Data Mining - Appunti

Francesco Lorenzoni

Febrero 2025

Contents

Part I

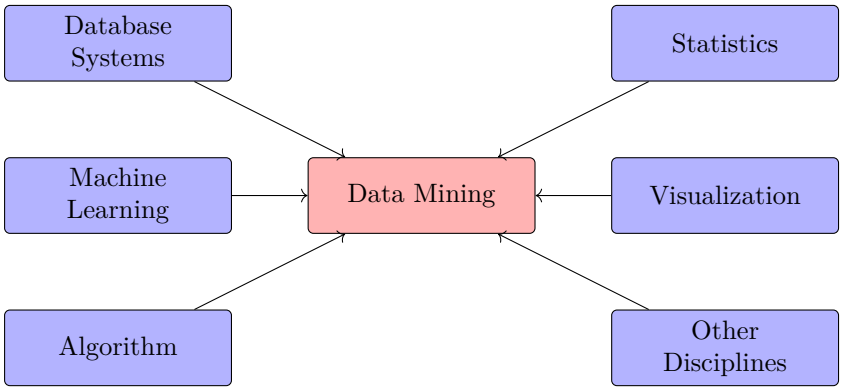
Introduction to Data Mining

Chapter 1

Introduction

Definition 1.1 (Data Mining) *is the use of efficient techniques for the analysis of very large collections of data and the extraction of useful and possibly unexpected patterns in data (hidden knowledge). The goal is to extract (human-readable) knowledge and insight from raw data.*

- ◇ Knowledge implies we are often not just trying to solve a task
- ◇ Insight implies that we should infer non-obvious knowledge
- ◇ Human-readable implies that knowledge should be (when possible) understood by humans: focus on interpretability!
- ◇ Raw data implies we'll need to clean it



1.1 Definitions

1.1.1 Knowledge Discovery Loop

Large collections tend to be heterogeneous in source, domain, language and refinement. The first step is to store the data, which however does not assess its heterogeneity. Data cleaning and integration tackle this problem, so that we get integrated sources, homogenous language, and data cleared of noise and outliers.

To look for insight on the data we have to answer questions on the data as a stakeholder. We may see patterns and ask ourselves their nature. Pattern extraction and validation lead to possible insight. Insight may lead to noticing that some data missing may be useful, and we may want to collect it, going back at previous steps.

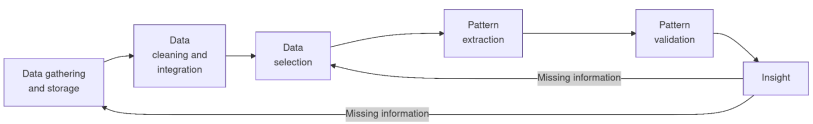


Figure 1.1: Knowledge Discovery Loop
This essentially summarizes the KDD process.

1.1.2 KDD Process

The KDD process consists of the following steps:

1. **Data Cleaning:** Remove noise and inconsistent data.

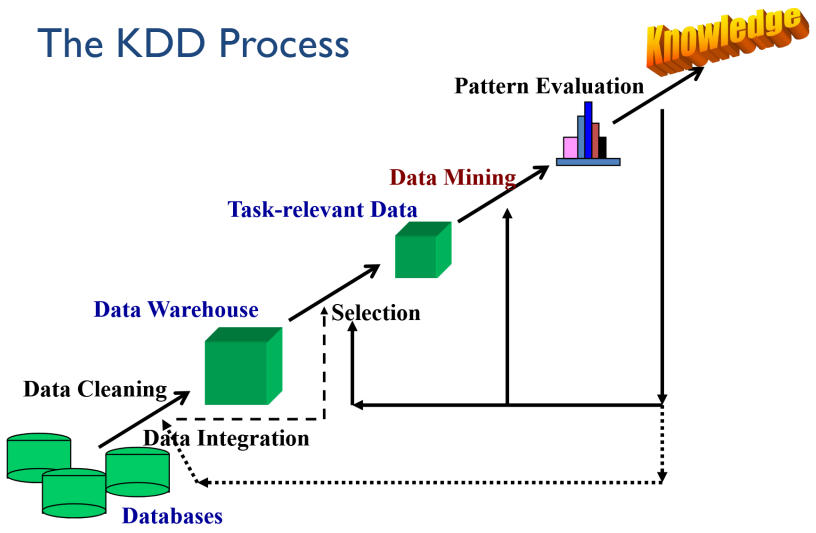
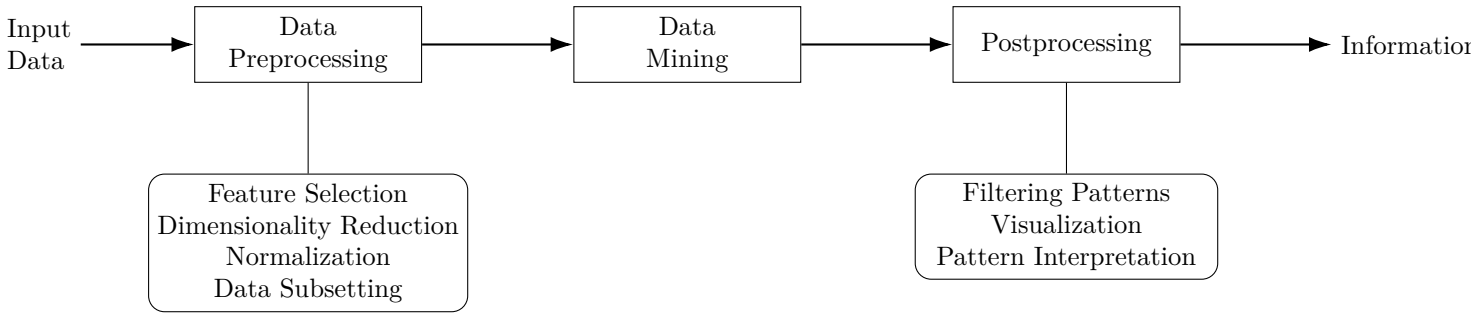


Figure 1.2: KDD Process

2. **Data Integration:** Combine multiple data sources.
   Involves the process of data understanding, data cleaning, merging data coming from multiple sources and transfoming them to load them into a **Data Warehouse**.
   Data Warehouse is a database targeted to answer specific business questions
3. **Data Selection:** Select relevant data for analysis.
4. **Data Transformation:** Transform data into suitable formats for mining (summary, aggregation, etc.).
5. **Data Mining:** Apply algorithms to extract patterns.
   - ◇ *Prediction Methods*
     Use some variables to predict unknown or future values of other variables.
   - ◇ *Description Methods*
     Find human-interpretable patterns that describe the data.
6. **Pattern Evaluation:** Identify truly interesting patterns.
7. **Knowledge Presentation:** Present the mined knowledge in an understandable way.

1.1.3 Data Mining Process



Definition 1.2 (Primary Data) *Original data that has been collected for a specific purpose.*
Primary data is not altered by humans

Definition 1.3 (Secondary Data) *Data that has been already collected and made available for other purposes.*
Secondary data may be obtained from many sources

Definition 1.4 (Association rule discovery) *Given a set of records each of which contain some number of items from a given collection.*
Produce dependency rules which will predict occurrence of an item based on occurrences of other items.

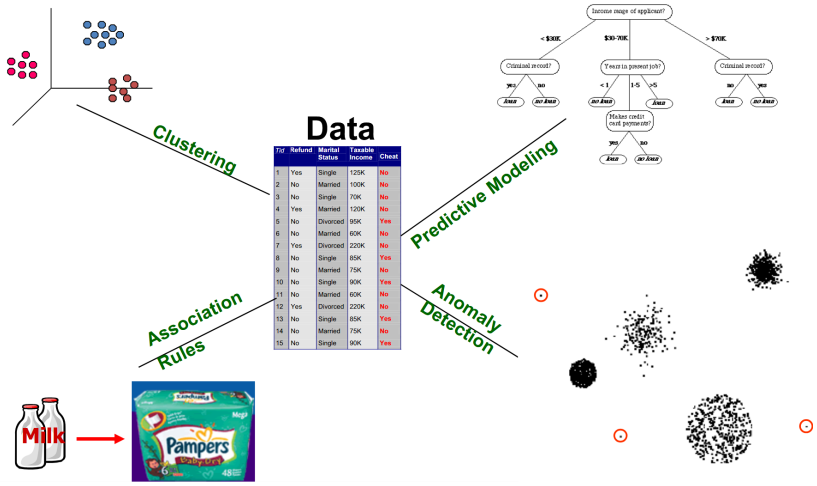


Figure 1.3: Data Mining methods

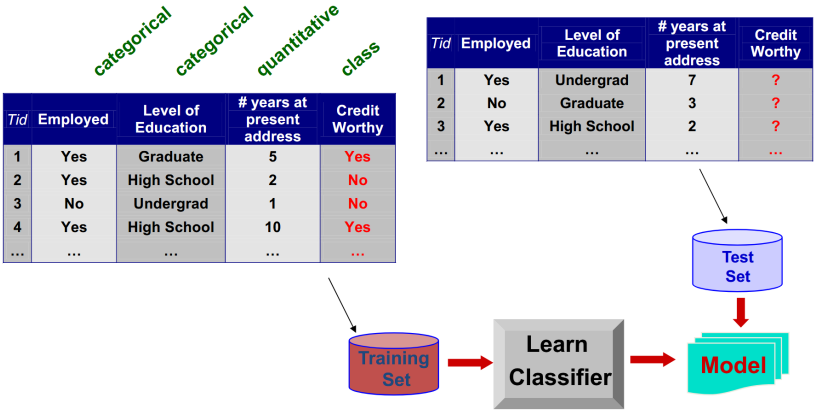


Figure 1.4: Classification Process

Association Use Cases

- **Market-basket analysis**
  Rules are used for sales promotion, shelf management, and inventory management
- **Telecommunication alarm diagnosis**
  Rules are used to find combination of alarms that occur together frequently in the same time period
- **Medical Informatics**
  Rules are used to find combination of patient symptoms and test results associated with certain diseases

1.2 Data Understanding

Definition 1.5 (Data) *Data is a collection of data objects and their attributes.*
An attribute is a property or characteristic of an object. A collection of attributes describe an object (record).

If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute.
Such data set can be represented by an $m \times n$ matrix, where there are m rows, one for each object, and n columns, one for each attribute.

