Clustering

Slides from Eamonn Keogh at UC Riverside

What is Clustering?

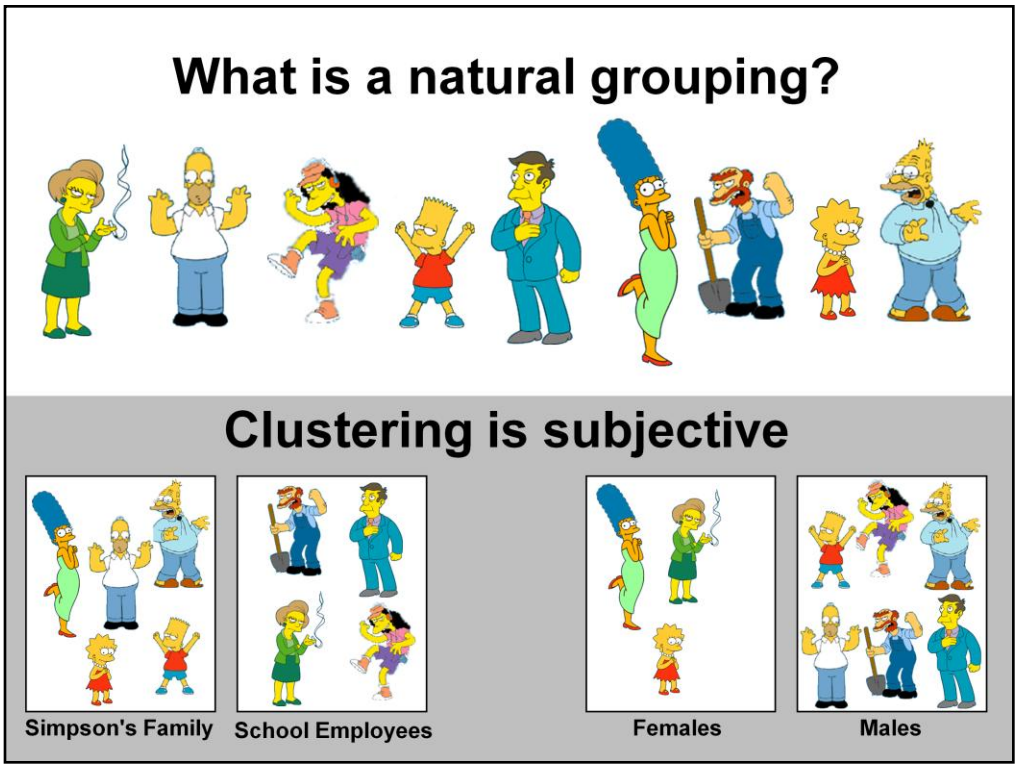
Organizing data into classes such that there is:

- **high intra-class similarity**
- **low inter-class similarity**

More informally, finding natural groupings among objects.

Also called unsupervised learning, sometimes called classification by statisticians, sorting by psychologists, and segmentation by people in marketing

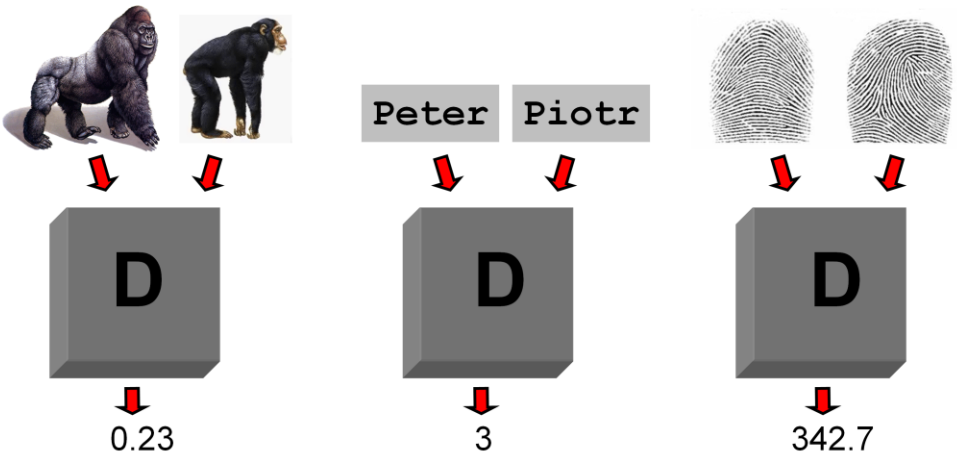
"Unsupervised" as compared to "supervised" learning, where you are given labeled examples (with which you can construct training/test sets)



How could you cluster these examples? By what traits?

Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$



This is a key representation problem in clustering.

What properties should a distance measure have?

$D(A,B) = D(B,A)$	***Symmetry***
$D(A,A) = 0$	***Self-Similarity***
$D(A,B) = 0$ iff $A = B$	***Positivity (Separation)***
$D(A,B) \leq D(A,C) + D(B,C)$	***Triangular Inequality***

Just imagine what this would be like if you *didn't* have these properties.

Desirable Properties of a Clustering Algorithm

- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records

Should be able to use different distance metrics, and different data.

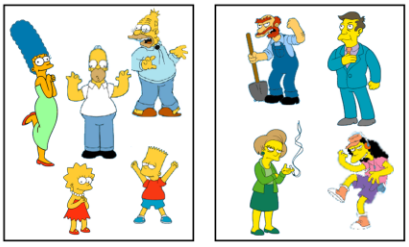
Noise is very common, and shouldn't completely break your algorithm.

Should not matter what order the data is given to the algorithm

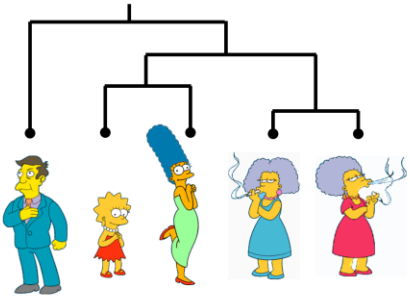
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion

