



High-Dimensional Data Analysis

Lecture 6 - Clustering

Fall semester - 2024

Dr. Eng. Valentin Leplat
Innopolis University
October 2, 2024

Outline

- 1 What is Clustering ?
 - Clustering, Classification and Prediction
 - Supervised versus unsupervised learning
 - Clustering
 - k -means clustering
- 2 Practical Considerations
- 3 How k -Means clustering is NP-hard
- 4 Advanced Topics
- 5 Summary
- 6 Appendices

What is Clustering ?

Outline

- 1 What is Clustering ?
 - Clustering, Classification and Prediction
 - Supervised versus unsupervised learning
 - Clustering
 - k -means clustering
- 2 Practical Considerations
- 3 How k -Means clustering is NP-hard
- 4 Advanced Topics
- 5 Summary
- 6 Appendices

Introduction

There are two broad categories for machine learning:

1. *supervised machine learning*, and
2. *unsupervised machine learning*.

In the first category:

- ▶ Algorithms use *labelled* datasets, and the training data is labeled by experts.
- ▶ examples of the input and output of a desired model are explicitly given, and regression methods are used to find the best model for the given labeled data, via optimization.
- ▶ This model is then used for prediction and classification using new data.

In the second:

- ▶ they must find *patterns* in the data in a principled way in order to determine how to cluster data and generate labels for predicting and classifying new data.
- ▶ discover patterns in the data embedded in the low-rank subspaces so that feature engineering or feature extraction can be used to build an appropriate model.

Clustering, Classification and Prediction

This lecture, we mainly present one of the most prominent unsupervised methods in use today: the *k-means clustering*, and the most popular heuristic to solve it.

We also highlight:

- ▶ how data mining can produce important data features (feature engineering) for later use in model building.
- ▶ the machine learning methods can be broadly used for **clustering** (*Today*) and **classification** (*Lecture 5*), as well as for building **regression models for prediction** (*Lecture 3*).

Critical to all of this machine learning architecture is finding low-rank/dimensional feature spaces that are informative and interpretable.

Feature selection and data mining

- ▶ To exploit data for diagnostics, prediction and control, dominant features of the data must be extracted.
 - ▶ We have introduced in previous lectures **SVD** and **PCA** as methods for determining the dominant correlated structures contained within a data set.
 - ▶ Recall the *Eigenfaces* example from Lecture 4:
 1. the dominant features of a large number of cropped face images were shown.
 2. Eigenfaces ordered by ability to account for *covariance* (or correlation) across face database.
 3. Best set of k features for reconstructing a face in an ℓ_2 -sense with rank- k truncation.
 4. Eigenface modes provide clear and interpretable features for identifying faces: eyes, nose, and mouth regions.
- Instead of working with the high-dimensional measurement space, the feature space allows one to consider a significantly reduced subspace where diagnostics can be performed.

Feature selection and data mining

General goal

The goal of data mining and machine learning is to construct and exploit the intrinsic low-rank feature space of a given data set.

Feature selection and data mining

General goal

The goal of data mining and machine learning is to construct and exploit the intrinsic low-rank feature space of a given data set.

- ▶ The feature space can be found in an unsupervised fashion by an algorithm,
- ▶ or it can be explicitly constructed by expert knowledge and/or correlations among the data.

Feature selection and data mining

General goal

The goal of data mining and machine learning is to construct and exploit the intrinsic low-rank feature space of a given data set.

- ▶ The feature space can be found in an unsupervised fashion by an algorithm,
- ▶ or it can be explicitly constructed by expert knowledge and/or correlations among the data.

For the *Eigenfaces* example:

- ▶ Features are the Principal Components (PC's) obtained either with SVD on centered data matrix X or with EigenValue decomposition on the sample covariance matrix $A = \frac{1}{m}X^T X$.
- ▶ Each PC is high-dimensional, but the weight of that component in representing a face is the *only important* quantity in feature space.
- ▶ With an k -rank truncation, any face needs only k features to represent it in feature space.

Feature selection and data mining - Example 1

- ▶ The [Fisher iris data set](#) consists of measurements of 150 irises of three varieties: setosa, versicolor, and virginica.
- ▶ Each variety includes 50 samples with measurements of sepal length, sepal width, petal length, and petal width in centimeters, so $x^{(i)} \in \mathbb{R}^4$ with $1 \leq i \leq m = 150$.
- ▶ The four features are defined based on interpretable properties of plant biology.
- ▶ For visualization, the following [Figure \(Section 1\)](#) considers only the first three features.

Feature selection and data mining - Example 1

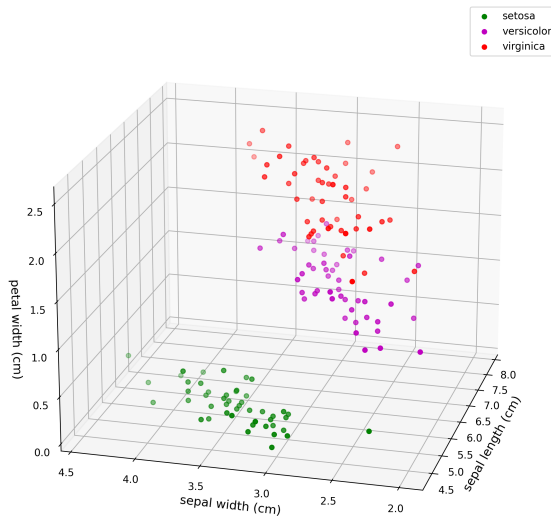


Figure: Fisher iris data set

Feature selection and data mining - Example 1

- ▶ the properties measured can be used as a good set of features for *clustering* and *classification* purposes.
- ▶ **Indeed:** the three iris varieties are well *separated* in this feature space.
- ▶ The setosa iris is most distinctive in its feature profile, while the versicolor and virginica have a small overlap among the samples taken.
- ▶ For this data set, machine learning is certainly not required to generate a good classification scheme.

However, data generally does not so readily reduce down to simple two- and three-dimensional visual cues. Rather, decisions about clustering in feature space occur with many more variables, thus requiring the aid of computational methods to provide good classification schemes.

Feature selection and data mining - Example 2

- ▶ Second example involves an image database of 80 [▶ dogs](#) . and 80 [▶ cats](#) ..
- ▶ **Goal:** to develop an automated classification method to distinguish between cats and dogs.
- ▶ Data for each cat and dog is the 64x64 pixel space of the image, with each image having 4096 measurements.
- ▶ Similar to eigenfaces, SVD will be used to extract dominant correlations among the images.
- ▶ **Demo:** [▶ Section 2](#) .

Feature selection and data mining - Example 2

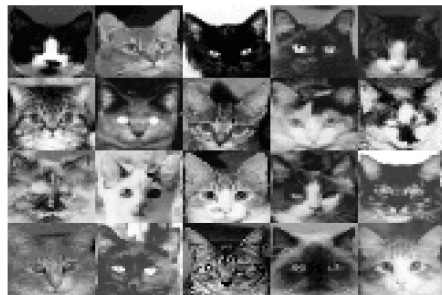
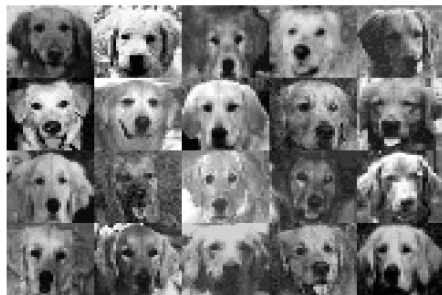


Figure: Example images of dogs (left) and cats (right).

