Supervised learning

Unsupervised learning: given labeled examples

Unsupervised learning

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

Given some example without labels, do something!

Unsupervised applications areas

- 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

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

Date from Garber et al. PNAS (98), 2001.
Search result clustering

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

Advertiser
Bidded Keyword
~10^9 nodes
Data visualization

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

A data set with clear cluster structure

What are some of the issues for clustering?

Issues for clustering

Representation 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: Initialize centers randomly

What points are closest?

K-means: assign points to nearest center

Where are the new centers?
K-means: readjust centers

What points are closest?

K-means: assign points to nearest center

K-means: assign points to nearest center

Where are the new centers?
When do we stop?

No changes: Done
**K-means**

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

How do we do this?

**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 Euclidian distance?

Which do you think should be closer?

Issues with Euclidean distance

The Euclidean distance between q and d2 is large but, the distribution of terms in q and d2 are very similar.

This is not what we want!

Cosine similarity

\[
sim(x, y) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{x \cdot y}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}
\]

correlated with the angle between two vectors.
cosine distance

cosine similarity ranges from 0 and 1, with things that are similar 1 and dissimilar 0

cosine distance:
\[ d(x, y) = 1 - \text{sim}(x, y) \]

- good for text data and many other "real world" data sets
- computationally friendly since we only need to consider features that have non-zero values for both examples

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:
\[ x + y = \sum_{i=0}^{n} x_i + y_i \quad \frac{x}{|C|} = \sum_{i=0}^{n} \frac{x_i}{|C|} \]
K-means loss function

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

$$\text{loss} = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \text{ where } \mu_k \text{ is cluster center for } x_i$$

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

$$\text{loss} = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \text{ where } \mu_k \text{ is cluster center for } x_i$$

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

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

$$\text{loss} = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \text{ where } \mu_k \text{ is cluster center for } x_i$$

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

Does this mean that k-means will always find the minimum loss/clustering?
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

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

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

K!

K-means: Initialize centers randomly

What would happen here?

Seed selection ideas?
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 points (not examples) 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 $K$:

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

K-means: Initialize furthest from centers

Pick a random point for the first center

K-means: Initialize furthest from centers

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

Furthest point from center
What point will be chosen next?

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

Any issues/concerns with this approach?

Furthest points concerns
Furthest points concerns

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

Doesn’t deal well with outliers

K-means++

$\mu_1 = \text{pick random point}$

for $k = 2$ to $K$:
  for $i = 1$ to $N$:
    $s_i = \min d(x_i, \mu_{i,k-1})$ // smallest distance to any center
  $\mu_k = \text{randomly pick point proportionate to } s_i$

How does this help?

Makes it possible to select other points
- if $\#\text{points} >> \#\text{outliers}$, we will pick good points
- Makes it non-deterministic, which will help with random runs
- Nice theoretical guarantees!

K-means++
K-means variations/parameters

Initial (seed) cluster centers

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

K!

How Many Clusters?

Number of clusters K must be provided
How should we determine the number of clusters?
How did we deal with models becoming too complicated previously?

Many approaches

Regularization!!!

Statistical test

k-means loss revisited

K-means is trying to minimize:

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

What happens when \( k \) increases?
k-means loss revisited

K-means is trying to minimize:

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

Loss goes down!

Making the model more complicated allows us more flexibility, but can “overfit” to the data

\[ \text{loss}_{BIC} = \text{loss}_{\text{knmeans}} + K \log N \] (where \( N \) = number of points)

\[ \text{loss}_{AIC} = \text{loss}_{\text{knmeans}} + KN \]

2 regularization options

AIC penalizes increases in \( K \) more harshly

Both require a change to the K-means algorithm

Tend to work reasonably well in practice if you don’t know \( K \)