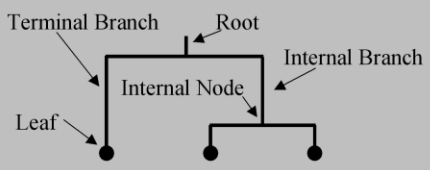
Partitional



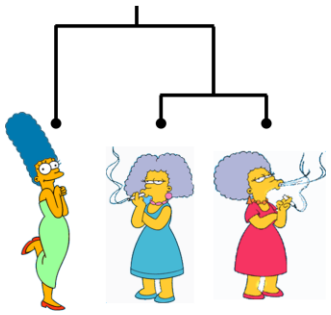
Hierarchical



Dendogram

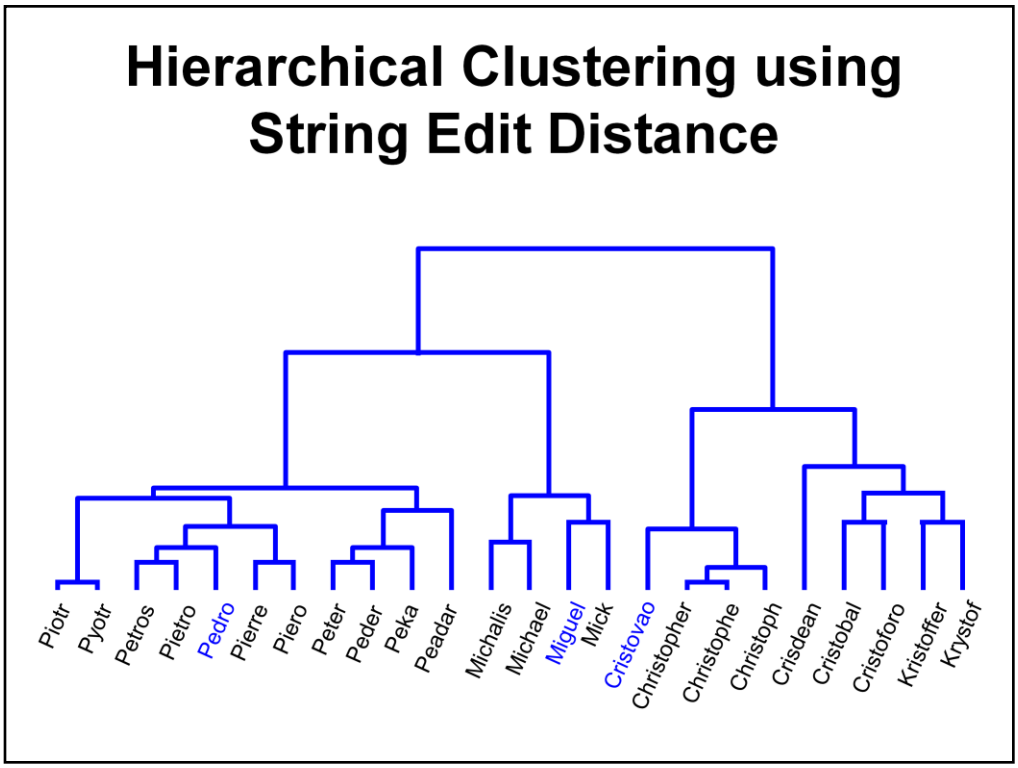


The similarity between two objects in a dendrogram is represented as the height of the lowest internal node they share.



Dendogram a lot like a binary tree

What should distance be? max(Distance to closest shared ancestor)

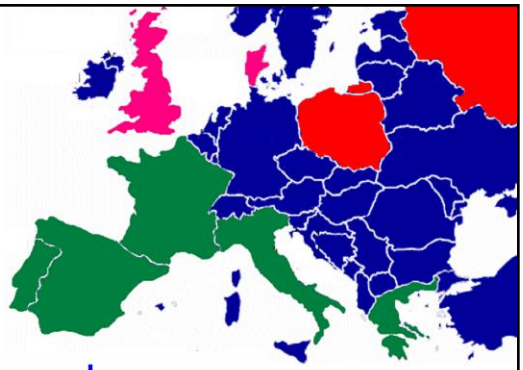


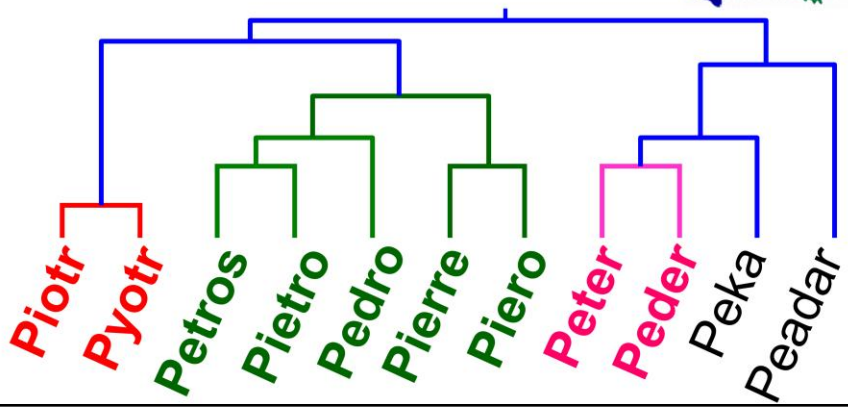
A clustering of the same name in different languages.

Pedro

(Portuguese/Spanish)

Petros (**Greek**), Peter (**English**), Piotr (**Polish**), Peadar (Irish), Pierre (**French**), Peder (**Danish**), Peka (Hawaiian), Pietro (**Italian**), Piero (**Italian Alternative**), Petr (Czech), Pyotr (**Russian**)

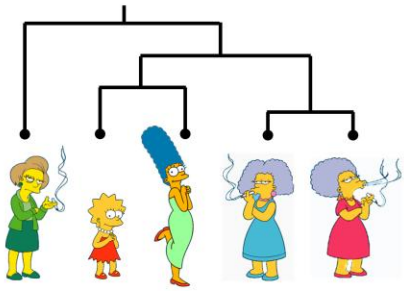




Hierarchical Clustering

The number of dendrograms with *n* leafs = $(2n-3)!/[(2^{(n-2)})(n-2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	...
10	34,459,425



Since we cannot test all possible trees we will have to heuristic search of all possible trees.

