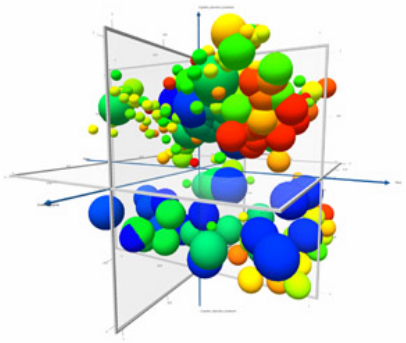


Clustering

Scott C-H Pegg, Ph.D.

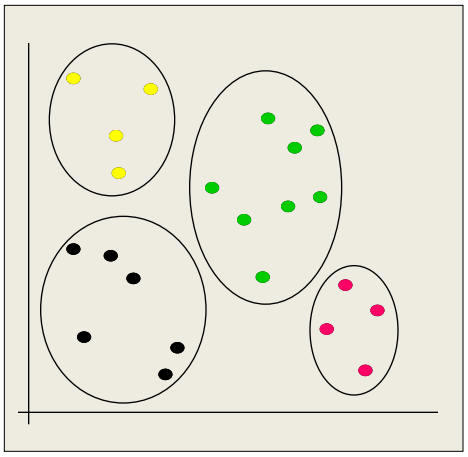
BMI203 January 24, 2018

Overview

- Relationships between objects
- Partitioning algorithms
- Hierarchical algorithms
- Choosing an algorithm
- Evaluating a clustering
- Homework instructions

What is clustering?

Clustering is the grouping of objects together



A meaningful clustering groups objects such that there's a relationship between objects in a group.

What's a Relationship?

Binary relations (relationships between two objects)

Binary relations can have all sorts of properties. Here are three simple ones:

Given objects $a_0, a_1, \ldots, a_n$ in set $\mathbf{A}$, and relation $\boldsymbol{R}$,

1. $a_i \boldsymbol{R} a_i$ for all a_i in $\mathbf{A}$ — Reflexitivity
2. $a_i \boldsymbol{R} a_j \implies a_j \boldsymbol{R} a_i$ — Symmetry
3. $a_i \boldsymbol{R} a_j$ and $a_j \boldsymbol{R} a_k \implies a_i \boldsymbol{R} a_k$ — Transitivity

Equivalence Relations

A binary relation that satisfies all three of these properties is an equivalence relation.

What does this have to do with clustering?

An equivalence relation on a set of objects defines a partitioning of that set (ie. a clustering).

Any partitioning (clustering) of a set of objects defines an equivalence relation on that set.

Equivalence Relations

You don't actually need an equivalence relation in order to perform a clustering.

So why are we talking about this?

- Clustering is really about relationships between objects.
- Considering relationships is a good place to both start and end when it comes to building and evaluating your clustering algorithms.

Equality

This one is simple. Just substitute the ***R*** for = .

Is this an equivalence relation?

Yes

1. $a_i = a_i$ for all a_i in **A** — Reflexive
2. $a_i = a_j \Longrightarrow a_j = a_i$ — Symmetric
3. $a_i = a_j$ and $a_j = a_k \Longrightarrow a_i = a_k$ — Transitive

Sequence Similarity

Let my objects in **A** be a set of protein sequences.

Let ***R*** be "is similar to" (using a threshold score from pairwise sequence alignment).

Is this an equivalence relation?

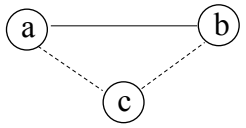
1. $a_i \boldsymbol{R} a_i$ for all a_i in **A** — Reflexive
2. $a_i \boldsymbol{R} a_j \Longrightarrow a_j \boldsymbol{R} a_i$ — Symmetric
3. $a_i \boldsymbol{R} a_j$ and $a_j \boldsymbol{R} a_k \not\Longrightarrow a_i \boldsymbol{R} a_k$ — NOT Transitive!

Distance Relations

These relations (functions, really) consider the distance between two objects in a geometric sense.

Distance relations must satisfy the following properties

1. $D(a,a) = 0$
2. $D(a,b) \geqslant 0$
3. $D(a,b) = D(b,a)$
4. $D(a,b) \leqslant D(a,c) + D(c,b)$



triangle inequality

Is sequence similarity a distance relation?

Distance Relations

Here are a couple of examples:

Euclidean distance

$$D(a,b) = \sqrt{(x_{a0} - x_{b0})^2 + (x_{a1} - x_{b1})^2 + \ldots + (x_{an} - x_{bn})^2}$$

(x_{a0}= the 0^{th} property of object x_a)

Manhattan (or city-block) distance

$$D(a,b) = |x_{a0} - x_{b0}| + |x_{a1} - x_{b1}| + \ldots + |x_{an} - x_{bn}|$$

Similarity Relations

There are lots of ways to define the similarity between objects.

