Programming for AI

Cindi Thompson[1]

[1]University of San Francisco
Email: cathompson4@usfca.edu

Unsupervised Learning
3 December, 2013

Learning Scenarios

Known classes/ outputs?	Data labels	Type of learning problem
Yes	Given for training data	Supervised Learning
Yes	Given for some but not all training data	Semi-supervised Learning
Yes, but delayed	Hints/feedback	Reinforcement learning
No	None	Unsupervised learning, Clustering

Unsupervised Learning

- What if we want to group instances, but we don't know their classes?
- We just want "similiar" instances to be in the same group.
- Examples:
  - Clustering documents based on text
  - Grouping users with similar preferences
  - Identifying demographic groups

Unsupervised Data Examples

Unlabeled data is everywhere

- Images
- Text
- Sound
- Species
- Diseases
- People!

The Setting

Given: a set, D, of (unlabeled) data $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$ where each $\mathbf{x_i}$ is a d-dimensional feature vector.
Find: ???

The Setting

Given: a set, D, of (unlabeled) data $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$ where each $\mathbf{x_i}$ is a d-dimensional feature vector.

Find: Patterns in the data. Usually viewed as a *clustering* problem

Clustering

- Partition unlabeled examples into disjoint subsets (called clusters), such that:
  - Examples within a cluster are similar
  - Examples in different clusters are different
- Discover new categories in an unsupervised manner (no sample category labels provided)

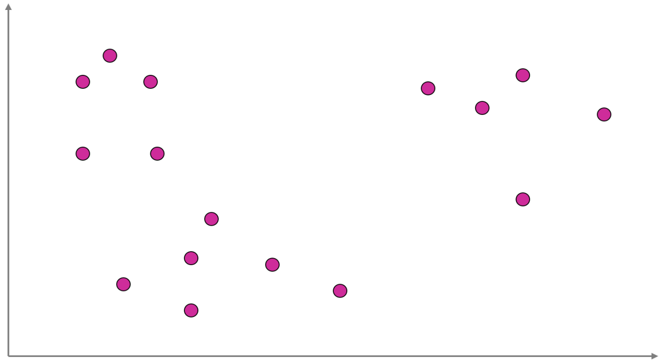
Applications of Clustering

- Cluster retrieved documents
  - to present more organized and understandable results to user – "diversified retrieval"
- Detect near duplicates
  - Entity resolution
    - E.g. IBM == International Business Machines
  - Cheating detection
- Exploratory data analysis
- Automated (or semi-automated) creation of taxonomies
- Compression

And... How do you know that those labels are right anyway?

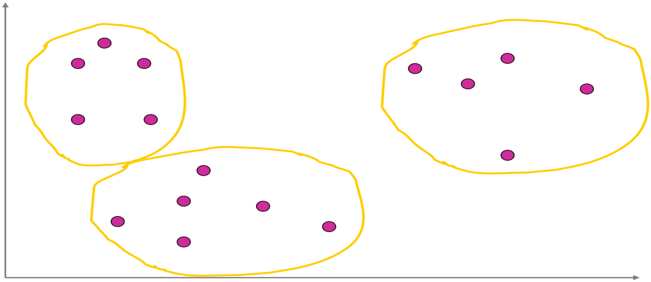
Example

How would you group these?

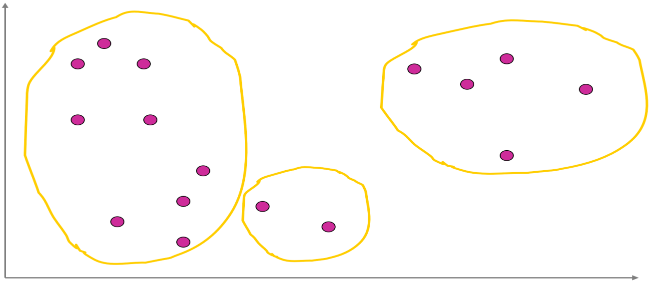


Example

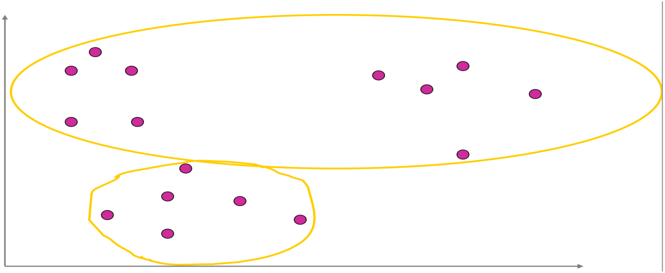
Example



Example



Example



Lots of issues

Divide examples into subsets of “similar” examples

- How to measure similarity
- How to “represent” clusters
- How to evaluate cluster quality
- How many subsets

What is a good clustering?