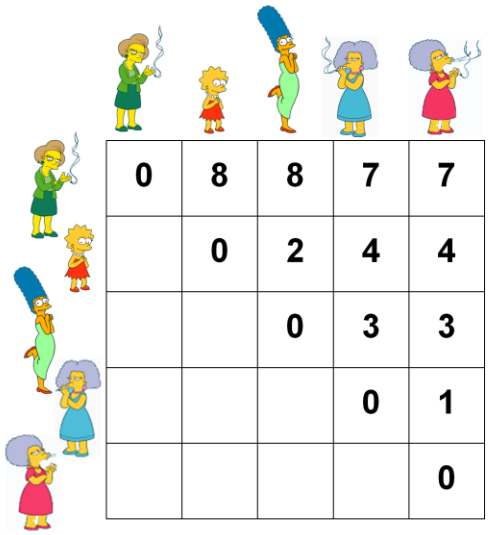
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

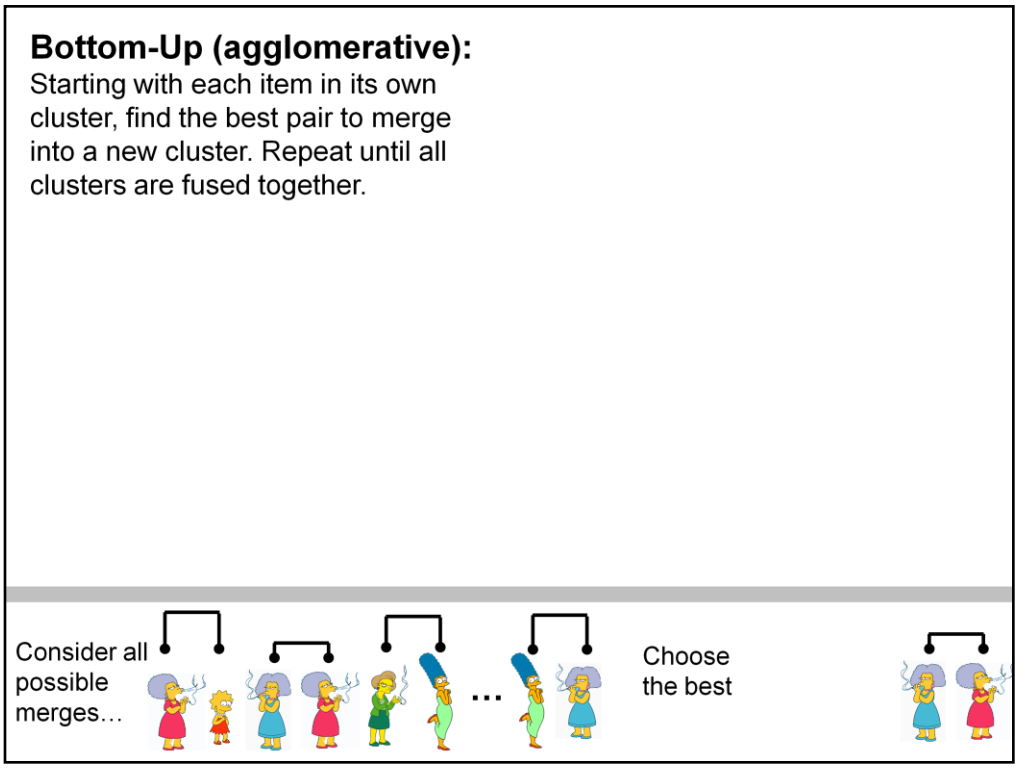
We begin with a distance matrix which contains the distances between every pair of objects in our database.