Feature selection and data mining - Example 2

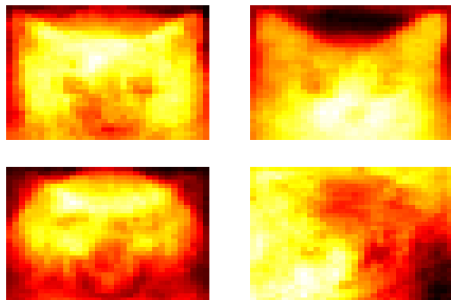


Figure: First four PC's (top left - top right - bottom left - bottom right) from the SVD of the 160 images of dogs and cats, i.e. $V_{1:4}$.

- ▶ Note that the first two PC's (two tops) show that the triangular ears are important features when images are correlated.
- ▶ This is certainly a distinguishing feature for cats, while dogs tend to lack this feature.

Feature selection and data mining - Example 2

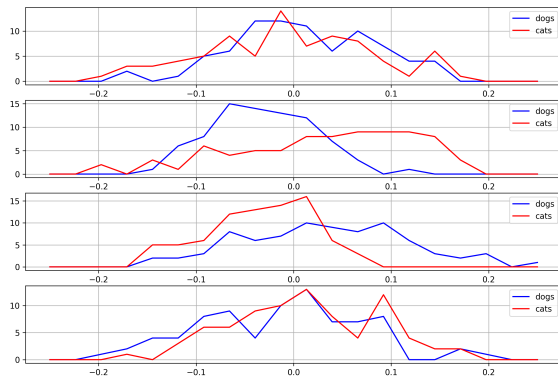


Figure: Histogram of the distribution of loadings for dogs (blue) and cats (red) on the first four dominant PC's (top to bottom)

Note the good separability between dogs and cats using the second mode.

Outline

1 What is Clustering ?

- Clustering, Classification and Prediction
- Supervised versus unsupervised learning
- Clustering
- k -means clustering

2 Practical Considerations

3 How k -Means clustering is NP-hard

4 Advanced Topics

5 Summary

6 Appendices

Some illustration first

- ▶ To illustrate the difference in supervised versus unsupervised learning, consider the [demo here \(Section 3\)](#).
- ▶ This shows a scatter plot of two Gaussian distributions.
 1. Case 1 - the data is well separated so that their means are sufficiently far apart and two distinct clusters are observed.
 2. Case 2 - the two distributions are brought close together so that separating the data is a challenging task.
- ▶ As stated previously: *The goal of unsupervised learning is to discover clusters in the data.*
- ▶ This is a trivial task by visual inspection, provided the two distributions are sufficiently separated.
- ▶ **Otherwise**, it becomes very difficult to distinguish clusters in the data.
- ▶ *Supervised learning* provides labels for some of the data: points are either labeled with green dots or magenta dots and the task is to classify the unlabeled data (grey dots) as either green or magenta.

Some illustration first

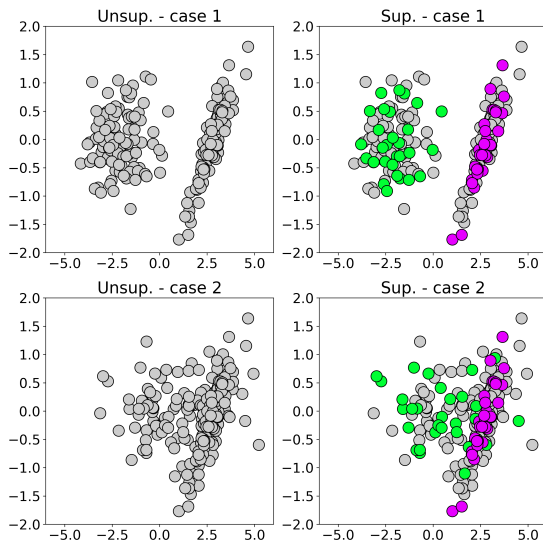


Figure: Illustration of unsupervised versus supervised learning.

Mathematically

Supervised and unsupervised learning can be stated mathematically. Let

$$\mathcal{D} \subset \mathbb{R}^n$$

so that \mathcal{D} is an open bounded set of dimension n .

Mathematically

Supervised and unsupervised learning can be stated mathematically. Let

$$\mathcal{D} \subset \mathbb{R}^n$$

so that \mathcal{D} is an open bounded set of dimension n .

Further, let

$$\mathcal{D}' \subset \mathcal{D}.$$

- ▶ The goal of classification is to build a classifier labeling all data in \mathcal{D} given data from \mathcal{D}' .
- ▶ More precisely: consider a set of data points $x^{(i)} \in \mathbb{R}^n$ and labels $y^{(i)}$ for each point where $j = 1, \dots, o$, where o is the number of points in \mathcal{D} .
- ▶ For the training, we have $m \ll o$ points/samples defining \mathcal{D}' .
- ▶ Labels for the data can come in many forms, from numeric values, including integer labels, to text strings, for simplicity here: $y^{(i)} \in \{-1, +1\}$

Mathematically

- For unsupervised learning:

Input: data set $\{x^{(i)}\}_{i=1}^o$ with $x^{(i)} \in \mathbb{R}^n$ for all i .

Output: labels $\{y^{(i)}\}_{i=1}^o$ with $y^{(i)} \in \{-1, +1\}$ for all i .

- The mathematical framing of unsupervised learning is focused on producing labels $y^{(i)} \in \{-1, +1\}$ for all the data.
- Generally, the data $x^{(i)}$ used for training the classifier is from \mathcal{D}' .
- The classifier is then more broadly applied, i.e. it generalizes, to the open bounded domain \mathcal{D} .

- Supervised learning **provides** labels for the training stage.

Input: data set $\{x^{(i)}\}_{i=1}^o$ and labels $\{y^{(i)}\}_{i=1}^m$

Output: labels $\{y^{(i)}\}_{i=1}^o$ with $y^{(i)} \in \{-1, +1\}$ for all i .

- all the labels are (usually) known in order to build the classifier on \mathcal{D}' .
- The classifier is then applied to \mathcal{D} .

In both cases: if the data used to build a classifier only samples a small portion of the larger domain, then it is often the case that the classifier will not generalize well.

Mathematically - example 1

- ▶ For the data sets considered in our feature selection and data mining section, we can consider in more detail the key components required to build a classification model: $x^{(i)}, y^{(i)}, \mathcal{D}$ and \mathcal{D}' .
- ▶ The Fisher iris data is a classic example for which we can detail these quantities.

Mathematically - example 1

- ▶ For the data sets considered in our feature selection and data mining section, we can consider in more detail the key components required to build a classification model: $x^{(i)}$, $y^{(i)}$, \mathcal{D} and \mathcal{D}' .
- ▶ The Fisher iris data is a classic example for which we can detail these quantities.
- ▶ We begin with the data collected

$$x^{(i)} := \{\text{sepal length, sepal width, petal length, petal width}\}$$

Thus each iris measurement contains four data fields, or features or variables.

