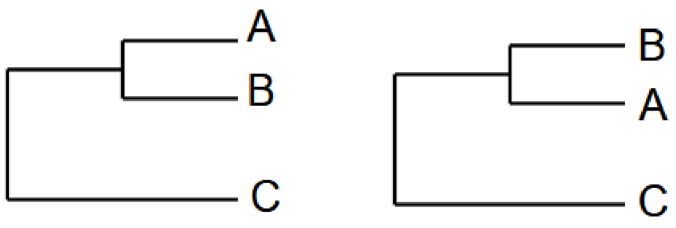
Ordering Leaves

Often more visual focus goes to the heatmap and the row ordering than the dendogram.



Question: Does hierarchical clustering uniquely determine an ordering of leaves (rows)?

- No! Two leaves can be swapped around without changing the structure of the hierarchical clustering.
- The two structures below have the same hierarchical clustering, but different orders.



Question: How many possible orderings are there for a hierarchical clustering of n data points?

- 2^{n-1}. This is because the hierarchical clustering always produces a binary tree. If there are n leaves, then there are 2^{n-1} internal nodes.
- Each internal node can be flipped.

Question: How should we select among possible orderings?

- Idea: pick an ordering that minimizes distance between adjacent leaves or equivalently maximizes similarity.

Optimal Ordering Leaves

Problem: Order leaves of hierarchical clustering dendrogram to minimize sum of distances between neighboring leaves or equivalently maximize similarity of neighboring leaves.

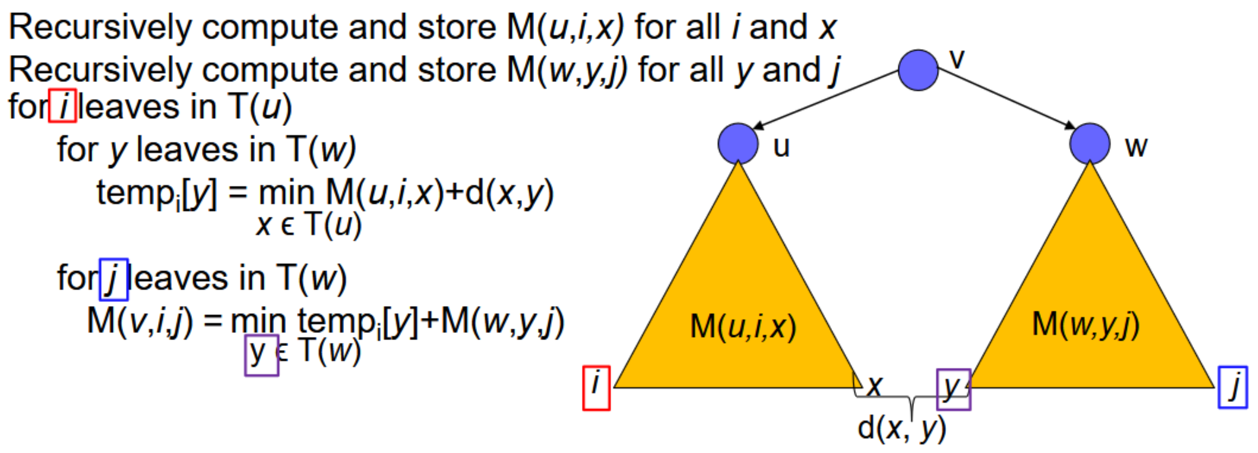
From Eisen et al, 1998 paper:

Ordering of Data Tables. For any dendrogram of n elements, there are 2^{n-1} linear orderings consistent with the structure of the tree (at each node, either of the two elements joined by the node can be ordered ahead of the other). An optimal linear ordering, one that maximizes the similarity of adjacent elements in the ordering, is impractical to compute.

- It turns out that solving this problem with brute force will take expoenntial time. However, it is possible to compute this in polynomial time - O(n^3).