D() = 8

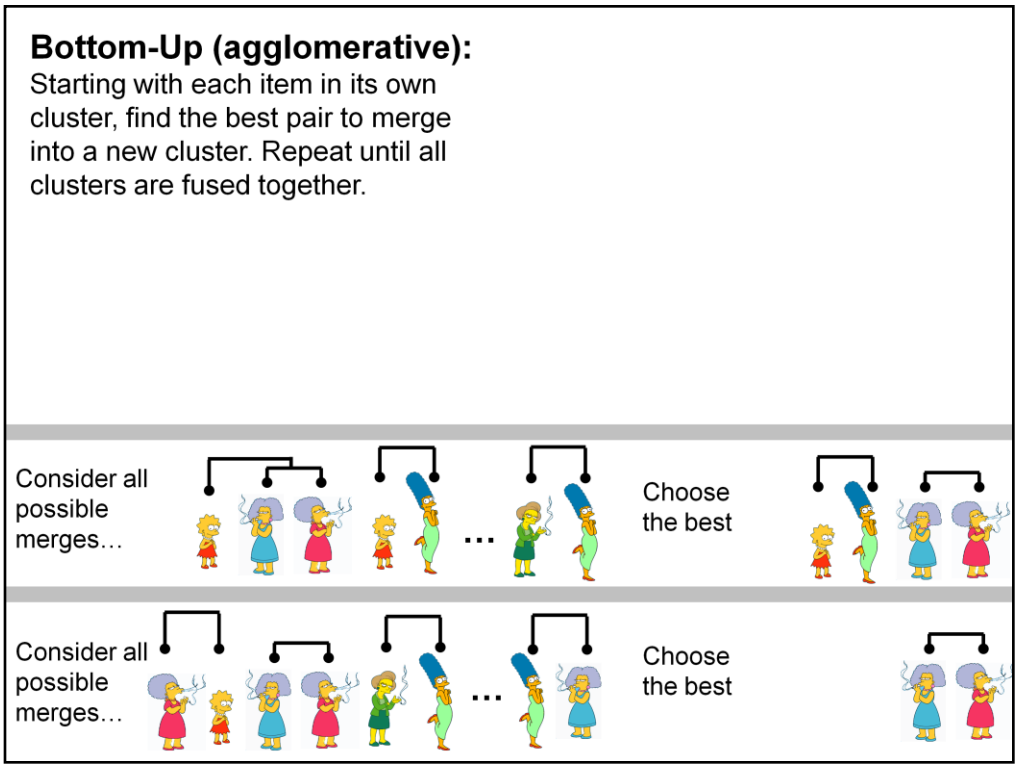
D() = 1



0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

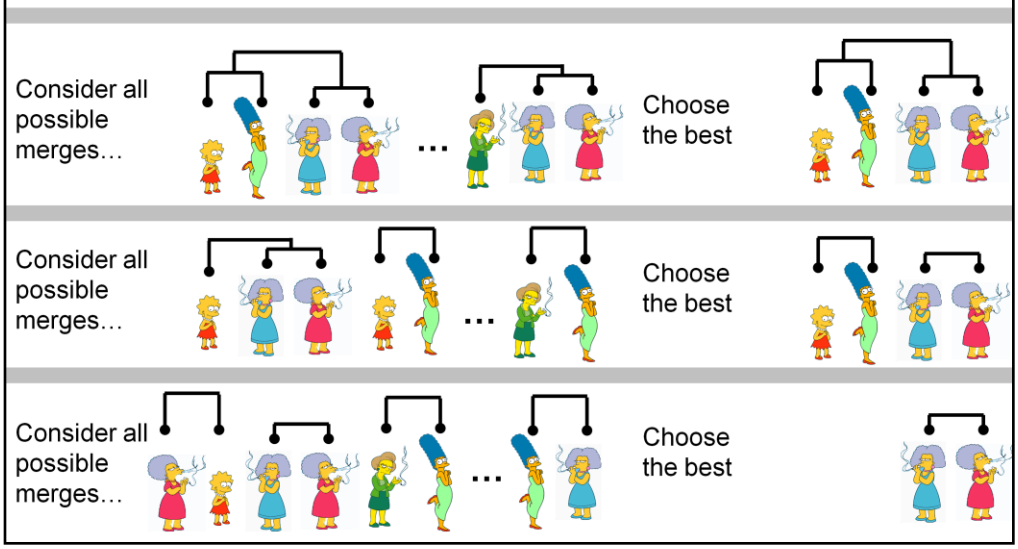


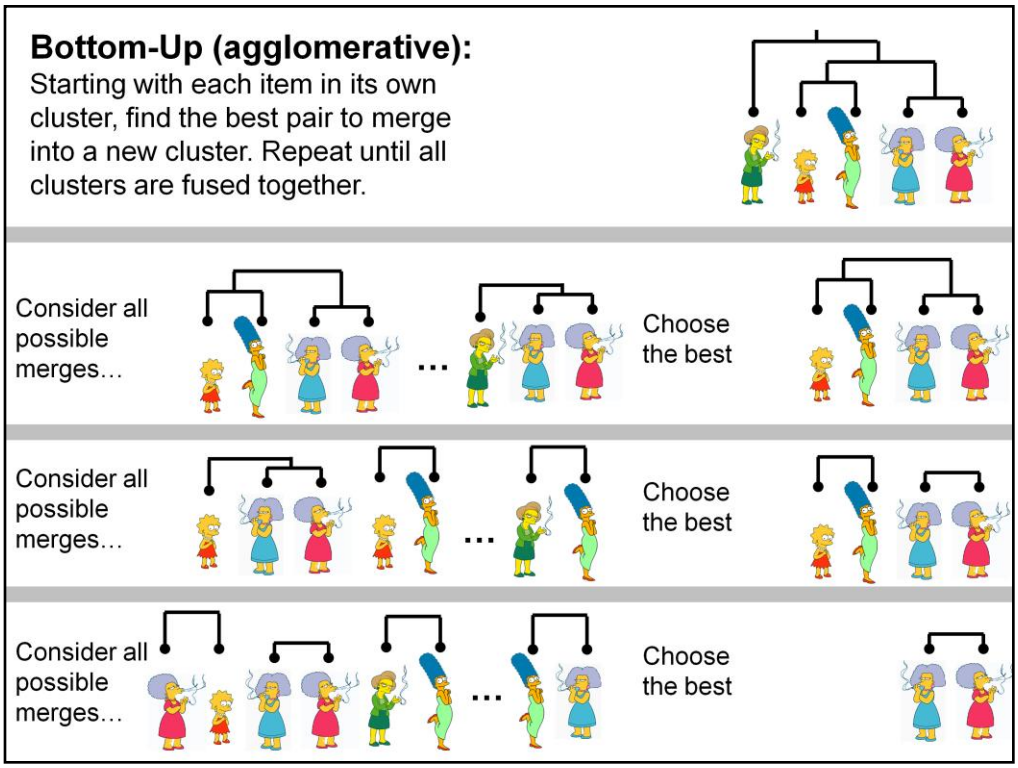
Start off with each point as a cluster. Then consider all pairs of clusters, and choose the best pair. Combine the best pair into a cluster.



Again, choose the best two clusters and combine them.

Bottom-Up (agglomerative):
Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

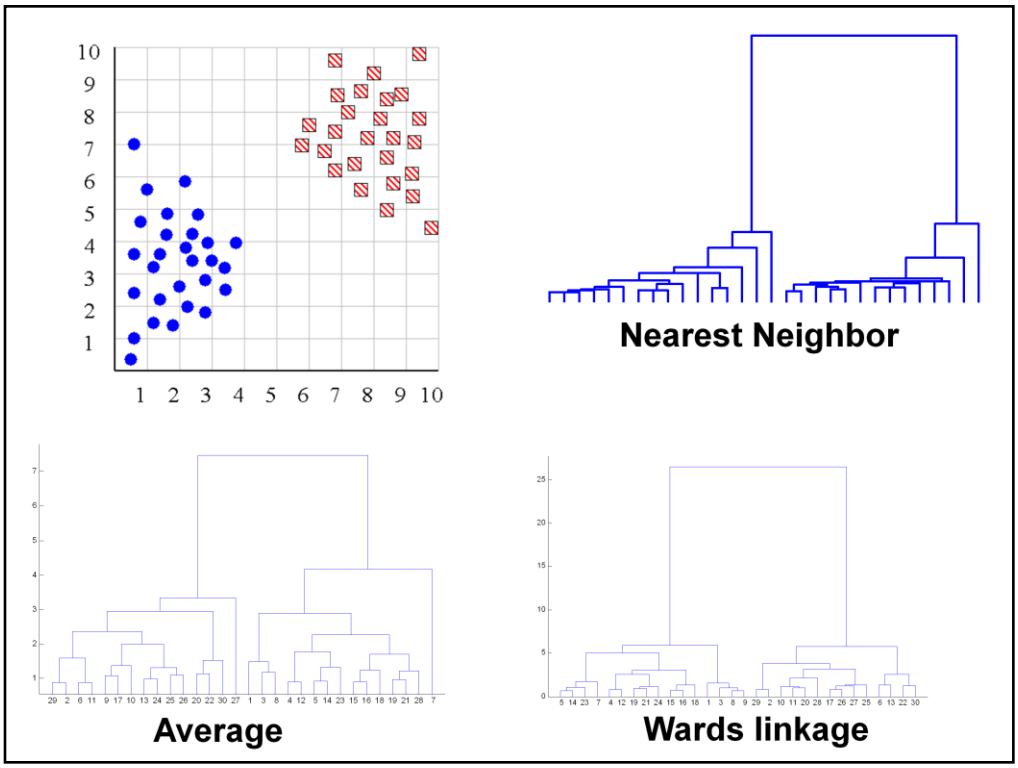



Bottom-Up (agglomerative):
Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.
Consider all possible merges…
Choose the best
Consider all possible merges…
Choose the best
Consider all possible merges…
Choose the best

We know how to measure the distance between two objects, but defining the distance between an object and a cluster, or defining the distance between two clusters is non obvious.