- Let's require that clusters, like classes, *partition* the data: each example belongs to exactly one cluster.
- What does an example's membership in a cluster tell us?
  - We want it to tell us something about the object.
- Examples within a cluster should be *similar* to each other
- And clusters should be different from each other.
- And there should be fewer of them than the number of examples!

Similarity (Distance) Measures

- Euclidean distance a common metric
- Manhattan distance - distance between two points following only a grid - sometimes used
- Several other possibilities, depending on the data

Ok, what do we actually do?

Two main approaches

- Non-Hierarchical (flat)
- Hierarchical

K-means Clustering

- Let's suppose we want to group our items into K clusters.
  - For the moment, assume K given.
- Approach 1:
- Choose K items at random. We will call these the *centers*.
- Each center is associated with a cluster.
- For each other item, assign it to the cluster that minimizes distance between it and a cluster center.
- This is called K-means clustering.

K-means Clustering

- To evaluate this, we measure the sum of all distances between instances and the center of their cluster.
- But how do we know that we picked good centers?

K-means Clustering

- To evaluate this, we measure the sum of all distances between instances and the center of their cluster.
- But how do we know that we picked good centers?
- We don't. We need to adjust them.

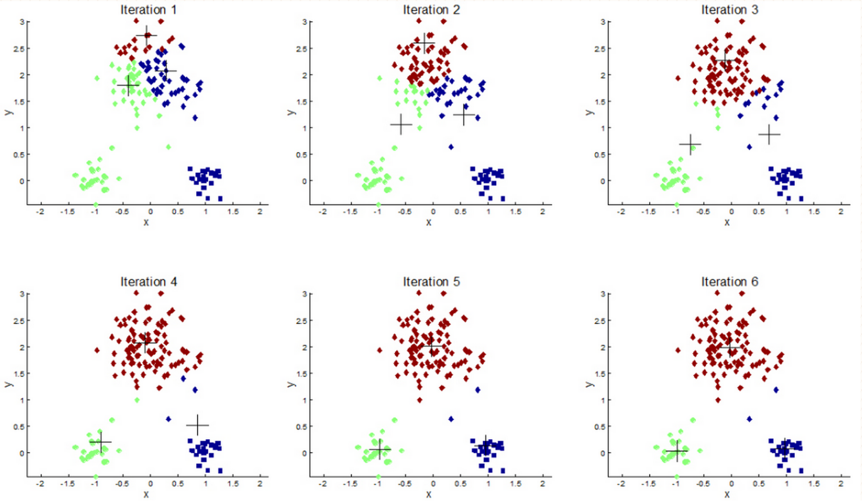
Tuning the centers

- For each cluster, find its centroid.
  - This is the point c that minimizes the total distance to all points in the cluster.
- $m = \frac{1}{n} \sum_{x \in C} x$
- where n is the number of points in a cluster C.
- But what if some points are now in the wrong cluster?

Iterate

- Check all points to see if they are in the correct cluster.
- If not, reassign them.
- Then recompute centers.
- Continue until no points change clusters.

Example


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
x
y

Strengths of K-means

- Easy to implement
- Reasonably scalable
- Remains the most widely used partitional clustering algorithm in practice

Weaknesses of K-means

- Can converge to local minima
- Slow on very large datasets
- Assumes instances are numeric vectors
- Sensitive to noise and outlier data points
- Results are dependent on initial centroid locations

Hierarchical Clustering

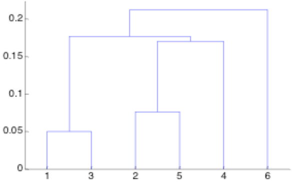
- K-means produces a flat set of clusters.
- Each example is in exactly one cluster.
- What if we want a tree of clusters?
- For example:
  - Topics and subtopics for documents
  - Relationships between clusters, such as species
- We can do this using *hierarchical clustering*

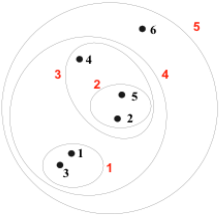
Hierarchical Clustering

Create a tree-based taxonomy; nested clusters

Can be visualized as a *dendogram*

- Tree-like diagram that records the structure of the hierarchy/taxonomy





Advantages of Hierarchical Clustering

- No assumptions on the number of clusters
  - Any desired number of clusters can be obtained by cutting the dendogram at the proper level
- Hierarchical clusterings may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

How might you go about building one?