- ▶ The labels can be one of the following

$$y^{(i)} := \{\text{setosa, versicolor, virginica}\}$$

- ▶ Finally, there is the domain of the data:
 - $\mathcal{D}' \in \{150 \text{ iris samples : 50 setosa, 50 versicolor, and 50 virginica}\}$
 - $\mathcal{D} \in \{\text{the universe of setosa, versicolor and virginica irises}\}$

Exercise: do the same for the dogs/cats examples.

Some additional thoughts

- ▶ Supervised and unsupervised learning methods aim to either create algorithms for classification, clustering, or regression.
- ▶ Obviously, and based on very simple examples: generalization can be very difficult.
- ▶ Deep neural networks are state-of-the-art machine learning algorithms for *regression* and *classification*.
- ▶ However, they have difficulty generalizing.
- ▶ Creating strong generalization schemes is at the forefront of machine learning research.
- ▶ **Challenging 2-D example:** [▶ Section 4](#).

Outline

1 What is Clustering ?

- Clustering, Classification and Prediction
- Supervised versus unsupervised learning
- Clustering
- k -means clustering

2 Practical Considerations

3 How k -Means clustering is NP-hard

4 Advanced Topics

5 Summary

6 Appendices

Definition's attempt for clustering

- ▶ Clustering is one of the main tasks in unsupervised learning. It is the process of detecting *clusters* in the dataset.
- ▶ Often the membership of a cluster can replace the role of a label in the training dataset.
- ▶ In general, clusters reveal a lot of information about the underlying structure of the data.
- ▶ In practice, we do not always have access to this additional label. Instead, one uses clustering algorithms to find natural clusterings of the data.
- ▶ This raises the question of what a “clustering” is, in the first place ?

Broad definition

Technically, any partition of the dataset \mathcal{D} into K subsets $\mathcal{C}_1, \dots, \mathcal{C}_K$ (with K “given” or to predict) can be called a clustering. That is:

$$\cup_{k=1}^K \mathcal{C}_k = \mathcal{D}, \text{ and } \cap_{k=1}^K \mathcal{C}_k = \emptyset$$

intuitively we understand that not all partitions are a natural clustering of the dataset; our goal therefore will be to define what a “good” clustering is.

Some Attempts to Define a “Good” Cluster

Definition of Cluster: Attempt 1

A “good” **cluster** is a subset of points which are closer to each other than to all other points in the dataset.

This is a good start but this definition does not apply to the clusters in the simple example below:

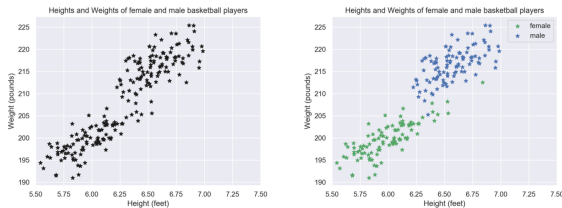


Figure: Height vs weight scatter plot of basketball players. In the plot on the right, the points in green and blue respectively correspond to female and male players.

What about the points in the middle of the plot which are far away from the points on the top right corner or the bottom left corner ?

Some Attempts to Define a “Good” Cluster

Definition of Cluster: Attempt 2

A “good” **cluster** is a subset of points which are closer to the **mean** of their own cluster than to the mean of other clusters.

Here Mean and Variance are defined as follows:

Definition: Mean and Variance of Clusters

Let \mathcal{C}_k be one of the clusters for a dataset \mathcal{D} . Let $m_k = |\mathcal{C}_k|$ denote the cluster size. The mean of the cluster \mathcal{C}_k is

$$\mu_k = \frac{1}{m_k} \sum_{x^{(i)} \in \mathcal{C}_k} x^{(i)}$$

and the variance within the cluster \mathcal{C}_k is

$$\sigma_k^2 = \frac{1}{m_k} \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2$$

Some Attempts to Define a “Good” Cluster

Definition of Cluster: Attempt 2

A “good” **cluster** is a subset of points which are closer to the **mean** of their own cluster than to the mean of other clusters.

Here Mean and Variance are defined as follows:

Definition: Mean and Variance of Clusters

Let \mathcal{C}_k be one of the clusters for a dataset \mathcal{D} . Let $m_k = |\mathcal{C}_k|$ denote the cluster size. The mean of the cluster \mathcal{C}_k is

$$\mu_k = \frac{1}{m_k} \sum_{x^{(i)} \in \mathcal{C}_k} x^{(i)}$$

and the variance within the cluster \mathcal{C}_k is

$$\sigma_k^2 = \frac{1}{m_k} \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2$$

Some Attempts to Define a “Good” Cluster

Have you noticed ?

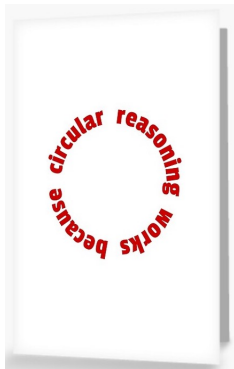


Figure: Definition Attempt 2 appears to be using circular reasoning: it defines clusters using the mean of the clusters, but the mean can only be calculated once the clusters have been defined.

Outline

1 What is Clustering ?

- Clustering, Classification and Prediction
- Supervised versus unsupervised learning
- Clustering
- *k*-means clustering

2 Practical Considerations

3 How *k*-Means clustering is NP-hard

4 Advanced Topics

5 Summary

6 Appendices

k-Means Clustering

In this section: we present a particular partition of the dataset called the *k-means clustering*.

- ▶ Given K , the desired number of clusters, the *k-means clustering* partitions \mathcal{D} into K clusters $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K$ so as to minimize the cost function $\sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2$ or equivalently, it boils down to solve the optimization problem:

$$\operatorname{argmin}_{\{\mu_k\}_{k=1}^K} \sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2 \quad (1)$$

- ▶ This can be seen as minimizing the average of the individual cost of the K clusters, where cost of \mathcal{C}_k is the cluster size times the variance.
- ▶ This idea is similar to our second attempt: we want the distance of each data point to the mean of the cluster to be small.

k -Means Clustering

- ▶ **Important:** The process of finding the optimal solution for Problem (1) is called the *k-Means clustering problem*.
- ▶ In general, solving the optimization problem as stated is *NP-hard*, making it *computationally intractable* even if there are only two clusters.¹
- ▶ However, there a number of *heuristic* algorithms that provide good performance despite not having a guarantee that they will converge to the globally optimal solution.
- ▶ Somewhat confusingly, the most famous *heuristic algorithm* that is used to solve the *k-Means clustering problem* is also called *k-Means*. It is also called the Lloyd algorithm, and was proposed in
"Stuart Lloyd. *Least squares quantization in PCM*. IEEE transactions on information theory, 28(2):129–137, 1982."

¹There is extensive research on finding near-optimal solutions to *k*-Means. Considering its NP-hardness, we believe that an algorithm that is guaranteed to produce the optimum solution on all instances must require exponential time.