Data Types:
- Document data
- Transaction data
- Graph data
- Ordered data
  - Spatial data
  - Temporal data

The type of the attribute depends on the following properties:
- Distinctness: $=\neq$
- Order: $<>$
- Differences are meaningful: $+-$
- Ratios are meaningful: $*/$

Attribute types:
- Nominal/Categorical: attribute values in a finite domain (*distinctness*)
- Binary: special case of nominal with two values
- Ordinal: attribute values have a total ordering (*distinctness* and *order*)
- Numeric: quantity (integer or real-valued) (*distinctness*, *order*, *differences*)
- Ratio-Scaled: we can speak of values as being an order of magnitude larger than the unit of measurement (*all 4 properties*)
  length, counts, elapsed time (A baseball game lasting 3 hours is 50% longer than a game lasting 2 hours)
- Discrete/Continuous: attribute values are discrete (finite or countably infinite) or continuous (real-valued).

Attribute Type	Description	Examples	Operations
Nominal	Nominal attribute values only distinguish. $(=, \neq)$	zip codes, employee ID numbers, eye color, sex: {male, female}	mode, entropy, contingency correlation, χ^2 test
Ordinal	Ordinal attribute values also order objects. $(<, >)$	hardness of minerals, {good, better, best}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, differences between values are meaningful. $(+, -)$	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
Ratio	For ratio variables, both differences and ratios are meaningful. $(*, /)$	temperature in Kelvin, monetary quantities, counts, age, mass, length, current	geometric mean, harmonic mean, percent variation

Table 1.1: Attribute types, examples and operations

1.2.1 Data Quality

Examples of data quality problems:

- ◇ Wrong data
- ◇ Duplicate data
- ◇ Noise and outliers
- ◇ Missing values

In order to know our data and discovery quality issues we need use descriptive statistics for getting a global picture and summarize properties of data and compare such statistics with the expected behaviour. All around we can exploit visualization techniques that can help in detecting general or unusual patterns and trends, as well as outliers.

1.2.1.1 Histograms

A histogram shows the frequency distribution for a numerical attribute. The range of the numerical attribute is discretized into a fixed number of intervals (**bins**).

The number of bins according to Sturges' rule is:

$$k = \lceil \log_2 n + 1 \rceil$$

where n is the number of records in the data set. Sturges' rule is suitable for data from normal distributions and from data sets of moderate size.

1.2.1.2 Statistics notions

Notoriouses Mean/Median/Mode…The degree in which data tend to spread is called the *dispersion*, or **variance** of the data.

The most common measures for data dispersion are **range** (The distance between the largest and the smallest values), **standard deviation**, the **five-number summary** (based on *quartiles*), and the **inter-quartile range**.

$$\text{variance}(x) = \sigma^2 = s_x^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2$$

Standard deviation σ is the square root of variance σ^2.

Because of outliers, other measures are often used:

- ◇ absolute average deviation (AAD)

$$\text{AAD}(x) = \frac{1}{m}\sum_{i=1}^{m}|x_i - \bar{x}|$$

- ◇ median average deviation (MAD)

$$\text{MAD}(x) = \text{median}\left(\{|x_1 - \bar{x}|, \ldots, |x_m - \bar{x}|\}\right)$$

1.2.1.3 Box-Plot

1.3 Data Understanding - Lab

Data comes from diverse sources, and generally is not tailor-made for some downstream task. We need to start from basics:

- ◇ What features are available?
- ◇ What are they measuring, exactly?
- ◇ What properties do they have?
- ◇ What are their relations?
- ◇ Are there outliers?
- ◇ …

Data can be of different nature which may co-occur:

- ◇ **Temporal**: the data describes events over time
- ◇ **Sequential**: the data spans some ordering
- ◇ **Relational**: the data describes event in between instances
- ◇ **Spatial**: the data describes space
- ◇ **Independent**: instances in data are independent observations

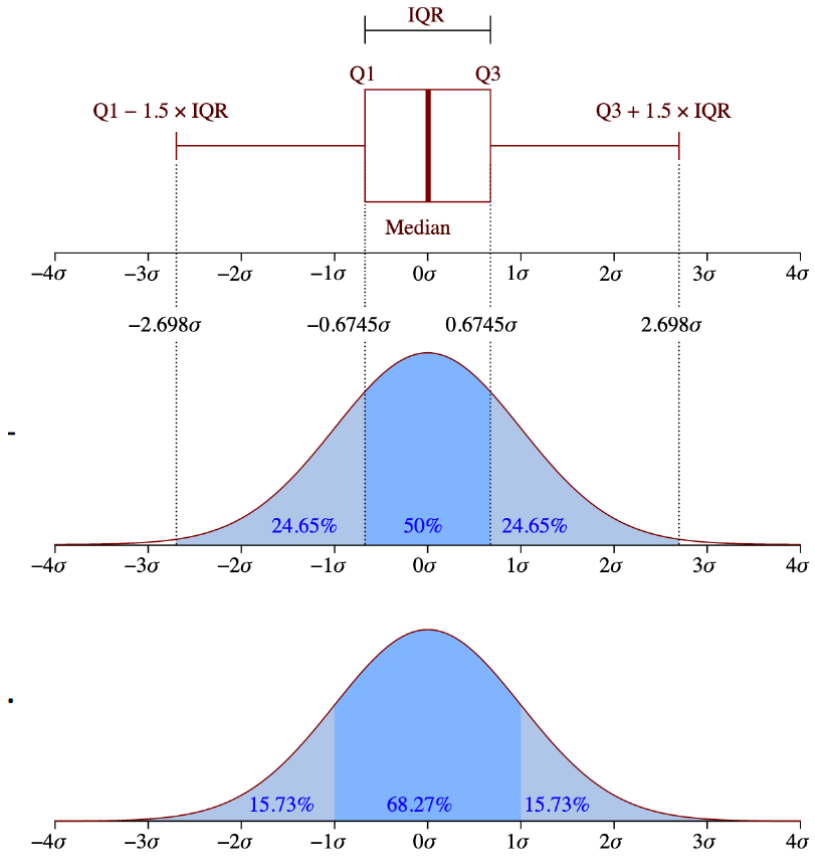


Figure 1.5: Box-Plot

1.3.1 Data Collections

We refer to single instances in the collections as objects/records/instances, which are described by attributes.

Id	Age	Income	Marital	Loan
0	30	2.5k	Married	Yes
1	24	1.4k	Single	No
...	...	...	...	...

Table 1.2: Grant Data

- ◇ Attributes: `Id , Age , Income , Marital,Loan grant`
- ◇ Records: `0, 30, 2.5k, Married, Yes , 1, 24,1.4k, Single, No`

1.3.2 Data Types

1.3.2.1 Tabular

When records are independent, and described by the same finite set of features, they are often represented in a tabular form: the data matrix. Each row is a record, each dimension is an attribute.

Records on the rows, attributes on the columns.

Id	Age	Bike used	Length	Duration	Date	Cyclist
0	28	Colnago VRS4	152.4	3:43:12	15-5-2025	Alessandro Covi
1	40	Cervelo RS5	72.4	2:55:01	4-3-2024	Gianni Affino

Table 1.3: Cyclist Data

1.3.2.2 Transaction

A feature contains a (multi)set of items.

PurchaseId	Cart	Bought on
0	Bread, Milk	17:12-15-5-2025
1	Notebook, Pens, Bread, Basil	8:04-4-3-2024

Table 1.4: Transaction Data

Records on the rows, attributes on the columns.

1.3.2.3 Graph

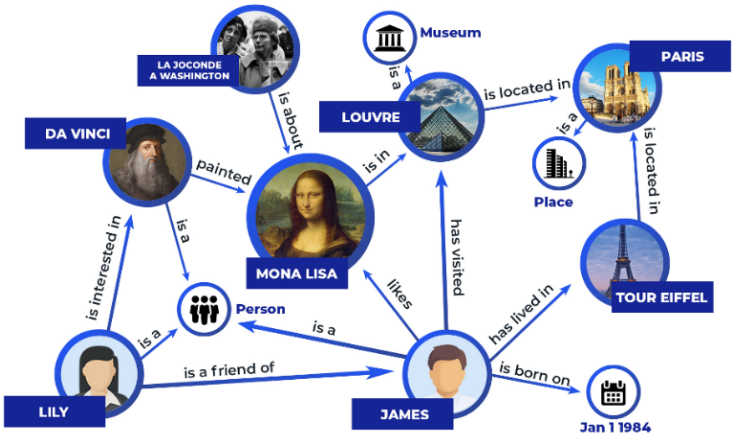


Figure 1.6: Graph Data

Data is linked, either on records or features. Records are nodes in a graph, attributes can vary wildly across records.

1.3.2.4 Sequential

Records are sequences (of variable length): attributes are indexed (order or time).

Image on the right lacks two images ☺

Figure 1.7: Sequential Data

1.3.2.5 Spatial

Records are associated with locations in space: attributes can include coordinates, regions, etc.

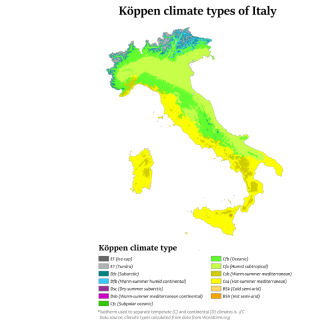


Figure 1.8: Spatial Data

1.3.2.6 Attribute types

Type	Description	Example
Numerical	Values have a total ordering, and represent some numerical quantity	Age, dates
Ordinal	Values have a total ordering, and represent some quantity	Dress size, Cup size
Binary	Values are one of two categories: no ordering	Boolean values
Categorical	Values of one of multiple categories: no ordering	Country, Job

Table 1.5: Types of Data

1.3.2.7 Values types

Values can be either:

- ◇ **Discrete** Defined in a finite or countably finite domain, e.g., country, job, cup size. Note: ordinal values may be discrete too!
- ◇ **Continuous** Defined in a continuous and infinite domain, e.g., distance.

1.3.3 Data Syntax and Semantics

Given the categorization of the records and attributes of your data, we can study its general behavior. We leverage some basic statistical tools, first of all by drawing the empirical distribution of the attributes.