- **Single linkage (nearest neighbor):** Distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.
- **Complete linkage (furthest neighbor):** 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:** Distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.
- **Wards Linkage**: Try to minimize the variance of the merged clusters

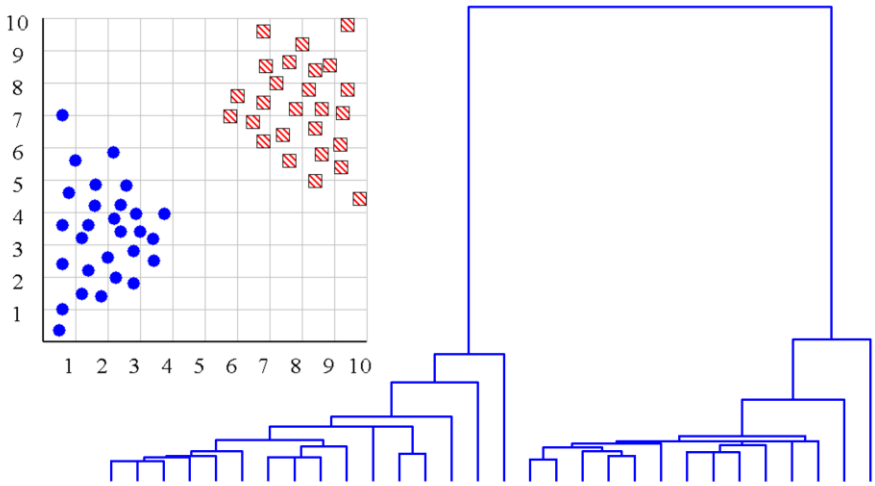
How do you determine the best pair of clusters? These are some metrics.



You get different clusters based on what distance metric you use.

All make the same split into two big clusters, but the smaller clusters differ a lot

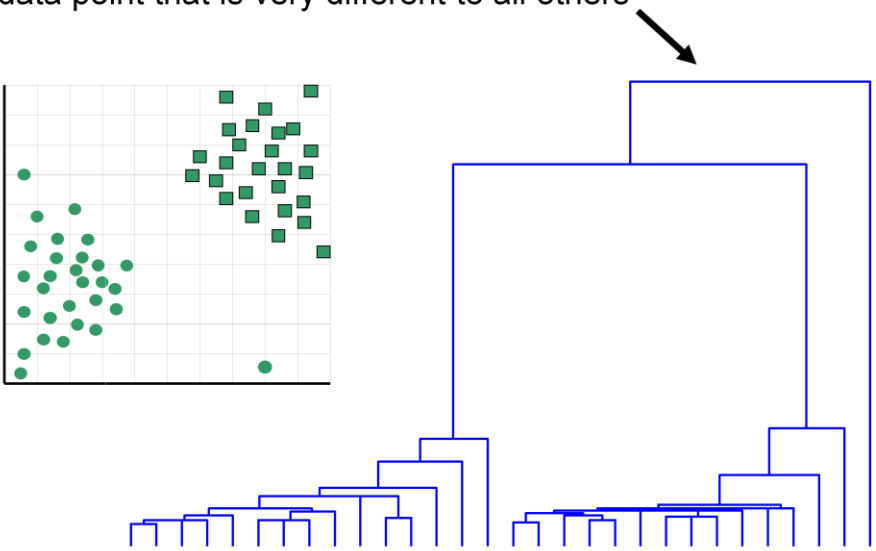
We can look at the dendrogram to determine the “correct” number of clusters. In this case, the two highly separated subtrees are highly suggestive of two clusters. (Things are rarely this clear cut, unfortunately)



Often you are not told beforehand how many clusters there are. Dendrograms are good at letting you know how many clusters there are.

Detecting Outliers

The single isolated branch is suggestive of a data point that is very different to all others



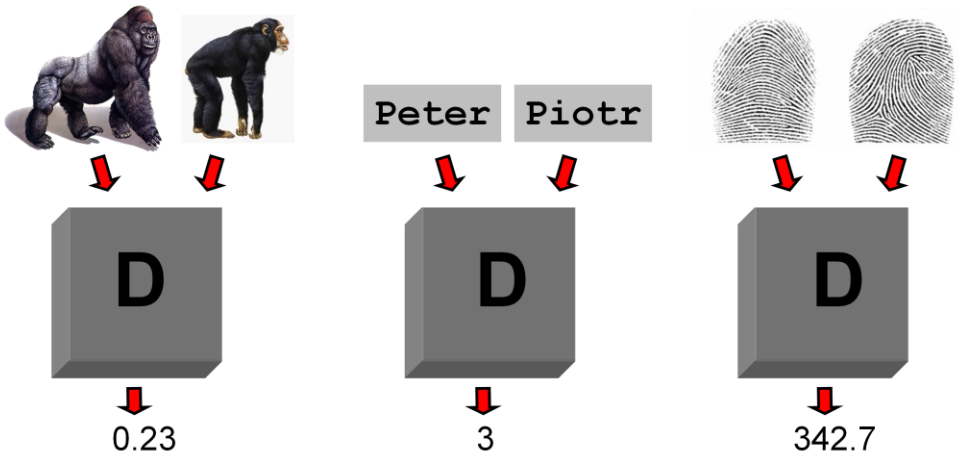
What to do with outliers? Well, we could ignore them!

Summary of Hierarchal Clustering Methods

- No need to specify the number of clusters in advance.
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects.
- Like any heuristic search algorithms, local optima are a problem.
- Interpretation of results is (very) subjective.

Up to this point we have simply assumed that we can measure similarity, but

How do we measure similarity?



Edit Distance

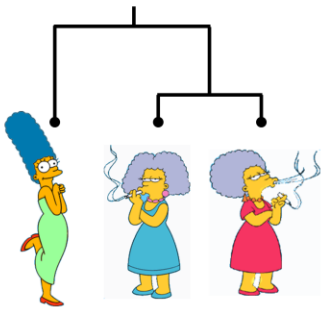
To measure the similarity between two objects, transform one of the objects into the other, and measure how much effort it took. The measure of effort becomes the distance measure.

The edit distance between Patty and Selma.

Change dress color,	**1 point**
Change earring shape,	**1 point**
Change hair part,	**1 point**
D(Patty,Selma) =	**3**

Marge and Selma.