Optimal Ordering Leaves in O(n^3) time

- T(v) - subtree rooted at v
- M(v,i,j) - cost of optimal tree rooted at v with start and end leaves i and j respectively where i,j are leaves in T(v)
- M(v,v,v) = 0
- $\mathrm{M}(v,i,j) = \min_{x\in \mathrm{T}(u), y\in \mathrm{T}(w)} \mathrm{M}(u,i,x) + \mathrm{d}(x,y) + \mathrm{M}(w,y,j)$
  - u is the left sub-child
  - w is the right sub-child
- Dynamic programming!



```
Recursively compute and store M(u,i,x) for all i and x
Recursively compute and store M(w,y,j) for all y and j
for i leaves in T(u)
    for y leaves in T(w)
        temp_i[y] = min_{x ∈ T(u)} M(u,i,x)+d(x,y)
    for j leaves in T(w)
        M(v,i,j) = min_{y ∈ T(w)} temp_i[y]+M(w,y,j)
```

Proof of O(n^3) time

- Let $F(n)$ be the total time to compute $M(v,i,j)$ for all i,j in a tree with n leaves.
- Let r be the number of leaves in subtree $T(u)$
- Let s be the number of leaves in subtree $T(w)$

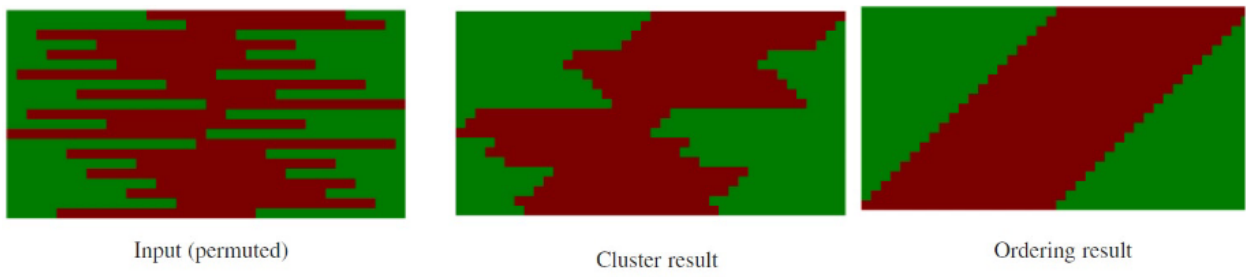
$$F(n) = F(r) + F(s) + O(r^2 s) + O(rs^2)$$

- We have $r + s = n$ and

$$(r^3 + s^3 + r^2 s + rs^2) \leq (r+3)^3 = n^3$$

- By induction it follows that $F(n)$ is O(n^3)