Figure 1.9: Data Syntax and Semantics

Useful statistics for data semantics

◇ **Expected value**

$$\mathbb{E}[X] = \sum_{x \in \text{dom}(X)} \Pr(X = x)\, x$$

A statistic representative of the value of an attribute, weighing values and their probability

◇ **Variance**

$$\sigma^2(X) = \mathbb{E}\left[\sum_{x \in \text{dom}(X)} \big(x - \mathbb{E}[X]\big)^2\right]$$

Distance from the expected value of all records: the data spread

◇ **Quantiles**

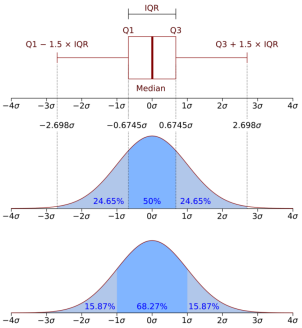
$$q^p = x \;\; s.t. \;\; \Pr(X \leq x) = q^p$$

Inflection points defining values for a threshold, e.g., if the 99-th percentile is , then we

◇ **Interquantile range**

$$q^{75} - q^{25}$$

Distance between quantiles: how spread are inflection points?



A Normal distribution, its q^{25} and q^{75} quantiles, and interquartile range between them. On top, a box plot visualization of the distribution.

Figure 1.7: Statistics Graph

Statistical summary of the distribution are typically accompanied by visual and semantic one.
Erroneous or weird values to be cleaned later may already pop up in these basic steps. Outlier values typically skew statistics. Variance is often replaced by absolute/median average deviation

1.4 Data Cleaning

There are some concepts to be aware of when dealing with data quality, hence data cleaning.

Data accuracy is the degree to which data correctly describes the "real world" object or event being described.

◇ Syntactic: values outside domain, e.g., Eataly in Country
◇ Semantic: values in domain, but semantically wrong, e.g., age is 3, and weight is 82kg

Completeness is the degree to which all required data is known.
Some attributes are not collected, or are collected partially, e.g., temperature was not recorded by the sensor.

Biased gathering is the degree to which data may be over/under-representative, e.g., the bank may only provide data about successful loan applicants.

Timeliness is the degree to which data is up to date.

Remember: *garbage in, garbage out!*[1] In a task-agnostic view, we are interested in addressing the above by tackling:

◇ **Duplicates**: skews the data distribution

[1] i.e. if you have garbage data, you'll get garbage results

- ◇ **Missing values**: give false/partial information
- ◇ **Noise**: uninformative of the data
- ◇ **Poor accuracy**: gives wrong data
- ◇ **Outliers**: skews the data distribution and models of the data

1.4.1 Handling Duplicates

Remove them... when appropriate! Not all duplicates are garbage, it depends on what insight you can gather from it.

Case A
You have data on registration to your website, with several duplicate e-mails. Insights:

- ◇ The "Sign in" button is hard to find
- ◇ The "Sign in" button is less visible than the "Sign up" button
- ◇ Your site is so anonymous people forget they signed up already

Case B
You have data on credit account opening from Poste (Italian postal service) with several duplicate e-mails. Insights:

- ◇ The client hacked the database and added themselves to ask more credit (unlikely)
- ◇ Poste's tech staff is underwhelming (very likely)

1.4.1.1 Duplicate Features

Duplicate features may be more tricky. Features convey similar, although not equal, information to others.
Examples:

- ◇ Resting heart rate and heart rate under continuous high effort
- ◇ Education level and reading skills
- ◇ Rent and available bank deposit

These pairs of features are not per se one duplicate of the other, but are strongly related: when one grows, so does the other, and when one goes down, so does the other.

Linear (and rank) relationships between two features X, Y can be quantified with their correlation. Correlation ranges in $[-1, 1]$, from perfectly negative to perfectly positive correlation.
Given two lists of values x^{i^n}, y^{i^n} we can compute two main correlation types.

- ◇ **Pearson correlation**
$$\rho_P^{X,Y} = \frac{\mathbb{E}\left[(x^i - \mathbb{E}[X])(y^i - \mathbb{E}[Y])\right]}{\sigma_X \sigma_Y}$$
Measures linear correlation between two numerical features and their values.
- ◇ **Spearman correlation**
$$\rho_S = \rho_P^{\text{rank}(X),\text{rank}(Y)}$$
Measures monotonic correlation between two ordinal or numerical features.

1.4.2 Handling Missing Values

Data may be missing for any number of reasons (at random or not at random).

- ◇ A record has a large and/or significant set of missing attributes
- ◇ An attribute has a large percentage of missing values

We have two choices: **dropping** or **imputing**.

Dropping
If a record has a large and/or significant set of missing attributes, or an attribute has a large percentage of missing values, we can drop the record/attribute.

- ◇ High percentage of missing values
- ◇ Missing values in critical attributes, e.g., a patient in cardiology has no heart rate data

Imputing
Imputing means replacing the missing value with a "best guess" value.
If a record has a small set of missing attributes, or an attribute has a small percentage of missing values, we can impute the missing values. We have to create a model to predict the missing value.

- ◇ Low percentage of missing values
- ◇ Reasonably good understanding of the attribute semantics/distribution

◇ Presence of related attributes

1.4.3 Outliers

Quantiles and distributions inform us on what values may be outlier. They are typically dropped, and unlike missing values, almost never imputed. We'll tackle algorithms later in the course.

1.4.3.1 Flower Example

There is a dataset with 5 attributes: sepal length, sepal width, petal length, petal width, and species (type).

Sepal L.	Sepal W.	Petal L.	Petal W.	Type
5.1	3.5	1.4	0.2	Setosa
7.0	3.2	4.7	1.4	Versicolor
...	...	...	...	...

Table 1.6: Flower Dataset Example

Figure 1.8: Flower Data plotted

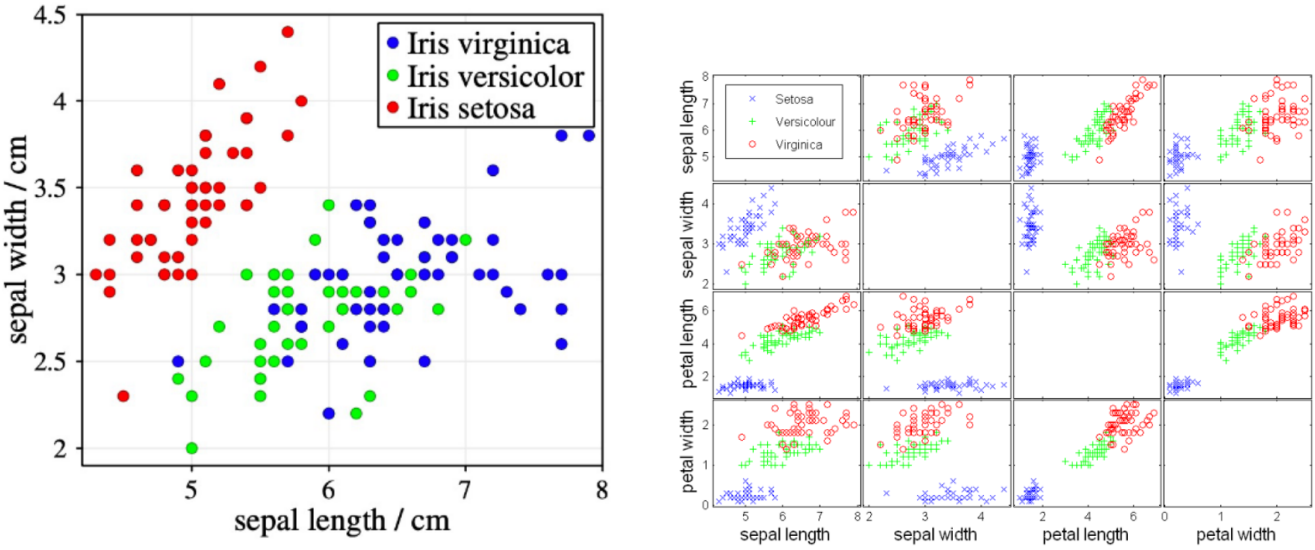


Figure 1.9: Scatter plot of sepal length and width, and scatter matrix: scatter plots of all pairs of attributes in the Iris dataset.

Plot bivariate (or trivariate) data, eyeing data correlation and outliers.

1.5 Data Preparation

We will delve into the following techniques of data preparation:

◇ Aggregation
◇ Data Reduction: Sampling
◇ Dimensionality Reduction

- ◇ Feature subset selection
- ◇ Feature creation
- ◇ Discretization and Binarization
- ◇ Attribute Transformation

1.5.1 Aggregation

Aggregation is the process of combining two or more attributes (or objects) into a single attribute (or object).

Purpose

- ◇ Data reduction
  - – Reduce the number of attributes or objects
- ◇ Change of scale
  - – Cities aggregated into regions, states, countries, etc.
  - – Days aggregated into weeks, months, or years
- ◇ More "stable" data
  - – Aggregated data tends to have less variability

1.5.2 Reduction

Reduction is simply reducing the amount of data We may reduce the number of **records** by sampling or clustering, or the number of **attributes** (*columns*) by selecting a subset of them, or by creating a new —smaller— set of attributes from the old one.

1.5.2.1 Sampling

Sampling is the main technique employed for data reduction.
It is often used for both the preliminary investigation of the data and the final data analysis.

Sampling is typically used in data mining because processing the entire set of data of interest is too expensive or time consuming.

The key principle for effective sampling is the following:

- ◇ Using a sample will work almost as well as using the entire data set, if the sample is representative
- ◇ A sample is representative if it has approximately the same properties (of interest) as the original set of data

- ◇ **Simple Random Sampling**
  - – There is an *equal probability* of selecting any particular item
  - – Sampling **without replacement**
    - ○ As each item is selected, it is removed from the population
  - – Sampling **with replacement**
    - ○ Objects are not removed from the population as they are selected for the sample.
    - ○ In sampling with replacement, the same object can be picked up more than once
  - – **Stratified sampling**
  - – Split the data into several partitions; then draw random samples from each partition
  - – Approximation of the percentage of each class
  - – Suitable for distribution with peaks: each peak is a **layer**

1.5.2.2 Dimensionality Reduction

This consists in reducing the number of attributes (or features) in the data. We want a selection of a subset of attributes that is as small as possible and sufficient for the data analysis.