A typical similarity function requires that

1. $0 \leqslant S(a,b) \leqslant 1$
2. $S(a,a) = 1$
3. $S(a,b) = S(b,a)$

Note that we can define dissimilarity as

$$D(a,b) = 1 - S(a,b)$$

Similarity Relations

Here are a couple of examples:

% identity of strings

$$S(a,b) = \frac{\text{\# of symbols in common}}{\text{\# of symbols in longer string}}$$

Tanimoto similarity of bitstrings

$$S(a,b) = \frac{\text{\# of 1's in common}}{\text{\# of 1's in both strings}}$$

Missing Values

Note that all of the functions we've looked at require a complete set of values for every object.

But we may not have values for every variable of every object.

So what do we do about missing values?

Missing Values

Here are a few options:

- Remove the objects with missing values from the overall clustering
- Replace the missing values of the variable with the average value of the variable over all the objects.
- Use prior knowledge of the variable's distribution and replace the missing value with the most likely value.

Normalizing Values

Let's go back to the Euclidean distance between two objects

$$D(a,b) = \sqrt{(x_{a0} - x_{b0})^2 + (x_{a1} - x_{b1})^2 + \ldots + (x_{an} - x_{bn})^2}$$

In some cases we're basing the distance on more than one variable. If these variables are not on the same scale (eg. Daltons vs. % identity), the variables will contribute unequally to the distance.

Normalizing Values

We can alleviate this by normalizing the values of the variables.

For example, we can divide by the mean absolute deviation

Change each x_i to z_i, where

$$z_i = \frac{x_i - m}{S}$$

$$m = 1/n \{x_0 + x_1 + \ldots + x_n\}$$

$$S = 1/n \{|x_0-m| + |x_1-m| + \ldots + |x_n-m|\}$$

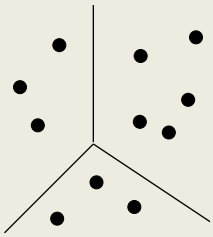
Lost?

- Binary relations
- Distance functions
- Similarity functions
- Scaling data

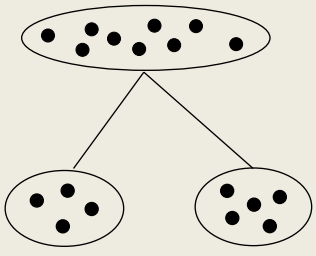
Clustering Algorithms

Clustering algorithms come in two basic flavors

Partitioning



Hierarchical



Partitioning Algorithms

Partitioning algorithms divide a set of objects into a given number of smaller sets. (You always get the number of clusters you asked for.)

But what if I don't know the number of clusters I want?

Run the program several times over a range of cluster numbers.

Partitioning Algorithms

The most common partitioning algorithms follow a 2-step process

1. Choose a set of "representative" objects.
2. Assign each remaining object to its nearest representative.

The critical part of these algorithms often lies in the first step.

The K-means Algorithm

One of the most widely used partitioning algorithms.

Input: A set of objects $\mathbf{X} = \{x_0, x_1, \ldots, x_n\}$, and an integer K.

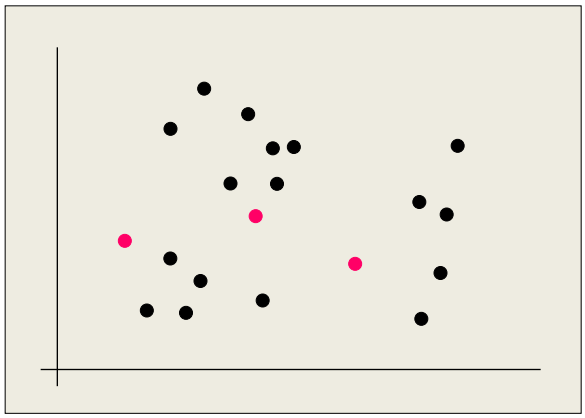
Output: A partition **S** of the objects that minimizes the sum of squared distance to the center of the cluster:

$$\sum_{j=1}^{K} \sum_{n \text{ in } S_j} D(X_n - \mu_n)^2$$

Where μ_j is the mean of cluster S_j.

