Unsupervised Learning

22c:145 Artificial Intelligence
The University of Iowa

Supervised learning vs. unsupervised learning

- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

What is Clustering?

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

- Organizing data into classes such that there is
  - high intra-class similarity
  - low inter-class similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

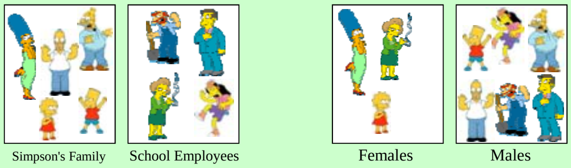
What is clustering for? (cont…)

- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

What is a natural grouping among these objects?

What is a natural grouping among these objects?

Clustering is subjective



What is Similarity?

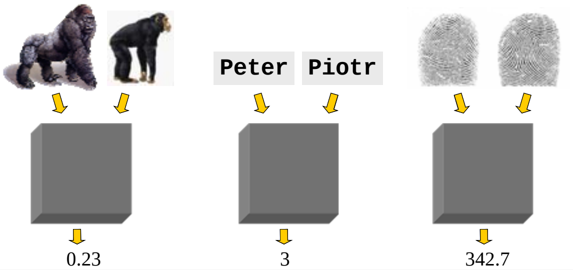
The quality or state of being similar; likeness; resemblance; as, a similarity of features.
Webster's Dictionary

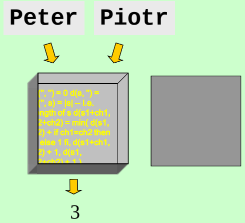
Similarity is hard to define, but… "*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

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 $D(O_1,O_2)$





When we peek inside one of these black boxes, we see some function on two variables. These functions might very simple or very complex.
In either case it is natural to ask, what properties should these functions have?

What properties should a distance measure have?

• D(A,B) = D(B,A)	*Symmetry*
• D(A,A) = 0	*Constancy of Self-Similarity*
• D(A,B) = 0 iff A= B	*Positivity (Separation)*
• D(A,B) ≤ D(A,C) + D(B,C)	*Triangular Inequality*

Intuitions behind desirable distance measure properties

D(A,B) = D(B,A) *Symmetry*
Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

D(A,A) = 0 *Constancy of Self-Similarity*
Otherwise you could claim "Alex looks more like Bob, than Bob does."

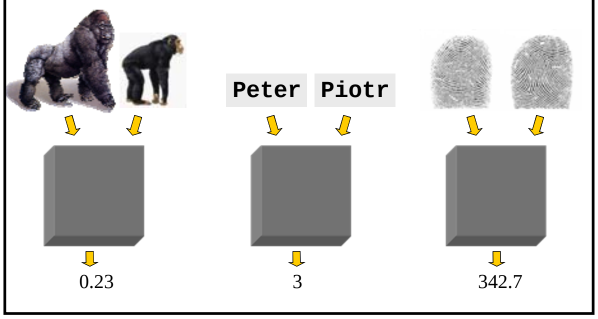
D(A,B) = 0 iff A=B *Positivity (Separation)*
Otherwise there are objects in your world that are different, but you cannot tell apart.

D(A,B) ≤ D(A,C) + D(B,C) *Triangular Inequality*
Otherwise you could claim "Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl."

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
- Incorporation of user-specified constraints
- Interpretability and usability

How do we measure similarity?



A generic technique for measuring similarity

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 distance between Patty and Selma.

```
Change dress color,    1 point
Change earring shape,  1 point
Change hair part,      1 point
```

D(Patty,Selma) = **3**

The distance between Marge and Selma.

```
Change dress color,    1 point
Add earrings,          1 point
Decrease height,       1 point
Take up smoking,       1 point
Lose weight,           1 point
```

D(Marge,Selma) = **5**

This is called the “edit distance” or the “transformation distance”

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*.

Note that for now we have ignored the issue of how we can find this cheapest transformation.

How similar are the names “Peter” and “Piotr”?

Assume the following cost function

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

D(`Peter`,`Piotr`) is 3

`Peter`

Substitution (i for e)

`Piter`

Insertion (o)

`Pioter`

Deletion (e)

`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.

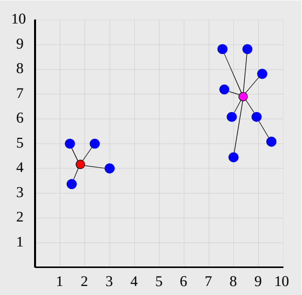
Minimize Squared Error

Distance of a point *i* in cluster *k* to the center of cluster *k*

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

Objective Function



K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) *D* be

  {$\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$},

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, …, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and *r* is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

Algorithm *k-means*

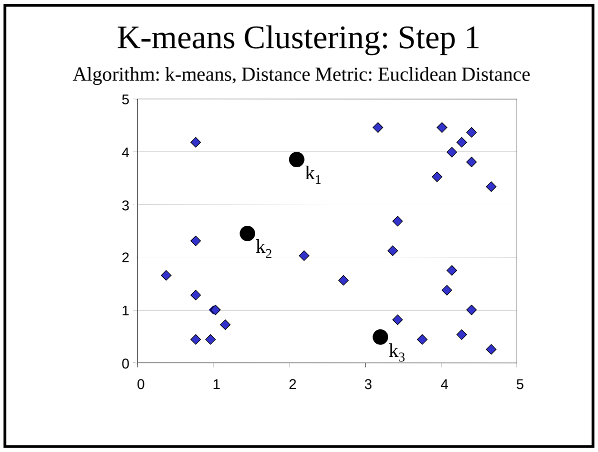
1. Decide on a value for *k*.