- ◇ removing (more or less) irrelevant features
  - – Contain no information that is useful for the data mining task at hand
  - – Example: students' ID is often irrelevant to the task of predicting students' GPA
- ◇ removing redundant features
  - – Duplicate much or all of the information contained in one or more other attributes
  - – Example: purchase price of a product and the amount of sales tax paid

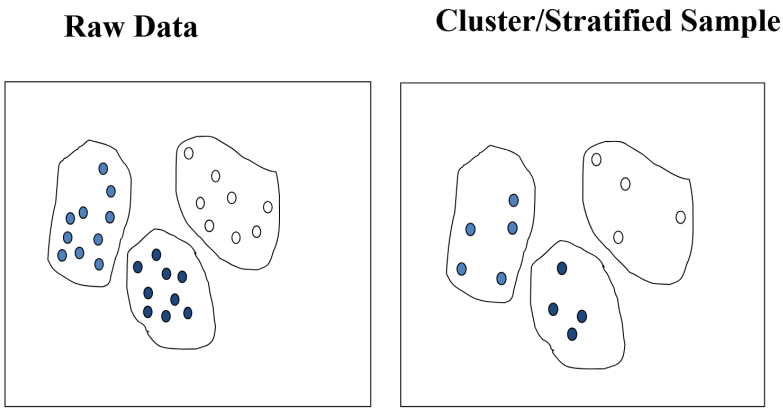


Figure 1.10: Stratified Sampling

Curse of Dimensionality

When dimensionality increases, data becomes **increasingly sparse** in the space that it occupies.

Definitions of density and distance between points, which are critical for clustering and outlier detection, become less meaningful.

This phenomenon is known as the **curse of dimensionality**.

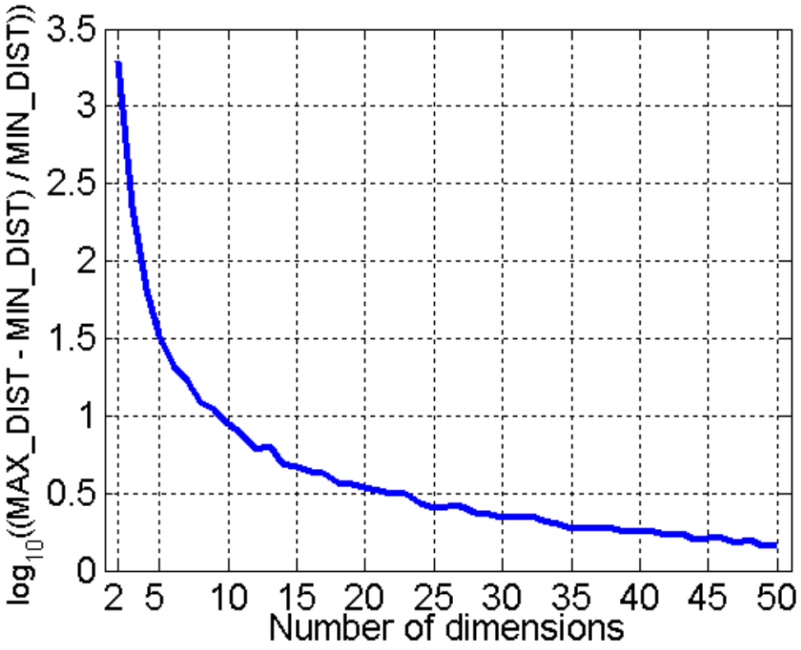


Figure 1.11: $log_{(MAXDIST-MINDIST)/MINDIST}$ decreases as the dimensionality increases, meaning that the difference between the farthest and nearest neighbor distances becomes less significant

Purposes of dimensionality reduction include:
- ◇ Avoid curse of dimensionality
- ◇ Reduce amount of time and memory required by data mining algorithms
- ◇ Allow data to be more easily visualized
- ◇ May help to eliminate irrelevant features or reduce noise

Techniques to do so include:
- ◇ Principal Components Analysis (PCA)
- ◇ Singular Value Decomposition
- ◇ Others: supervised and non-linear techniques

1.5.2.3 Feature Subset Selection

Feature subset selection consists in selecting a subset of the original features. The goal is to find a minimal subset of features that is as good as the entire set of features for the data analysis task at hand.

For removing irrelevant features, it is needed a **performance measure** indicating how well a feature or subset of features performs w.r.t. the considered data analysis task.

For removing **redundant features**, either a *performance measure* for subsets of features or a *correlation measure* is needed.

Filter Methods

- ◇ Selection after analyzing the **significance** and **correlation** with other attributes
- ◇ Selection is independent of any data mining task
- ◇ The operation is a pre-processing

Wrapper Methods

- ◇ Selecting the top-ranked features using as reference a DM task
- ◇ Incremental Selection of the "best" attributes
  "Best" = with respect to a specific measure of statistical significance (e.g.: information gain)

Embedded Methods

- ◇ Selection as part of the data mining algorithm
- ◇ During the operation of the DM algorithm, the algorithm itself decides which attributes to use and which to ignore (e.g. Decision tree)

Feature Selection Techniques

- ◇ **Selecting the top-ranked features**: Choose the features with the best evaluation when single features are evaluated.
- ◇ **Selecting the top-ranked subset**: Choose the subset of features with the best performance. This requires exhaustive search and is impossible for larger numbers of features. (For 20 features there are already more than one million possible subsets.)
- ◇ **Forward selection**: Start with the empty set of features and add features one by one. In each step, add the feature that yields the best improvement of the performance.
- ◇ **Backward elimination**: Start with the full set of features and remove features one by one. In each step, remove the feature that yields to the least decrease in performance.

Chapter 2

Data Representation

- By **correlation**
  I want to represent data according to the correlation of the dataset
  Algorithm: `PCA`
- By **neighborhood**
  I want to represent the data so that similar instances are similar
  Algorithm: `t-SNE`
- By **manifold**
  I want to represent the data so that its manifold is preserved
  Algorithm: `UMAP`

2.1 Principal Component Analysis (PCA)

PCA is a linear dimensionality reduction technique that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Essentially, PCA exploits spectral decomposition of the whole dataset to find a new basis for the data.

Data can often be correlated, and linear dependencies can exist among variables, e.g.,

- Rent is linearly dependent on salary and food expenses
- Bank deposit is linearly dependent on salary and work
- Cardio is linearly dependent on VO_2max

Vectors are m-dimensional elements in a field, and enjoy both addition and multiplication by scalar.
Composing these two, we can generate an infinite number of vectors: this is a **vector space**, and is defined by the basis vectors involved in the composition.

A matrix A defines a space…and thus a linear transformation! Av linearly combines the columns of with coefficients given by v.

Eigenpairs (λ, v) of a square matrix A are defined by the equation $Av = \lambda v$.

The eigenvectors $v_1, \dots, v_m$ of a matrix A define the stretching of the space, and their eigenvalues $\lambda_1 > \cdots > \lambda_m$ define the stretching factor.

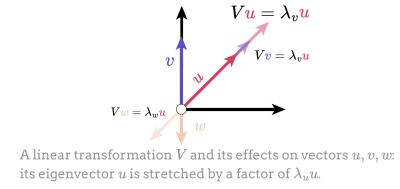
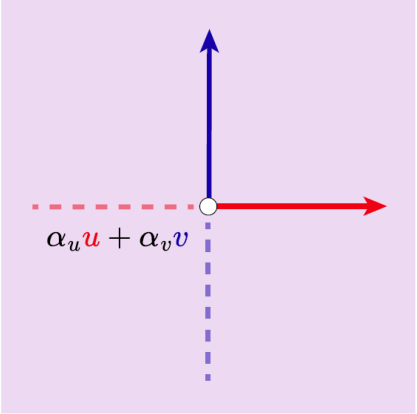


Figure 2.2: Eigenvectors of a matrix

PCA projects some data X to $\hat{X}$ through a linear transformation A: $AX = \hat{X}$.

Fun fact #1: for a mean-centered $\bar{X}$, the slope is directly proportional to the covariance!



Two vectors u, v (in red and blue), and the plane spanned by all their linear combinations $\alpha_u u + \alpha_v v$ (in purple).

Figure 2.1: Vector space spanned by two vectors

$$\bar{\Sigma} = \begin{bmatrix} \sigma^2_{\bar{X}^1} & \cdots & \text{cov}(\bar{X}^1, \bar{X}^n) \\ \cdots & \cdots & \cdots \\ & & \sigma^2_{\bar{X}^n} \end{bmatrix}$$

There is some pretty complicated linear algebra behind PCA, but the main steps are the following:

1. Mean-center your data X to get $\bar{X}$
2. Compute its eigenvectors matrix $\bar{V}$
3. Transpose V to obtain the transformation V^T
4. Project the data: $\bar{X}$ through $V^T\bar{X}$, obtaining the PCA-transformed data $\hat{X}$

In simple terms: PCA looks at your data and finds the "most important directions" - imagine you have a cloud of points and you want to find the best line that captures the main trend. PCA finds not just one line, but multiple directions ordered by importance. It then rotates your data so that the first dimension captures the most variation, the second dimension captures the second most variation. This allows you to keep only the first two dimensions while retaining most of the information, effectively reducing the complexity of your data while preserving its essential structure.

2.1.1 Observations

- ◇ PCA redefines data by removing collinearity: if your data has low covariance, the transformation will have minimal effect.
- ◇ PCA performs a linear transformation to tackle linear relationships between variables. Nonlinear relationships are not influenced.
- ◇ PCA does not work very well for high complexity data.

Uses

- ◇ **Feature selection**: high covariance of a feature may indicate disposability.
- ◇ **Dimensionality reduction**: trimming columns of lets us reduce the dimension of the resulting data.
- ◇ **Clustering preprocessing**: correlated features inflate object similarity

2.2 t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding) is a nonlinear dimensionality reduction technique particularly well suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. It works by modeling the data as a distribution of points in a high-dimensional space and then finding a lower-dimensional representation that preserves the pairwise similarities between points.

`t-SNE` focuses on data clusters rather than subspace representation, and again maps the original data X to a representation $\hat{X}$.

`t-SNE` tackles this problem in two phases:

1. **Similarity phase** In the original space $\mathcal{X}$, how similar is x_i to x_j?

How similar is x_i to x_j? Even better, what is the probability that x_i is a neighbor of x_j?