Lloyd's Algorithm - assignments

- ▶ Assign every one the data points to one of the K clusters.
- ▶ We represent these assignments by giving every one of the m data points a binary *responsibility vector* $r^{(i)}$.
- ▶ This vector is all zeros except one component, which corresponds to the cluster it is assigned to.
- ▶ That is, if $x^{(i)}$ is assigned to cluster k , then $r^{(i)}[k] = 1$ and all of the other entries in $r^{(i)}$ are zero.
- ▶ This an example of *one-hot coding* in which an integer between 1 and K is encoded as a length- K binary vector that is zero everywhere except one place.

Lloyd's Algorithm - principle

- ▶ It is given some initial clustering (we discuss some choices for initialization later)
- ▶ Lloyd's Algorithm is an iterative algorithm that loops until it converges to a (locally optimal) solution.
- ▶ Within each loop, it makes two kinds of updates:
 1. we find the mean of each current cluster.
 2. Then for each data point, we assign it to the cluster whose mean is the closest to the point, without updating the mean of the clusters.
- ▶ In case there are multiple cluster means that the point is closest to, we apply the *tie-breaker rule* that the point gets assigned to the current cluster if it is among the closest ones; otherwise, it will be randomly assigned to one of them.
- ▶ We repeat this process until there is no point that is mis-assigned.

Lloyd's Algorithm

input: Point set $\{x^{(i)}\}_{i=1}^m$, number of clusters K

for $i \leftarrow 1$ **to** m **do**

$r^{(i)} \leftarrow (0, 0, \dots, 0)$

Zero out the responsibilities.

$k' \leftarrow \text{RandomInteger}(1, K)$

Make one of them randomly one to initialize.

$r^{(i)}[k'] \leftarrow 1$

end

while *one of the $r^{(i)}$ changes* **do**

for $k \leftarrow 1$ **to** K **do**

$N_k \leftarrow \sum_{i=1}^m r^{(i)}[k]$

Compute the number assigned to cluster k .

$\mu_k \leftarrow \frac{1}{N_k} \sum_{i=1}^m r^{(i)}[k] x^{(i)}$

Compute the mean of the k th cluster.

end

for $i \leftarrow 1$ **to** m **do**

$r^{(i)} \leftarrow (0, 0, \dots, 0)$

Zero out the responsibilities.

$k' \leftarrow \text{argmin}_k \|x^{(i)} - \mu_k\|_2^2$

Find the closest mean.

$r^{(i)}[k'] \leftarrow 1$

end

end

output: assignments $\{r^{(i)}\}_{i=1}^m$ for each datum, and cluster means $\{\mu_k\}_{k=1}^K$

Lloyd's Algorithm - updates

- ▶ The *Lloyd's algorithm* is an iterative scheme with two main steps².
- ▶ In *numerical optimization's* terminology, the algorithm is a *coordinate descent* scheme, i.e., it alternates between 1) minimizing each of the μ_k , and 2) minimizing each of the $r^{(i)}$.
- ▶ To ease the derivation of the updates, let us reformulate a bit the Problem (1) as follows:

$$\operatorname{argmin}_{\{r^{(i)}\}_{i=1}^m, \{\mu_k\}_{k=1}^K} \sum_{i=1}^m \sum_{k=1}^K r^{(i)}[k] \|x^{(i)} - \mu_k\|_2^2 \quad (2)$$

²putting aside the initialization step

Lloyd's Algorithm - updates

Minimizing the $r^{(i)}$,

- ▶ Looking at Problem (2): the $r^{(i)}$ only appears in one of the outer sums, because it only affects one of the data points.
- ▶ There are only K possible values for $r^{(i)}$ and so we can minimize it (holding everything else fixed) by choosing $r^{(i)}[k] = 1$ for the cluster that has the smallest distance:

$$r^{(i)}[k] = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_{k'} \|x^{(i)} - \mu_{k'}\|_2^2 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Lloyd's Algorithm - updates

Minimizing the μ_k ,

- ▶ Looking at Problem (2) and having fixed everything else, we note that each μ_k only depends on one of the parts of the inner sums.
- ▶ We can think about the Problem written in terms of only one of these:

$$\operatorname{argmin}_{\mu_k} \sum_{i=1}^m r^{(i)}[k] \|x^{(i)} - \mu_k\|_2^2 := \sum_{i=1}^m r^{(i)}[k] \langle x^{(i)} - \mu_k, x^{(i)} - \mu_k \rangle \quad (4)$$

- ▶ To minimize, we differentiate this objective with respect to μ_k , set the resulting gradient to zero, and then solve for μ_k ³.
- ▶ We obtain:

$$\mu_k = \frac{\sum_{i=1}^m r^{(i)}[k] x^{(i)}}{\sum_{i=1}^m r^{(i)}[k]} \quad (5)$$

³Exercise: verify this

Lloyd's Algorithm - updates

Minimizing the μ_k ,

- ▶ Thus, for Euclidean distances anyway, taking the average of the assigned data gives the mean that minimizes the variance (holding everything else fixed).
- ▶ It can also be useful to think of the *k-Means/Lloyd's* algorithm as finding a **Voronoi** partition of the data space.

Definition - Voronoi diagram

A Voronoi diagram/partition divides the plane into separate regions where

- ▶ each region contains exactly one generating point, called the seed, and
- ▶ every point in a given region is closer to its seed than any other.

Illustration: ▶ [Section 5](#).

Lloyd's Algorithm - examples

- ▶ applying Lloyd's algorithm to a set of 30991 articles from Grolier's Encyclopedia [▶ website](#).
- ▶ The articles are represented as a sparse vector of word counts with a vocabulary of the 15276 most common words, excluding stop words such as “the” and ”and”.
- ▶ these counts were treated as the features directly, leading to a very simple notion of distance.
- ▶ Here: algo used for $K = 12$ and initialized with *k-Means++*⁴ before applying Lloyd's algorithm.

⁴discussed later and pseudo-code given in Appendix

Lloyd's Algorithm - examples

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
education	south	war	war	art	light
united	population	german	government	century	energy
american	north	british	law	architecture	atoms
public	major	united	political	style	theory
world	west	president	power	painting	stars
social	mi	power	united	period	chemical
government	km	government	party	sculpture	elements
century	sq	army	world	form	electrons
schools	deg	germany	century	artists	hydrogen
countries	river	congress	military	forms	carbon
Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12
energy	god	century	city	population	cells
system	world	world	american	major	body
radio	century	water	century	km	blood
space	religion	called	world	mi	species
power	jesus	time	war	government	cell
systems	religious	system	john	deg	called
television	steel	form	life	sq	plants
water	philosophy	united	united	north	animals
solar	science	example	family	south	system
signal	history	life	called	country	human

Figure: result of a simple application of Lloyd's Algorithm to a set of Grolier encyclopedia articles. Shown above are the words with the highest “mean counts” in each of the cluster centers, with clusters as columns. Even with this very simple approach, Lloyd's Algorithm identifies groups of words that seem conceptually related.

Why Does Lloyd's Algorithm Terminate in Finite time?

For the following, we consider the notations used in Problem (1)⁵

- ▶ The k -means/Lloyd's algorithm is a coordinate descent iterative scheme, and the iterations are trying to improve the cost.
- ▶ Based on Problem (4) and its solution given in Equation (5), we can easily derive the following Lemma:

Lemma

Given a set of points $x^{(1)}, \dots, x^{(m)}$ their mean $\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$ is the point that minimizes the average squared distance to the points.