Hierarchical Agglomerative Clustering (HAC)

Basic algorithm:
Compute the distance matrix between the input data points
Let each data point be a cluster
Repeat
  Merge the two closest clusters
  Update the distance matrix
Until only a single cluster remains

Key operation is the computation of the distance between two clusters

- Different definitions of the distance between clusters lead to different algorithms

Distance between two clusters

Each cluster is a set of points

How do we define distance between two *sets* of points

- Lots of alternatives
- Not an easy task

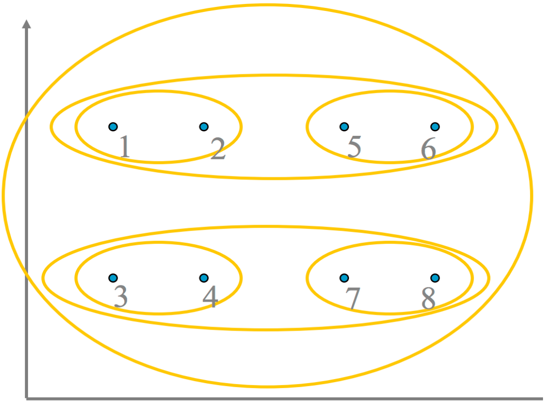
One approach

Single-link distance between clusters Ci and Cj is the minimum distance between any object in Ci and any object in Cj
The cluster distance is defined by the two most similar objects

$$D_{sl}(C_i, C_j) = min_{x,y}\{d(x,y) | x \in C_i, y \in C_j\}$$

Can result in “straggly” (long and thin) clusters

Example

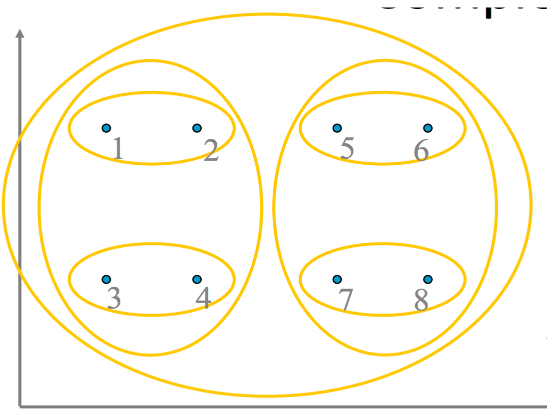


Another Approach

Complete-link distance between clusters Ci and Cj is the maximum distance between any object in Ci and any object in Cj
The distance is defined by the two most dissimilar objects

$$D_{cl}(C_i, C_j) = max_{x,y}\{d(x, y)|x \in C_i, y \in C_j\}$$

Example



Group average cluster distance

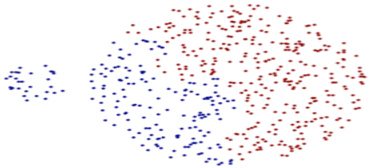
Group average distance between clusters Ci and Cj is the average distance between any object in Ci and any object in Cj

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

Tradeoffs

- Single link:
  - Can handle elliptical shapes
  - But, Sensitive to noise and outliers and produces long, elongated clusters
- Complete link:
  - More balanced clusters (with equal diameter); Less susceptible to noise
  - But: Tends to break large clusters
    All clusters tend to have the same diameter – small clusters are merged with larger ones

Complete link Bad clustering



Average-link Clustering

- Compromise between Single and Complete Link
- Strength: Less susceptible to noise and outliers
- Limitation: Biased towards globular clusters

Complexity of hierarchical clustering

Distance matrix is used for deciding which clusters to merge/split
At least quadratic in the number of data points
Not usable for large datasets
(not to mention curse of dimensionality - similar to K-nn)

Hierarchical Clustering

Advantages:

- Simple and outputs a hierarchy, a structure that is more informative
- Does not require us to pre-specify the number of clusters

Disadvantages:

- Selection of merge points is critical as once a group of objects is merged, it will operate on the newly generated clusters and will not undo what was done previously.
- Thus merge decisions if not well chosen may lead to low-quality clusters
- And as for clustering in general, doesn't scale well to high-dimensional data (large numbers of attributes)