Step 1: Arbitrarily choose from the set of objects K initial cluster centers,

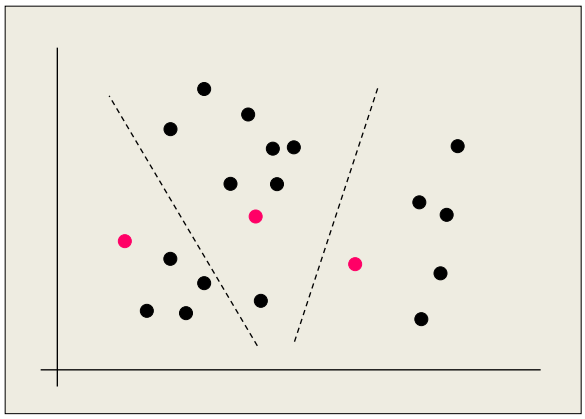
$$M_1^{(0)}, M_2^{(0)}, \ldots, M_K^{(0)}$$



Step 2: Assign each of the objects in $\mathbf{X}$ to the cluster with the nearest cluster center.

X_i in S_j if

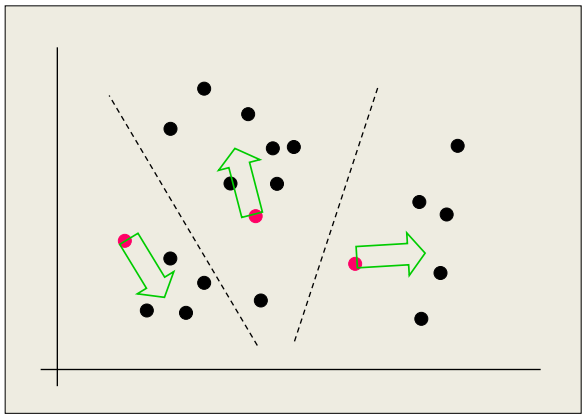
$$D(X_i, M_j^{(l)}) = \min\{ D(X_i, M_h^{(l)}, h = 1, \ldots, k) \}$$



Step 3: Recalculate the cluster centers to get $M_j^{(l+1)}$

$$M_j^{(l+1)} = 1/N_j^{(l)} \sum_{X \text{ in } S_j} X$$

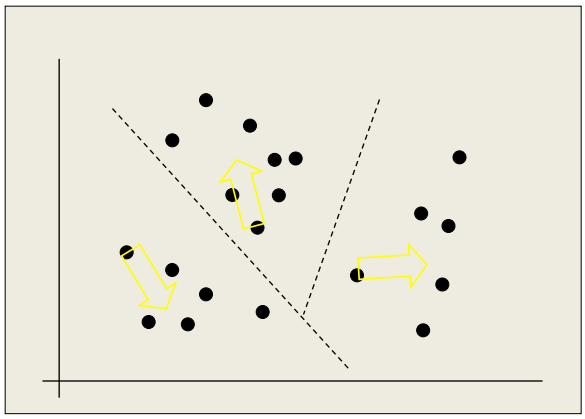
where $N_j^{(l)}$ is the number of samples in cluster $S_j^{(l)}$



Step 4: If the clusters have not changed (or a given number of iterations has been reached)

$$M_j^{(l+1)} = M_j^{(l)}$$

Terminate. Else, go to step 2.



Step 1: Arbitrarily choose from the set of objects K initial cluster centers.

Step 2: Assign each of the objects in $\mathbf{X}$ to the cluster with the nearest cluster center.

Step 3: Recalculate the cluster centers.

Step 4: If the clusters have not changed (or a given number of iterations has been reached), terminate, else go to step 2.

What's the big O? $O(n^2)$ In the worst case, you need to calcluate the distance between all n objects

K-means Variants

Recall that with K-means, we're minimizing the sum

$$\sum_{j=1}^{K} \sum_{n \text{ in } S_j} D(X_n - \mu_n)^2$$

where μ_n is the mean value of the X's in S_j

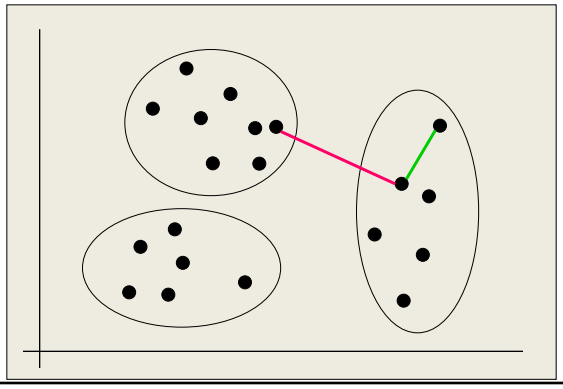
You can replace μ_n with other types of values to make variants.