Proof: set $K = 1$, and $r^{(i)} = 1$ for all i for Problem (4) then follow the same rationale as for deriving Equation (5).

⁵no use of *assignments vectors* to ease the notations.

Why Does Lloyd's Algorithm Terminate in Finite time?

Theorem

Each iteration of the Lloyd's Algorithm, possibly except for the last iteration before termination, strictly decreases the total cluster cost from (1).

Proof The total cost at the end of one iteration (the current iteration) is:

$$\sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}'_k} \|x^{(i)} - \mu'_k\|_2^2$$

where: μ'_k is the mean of the **newly** defined cluster \mathcal{C}'_k . Recall that:

1. The first steps compute the optimal μ'_k given the clusters \mathcal{C}_k (and assignments).
2. The second step finds the optimal clusters (and then assignments) \mathcal{C}'_k based on the μ'_k computed at step 1.

Why Does Lloyd's Algorithm Terminate in Finite time?

Proof - continued During the second step of the current iteration, for a data point $x^{(i)}$, two scenarios may have occurred given the *tie-breaker* rule:

1. $x^{(i)}$ was in \mathcal{C}_k and is now assigned to \mathcal{C}'_k , then μ'_k was one of the closest cluster means, i.e., $x^{(i)}$ gets assigned to the current cluster. When this happens for all $x^{(i)}$, the algorithm terminates immediately after this iteration since no point is reassigned to a different cluster.
2. $x^{(i)}$ is assigned to another cluster \mathcal{C}'_k

Therefore, we can write:

$$\sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}'_k} \|x^{(i)} - \mu'_k\|_2^2 \leq \sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu'_k\|_2^2$$

The equality holds at last iteration, i.e., when the algorithm terminates, otherwise we have a strict inequality (at least $x^{(i)}$ is assigned to another cluster). Hence, except at the last iteration before termination we have:

$$\sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}'_k} \|x^{(i)} - \mu'_k\|_2^2 < \sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu'_k\|_2^2 \quad (6)$$

Why Does Lloyd's Algorithm Terminate in Finite time?

Proof - continued We focus now on the terms $\sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k'\|_2^2$ which we can upper-bound using the previous Lemma: each sum is smaller than the sum of squared distance between the same set of points to any other point, including the mean μ_k at the beginning of the iteration!⁶ Hence:

$$\sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k'\|_2^2 \leq \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2$$

for any k . Summing over all the clusters k , we have:

$$\sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k'\|_2^2 \leq \sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2 \quad (7)$$

Combining Equations (6) and (7), except the the last iteration before termination, we have:

$$\sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k'} \|x^{(i)} - \mu_k'\|_2^2 < \sum_{k=1}^K \sum_{x^{(i)} \in \mathcal{C}_k} \|x^{(i)} - \mu_k\|_2^2$$

Note that the right hand side of the inequality is the total cost at the beginning of the iteration. QED.

⁶recall that μ_k' is the optimal mean for the cluster \mathcal{C}_k

Why Does Lloyd's Algorithm Terminate in Finite time?

Now we are ready to prove that the Lloyd's algorithm is guaranteed to terminate in finite time.

1. Since each iteration strictly reduces the cost, we conclude that the current clustering (i.e., partition) will never be considered again, except at the last iteration when the algorithm terminates.
2. Since there is only a finite number of possible partitions of the dataset \mathcal{D} ⁷,
then the algorithm must terminate in finite time !

⁷even if this number C_m^K (the number of ways, disregarding order, that K objects can be chosen among m objects) is big, this is still finite.

Why Does Lloyd's Algorithm Terminate in Finite time?

Crucial remarks

- ▶ **TRUE implication:** the algorithm will never enter in an *infinite* loop, it will end in **finite** time.
- ▶ **FALSE implication:** the algorithm goes through all the possible partitions.
- ▶ **What happens:**
Given an initialization, the algorithm will go through a finite sequence of partitions and stops.

Practical Considerations

How to Pick the Initial Clustering ?

- ▶ The choice of initial clusters greatly influences the quality of the solution found by the k -means/Lloyd's algorithm.
- ▶ Numerical Optimization's Reason: the objective function in Problem (1) is highly non-convex, with many local minima.
- ▶ Some of them are very easy to see: for example, you could clearly permute the indices of the clusters and wind up in a “different” solution that was just as good.
- ▶ **Recall** that the k -means/Lloyd's algorithm is a just an heuristic, it represents just one way to solve the k -Means clustering problem given in (1) ⁸

⁸even this problem represents a particular partition, based on the "means" and the Euclidean distance!

How to Pick the Initial Clustering ?

Naive method

- ▶ The most naive method is to pick k data points randomly to serve as the initial cluster centers and create k clusters by assigning each data point to the closest cluster center.
- ▶ However, this approach can be problematic.
 - Suppose there exists some “ground truth” clustering of the dataset.
 - By picking the initial clusters randomly, we may end up splitting one of these ground truth clusters.⁹
 - The final clustering ends up being very sub-optimal.

Thus one tries to select the initial clustering more intelligently.

⁹e.g., two initial centers are drawn from within the same ground truth cluster

How to Pick the Initial Clustering ?

Random restarts

- ▶ running the algorithm several times (e.g., 10 or 20), with different random seeds so that the initial $r^{(i)}$ might land in better places.
 - ▶ Then one looks at the final value of the objective to choose the best solution.
- Another practical strategy for larger data sets is to do these restarts with a smaller subset of the data in order to find some reasonable cluster centers before running the full iteration.

How to Pick the Initial Clustering ?

k-Means++

- ▶ the popular *k*-Means++ is an initialization procedure introduced in D. Arthur and S. Vassilvitskii. *k*-Means++: *The advantages of careful seeding*. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1027–1035. [▶ pdf](#)
- ▶ It does the following:
 1. Choose one center uniformly at random among all data points.
 2. For each data point $x^{(i)}$: compute $D(x^{(i)})$, the distance between $x^{(i)}$ and the nearest center which has already been chosen.
 3. Choose a new data point at random as a new center, where a point $x^{(i)}$ is chosen with probability proportional to $(D(x^{(i)}))^2$.
 4. Repeat steps 2 and 3 until K centers have been chosen.
- ▶ Works nicely: in fact, authors show that *k*-Means++ can do well even without using Lloyd's algorithm at all.

Choice of K ?

- ▶ Above we assumed that the number of clusters K is given, but in practice you have to choose the appropriate number of clusters K .
- ▶ **Puzzle 1:** Is there a value of k that guarantees an optimum cost of 0?

Choice of K ?

- ▶ Above we assumed that the number of clusters K is given, but in practice you have to choose the appropriate number of clusters K .
- ▶ **Puzzle 1:** Is there a value of k that guarantees an optimum cost of 0?
A: Yes ! Just set $K = m$ (i.e., each point is its own cluster). Of course, this is useless from a modeling standpoint!
- ▶ **Puzzle 2:** Argue that the optimum cost for $K + 1$ clusters is no more than the optimum cost for K clusters.
- ▶ Note that Puzzle 2 only concerns the optimum cost, which as we discussed may not be attained by the Lloyd's algorithm. Nevertheless, it does suggest that we can try various values of K and see when the cost is low enough to be acceptable.