Change dress color,	**1 point**
Add earrings,	**1 point**
Decrease height,	**1 point**
Take up smoking,	**1 point**
Gain weight,	**1 point**
D(Marge,Selma) =	**5**



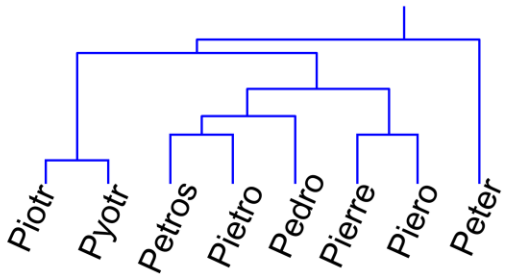
Edit distance between A and B is how much work you need to do to convert A to B, or B to A. Easy to define with strings, harder (but not impossible) to do with other things.

Edit Distance Example

It is possible to transform any string *Q* into string *C*, using only *Substitution*, *Insertion* and *Deletion*.

Assume that each of these operators has a cost associated with it.

The similarity between two strings can be defined as the cost of the cheapest transformation from *Q* to *C*.



How similar are the names "Peter" and "Piotr"?

Substitution	**1 Unit**
Insertion	**1 Unit**
Deletion	**1 Unit**

Peter

↓ Substitution

Piter

↓ Insertion

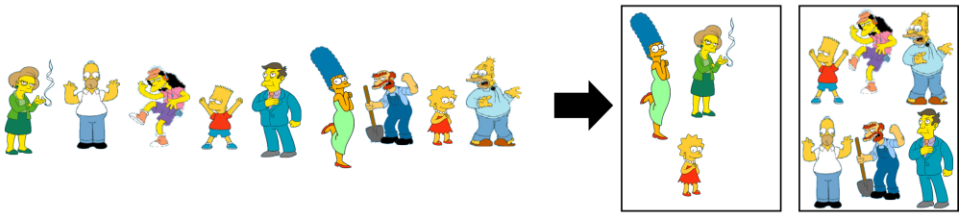
Pioter

↓ Deletion

Piotr

Partitional Clustering

- Nonhierarchical, each instance is placed in exactly one of K nonoverlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K.



Usually you have to tell the algorithm how many clusters you want.

Algorithm *K-means*

1. Decide on a value for *K*.

2. Initialize the *K* cluster centers (randomly, if necessary).

3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *K* cluster centers, by assuming the memberships found above are correct.

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

This step 1 is the lame (really difficult) part of this.

Step 2: Just pick a couple random points. But why cant you stop after that? Because, you don't really know that the two points you picked are any good.

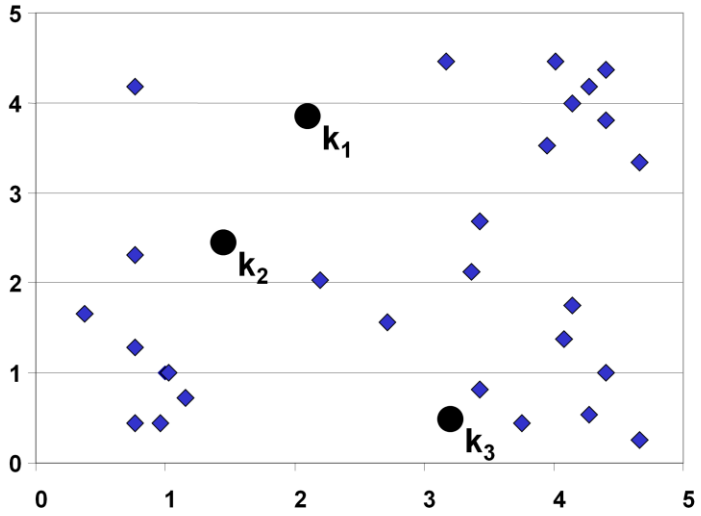
That's where step 3 comes in.

Sometime you won't converge (a point will dither between two or more clusters). In this case you need some more advanced stopping condition.

Sometime initial means are really bad, and have to be reinitialized.