Examples: K-medoids, K-centroids

Covering Algorithms

Covering algorithms are also based on choosing a representative set of objects, but don't require a pre-determined number of clusters.

Instead, they use a distance threshold for objects within a cluster.



Partitioning lots of objects

With a run time of $O(n^2)$, partitioning methods can take a long time for a large number of objects. How can we deal with this?

One common trick is to cluster only a small sample of the total set of objects, and then go back and assign the remaining objects to the representatives used in the sample clustering.

Since this depends on the original sample, you should do this several times with different samples.

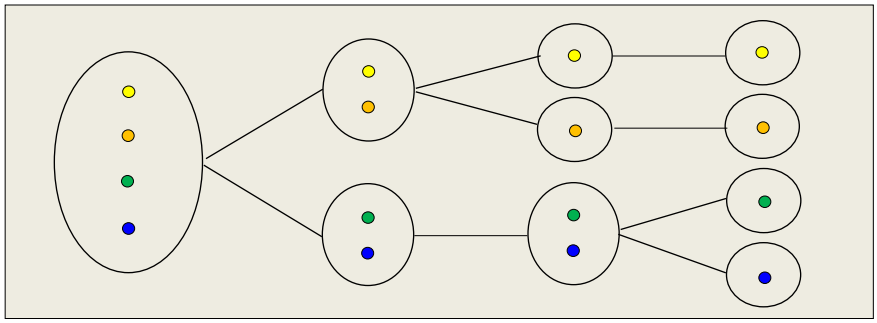
Lost?

- K-means algorithm
- Covering algorithms
- Partitioning large data sets

Hierarchical Algorithms

There are two types of hierarchical clustering algorithms

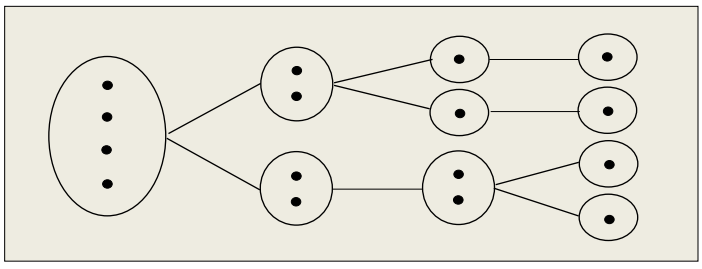
Divisive → ← Agglomerative



Note that the output here is not a single clustering, but a set of clusterings at different resolutions.

Hierarchical Algorithms

The key step in hierarchical algorithms is the decision of which sets to join (or split).



One of the drawbacks is that once you've made this decision, you can't go back and split (or re-join) clusters.

Agglomerative Algorithms

Agglomerative algorithms work as follows:

Step1: Put each object in its own cluster.

Step2: Choose (via a linkage criterion), two clusters and merge them.

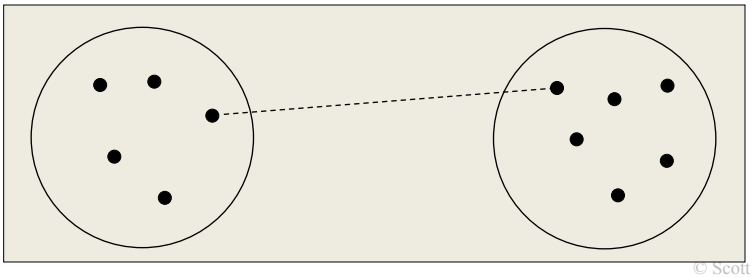
Step3: If there is only one cluster left, stop. Else, go to step 2.

Single Linkage

One of the oldest and simplest methods.

aka "Nearest-Neighbor"

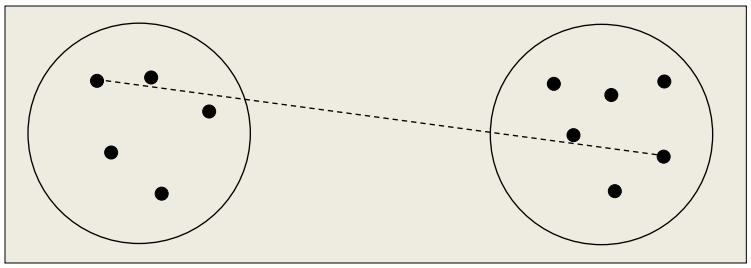
Choose the clusters with the shortest distance between the closest object in one cluster and the closest object in the other cluster.