Choice of K ?

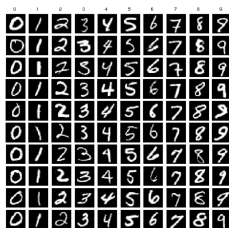
A frequent heuristic is the **elbow method**:

- ▶ create a plot of the number of clusters vs. the final value of the cost,
- ▶ look for an “elbow” where the objective tapers off.
- ▶ Note that if the dataset is too complicated for a simple Euclidean distance cost, the data might not be easy to cluster “nicely” meaning there is no “elbow” shown on the plot.

Choice of K ?

Examples **elbow method**:

- ▶ We use the MNIST hand-written digits dataset.



- ▶ Each image, which depicts some digit between 0 and 9, is represented as an 8×8 matrix of pixels and each pixel can take on a different luminosity value from 0 to 15.
- ▶ We can apply Lloyd's algorithm with increasing values of K to differentiate between images depicting
 1. two digits; the digit "1" and the digit "0."
 2. the ten digits.

Choice of K ?

Examples **elbow method**:

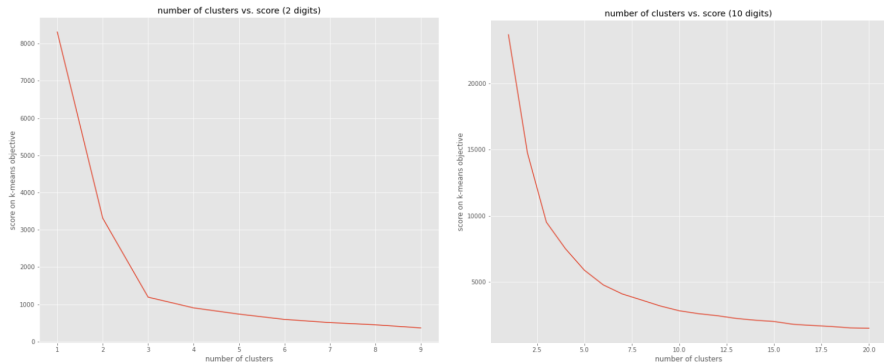


Figure: Two graphs of number of clusters vs. final value of cost. There is a distinct elbow on the left, but not on the right.

How k -Means clustering is NP-hard

Reformulation of the k -means clustering problem

- ▶ Recall from computational complexity theory that NP (nondeterministic polynomial time) is a class of decision problems, i.e., problems where the answer is "yes" or "no".
- ▶ Whether k -means clustering problem is in NP depends on how we formulate it.
- ▶ One standard formulation would be as the optimization problem (1), in the line of Definition attempt 2.
- ▶ Optimization problems like this aren't technically in NP , just because the output is not just "yes" or "no".
- ▶ However, we call them NP *optimization problems* because the natural corresponding decision problem is in NP . Here's the natural corresponding decision problem for the above formulation:

NP reformulation

Given m points and a number d , is there a partition into K clusters such that the sum of squared distances between each point and the center of its cluster is at most d ?

Why This Reformulation Works ?

let us discuss the *Relations* between problems

- ▶ The decision problem is closely related to the optimization problem:
 - \implies : if you can solve the decision problem, you can determine whether solutions exist for various values of d : by binary searching over d , you can effectively find the minimum sum of squared distances that can be achieved for the clustering.
 - \Longleftarrow : solving the optimization problem gives the minimum distance d^* , then you can easily answer the decision problem by checking if $d^* \leq d$.
- ▶ *Verification Process*: To verify the partition, you compute the center of each cluster and then calculate the sum of squared distances. This is a series of linear operations (addition and multiplication), which can be performed in **polynomial time**.
- ▶ *Certificate for Decision Problem*: The certificate (or proof) for the decision problem is a specific partition of the points into clusters. This partition can be represented using $m \log K$ bits, where m is the number of points and K is the number of clusters.

NP-hardness of clustering problems

- ▶ Several NP-hardness complexity results have been derived over the past decades for *several clustering Problems*.
- ▶ Once of the most famous is the *Euclidean K-Center Problem*:

Definition

Given m demand points in the plane, the K -center problem is to find K supply points (anywhere in the plane) so as to minimize the maximum distance from a demand point to its respective nearest supply point.

Proof in

N. Megiddo and K. J. Supowit. *On the complexity of some common geometric location problems*. SIAM J. COMPUT. Vol. 13, No. 1, February 1984. [▶ pdf](#)

- For the NP-hardness proof they establish a (polynomial-time) reduction from 3-satisfiability (famous *NP-complete* decision problem).
- They also prove the NP-hardness of famous variants: *K-center problem with ℓ_1 -norm, the K-median problem both in ℓ_2 and ℓ_1 -norms*.

- ▶ [Online demo](#)

NP-hardness of the k -means clustering problem

- ▶ NP-hardness of k -means clustering problem, AKA the Euclidean minimum sum of-squares clustering (MSSC), was proved (correctly) in 2009 in
D Aloise, A. Deshpande and P. Hansen. *NP-hardness of Euclidean sum-of-squares clustering*.
Mach Learn, 75: 245–248, 2009. [▶ pdf](#)

Theorem

MSSC in general dimension is NP-hard for $K = 2$.

- ▶ *Proof*: The reduction is from the densest cut problem, whose objective is to maximize for a given graph $G = (V, E)$ the ratio $|E(P, Q)|/|P||Q|$ over all bipartitions (P, Q) of the vertices in G , where $E(P, Q)$ denotes the edge set of the cut. The problem is equivalent to the sparsest cut problem on the complement graph, which was shown to be NP-hard in Matula and Shahrokhi (1990).

Advanced Topics

How Many Clusters?

- ▶ there are various useful heuristics in the literature. For a review and comparison, see Allan D. Gordon. *Null models in cluster validation*. In *From data to knowledge*, pages 32–44. Springer, 1996.
- ▶ More recent work includes:
Greg Hamerly and Charles Elkan. *Learning the k in k -means*. Advances in neural information processing systems, 16:281, 2004.
- ▶ One principled approach is to construct a **null hypothesis** for the clustering.
For example, the idea of the *gap statistic*:
R Tibshirani, G. Walther, and T. Hastie. *Estimating the number of clusters in a data set via the gap statistic*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2):411–423, 2001. [▶ pdf](#)

How Many Clusters?

Gap statistic:

- ▶ to compare a dispersion statistic of the clustered fit to the same statistic fit to synthetic data that are known not to have clusters.
- ▶ When we have the right number of clusters, we would expect to have less dispersion than random.

How Many Clusters?

Gap statistic:

- ▶ Let's imagine that we have run Lloyd's algorithm and we now have a set of responsibilities $\{r^{(i)}\}_{i=1}^m$ and cluster centers $\{\mu_k\}_{k=1}^K$.
- ▶ We define the **within-cluster dispersion** to be the sum of squared distances between all pairs in a given cluster:

$$D_k = \sum_i^m \sum_j^m r^{(i)}[k] r^{(j)}[k] \|x^{(i)} - x^{(j)}\|_2^2$$

- ▶ The **dispersion** for a size- K clustering is the normalized sum of the within-cluster dispersion over all of the clusters:

$$W_K = \sum_{k=1}^K \frac{D_k}{N_k}$$

This is basically a measure of “tightness” of the fit normalized by the size of each cluster. It will be smaller when the clustered points group together closely.