2. **Embedding phase** In the mapped space $\hat{\mathcal{X}}$, how similar is $\hat{x}_i$ to $\hat{x}_j$?

2.2.1 Similarity phase

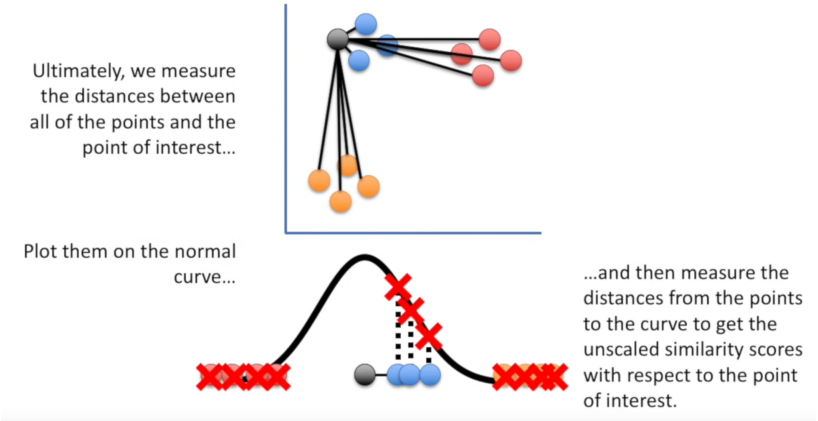


Figure 2.2: t-SNE plotting distance on an X axis and then projecting it on a normal distribution curve, to get the probability of being a neighbor

The similarity phase computes the similarity between points in the original high-dimensional space. This is typically done by converting the Euclidean distances between points into conditional probabilities that represent similarities. The probability that point x_j is a neighbor of point x_i is given by a Gaussian distribution centered at x_i. The variance of this Gaussian is controlled by a parameter called **perplexity**, which can be thought of as a smooth measure of the effective number of neighbors.

This yields a neighboring matrix P where each entry p_{ij} represents the probability that point x_j is a neighbor of point x_i in the original high-dimensional space.

The conditional probability is computed as:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

where σ_i is the variance of the Gaussian centered at point x_i. The perplexity parameter determines σ_i through a binary search to match the desired effective number of neighbors.

2.2.2 Embedding phase

In the embedding phase, t-SNE defines a similar probability distribution over the points in the low-dimensional map. However, instead of using a Gaussian distribution, it uses a Student's t-distribution with one degree of freedom (heavy-tailed distribution) to avoid the "crowding problem" where moderate distant points are forced to be too far apart in the low-dimensional representation.

The probability in the low-dimensional space is:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

where y_i and y_j are the low-dimensional counterparts of x_i and x_j.

2.2.3 Optimization

t-SNE minimizes the Kullback-Leibler divergence between the probability distributions P (high-dimensional) and Q (low-dimensional):

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

The algorithm uses gradient descent to find the low-dimensional representation Y that minimizes this cost function, effectively preserving the neighborhood structure of the original high-dimensional data.

Key advantages:

- ◇ Excellent for visualization of high-dimensional data
- ◇ Preserves local neighborhood structure
- ◇ Can reveal clusters and patterns not visible in linear methods

Key limitations:

- ◇ Computationally expensive (quadratic in the number of points)
- ◇ Non-deterministic (different runs can give different results)
- ◇ Sensitive to hyperparameters, especially perplexity
- ◇ Not suitable for embedding new data points (no explicit mapping function)

2.3 UMAP

UMAP (Uniform Manifold Approximation and Projection) is a nonlinear dimensionality reduction technique that is particularly effective for visualizing high-dimensional data in a low-dimensional space. It is based on manifold learning and topological data analysis, aiming to preserve both local and global structure of the data.

The computed distances induce a connectivity graph, and thus an adjacency matrix A, its edges measuring distances among instances. After turning distances into probabilities, UMAP optimizes a distance on A, to make it so that all and only the edges on the original manifold also appear in the transformed manifold with the same magnitude.

For the set of edges E, UMAP minimizes

$$-\sum_{e\in E}\left(\underbrace{\Pr(e;X)\log(\Pr(e;Z))}_{\text{existing edges}}+\underbrace{(1-\Pr(e;Z))\log(1-\Pr(e;X))}_{\text{non-existing edges}}\right),$$

where $\Pr(e;X), \Pr(e;Z)$ indicate the probability of edge e in the original and transformed representation, respectively.

Chapter 3

Data Cleaning

Key Points

- How to handle **anomalous values**
- How to handle **outliers**
- **Data Transformations**

3.1 Anomalous Values

- **Missing** values - NULL, ?
- **Unknown** Values - Values without a real meaning
- **Not Valid** Values - Values not significant

We can handle anomalous values with various techniques. First of all we can **eliminate** the records with anomalous values.
We could also **substitute** anomalous values, knowing, however, that it could influence the original distribution of numerical values.
To mitigate this effect, we can use **mean/median/mode** to substitute missing values, or estimate missing values using the **probability distribution** of existing values.
We could also **segment data** and apply the above for every segment where there happen to be missing values, hence using mean/mode/median or the probability distribution of the segment.
Finally, we could use *predictive models* (**classification/regression**) to estimate missing values, using the other attributes as predictors.

3.1.1 Discretization

*Discretization is the process of converting a **continuous** attribute into an **ordinal** attribute.*

- A potentially infinite number of values are mapped into a small number of categories
- Discretization is commonly used in classification
- Many classification algorithms work best if both the independent and dependent variables have only a few values

Unsupervised Discretization

- No label for instances
- The number of classes is unknown

Techniques of binning:

- **Natural** binning - Intervals with the same width
- **Equal Frequency** binning - Intervals with the same frequency
- **Statistical** binning - Use statistical information (Mean, variance, Quartile)

3.1.1.1 Natural Binning

This is a fairly simple approach: we sort the values, subdividing the range of values into k parts with the same size.

$$\sigma = \frac{x_{max} - x_{min}}{k}$$

Then, element x_j is assigned to bin —i.e. belong to the class— i if:

$$x_j \in [x_{min} + i\sigma, x_{min} + (i+1)\sigma)$$

Note however that this approach is sensitive to outliers, which can skew the range of values, generating very unbalanced distributions.

3.1.1.2 Equal Frequency Binning

Sort and count the elements, definition of k intervals of f, where, having N elements:

$$f = \frac{N}{k}$$

The element x_j is assigned to bin i if:

$$i \times f \leq j < (i+1) \times f$$

This is not always suitable for highlighting interesting correlations.

3.1.1.3 How many bins?

In both cases, the number of bins k is a parameter to be set. A rule of thumb is to set k accorgin to Sturges' optimal number of classes C for N elements:

$$C = 1 + \frac{10}{3}\log_{10}(N)$$

where N is the number of elements. The optimal width of the classes depends on the variance and the number of data points:

$$h = \frac{3.5 \cdot s}{\sqrt{N}}$$

3.2 Supervised discretization

3.2.1 Entropy-based Discretization

Minimizes the entropy wrt a label, with the goal of maximizing the purity of intervals (information gain).

3.3 Binarization

Binarization is the process of converting a continuous attribute into a binary attribute.
This can be useful in various scenarios, such as when we want to simplify the model or when we need to handle categorical variables. It is common to apply this for **association** analysis.

There are several techniques for binarization of continuous attributes:

- ◇ **Thresholding** - Assigning values above a certain threshold to one class and values below to another.
- ◇ **One-Hot Encoding** - Creating binary columns for each category in a categorical variable.
- ◇ **Binarization with Decision Trees** - Using decision tree algorithms to find optimal splits for binarization.

There are also techniques for binarization of categorical attributes:

Table 3.1: Conversion of a categorical attribute to three binary attributes.

Categorical Value	**Integer Value**	x_1	x_2	x_3
awful	0	0	0	0
poor	1	0	0	1
OK	2	0	1	0
good	3	0	1	1
great	4	1	0	0

This however leads to highlighting associations that are not really there, for example, in the table above, x_2 and x_3 may look correlated, but, in fact, they are not.

Table 3.2: Conversion of a categorical attribute to five asymmetric binary attributes.

Categorical Value	**Integer Value**	x_1	x_2	x_3	x_4	x_5
awful	0	1	0	0	0	0
poor	1	0	1	0	0	0
OK	2	0	0	1	0	0
good	3	0	0	0	1	0
great	4	0	0	0	0	1

3.3.1 Attribute Transformation

An attribute transform is a function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values.

- ◇ Simple functions: x^k, $log(x)$, e^x, $|x|$
- ◇ **Normalization** Refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range
- ◇ Take out unwanted, common signal, e.g., seasonality
- ◇ In statistics, **standardization** refers to subtracting off the means and dividing by the standard deviation

The transformation $Y = T(X)$ should:
- ◇ preserve the relevant information of X
- ◇ eliminates at least one of the problems of X
- ◇ is more useful of X

3.3.1.1 Normalization

- ◇ Min-Max Normalization

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

- ◇ z-score Normalization

$$v' = \frac{v - \mu}{\sigma} = \frac{v - mean_A}{stddev_A}$$

- ◇ Decimal Scaling

$$v' = \frac{v}{10^j} \qquad \textit{where } j \textit{ is the smallest integer such that } max(|v'|) < 1$$

3.3.1.2 Transformation functions

- ◇ Exponential Transformation

$$T_p(x) = \begin{cases} ax^p + b & (p \neq 0) \\ c \log x + d & (p = 0) \end{cases}$$

where $a, b, c, d, p \in \mathbb{R}$
  - – Preserve the order
  - – Preserve some basic statistics
  - – They are continuous functions
  - – They are derivable
  - – They are specified by simple functions

◇ Logarithmic Transformation - Stabilizing Variance

$$T(x) = c \log x + d$$

- Applicable to positive values
- Makes homogenous the variance in log-normal distributions
  E.g.: normalize seasonal peaks

◇ ??? - Stabilizing Variance

$$T(x) = ax^p + b$$

- Square-root Transformation
  - p = 1/c, c integer number
  - To make homogenous the variance of particular distributions e.g., Poisson Distribution
- Reciprocal Transformation
  - p < 0
  - Suitable for analyzing time series, when the variance increases too much wrt the mean

Chapter 4

Cluster analysis

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups.

The aim of clustering is to ease data **understanding** and **summarization**, i.e. reduce the size of data sets.