Complete Linkage

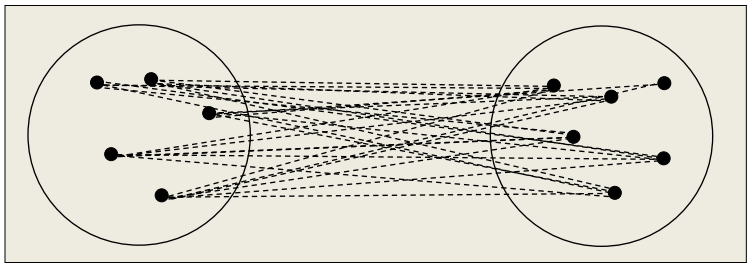
aka “Furthest-Neighbor”

Choose the clusters with the shortest distance between the furthest object in one cluster and the furthest object in the other cluster.



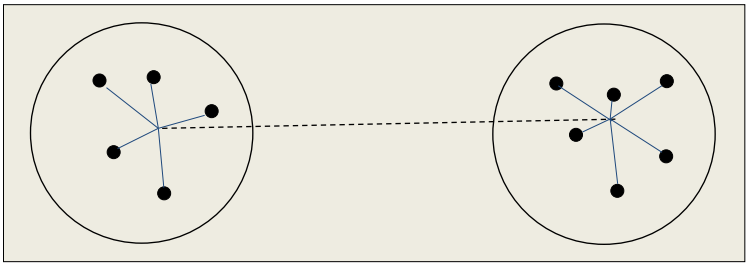
Average Linkage

Choose the clusters with the shortest average distance between all objects in one cluster and all objects in the other cluster.



Centroid Linkage

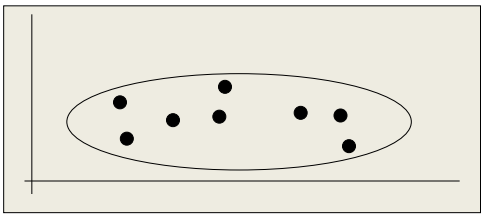
Choose the clusters with the shortest distance between the centroid of one cluster and the centroid of the other cluster.



Some Comparisons

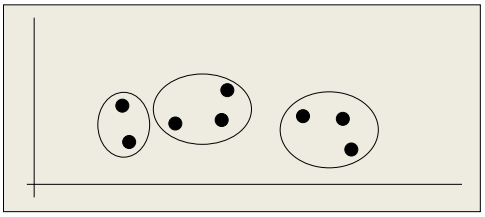
Single linkage

Long,drawn-out clusters



Complete linkage

Makes compact clusters



Centroid linkage

Somewhere in between

Complexity

Almost all implementations of hierarchical clustering are agglomerative because of the theoretical run times.

In an agglomerative algorithm, the first step considers all of the possible fusions of two (single element) clusters,

$$\frac{n(n-1)}{2} = O(n^2)$$

Complexity

A divisive algorithm must first consider all divisions of the entire data set into two non-empty sets,

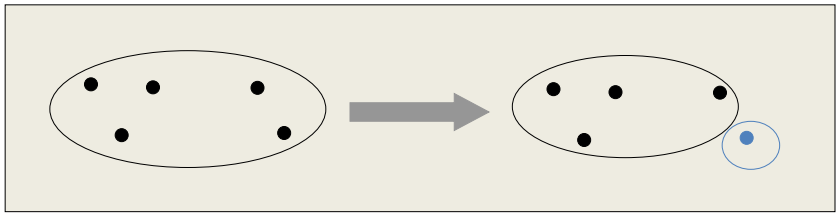
$$2^{n-1} - 1 = O(2^n)$$

There do exist some divisive algorithms that use tricks to get around the expensive first steps.

The Macnaughton-Smith Method

Step 1: Find the object with the highest average distance from all of the others.

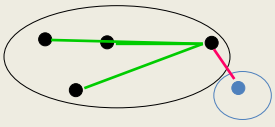
This object forms a splinter set.



The Macnaughton-Smith Method

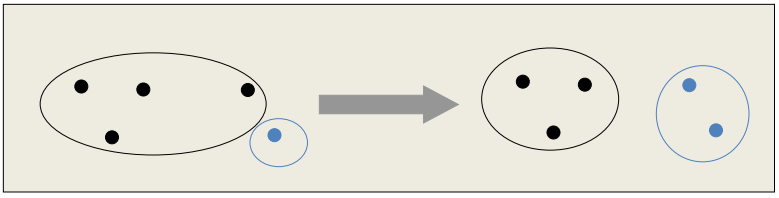
Step 2: For each object left in the original set, calculate the difference between the average distance from objects in the set and the average distance from objects in the splinter set.

$$\text{Diff}(X) = \frac{1}{|\text{orig set}|} \sum_{A \text{ in orig set}} D(X, A) - \frac{1}{|\text{splinter set}|} \sum_{B \text{ in orig set}} D(X, B)$$



The Macnaughton-Smith Method

Step 3: If all the differences are negative, stop. Else, move the object with the most positive difference to the splinter set and go to Step 2.

