Supervised learning

Unsupervised learning

Unsupervised learning: given data, i.e. examples, but no labels

Administrative

Final project

No office hours today
Unsupervised learning

Given some example without labels, do something!

Unsupervised learning applications

- learn clusters/groups without any label
- customer segmentation (i.e. grouping)
- image compression
- bioinformatics: learn motifs
- find important features
- ...

Unsupervised learning: clustering

Raw data
features
extract features

No "supervision": we're only given data and want to find natural groupings

Unsupervised learning: modeling

Most frequently, when people think of unsupervised learning they think clustering

Another category: learning probabilities/parameters for models without supervision
- Learn a translation dictionary
- Learn a grammar for a language
- Learn the social graph
Clustering

Clustering: the process of grouping a set of objects into classes of similar objects

Applications?

Gene expression data

Data from Garber et al. PNAS (98), 2001.

Face Clustering

Face clustering
Search result clustering

Google News

Clustering in search advertising

- Find clusters of advertisers and keywords
  - Keyword suggestion
  - Performance estimation

Clustering applications

- Find clusters of users
  - Targeted advertising
  - Exploratory analysis

- Clusters of the Web Graph
  - Distributed pagerank computation

Advertisement

- Advertiser
- Bidded

Keyword

~10M nodes

~100M nodes
Data visualization

Wise et al, “Visualizing the non-visual” PNNL

ThemeScapes, Cartia
- Mountain height = cluster size

A data set with clear cluster structure

What are some of the issues for clustering?
- How do we represent an example: features, etc.
- Similarity/distance between examples

Issues for clustering
- How do we represent an example: features, etc.
- Similarity/distance between examples

Flat clustering or hierarchical

Number of clusters
- Fixed a priori
- Data driven

Clustering Algorithms

Flat algorithms
- Usually start with a random (partial) partitioning
- Refine it iteratively
- K means clustering
- Model based clustering
- Spectral clustering

Hierarchical algorithms
- Bottom-up, agglomerative
- Top-down, divisive
Hard vs. soft clustering

Hard clustering: Each example belongs to exactly one cluster

Soft clustering: An example can belong to more than one cluster (probabilistic)
- Makes more sense for applications like creating browsable hierarchies
- You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes

K-means

Most well-known and popular clustering algorithm:

Start with some initial cluster centers

Iterate:
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

K-means: an example

K-means: Initialize centers randomly
K-means: assign points to nearest center

K-means: readjust centers
K-means: assign points to nearest center

K-means: readjust centers

Iterate:
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

K-means

No changes: Done

How do we do this?
K-means

Iterate:
- Assign/cluster each example to closest center
  - get distance to each cluster center
  - assign to closest center (hard cluster)
- Recalculate centers as the mean of the points in a cluster

What distance measure should we use?

Distance measures

Euclidean:

\[ d(x, y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \]

good for spatial data

Clustering documents (e.g. wine data)

One feature for each word. The value is the number of times that word occurs.

Documents are points or vectors in this space.
When Euclidean distance doesn’t work

Which document is closest to \( q \) using Euclidean distance?

Which do you think should be closer?

Issues with Euclidean distance

The Euclidean distance between \( q \) and \( d_2 \) is large

but, the distribution of terms in the query \( q \) and the distribution of terms in the document \( d_2 \) are very similar.

This is not what we want!

Cosine similarity

\[
sim(x, y) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}}
\]

correlated with the angle between two vectors

Cosine distance

Cosine similarity is a similarity between 0 and 1, with things that are similar 1 and not 0

We want a distance measure, cosine distance:

\[
d(x, y) = 1 - \sim(x, y)
\]

- good for text data and many other “real-world” data sets
- is computationally friendly since we only need to consider features that have non-zero values both examples
K-means
Iterate:
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

Where are the cluster centers?

K-means
Iterate:
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

How do we calculate these?

K-means
Iterate:
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

Mean of the points in the cluster:

\[
\mu(C) = \frac{1}{|C|} \sum_{x \in C} x
\]

where:

\[
x + y = \sum_{i=1}^{n} x_i + y_i \quad \frac{X}{|C|} = \sum_{i=1}^{n} \frac{x_i}{|C|}
\]

K-means loss function

K-means tries to minimize what is called the “k-means” loss function:

\[
loss = \sum_{i=1}^{n} d(x_i, \mu_j)^2 \quad \text{where } \mu_j \text{ is cluster center for } x_i
\]

that is, the sum of the squared distances from each point to the associated cluster center
Minimizing k-means loss

Iterate:
1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

\[ \text{loss} = \sum_{i} d(x_i, \mu_k)^2 \] where \( \mu_k \) is cluster center for \( x_i \)

Does each step of k-means move towards reducing this loss function (or at least not increasing)?

Minimizing k-means loss

Iterate:
1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

\[ \text{loss} = \sum_{i} d(x_i, \mu_k)^2 \] where \( \mu_k \) is cluster center for \( x_i \)

This isn’t quite a complete proof/argument, but:
1. Any other assignment would end up in a larger loss
2. The mean of a set of values minimizes the squared error

Minimizing k-means loss

Iterate:
1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

\[ \text{loss} = \sum_{i} d(x_i, \mu_k)^2 \] where \( \mu_k \) is cluster center for \( x_i \)

No! It will find a minimum.

Unfortunately, the k-means loss function is generally not convex and for most problems has many, many minima

We’re only guaranteed to find one of them
K-means variations/parameters

Start with some initial cluster centers

Iterate:
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

What are some other variations/parameters we haven’t specified?

K-means variations/parameters

Initial (seed) cluster centers

Convergence
- A fixed number of iterations
- partitions unchanged
- Cluster centers don’t change

Seed choice

Results can vary drastically based on random seed selection

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings

Common heuristics
- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- Try out multiple starting points
- Initialize with the results of another clustering method
Furthest centers heuristic

\[ \mu_1 = \text{pick random point} \]

for \( i = 2 \) to \( K \):

\[ \mu_i = \text{point that is furthest from any previous centers} \]

\[ \mu_i = \arg \max_x \min_{j: 1 < j < i} d(x, \mu_j) \]

K-means: Initialize furthest from centers

Pick a random point for the first center

What point will be chosen next?

Furthest point from center

What point will be chosen next?
K-means: Initialize furthest from centers

Furthest point from center

What point will be chosen next?

Any issues/concerns with this approach?

Furthest points concerns

If \( k = 4 \), which points will get chosen?

If we do a number of trials, will we get different centers?
Furthest points concerns

- Doesn’t deal well with outliers

K-means++

\[ \mu_1 = \text{pick random point} \]

for \( k = 2 \) to \( K \):
  
  for \( i = 1 \) to \( N \):
    
    \[ s_i = \min d(x_i, \mu_{1..k-1}) \] // smallest distance to any center

\[ \mu_k = \text{randomly pick point proportionate to } s \]

How does this help?

- Makes it possible to select other points
- If #points >> #outliers, we will pick good points
- Makes it non-deterministic, which will help with random runs
- Nice theoretical guarantees!