The following are *NOT* cluster analysis:

- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name
- Results of a query
  - Groupings are a result of an external specification
  - Clustering is a grouping of objects based on the data
- Supervised classification
  - Have class label information
- Association Analysis
  - Local vs. global connections

These ain't clustering essentially because there is no **similarity** measure involved, which instead is the core of clustering.

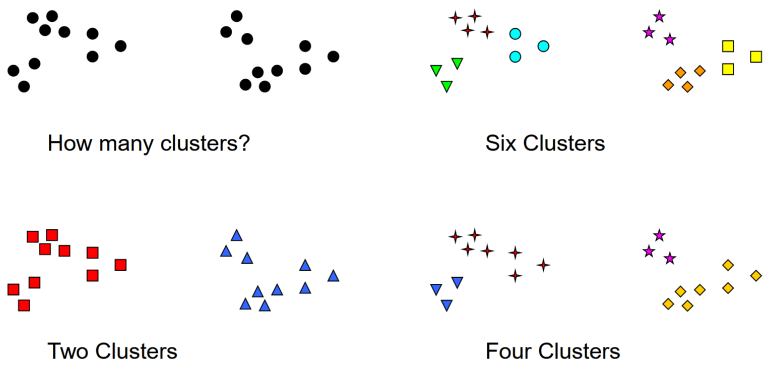


Figure 4.1: "Cluster" may be an amibguous term. How to define a cluster? How big is a cluster? How many clusters are there?

4.1 Definitions

A **clustering** is a set of clusters. There is an important distinction between hierarchical and partitional sets of clusters.

- *Partitional* Clustering
  A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.
- *Hierarchical* Clustering

A set of nested clusters organized as a hierarchical tree.

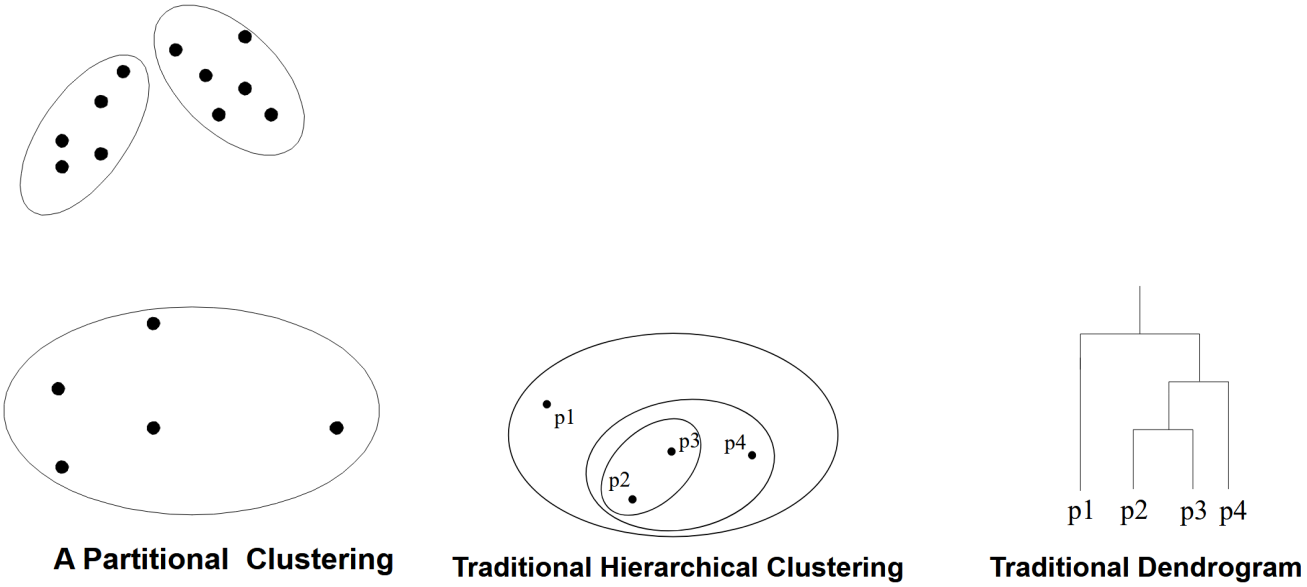


Figure 4.2: Partitional vs Hierarchical clustering

There are other distinctions among clusterings:

- ◇ Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or 'border' points
- ◇ Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- ◇ Partial versus complete
  - In some cases, we only want to cluster some of the data
- ◇ Heterogeneous versus homogeneous
  - Clusters of widely different sizes, shapes, and densities

4.2 Types of Clustering

- ◇ Well-separated clusters A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster
- ◇ Center-based clusters

Figure 4.3: Center-based clusters

  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster

- ◇ Contiguous clusters (Nearest neighbor or Transitive)
  - Each point is closer to at least one point in its cluster than to any point in another cluster.
  - Graph based clustering
  - This approach can have trouble when noise is present since a small bridge of points can merge two distinct clusters
- ◇ Density-based clusters

Figure 4.3: Contiguity-based clusters

  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.
  - Points which are not classfied as part of any cluster are considered noise. In the figure they may be the gray blackground points.

Figure 4.4: Density-based clusters

- ◇ Property or Conceptual
- ◇ Described by an Objective Function
  - Finds clusters that minimize or maximize an objective function.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the 'goodness' of each potential set of clusters by using the given objective function.
  - NP Hard)
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives

4.3 Similarity

- ◇ Similarity
  - – Numerical measure of how alike two data objects are.
  - – Is higher when objects are more alike.
  - – Often falls in the range [0,1]
- ◇ Dissimilarity
  - – Numerical measure of how different are two data objects
  - – Lower when objects are more alike
  - – Minimum dissimilarity is often 0
  - – Upper limit varies
- ◇ Proximity refers to a similarity or dissimilarity

4.3.1 Similarity and Dissimilarity for Different Attribute Types

Table 4.1: Similarity and dissimilarity for simple attributes

Attribute Type	**Dissimilarity**	**Similarity**
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d-min\_d}{max\_d-min\_d}$

p and q are the attribute values for two data objects.

4.3.2 Euclidean Distance

The most common way to measure distance between two data objects is the Euclidean distance:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

where n is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) or data objects $\mathbf{x}$ and $\mathbf{y}$. Standardization is necessary, if scales differ.

- ◇ Standardization is necessary, if scales differ.

4.3.2.1 Minkowski Distance

Minkowski Distance is a generalization of Euclidean Distance. r is a parameter that defines the type of distance, n is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) or data objects $\mathbf{x}$ and $\mathbf{y}$.

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{\frac{1}{r}}$$

- ◇ For $r = 1$, it is the Manhattan distance (or city block/Hamming distance)
- ◇ For $r = 2$, it is the Euclidean distance
- ◇ For $r \to \infty$, it is the supremum distance (or Chebyshev distance)

Metrics and Similiraties

1. $d(x, y) \geq 0$ for all x and y and $d(x, y) = 0$ only if $x = y$. (Positive definiteness)
2. $d(x, y) = d(y, x)$ for all x and y. (Symmetry)
3. $d(x, z) \leq d(x, y) + d(y, z)$ for all points x, y, and z (Triangle Inequality).

A *distance* d is a *metric* if it satisfies the three conditions above.

1. $s(x, y) = 1$ (or maximum similarity) only if $x = y$.
2. $s(x, y) = s(y, x)$ for all x and y. (Symmetry)

These instead are properties of a *similarity* measure.

4.3.3 Binary Similarity

Computing similarity among objects described by binary attributes is slightly different from numerical attributes.

◇ Simple Matching Coefficient (SMC)

$$SMC = \frac{\texttt{\#matches}}{\texttt{\#attributes}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

◇ Jaccard Coefficient

$$J = \frac{\texttt{\#11matches}}{\texttt{\#non-zero attribute values}} \frac{f_{11}}{f_{11} + f_{10} + f_{01}}$$

$$\begin{aligned}
p &= 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
q &= 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \\
f_{01} &= 2 \text{ (\#attributes where } p \text{ was 0 and } q \text{ was 1)} \\
f_{10} &= 1 \text{ (\#attributes where } p \text{ was 1 and } q \text{ was 0)} \\
f_{00} &= 7 \text{ (\#attributes where } p \text{ was 0 and } q \text{ was 0)} \\
f_{11} &= 0 \text{ (\#attributes where } p \text{ was 1 and } q \text{ was 1)} \\
SMC &= \frac{0+7}{10} = 0.7 \\
J &= \frac{0}{0+1+2} = 0
\end{aligned}$$

4.3.4 Cosine Similarity

Used often in text mining, where each dimension corresponds to a term (word) and the value of the dimension is the frequency of the term in the document.

If d_1 and d_2 are two document vectors, then

$$\cos(d_1, d_2) = (d_1 \cdot d_2)/||d_1||||d_2||$$

where $\cdot$ indicates vector dot product and $||d||$ is the length of vector d.

Example:

$$\begin{aligned}
d_1 &= 3\ 2\ 0\ 5\ 0\ 0\ 0\ 2\ 0\ 0 \\
d_2 &= 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 2 \\
d_1 \cdot d_2 &= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5 \\
||d_1|| &= \sqrt{3^2 + 2^2 + 0^2 + 5^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = \sqrt{42} \\
||d_2|| &= \sqrt{1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 2^2} = \sqrt{6} \\
\cos(d_1, d_2) &= \frac{5}{\sqrt{42} * \sqrt{6}} = \frac{5}{\sqrt{252}} \approx 0.315
\end{aligned}$$

4.3.5 Correlation

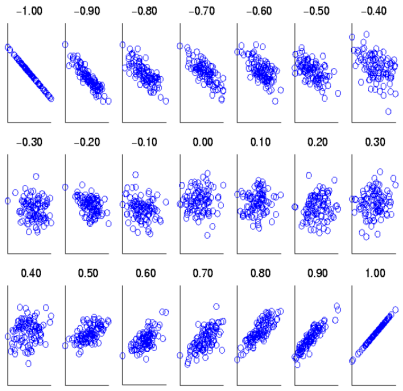


Figure 4.4: Scatter plots showing the similarity from -1 to 1

Correlation measures the linear relationship between objects (binary or continuous). To compute correlation, we standardize data objects, p and q, and then take their dot product (covariance/standard deviation).

$$corr(\mathbf{x}, \mathbf{y}) = \frac{covariance(\mathbf{x}, \mathbf{y})}{stddev(\mathbf{x}) * stddev(\mathbf{y})} = \frac{\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{n}(y_k - \bar{y})^2}}$$

4.3.6 Entropy

Information relates to possible outcomes of an event such as transmission of a message, flip of a coin, or measurement of a piece of data. The amount of information is inversely related to the probability of an event. Entropy is the commonly used measure of information content. For a discrete random variable X with n possible values $x_1, x_2, ..., x_n$ each having probability $p_1, p_2, \ldots, p_n$, the entropy $H(X)$ is defined as:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Entropy is measured in bits and is $0 \leq H(X) \leq \log_2(n)$. Thus, entropy is a measure of how many bits it takes to represent an observation of X on average.