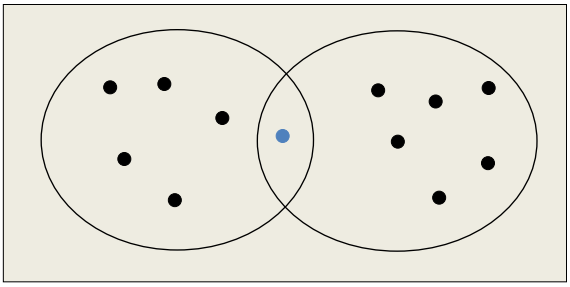


This algorithm takes $O(n^2)$ instead of $O(2^n)$ to split a group

"Fuzzy" Variants

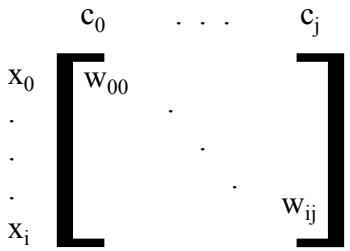
There are "fuzzy" variants to both partitioning and hierarchical methods.

These variants allow objects to be assigned to more than one cluster at a time.



“Fuzzy” Variants

Each object is assigned a vector of weights representing membership in a cluster.



When is this useful?

Lost?

- Agglomerative algorithms
- Linkage methods
- Divisive algorithms
- Runtimes

A Sample Algorithm

Input: A set of data points, $\{x_1, x_2, \ldots, x_n\}$, and a threshold t.

```
i = 1,  k = 1
assign x_1 to cluster C_1
if not all points assigned:
    i = i + 1
    dist =  min(distance between x_i and all x assigned)
    C_m = cluster corresponding to minimum distance
    if dist < t:  assign x_i to C_m
    else:
        k = k + 1
        assign x_i to C_k
```

Is this a partitioning algorithm, or a hierarchical algorithm?

What parts of the algorithm effect the final clustering?

What's the runtime of this algorithm?

Choosing an Algorithm

Often, there is no clear choice of clustering algorithm to use.

Considerations include:

- Total number of objects
- Number of likely clusters
- Shape of clusters

You may want to try more than one algorithm.

Evaluating Clusterings

How do you know when you've clustered correctly?

Unless you know the answer ahead of time, in general you don't.

But that doesn't mean that people haven't tried to evaluate clusterings mathematically.

INTERNAL CLUSTERING VALIDATION MEASURES

	Measure	Notation	Definition	Optimal value
1	Root-mean-square std dev	$RMSSTD$	$\{\sum_i \sum_{x \in C_i} \parallel x - c_i \parallel^2 / [P \sum_i (n_i - 1)]\}^{\frac{1}{2}}$	Elbow
2	R-squared	RS	$(\sum_{x \in D} \parallel x - c \parallel^2 - \sum_i \sum_{x \in C_i} \parallel x - c_i \parallel^2) / \sum_{x \in D} \parallel x - c \parallel^2$	Elbow
3	Modified Hubert Γ statistic	Γ	$\frac{2}{n(n-1)} \sum_{x \in D} \sum_{y \in D} d(x,y) d_{x \in C_i, y \in C_j}(c_i, c_j)$	Elbow
4	Calinski-Harabasz index	CH	$\frac{\sum_i n_i d^2(c_i,c)/(NC-1)}{\sum_i \sum_{x \in C_i} d^2(x,c_i)/(n-NC)}$	Max
5	I index	I	$(\frac{1}{NC} \cdot \frac{\sum_{x \in D} d(x,c)}{\sum_i \sum_{x \in C_i} d(x,c_i)} \cdot \max_{i,j} d(c_i, c_j))^p$	Max
6	Dunn's indices	D	$\min_i \{\min_j (\frac{\min_{x \in C_i, y \in C_j} d(x,y)}{\max_k \{\max_{x,y \in C_k} d(x,y)\}})\}$	Max
7	Silhouette index	S	$\frac{1}{NC} \sum_i \{\frac{1}{n_i} \sum_{x \in C_i} \frac{b(x)-a(x)}{\max[b(x),a(x)]}\}$ $a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x} d(x,y), b(x) = \min_{j, j \neq i} [\frac{1}{n_j} \sum_{y \in C_j} d(x,y)]$	Max
8	Davies-Bouldin index	DB	$\frac{1}{NC} \sum_i \max_{j, j \neq i} \{[\frac{1}{n_i} \sum_{x \in C_i} d(x,c_i) + \frac{1}{n_j} \sum_{x \in C_j} d(x,c_j)] / d(c_i, c_j)\}$	Min
9	Xie-Beni index	XB	$[\sum_i \sum_{x \in C_i} d^2(x,c_i)] / [n \cdot min_{i, j \neq i} d^2(c_i, c_j)]$	Min
10	SD validity index	SD	$Dis(NC_{max}) Scat(NC) + Dis(NC)$ $Scat(NC) = \frac{1}{NC} \sum_i \parallel \sigma(C_i) \parallel / \parallel \sigma(D) \parallel, Dis(NC) = \frac{max_{i,j} d(c_i,c_j)}{min_{i,j} d(c_i,c_j)} \sum_i (\sum_j d(c_i,c_j))^{-1}$	Min
11	S_Dbw validity index	S_Dbw	$Scat(NC) + Dens\_bw(NC)$ $Dens\_bw(NC) = \frac{1}{NC(NC-1)} \sum_i [\sum_{j, j \neq i} \frac{\sum_{x \in C_i \cup C_j} f(x,u_{ij})}{max\{\sum_{x \in C_i} f(x,c_i), \sum_{x \in C_j} f(x,c_j)\}}]$	Min