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.

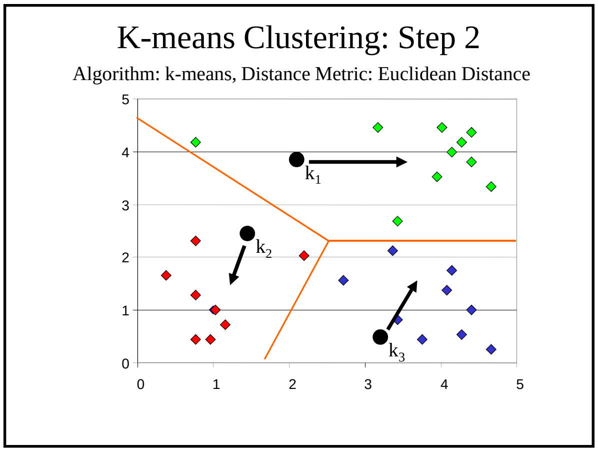
4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.

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



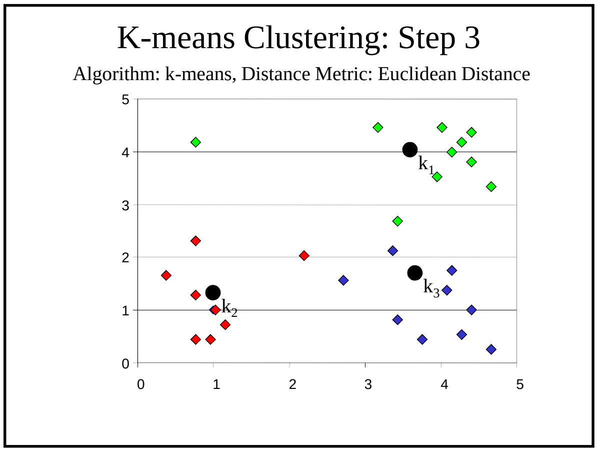
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



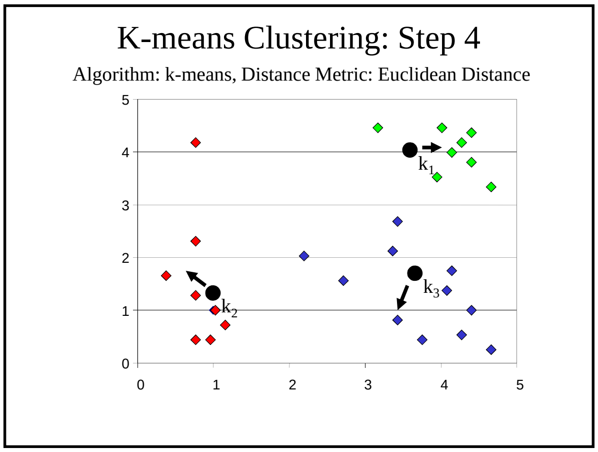
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



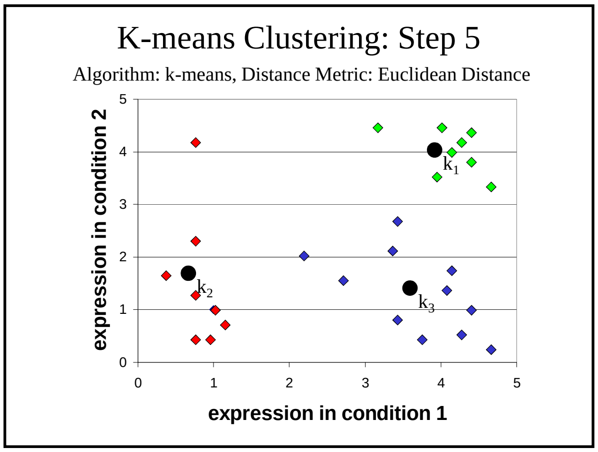
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance

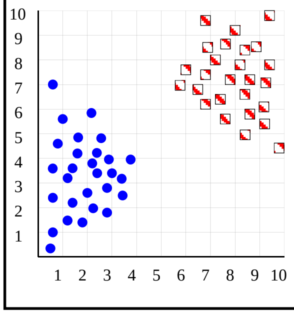


K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance

How can we tell the *right* number of clusters?

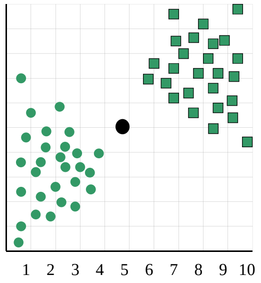
In general, this is an unsolved problem. However there are many approximate methods. In the next few slides we will see an example.