How Many Clusters?

Gap statistic:

- ▶ The gap statistic uses a **null distribution** for the data we have clustered, from which we generate *reference data*.
- ▶ You can think of this as a distribution on the same space, with similar coarse statistics, but in which **there aren't** clusters.
- ▶ To compute the gap statistic, we now generate several sets of reference data, cluster each of them, and compute their dispersions.
- ▶ This gives us an idea (with error bars) as to what the expected dispersion would be based on the coarse properties of the data space, for a given K .

How Many Clusters?

Gap statistic:

- ▶ We compute the gap statistic as follows:

$$\text{Gap}_m(K) = \mathbb{E}_{\text{null}}(\log W_K) - W_K$$

here:

- the first term is the expected log of the dispersion under the null distribution – something we can compute by averaging the log dispersions of our reference data.
- We subtract from this the dispersion of our actual data.
- ▶ We can then look for the K which maximizes $\text{Gap}_m(K)$. We choose the smallest one that appears to be statistically significant.

How Many Clusters?

Gap statistic:

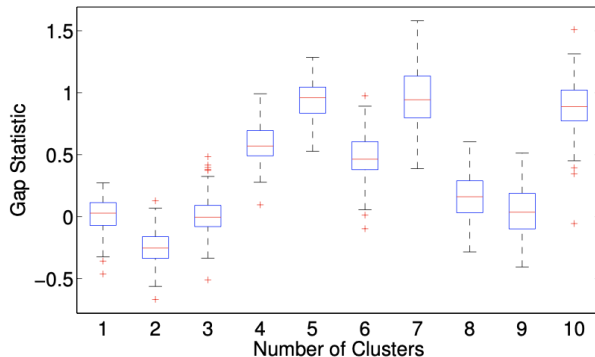


Figure: The gap statistic computed for the fruit data. Here, it seems reasonable to choose $K = 5$. This used 100 reference data sets, each from a Gaussian Maximum Likelihood Estimation fit for the null model.

Hierarchical Clustering: Dendrogram

- ▶ Another commonly used unsupervised algorithm for clustering data is a *dendrogram*.
- ▶ Like k -means clustering, dendrograms are created from a simple hierarchical algorithm, allowing one to efficiently visualize if data is clustered without any labeling or supervision.
- ▶ Hierarchical clustering methods are generated either from a top-down or a bottom-up approach. Specifically, they are one of two types:
 1. **Agglomerative:** Each data point $x^{(i)}$ is its own cluster initially. The data is merged in pairs as one creates a hierarchy of clusters. The merging of data eventually stops once all the data has been merged into a single *ubercluster*. This is the bottom-up approach in hierarchical clustering.
 2. **Divisive:** all the observations $x^{(i)}$ are initially part of a single giant cluster. The data is then recursively split into smaller and smaller clusters. The splitting continues until the algorithm stops according to a user specified objective. The divisive method can split the data until each data point is its own node.

Hierarchical Clustering: Dendrogram

- ▶ the merging and splitting of data is accomplished with a *heuristic, greedy* algorithm which is easy to execute computationally.
- ▶ The results of hierarchical clustering are usually presented in a dendrogram.
- ▶ For the following: we will focus on **agglomerative** hierarchical clustering.
- ▶ Like the Lloyd algorithm for k -means clustering, building the dendrogram proceeds from a simple algorithmic structure based on computing the distance between data points. Example of distances:

Squared Euclidean distance $\|x^{(i)} - x^{(j)}\|_2^2$

ℓ_1 - norm $\|x^{(i)} - x^{(j)}\|_1$

ℓ_∞ - norm $\|x^{(i)} - x^{(j)}\|_\infty$

Mahalanobis distance $\sqrt{(x^{(i)} - x^{(j)})^T C^{-1} (x^{(i)} - x^{(j)})}$

where C is the sample covariance matrix.

The choice of norm can make a tremendous difference for exposing patterns in the data that can be exploited for clustering and classification.

Hierarchical Clustering: Dendrogram

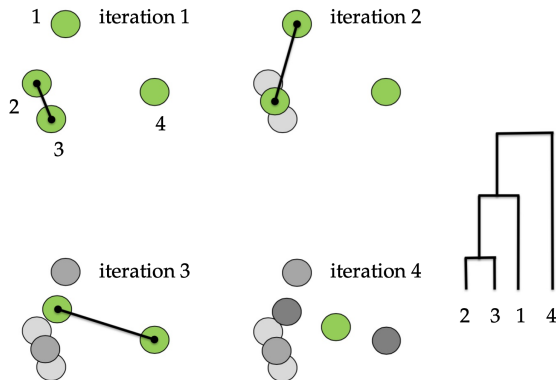


Figure: Illustration of the agglomerative hierarchical clustering scheme applied to four data points.

Hierarchical Clustering: Dendrogram

- ▶ In the algorithm, the distance between the four data points is computed.
- ▶ Initially the Euclidian distance between points 2 and 3 is closest.
- ▶ Points 2 and 3 are now merged into a point mid-way between them and the distances are once again computed.
- ▶ The dendrogram on the right shows how the process generates a summary (dendrogram) of the hierarchical clustering.
- ▶ Note that the length of the branches of the dendrogram tree are directly related to the distance between the merged points.

2-D example: ▶ [Section 6](#).

Summary

Summary

We have seen :

- ▶ Few words about: **clustering**, **classification** and **regression**.
- ▶ Feature selection and data mining: construct and exploit the intrinsic low-rank feature space of a given data set that is *informative* and *interpretable*.
- ▶ **Definition** of
 - a *good cluster*: a subset of points which are closer to the mean of their own cluster than to the mean of other clusters
 - of the *mean* and the *variance* of clusters
- ▶ Presentation of the *k-Means clustering* problems: finding optimal solution of a specific optimization problem.
- ▶ The **Lloyd's** algorithm: an *heuristic* algorithm with *finite-time* convergence guarantee to a *local* minimum.
- ▶ Practical considerations: choose the initial clusters, and $K +$ hierarchical clustering.
- ▶ *k*-Means clustering problem is *NP-hard*.

Preparation for the lab and outlooks

- ▶ Review the lecture :).
- ▶ Practice a bit the coding of Lloyd's algorithm and kmeans++ for simple 2-D clustering.

Next topics are outside the scope of this class, but are good things to look into next if you find clustering to be an interesting topic.

1. **Mixture of Gaussians** and the expectation-maximization algorithm, and the links with the k -means/Lloyd's algorithm: [▶ Stanford Lecture notes](#).
2. **Spectral Clustering**: Clusters in data are often more complicated than simple isotropic blobs. Spectral clustering constructs a graph over the data first and then performs operations on that graph using the Laplacian. The idea is that data which are close together tend to be in the same group, even if they are not close to some single prototype. [▶ Stanford Lecture notes](#).
3. **Affinity Propagation**: B. J Frey and D. Dueck. *Clustering by passing messages between data points*. science, 315(5814):972–976, 2007.
4. **Biclustering**: J. A Hartigan. *Direct clustering of a data matrix*. Journal of the american statistical association, 67(337):123–129, 1972.
Algorithm for simultaneously grouping both rows and columns of a matrix, in effect discovering blocks. Variants of this technique have become immensely important to biological data analysis.