D: data set; n: number of objects in D; c: center of D; P: attributes number of D; NC: number of clusters; C_i: the i–th cluster; n_i: number of objects in C_i; c_i: center of C_i; $\sigma(C_i)$: variance vector of C_i; $d(x,y)$: distance between x and y; $\parallel X_i \parallel = (X_i^T \cdot X_i)^{\frac{1}{2}}$

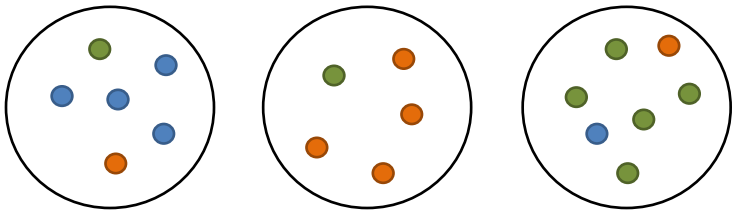
Sum of Distances Method

This often-used quantitative method looks at the sum of distances between objects within a cluster (or to a representative object).

$$Q = \frac{1}{K} \sum_{n=1}^{K} \sum_{i,j \text{ in } S_n} D(X_i - X_j) + C$$

The lower the sum, the tighter (and often "better") the clustering.

Purity



Each cluster is assigned a class based on the object label that appears most frequently in the cluster.

$$\text{Purity} = \frac{\text{\# of correctly assigned objects}}{\text{total \# of objects}}$$

Rand Index

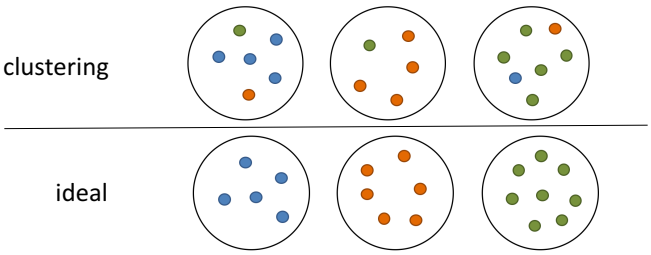
Similar to the Purity measure, except now we can penalize incorrect assignments.

Consider the process of clustering as a set of binary decisions: for each pair of objects, they either belong in the same cluster, or different clusters.

TP = True Positives = # of pairs correctly in same cluster
TN = True Negatives = # of pairs correctly in different clusters
FP = False Positives = # of pairs incorrectly in same cluster
FN = False Negatives = # of pairs incorrectly in different clusters

$$\text{Rand Index} = \frac{TP + TN}{TP + FP + TN + FN}$$

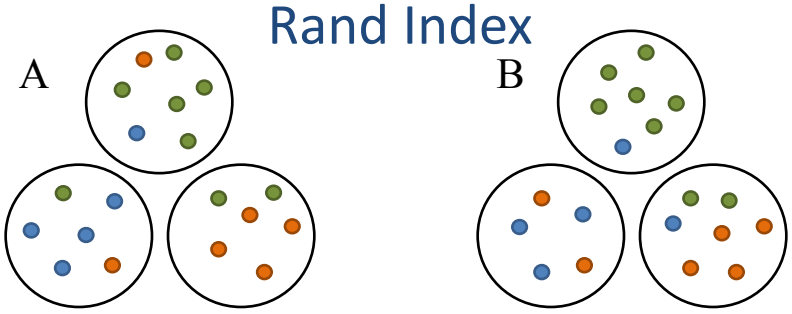
Rand Index



TP = (6) + (6) + (10) = 22
TN = (4+16+1) + (2+0+24) + (2+25+5) = 79
FP = (5+5) + (4) + (5+5) = 24
FN = (4+4+1) + (0+4+5) + (4+1+5) = 28