The information one variable provides about another is called mutual information. The mutual information $I(X;Y)$ of two discrete random variables X and Y is defined as:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) = \sum_{x \in X}\sum_{y \in Y} p(x,y) \log_2 p(x,y)$$

where $H(X,Y)$ is the joint entropy of X and Y, and $p(x,y)$ is the probability that x and y occur together.

4.4 K-Means

- ◇ *Partitional* clustering approach
- ◇ The number of clusters K, must be specified
- ◇ Each cluster is associated with a **centroid** (center point), typically being the mean of the points in the cluster
- ◇ Each point is assigned to the cluster with the closest centroid. This may measured as Euclidean distance, cosine similarity, correlation, etc.
  Using these measures makes K-Means to converge in the first few iterations.
- ◇ Complexity is $O(nKId)$, where n is the number of data points, K is the number of clusters, I is the number of iterations, and d is the number of dimensions (attributes)
- ◇ The basic algorithm is very simple

Algorithm 1 K-Means Clustering Algorithm

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

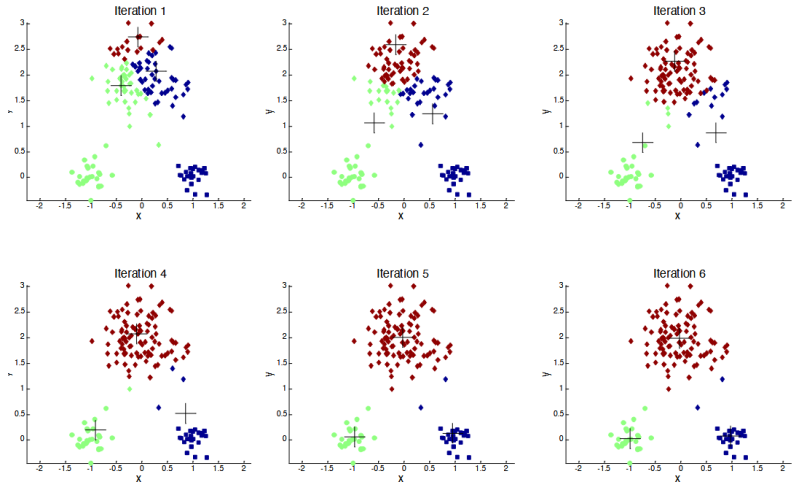


Figure 4.5: K-Means Clustering Iterations

4.4.1 Evaluating K-Means clusters

The most common measure to evaluate the quality of K-Means clusters is the **Sum of Squared Errors** (SSE), also known as **Inertia**. It quantifies how tightly the data points in a cluster are grouped around their centroid. The SSE is calculated as follows:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$

- K is the number of clusters,
- C_i is the set of points in cluster i,
- x is a data point in cluster C_i,
- μ_i is the centroid of cluster C_i,
- $||x - \mu_i||^2$ is the squared Euclidean distance between point x and centroid μ_i.

4.4.2 Limitations of K-Means

K-means has problems when clusters are of differing

- Clusters have different sizes
- Clusters have different densities
- Clusters have Non-globular shapes
- The data contains **outliers**.

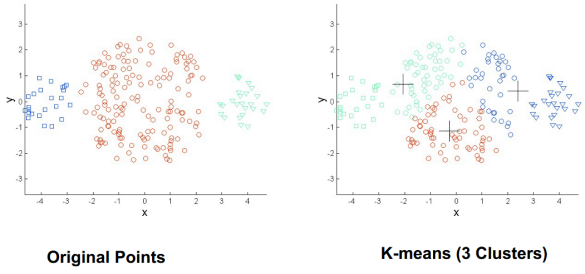


Figure 4.6: K-Means with different cluster sizes
A solution may be to use more clusters, hence increasing K, and then merging them.

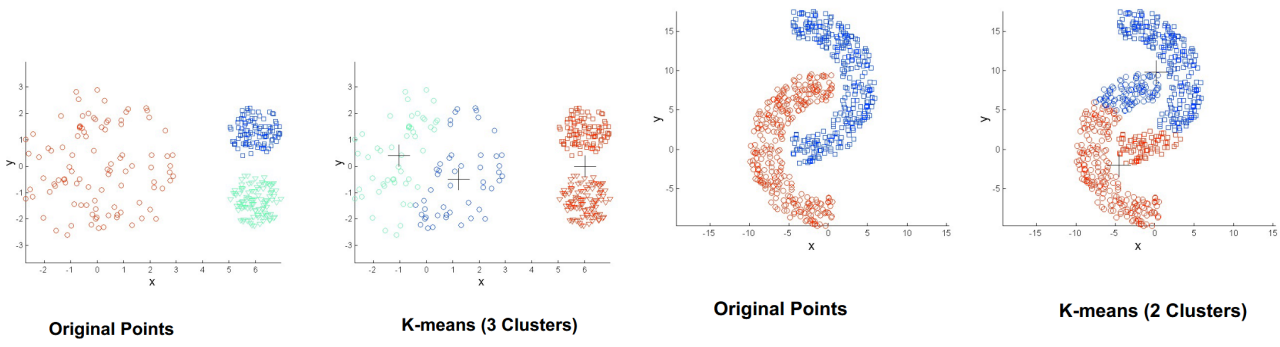


Figure 4.6: K-Means with different cluster densities and non-globular shapes

4.4.2.1 Empty Clusters

K-Means sometimes produces empty clusters. This happens when no points are assigned to a cluster during the assignment step. This can occur if the initial centroids are poorly chosen or if the data distribution is such that certain centroids end up being too far from any data points.

Solutions include the following strategies, which may be iterated until no empty clusters remain:

- ◇ Choose a point and assign it to the cluster
  - – Choose the point that contributes most to SSE
  - – Choose a point from the cluster with the highest SSE

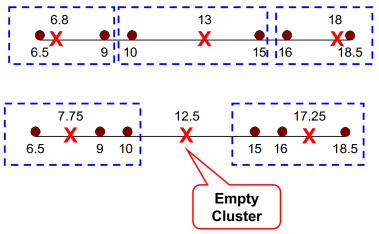


Figure 4.7: K-Means with Empty Clusters

4.4.3 Workflow

1. Data Preprocessing
   - ◇ Handle missing values
   - ◇ Normalize/standardize data
2. Choose the number of clusters K
3. Post processing and validation
   - ◇ Eliminate small clusters that may represent outliers
   - ◇ Split 'loose' clusters, i.e., clusters with relatively high SSE
   - ◇ Merge clusters that are 'close' and that have relatively low SSE
   - ◇ Can use these steps during the clustering process
     `ISODATA`

4.4.4 Choosing the number of clusters K

If there are K "real" clusters then the chance of selecting one centroid from each cluster is small, especially if K is large.

A solution is to run K-Means multiple times with different initial centroids and choose the best result, but probability of getting a good result is still small.

A different approach is to use **hierarchical clustering** to determine initial centroids for K-Means.

Another option is to **incrementally** update centroids as points are assigned to clusters, instead of waiting until all points are assigned. This is more expensive and introduces order dependence.

We can define a goodness measure of a cluster c (hence of the set of Clusters C) as the SSE from the cluster centroid:

$$SSE_C(c, s) = \sum_{i=1}^{n} (d_i, s_c)^2 \tag{4.1}$$

$$G(C, s) = \sum_{c \in C} SSE_C(c, s) \tag{4.2}$$

where s_c is the centroid of cluster c, d_i is a point in cluster c, and C is the set of clusters.

Re-assignment of points to clusters and re-computation of centroids is guaranteed to *not increase* (or *monotonically decreases*) $G(C, s)$, hence K-Means converges to a local minimum.

At any step we have some value for G(C,s),

1. Fix s, optimize $C \rightarrow$ Assignment step: assign d_i to the closest centroid $\Rightarrow G(C', s) \leq G(C, s)$
2. Fix C', optimize $s \rightarrow$ Update step: recompute centroids $\Rightarrow G(C', s') \leq G(C', s) \leq G(C, s)$

In this way the new cost results smaller than the original one, leading to convergence to a local minimum.

4.5 Hierarchical Clustering

Hierarchical clustering produces a set of nested clusters organized as a hierarchical tree can be visualized as a dendrogram, which is a tree-like diagram that records the sequences of merges or splits.

We do not have to assume any particular number of clusters, any desired number of clusters can be obtained by "cutting" the dendrogram at the proper level, each possibly corresponding to meaningful taxonomies.

4.5.1 Agglomerative vs Divisive

There are two types of hierarchical clustering, both exploiting a similarity or distance matrix:

- ◇ **Agglomerative** (bottom-up)
  Start with the points as individual clusters, and, at each step, merge the closest pair of clusters until only one cluster (or k clusters) remain.
- ◇ **Divisive** (top-down)
  Start with all points in one cluster and, at each step, split the cluster into smaller clusters until each point is its own cluster (or k clusters are formed).

Algorithm 2 Agglomerative Hierarchical Clustering Algorithm

```
Compute the proximity matrix
Let each data point be a cluster
repeat
    Merge the two closest clusters
    Update the proximity matrix
until only a single cluster remains
```

Proximity matrix is a square matrix that contains the distances (or similarities) between each pair of data points. "Proximity" is just a general term that can refer to either distance or similarity, depending on the context.