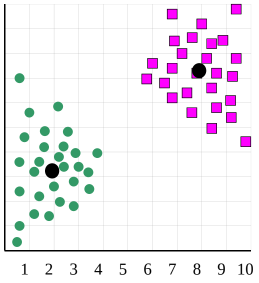
For our example, we will use the dataset on the left.

However, in this case we are imagining that we do NOT know the class labels. We are only clustering on the X and Y axis values.

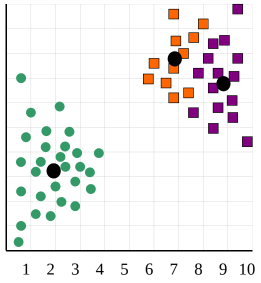
When k = 1, the objective function is 873.0



When k = 2, the objective function is 173.1

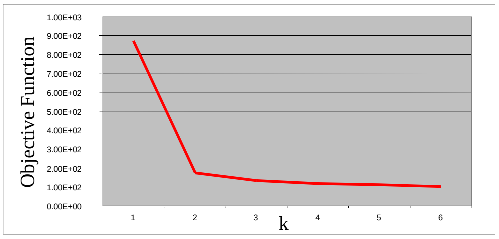


When k = 3, the objective function is 133.6



We can plot the objective function values for k equals 1 to 6…

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



Note that the results are not always as clear cut as in this toy example

“elbow finding” implementation

$$se_{K_i} = \sum_{j=1}^{m} ||t_{ij} - C_k||^2$$

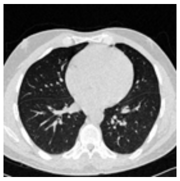
$$se_K = \sum_{j=1}^{k} se_{K_j}$$

New objective function: Find *k* such that

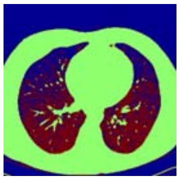
$se_K + ck$

is minimized, where *c* is a constant.

Image Segmentation Results



An image (I)



Three-cluster image (J) on gray values of I

Note that K-means result is "noisy"

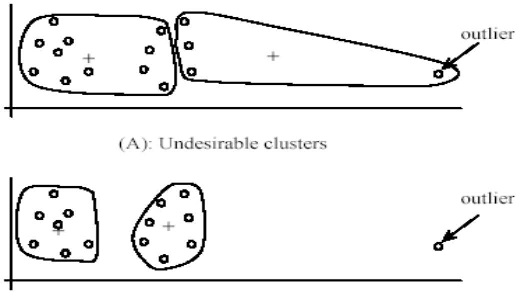
Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
    where n is the number of data points,
    k is the number of clusters, and
    t is the number of iterations.
  - Since both k and t are small, k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, k-mode - the centroid is represented by most frequent values.
- The user needs to specify k.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters

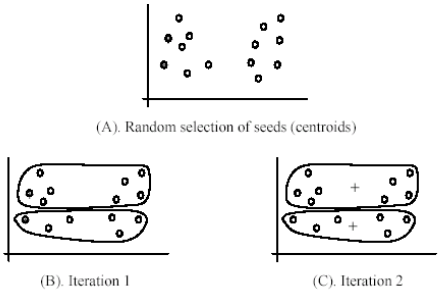
(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

- The algorithm is sensitive to initial seeds.



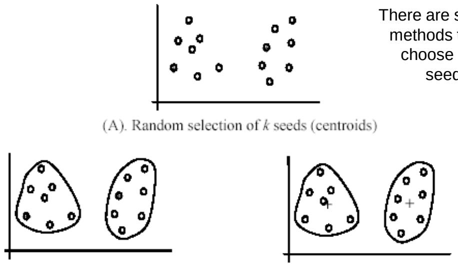
(A). Random selection of seeds (centroids)

(B). Iteration 1

(C). Iteration 2

Weaknesses of k-means (cont …)

- If we use different seeds: good results


(A). Random selection of *k* seeds (centroids)

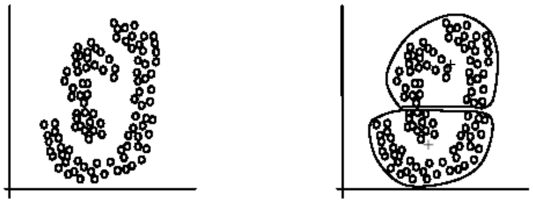
(B). Iteration 1

(C). Iteration 2

There are some methods to help choose good seeds

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).


(A): Two natural clusters

(B): *k*-means clusters

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

K-Means Summary (cont)

- Strength
  - *Relatively efficient training*: *O*(*tknm*), where *n* is # objects, *m* is size of an object, *k* is # of clusters, and *t* is # of iterations. Normally, *k*, *t* << *n*.
  - *Efficient decision*: O(*km*)
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques exhaustive search.
- Weakness
  - Applicable only when *mean* is defined. 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*