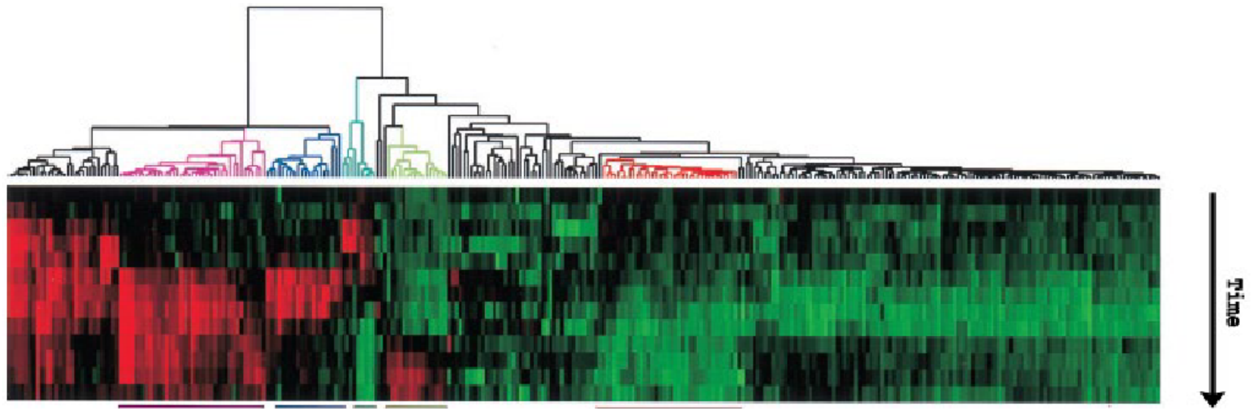
Ordering Leaves Optimally



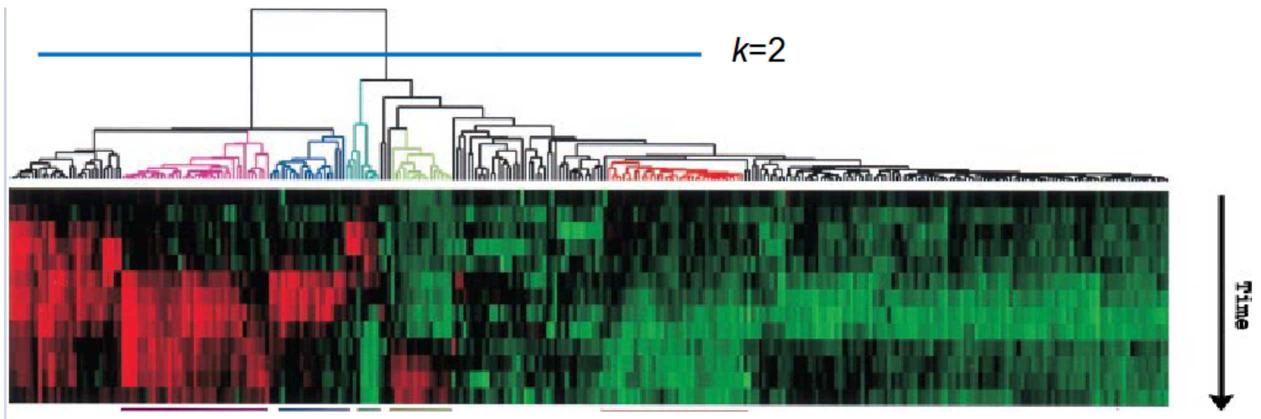
Rows are genes. Columns experiments
Green corresponds to -1 values
Red corresponds to 1 values

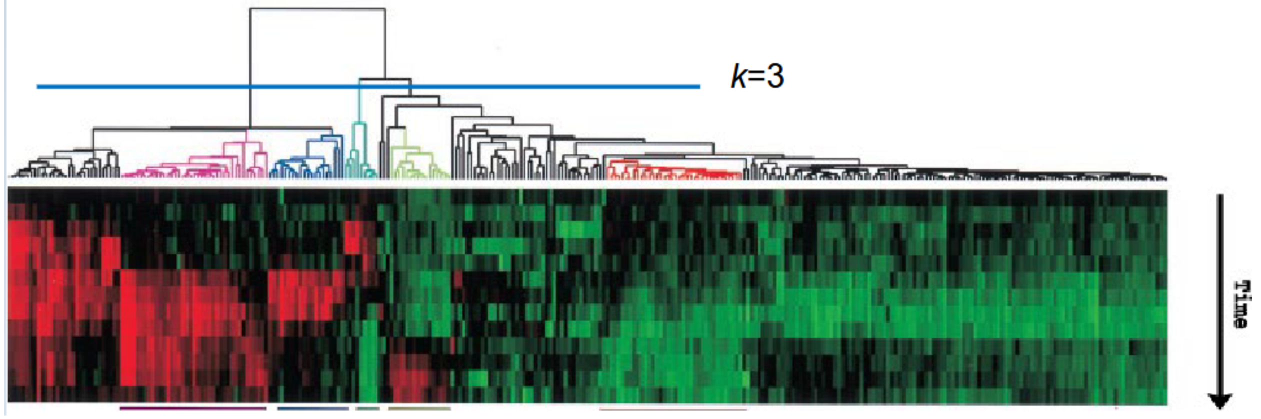
Getting Clusters from Hierarchical Clustering

How can we get k clusters from hierarchical clustering?



- Idea: cut tree to undo last $k - 1$ merges.





- This method is popular since it shows all the data in a hierarchy, but limited theoretical basis for resulting clusters
  - (i.e., no associated optimization criteria or statistical model)

K-means Clustering

The K-means Objective Function

$$\operatorname{argmin} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

- μ_i = Mean of cluster i
- S_i = Data points assigned to cluster i

Can be motivated by minimizing loss of information in compression

- an encoder function: ENCODE : $\Re^d \rightarrow [1 \ldots k]$
- a decoder function: DECODE : $[1 \ldots k] \rightarrow \Re^d$
- We define distortion to be

$$\text{Distortion} = \sum_{i=1}^{n} (x_i - \text{DECODE}[\text{ENCODE}(x_i)])^2$$

After initializing clustering centers, iterate between two steps:

- Re-assign data points to its closest cluster mean
- Recompute the cluster mean of the points assigned to each cluster