$$\text{Rand Index} = \frac{22 + 79}{22 + 24 + 79 + 28} = 0.660$$

Comparing Clusterings using the Rand Index



TP = # of pairs of objects in the data set that are in the same cluster in both clustering A and clustering B

TN = # of pairs of objects in the data set that are in different clusters in both clustering A and clustering B

FP = # of pairs of objects in the data set that are in the same cluster in clustering A, but different clusters in clustering B

FN = # of pairs of objects in the data set that are in different clusters in clustering A, but the same cluster in clustering B

F Measure

Sometimes we don't want to treat false positives and false negatives equally.

$$\text{P = Precision} = \frac{TP}{TP + FP} \qquad \text{R = Recall} = \frac{TP}{TP + FN}$$

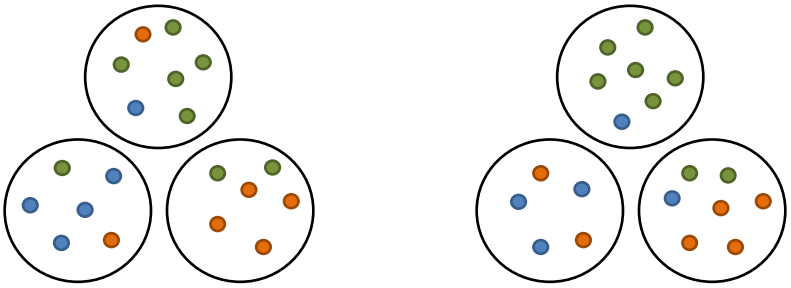
$$F_\beta = \frac{(\beta^2 + 1)\, P\, R}{\beta^2 P + R} \qquad 0 \le \beta \le +\infty$$

$F_{\beta<1}$ is precision oriented

$F_{\beta=1}$ is the harmonic mean of P and R

$F_{\beta>1}$ is recall oriented

Comparing Clusterings



Both the Rand Index and the *F* Measure can be used to compare two clusterings.

(Included this week is a PDF that describes several other interesting methods.)

Back to Binary Relations

One qualitative way to look at the results of a clustering is to look at the equivalence relation defined by the clustering itself.

Is it essentially the same as the relationship between objects that the clustering algorithm was using?

Does it make sense in the context of your particular problem?

Cautions

Clustering is often used as a "fishing" tool--cluster first and ask questions later.

- Think about the question you're trying to answer and choose a relationship and algorithm accordingly.

Boundaries between clusters are almost never "nice".

- Don't take them as gospel, especially if they change when different algorithms are used on the same data.

Clustering in Bioinformatics

Objects	Variables	Algorithms
genes	sequence, organism, promoters	agglomerative
proteins	sequence, structure, function	partitioning, hierarchical
microarray data	fluorescence, time, conditions	agglomerative
small compounds	structure, bioactivity	partitioning, hierarchical
patients	treatment, outcome, location	partitioning, hierarchical

Summary

What have we learned today?

- Some relational algebra
- Some useful relations between objects
- Data handling
- Partitioning methods (K-means, etc.)
- Hierarchical methods (single linkage, etc.)
- How to choose an algorithm
- How to evaluate a clustering

Homework Assignment

You will be given a set (~130) enzyme active sites to be clustered.

You will implement:

A similarity metric between active sites

There's no "right" answer, but you should think hard about choosing something biologically meaningful.

A partitioning algorithm to cluster the set of active sites

A hierarchical algorithm to cluster the set of active sites

A function to measure the quality of your clusterings

A function to compare your two clusterings

I will provide you with Python code which performs the I/O and contains some useful classes.

Homework Assignment

Answer the following questions

1. Explain your similarity metric, and why it makes sense biologically.
2. Explain your choice of partitioning algorithm.
3. Explain your choice of hierarchical algorithm.
4. Explain your choice of quality metric. How did your clusterings measure up?
5. Explain your function to compare clusterings? How similar were your two clusterings using this function?
6. Did your clusterings have any biological meaning?

Turn in your code and question answers to Tamas

Due Date: 10:29 AM 2/7/2018