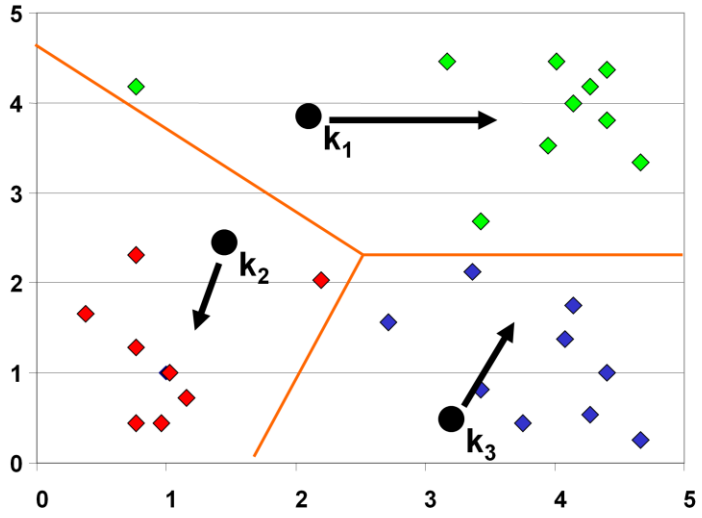
K-means Clustering: Step 1

Algorithm: K-means, Distance Metric: Euclidean Distance



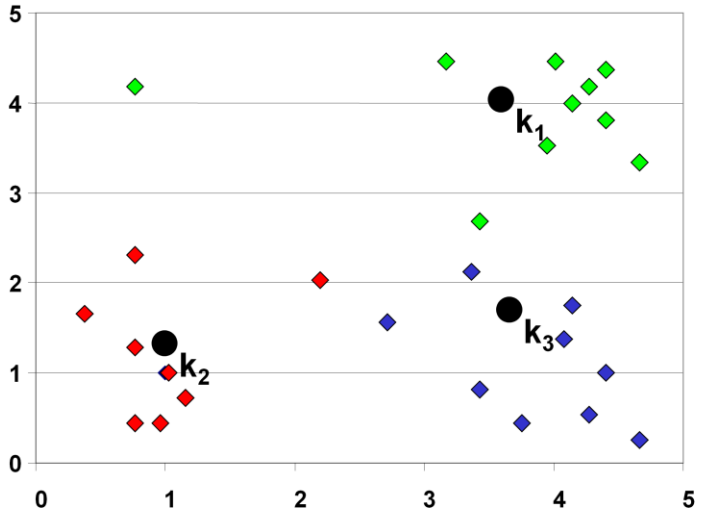
K-means Clustering: Step 2

Algorithm: K-means, Distance Metric: Euclidean Distance



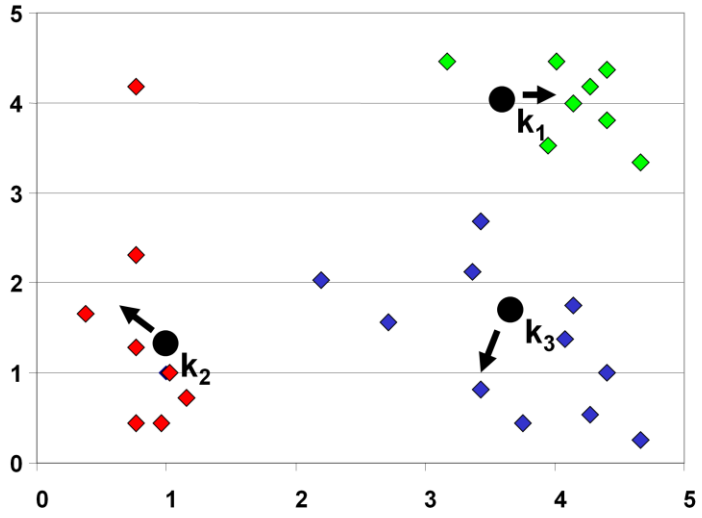
K-means Clustering: Step 3

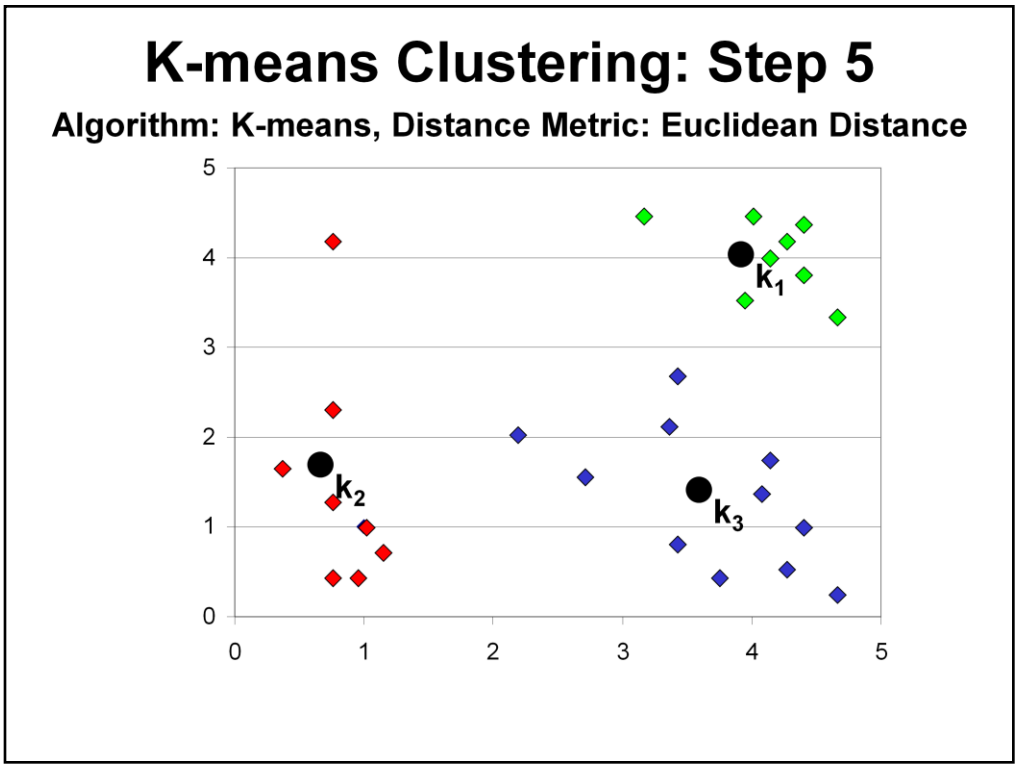
Algorithm: K-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 4

Algorithm: K-means, Distance Metric: Euclidean Distance





“All kinds of bad things can happen, but hey, it sort of works”.

That’s pretty much something you’ll run into with unsupervised learning. When you cant’ even tell the algorithm what’s *supposed* to be right, it can come up with a lot of creative ways to screw it up.

You *can* select your centers not at random, but we trust random more because humans have their own biases.

Comments on the *K-Means* Method

- Pros
  - Relatively efficient: O(tkn), where n is # objects, k is # clusters, and t is # iterations. Normally, k, t << n.
  - Often terminates at a local optimum. The global optimum may be found using techniques such as: deterministic annealing and genetic algorithms
- Cons
  - Applicable only when mean is defined, then what about categorical data?
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

Fortunately, there are a lot of domains where clusters *are* convex. Gaussians, for instance. (Of course, when you actually know the statistical distribution, there are better methods)

You can't guarantee convergence.

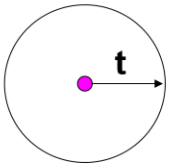
What happens if the data is “streaming” (coming in one at a time)…

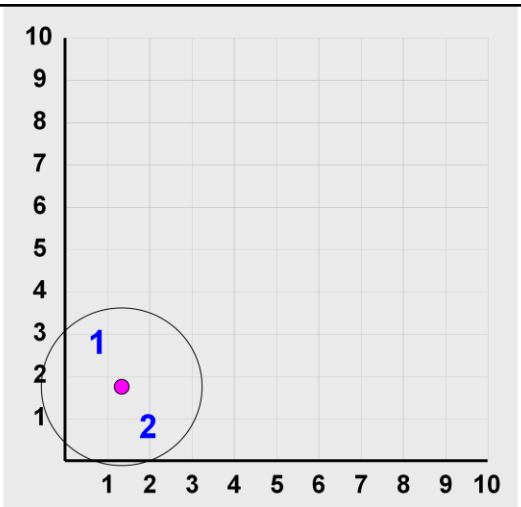
Nearest Neighbor Clustering

- **Items are iteratively merged into the existing clusters that are closest.**
- **Threshold, t, used to determine if items are added to existing clusters or a new cluster is created.**

Useful if you don't start off with all the data

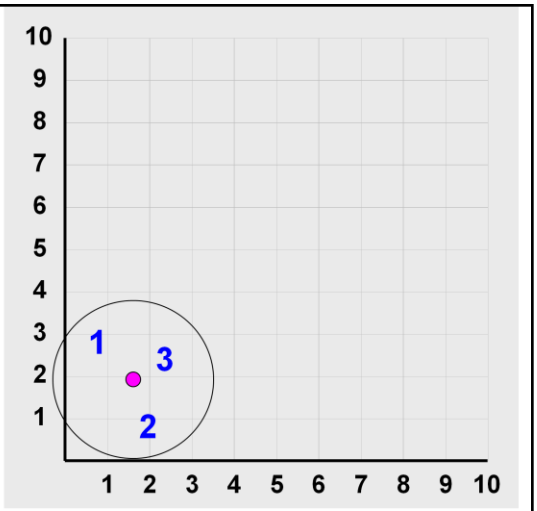
Threshold t





New data point arrives…

It is within the threshold for cluster 1, so add it to the cluster, and update cluster center.