K-means Algorithm Example

1. Ask user how many clusters they'd like.
    - E.g., $k = 5$
2. Randomly guess k cluster center locations

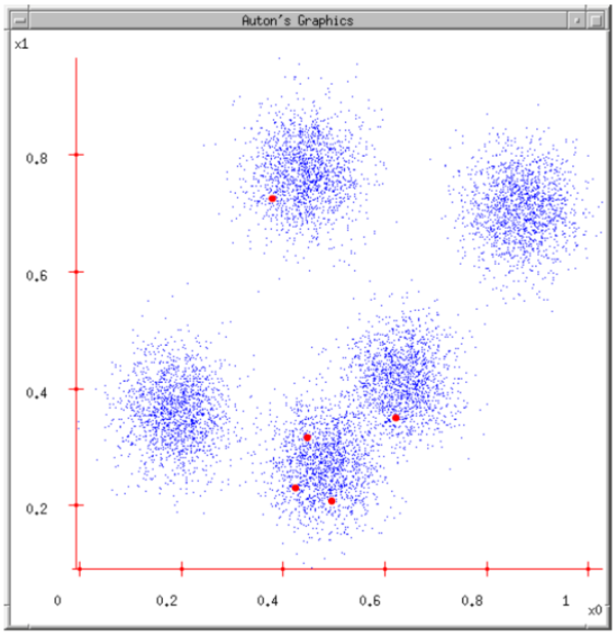


Image from Andrew Moore slides

- Above: blue dots are data points, red dots are randomly guessed centers.

3. Each datapoint finds out which center it's closest to. (Thus, each center "owns" a set of datapoints)

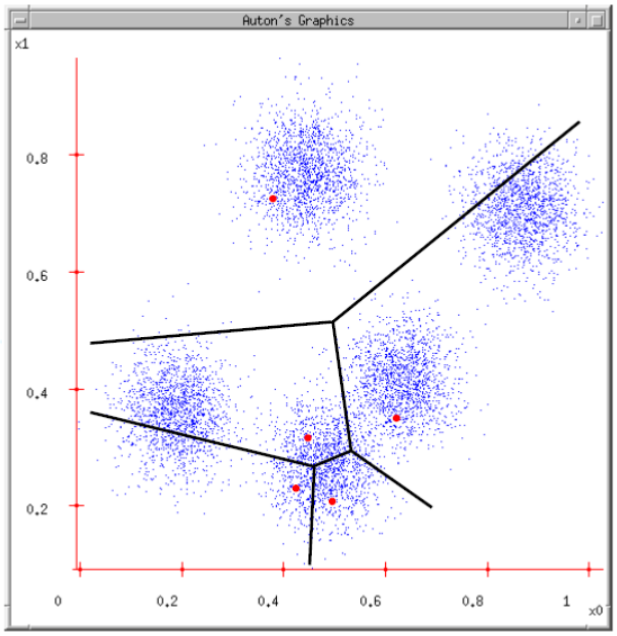


Image from Andrew Moore slides

4. Each center finds the centroid of the points it owns, and moves there.

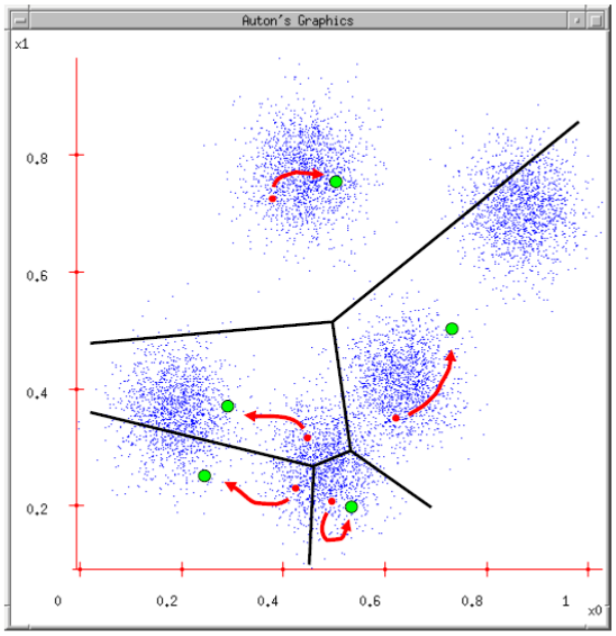


Image from Andrew Moore slides

5. Repeat 3-4 until the centers no longer move, or the program is terminated.

K-means Convergence

Is the k-means algorithm guaranteed to converge (i.e., no change in objective cost)?