Appendices

Kmeans++ pseudo code

input: Point set $\{x^{(i)}\}_{i=1}^m$, number of clusters K
 $l \leftarrow \text{RandomInteger}(1, m)$
 $\mu_1 \leftarrow x^{(l)}$
for $k \leftarrow 2$ **to** K **do**
 for $i \leftarrow 1$ **to** m **do**
 $d_i \leftarrow \min_{k' < k} \|x^{(i)} - \mu_{k'}\|_2^2$
 end
 for $i \leftarrow 1$ **to** m **do**
 $p_i \leftarrow \frac{d_i^2}{\sum_j d_j^2}$
 end
 $l \leftarrow \text{Discrete}(p_1, p_2, \dots, p_m)$
 $\mu_k \leftarrow x^{(l)}$
end
output: cluster means $\{\mu_k\}_{k=1}^K$

Choose a datum at random.

Make this random datum the first cluster center.

Compute the distance to the closest center.

Compute a distribution proportional to d_i^2 .

Draw a datum from this distribution.

Make this datum the next center.

Standardizing Data

- ▶ It is necessary to think carefully about what distances mean in the context of K -Means clustering problems.
- ▶ **For example:** imagine that we wanted to cluster cars using two quantities: wheelbase (a length) and performance (horsepower):
 1. Consumer cars range in horsepower from about 150hp to perhaps 400hp for sports cars; exotics such as the Bugatti Veyron might go up to 1000hp.
 2. Wheelbase (the distance between the front and rear centers of the wheels) we might reasonably measure in meters, typically between 2.5m and 3.5m for passenger cars.
- ▶ If we use these numbers as our inputs, we might get into trouble because the horsepower will dominate the distance, as it is two orders of magnitude larger in scale¹⁰.

¹⁰This is very unsatisfying because these are just arbitrary units; we could've just as easily used millimeters for wheelbase and then it would dominate over horsepower!

Standardizing Data

- ▶ A typical way to deal with this is to *standardize* each of the dimensions by finding the transformation so that each dimension has mean zero and standard deviation one.
- ▶ This causes each of the features to be centered at zero and have approximately the same scale (as determined by the data).
- ▶ Algorithm in the next slide shows such a standardization.

Standardizing Data

input: Point set $\{x^{(i)}\}_{i=1}^m$

for $j \leftarrow 1$ **to** n **do**

$s_1 \leftarrow 0$

$s_2 \leftarrow 0$

for $i \leftarrow 1$ **to** m **do**

$s_1 \leftarrow s_1 + x^{(i)}[j]$

$s_2 \leftarrow s_2 + (x^{(i)}[j])^2$

end

$\mu \leftarrow \frac{s_1}{m}$

$\sigma^2 \leftarrow \frac{s_2}{m} - \mu^2$

for $i \leftarrow 1$ **to** m **do**

$x^{(i),'}[j] \leftarrow \frac{(x^{(i)}[j] - \mu)}{\sigma}$

end

end

output: transformed point set $\{x^{(i),'}\}_{i=1}^m$

For storing the total of the values.

For storing the total of the squared values.

Compute sample mean.

Compute sample variance.

Shift by mean, scale by standard deviation.

Missing Data

- ▶ In practical data analysis problems, we often have missing data.
- ▶ That is, some dimensions of some of the data may be completely missing.
- ▶ How do we deal with this in K -Means?
- ▶ There are a variety of possibilities, but the two main options are *imputation* and *marginalization*.

Missing Data

Imputation:

- ▶ uses a procedure for inventing the missing data.
- ▶ **For example:**
 - we might replace the missing data with the mean of the remaining data (if we standardized, this will be zero), or
 - we might regress the missing values on the other features.
- ▶ Imputation can be useful, but it can also be dangerous if it ends up influencing the conclusions that you draw from the data.

Missing Data

Marginalization:

- ▶ is a more principled approach to dealing with missing data.
- ▶ We try to account for all possible values of the unknown dimension,
- ▶ that is, we could compute the expectation of the distance between two points, integrating over the missing values.
- ▶ If we've standardized the data (and there aren't too many missing values) then we might reasonably assume that the missing data have a $\mathcal{N}(0, 1)$ distribution.
- ▶ The expectation of the squared Euclidean distance, assuming that the d th dimension is missing, is


$$\begin{aligned}\mathbb{E}[\|x - \mu\|_2^2] &= \int_{-\infty}^{\infty} \sum_j^n (x_j - \mu_j)^2 \mathcal{N}(x_d|0, 1) dx_d \\ &= \left[\sum_{j \neq d}^n (x_j - \mu_j)^2 \right] + \int_{-\infty}^{\infty} (x_d - \mu_d)^2 \mathcal{N}(x_d|0, 1) dx_d \\ &= \left[\sum_{j \neq d}^n (x_j - \mu_j)^2 \right] + 1 + \mu_d^2\end{aligned}$$

Basically, this simple marginalization results in us adding a bit to the distance.

Vectorizing Code

- ▶ As with almost all the numeric code you write to do data analysis, if you're using a high-level language like Matlab or Python, you'll want to vectorize things as much as possible.
- ▶ That means that rather than writing the simple loops such as I show in these algorithm boxes, you'll want to frame things in terms of operations on vectors and matrices.
- ▶ For example, instead of writing the loops for Data Standardization, you might write a Python function that takes advantage of vectorized built-ins

```
1 def standardize(data):  
2 #Take an mxn numpy matrix as input and return a standardized version.  
3     mean = np.mean(data, axis=0)  
4     std = np.std(data, axis=0)  
5     return (data - mean)/std  
6
```

There are no loops in sight, which means that they are (hopefully) being performed in the underlying *Fortran* or *C* libraries. These computations can be done very quickly using , for example.

Vectorizing Code

- ▶ Another very useful trick to know is for computing distances.
- ▶ When the data are one- dimensional, it seems pretty easy. However, with higher dimensional data, it's less obvious how to compute a matrix of distances without looping.
- ▶ Let's imagine that we have an $K \times n$ matrix X and an $L \times n$ matrix Y and that we want to compute an $K \times L$ matrix D with all of the squared distances.
- ▶ One entry in this matrix is given by: $D[k, l] = \langle x_k - y_l, x_k - y_l \rangle = x_k^T x_k - 2\langle x_k, y_l \rangle + y_l^T y_l$ where $x_k^T, y_l^T \in \mathbb{R}^n$ respectively denote the k and l -th row vector of X and Y .
- ▶ this is a sum of inner products. Using broadcasting, which *NumPy* does naturally and can be done in *Matlab* using *bsxfun*, this gives us a rapid way to offload distance calculations to *BLAS* (or equivalent) without looping in our high-level code.
Example [▶ here](#) of built-in functions in *SciPy*.

Goodbye, So Soon

THANKS FOR THE ATTENTION

- ▶ v.leplat@innopolis.ru
- ▶ sites.google.com/view/valentinleplat/