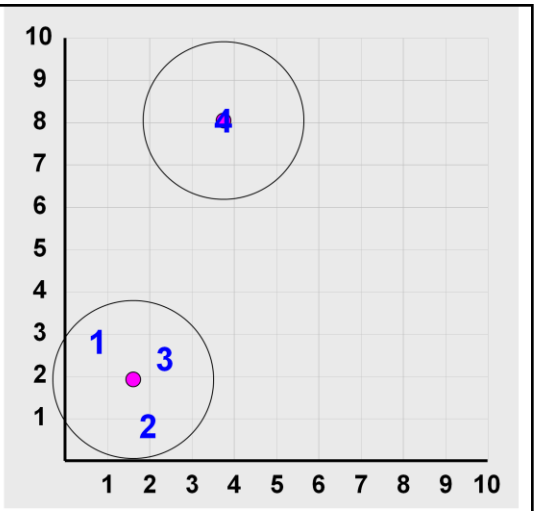


New data point arrives…

It is not within the threshold for cluster 1, so create a new cluster, and so on..

Algorithm is highly order dependent…

It is difficult to determine to in advance…

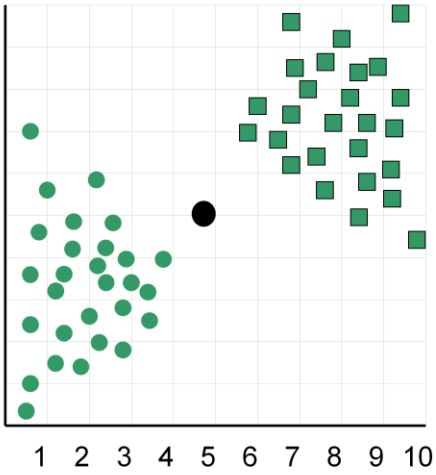


The order the data arrives in will therefore make a big difference. That's not great. Also the clusters can overlap (maybe just add the point to the nearest cluster?).

How can we tell the right number of clusters?

In general, this is a unsolved problem. However there are many approximate methods.

When k = 1, the objective function is 873.0



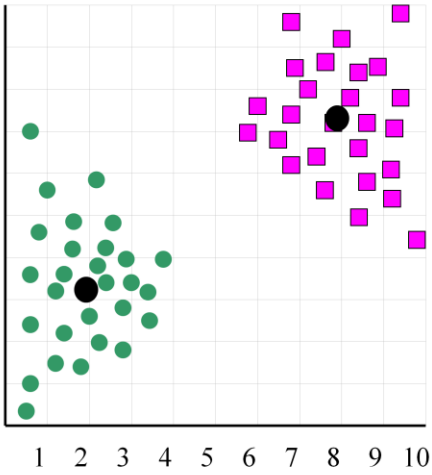
Note, finding the right number of clusters is an unsolved problem.

The objective function is a function we would like to minimize. You can try to minimize the variance, for instance. But then you're just going to keep going until you get N clusters and variance is 0.

How can we tell the right number of clusters?

In general, this is a unsolved problem. However there are many approximate methods.

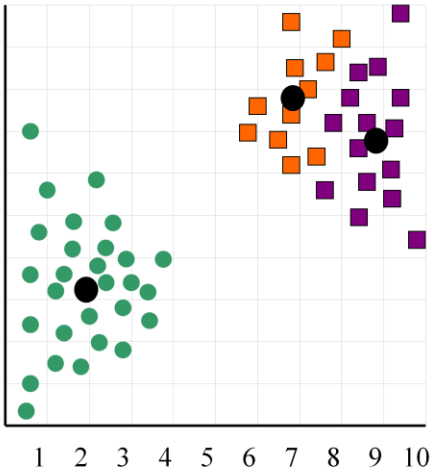
When k = 2, the objective function is 173.1



How can we tell the right number of clusters?

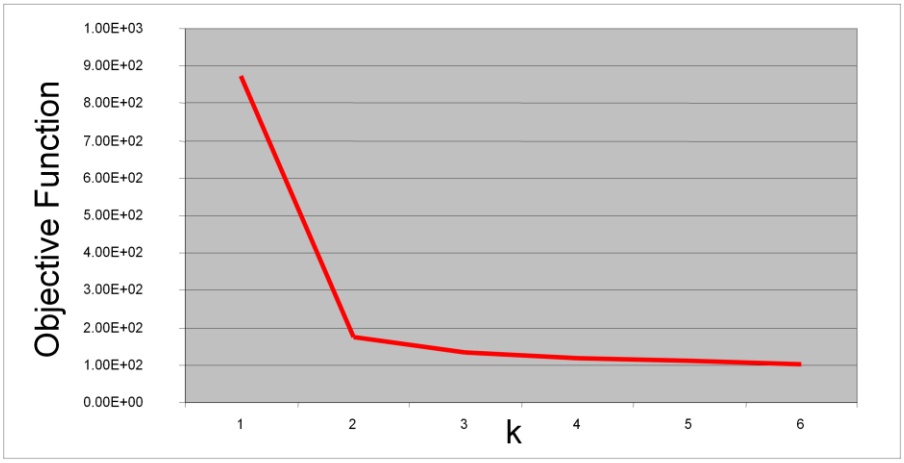
In general, this is a unsolved problem. However there are many approximate methods.

When k = 3, the objective function is 133.6



We can plot the objective function values against K

The abrupt change at K = 2, is highly suggestive of two clusters in the data. This technique is known as “knee finding” or “elbow finding”.



Stop when increasing the number of clusters doesn't help you very much.

Why shouldn't we use more clusters? Then clusters become meaningless

For other methods, see “model selection”. You can, for instance, decide to penalize the objective with a regularization term, etc.

The End