- **Yes.** The algorithm will converge since there are only a finite number of ways of partitioning the set of data points into k-groups. Each iteration would need to visit a new configuration since it cannot increase objective value being minimized between iterations.

Going back to the objective function, to prove convergence, we need to claim that neither step (out of reassigning data points and moving center) would increase the objective function.

For the reassign step:

- For each point $x_j \in S_i$, either
  - it is closer to its current center $i \rightarrow$ no change
  - it is closer to another center $m \rightarrow$ objective function improves since $\|x_j - \mu_m\|^2 < \|x_j - \mu_i\|^2$

For the recompute cluster mean step:

- To find the minimum, take partial derivatives and set them equal to 0.

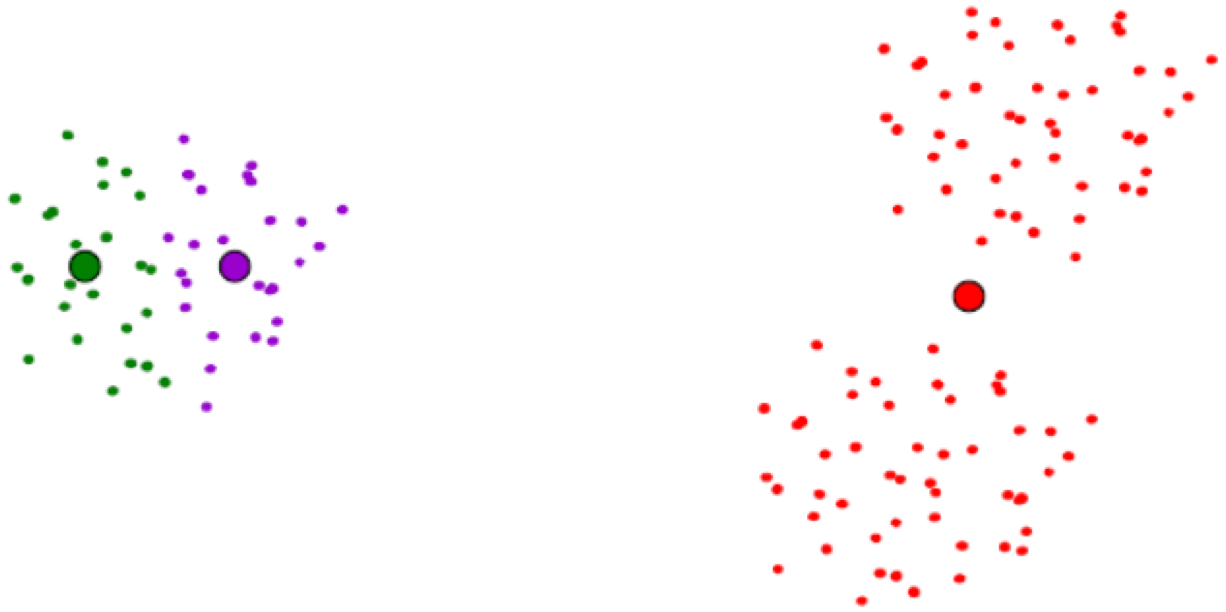
$$\frac{\partial}{\partial \mu_{i,d}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 = \frac{\partial}{\partial \mu_{i,d}} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 = -2 \sum_{x_j \in S_i} (x_{j,d} - \mu_{i,d})$$

- The above is 0 when $\mu_{i,d} = \frac{\sum_{x_j \in S_i}(x_{j,d})}{|S_i|}$ (i.e., cluster center)

K-means Optimal Solution

Is the k-means algorithm guaranteed to find an optimal solution?

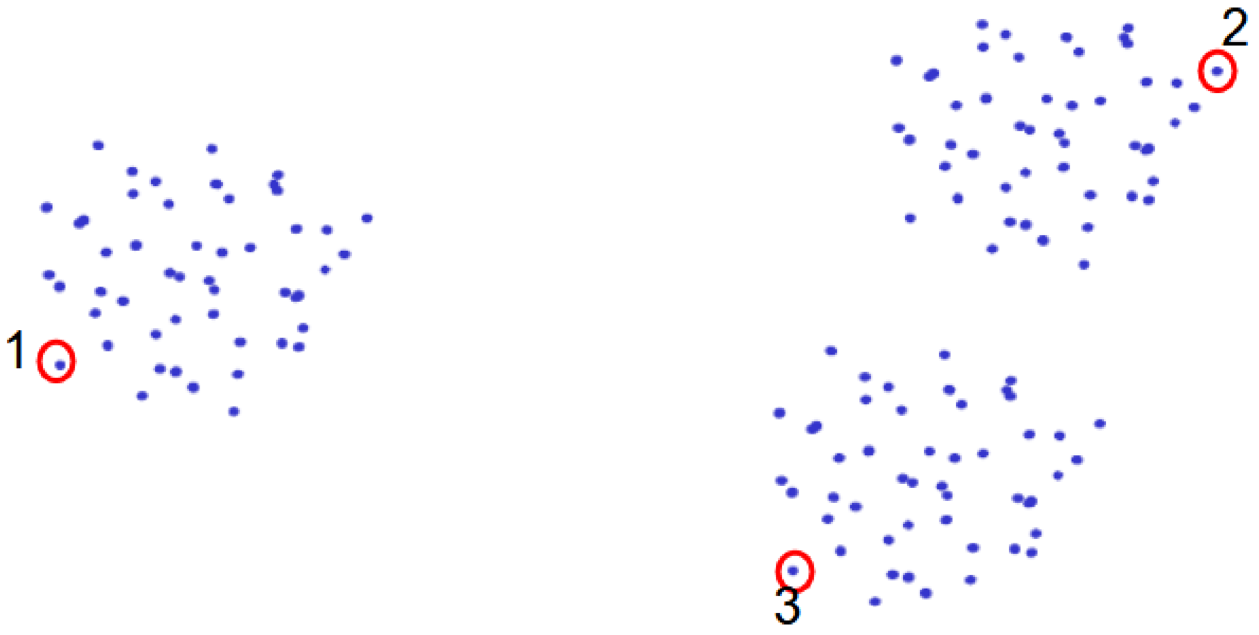
- **No.** Consider if the clusters were initiallized like below:



Is there a better way to initialize clusters, so this doesn't happen?

One Idea - Furthest Point Heuristic

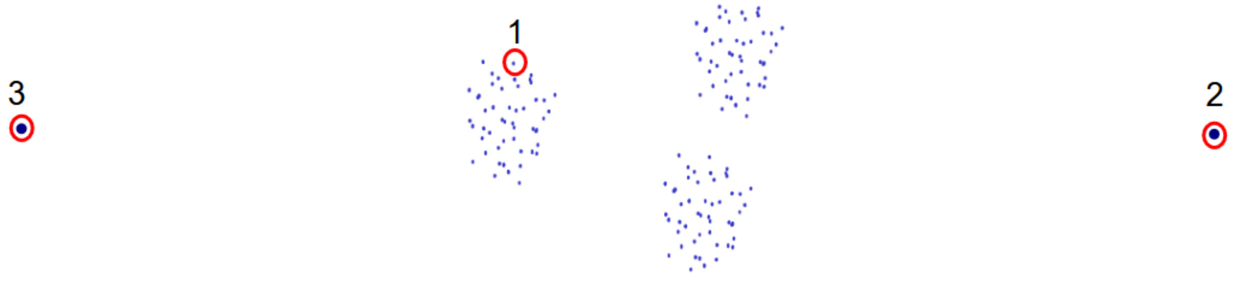
- Pick the first cluster center to be one arbitrarily selected point
- Iteratively pick cluster centers to be remaining points that are furthest from any selected point



Approximation algorithm to k-centers problem - minimize maximum distance of any point to its closest center

Question: Can this fail to lead to a good clustering?

- **Yes** - suppose there are two very far out points:



Question: Any ideas for initializing the centers that would spread the points while being less sensitive to outliers?

- Pick first cluster center to be one arbitrarily selected point
- Iteratively select a cluster center x' to be a point in the data probabilistically, where the probabilities are determined by

$$\frac{D(x')^2}{\sum_{x \in X} D(x)^2}$$

  - where $D(x)$ is the distance of x to its closest currently selected cluster center and X is the set of all points

Run-time of k-means algorithm

- Let n be the number of data points
- Let d be the number of dimensions in the data

- Let k be the number of clusters
- Let i be the number of iterations
- The run-time of the k-means algorithm is $\mathrm{O}(ndki)$
- In the worst case, the number of iterations is $\mathrm{O}(2^{\Omega(\sqrt{n})})$
- in practice, we will need fewer iterations and can terminate early based on a fixed number of iterations or small changes to objective.