4.5.1.1 Updating Proximity Matrix

When two clusters c_i and c_j are merged into a new cluster c_k, we need to update the proximity matrix to reflect the distances between the new cluster c_k and all other existing clusters. There are several methods to do this, each defining the distance between clusters differently:

- ◇ **Single Linkage** (or Nearest Neighbor) The distance between two clusters is defined as the minimum distance between any single point in the first cluster and any single point in the second cluster.
$$d(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y)$$
- ◇ **Complete Linkage** (or Farthest Neighbor) The distance between two clusters is defined as the maximum distance between any single point in the first cluster and any single point in the second cluster.
$$d(c_i, c_j) = \max_{x \in c_i, y \in c_j} d(x, y)$$
- ◇ **Average Linkage** The distance between two clusters is defined as the average distance between all pairs of points, one from each cluster.
$$d(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{x \in c_i} \sum_{y \in c_j} d(x, y)$$
  Note that **Single**, **Complete** and **Average** Linkage have all the same complexity! $O(n^3)$, where n is the number of data points.
- ◇ **Centroid Linkage** The distance between two clusters is defined as the distance between their centroids.
$$d(c_i, c_j) = d(\mu_i, \mu_j)$$
  where μ_i and μ_j are the centroids of clusters c_i and c_j, respectively.
- ◇ **Ward's Method** This method minimizes the total within-cluster variance. When two clusters are merged, the increase in the total within-cluster variance is calculated, and the pair of clusters that results in the smallest increase is merged.
$$d(c_i, c_j) = \frac{|c_i||c_j|}{|c_i| + |c_j|} ||\mu_i - \mu_j||^2$$
  where $|c_i|$ and $|c_j|$ are the sizes of clusters c_i and c_j, and μ_i and μ_j are their centroids.

4.5.2 Divisive Hierarchical Clustering

First of all, build MST (Minimum Spanning Tree) of the data points, then iteratively remove the longest edge in the MST to split the data into clusters.
Start with a single cluster which consists of any point, and at each step, look for the closest pair of points p, q such that p is in the current tree while q is not. Add q to the tree and the edge (p, q) to the MST. Repeat until all points are in the tree.

Algorithm 3 Divisive Hierarchical Clustering Algorithm

1: Compute a MST for the proximity graph
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the longest edge in the MST (smallest proximity)
4: **until** only singleton clusters remain

4.5.3 Complexity and Limitations

- ◇ Time Complexity - $O(n^3)$ for naive implementations of hierarchical clustering
  - $O(n^2)$ to compute the proximity matrix
  - $O(n)$ to find the closest pair of clusters
  - $O(n)$ to update the proximity matrix
  - Repeated n times

  More efficient algorithms can achieve $O(n^2 \log n)$.
- ◇ Space Complexity - $O(n^2)$ for storing the proximity matrix

4.5.3.1 Limitations

Once a decision is made to combine two clusters, it cannot be undone.
No global objective function is directly minimized.
Different schemes have problems with one or more of the following:

- ◇ Sensitivity to noise and outliers
- ◇ Difficulty handling clusters of different sizes and non- globular shapes
- ◇ Breaking large clusters

4.6 Density based - DBSCAN

Definition 4.1 (Density) ***Density*** *is the number of points within a specified radius (Eps or ϵ).*

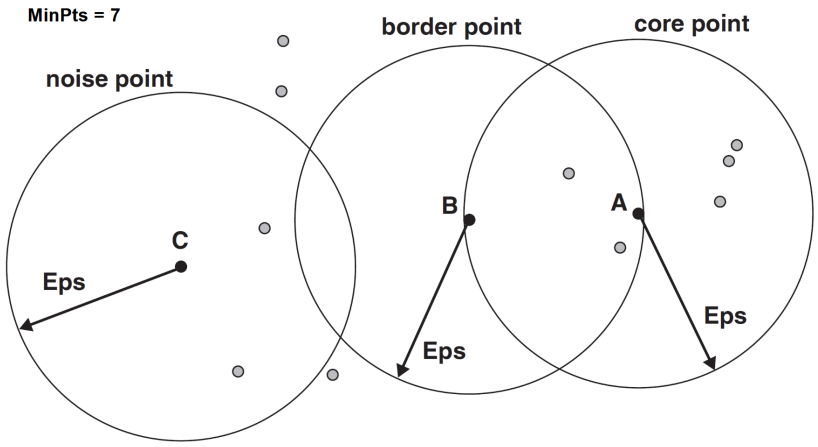


Figure 4.7: Core, Border, and Noise Points in DBSCAN

A point is a **core point** if it has at least MinPts points within distance Eps (including itself). These are points which are at the interior of a cluster.
A point is a **border point** if it has fewer than MinPts points within distance Eps, but is in the neighborhood of a core point. These are points which are on the edge of a cluster.
A point is a **noise point** (or outlier) if it is neither a core point nor a border point. These are points which are not part of any cluster.

DBSCAN

1. Label points as core, border, or noise based on thresholds R (radius of neighborhood) and `min_pts` (min number of neighbors)
2. Connect core points that are within R of each other, hence are neighbors, putting them in the same cluster
3. Associate border points to the nearest core point, hence to the same cluster as the core point, and remove noise.

Algorithm 4 DBSCAN Algorithm

```
current_cluster_label ← 0
for all core points do
    if the core point has no cluster label then
        current_cluster_label ← current_cluster_label + 1
        Label the current core point with cluster label current_cluster_label
    for all points in the Eps-neighborhood, except i^th the point itself do
        if the point does not have a cluster label then
            Label the point with cluster label current_cluster_label
```

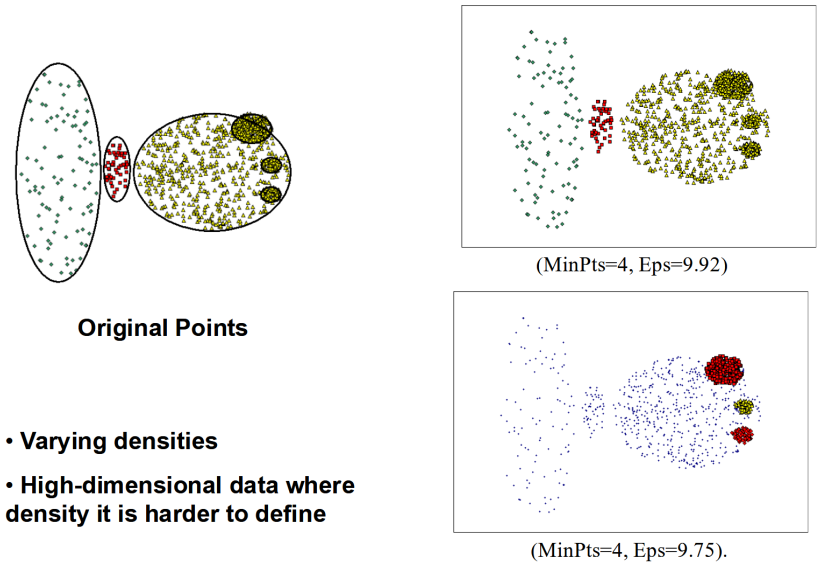


Figure 4.8: DBSCAN Not behaving well in given situations

Determining Eps and MinPts

To determine Eps and MinPts, we can use the k-distance graph.

The idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance Noise points have the k^{th} nearest neighbor at distance.

So, plot sorted distance of every point to its k^{th} nearest neighbor.

4.7 Cluster Validity

For supervised classification we have a variety of measures to evaluate how good our model is. such as accuracy, precision, recall, etc…

" For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters? " —

"Clusters are in the eye of the beholder", but we still want to evaluate them to avoid finding patterns in noise, compare clustering algorithms, or to compare clusters or sets of clusters.

4.8 Towards cluster validation

1. Determining the **clustering tendency** of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data without reference to external information.
4. Comparing the results of two different sets of cluster analyses to determine which is better
5. Determining the "correct" number of clusters.

Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types:

- ◇ **External** Index: Used to measure the extent to which cluster labels match externally supplied class labels.
  Entropy
- ◇ **Internal** Index: Used to measure the goodness of a clustering structure without respect to external information.
  Sum of Squared Error (SSE)
- ◇ **Relative** Index: Used to compare two different clusterings or clusters.
  Often an external or internal index is used for this function, e.g., SSE or entropy

Sometimes these are referred to as ***criteria*** instead of *indices*; however, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

4.8.1 Measuring validity through correlation

Two matrices:

- ◇ Proximity matrix
- ◇ Ideal similarity matrix
  - One row and one column for each data point
  - An entry is 1 if the associated pair of points belong to the same cluster
  - An entry is 0 if the associated pair of points belongs to different clusters

Compute the correlation between the two matrices; note that since the matrices are symmetric, only the correlation between $n(n-1)/2$ entries needs to be calculated.

4.8.2 Internal measures

4.8.2.1 SSE - Sum of Squared Error

SSE (Sum of Squared Error) is a common measure of cluster cohesion, defined as:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$$

where m_i is the centroid of cluster C_i.

This is a measure of how tightly the data points in a cluster are grouped together. Lower values of SSE indicate more cohesive clusters. It can be a nice measure which does not need external information, and may also be used to determine the appropriate number of clusters by plotting the SSE for different values of k and looking for an "elbow" in the graph.

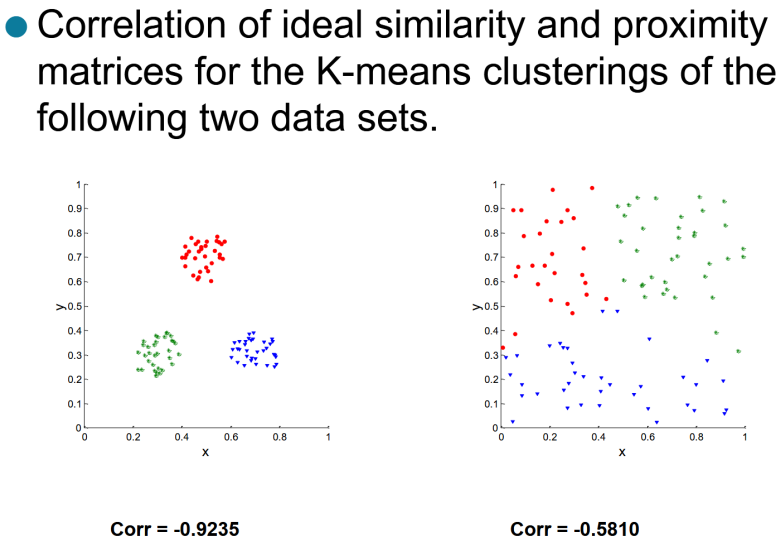


Figure 4.9: Correlation of two sets. Actually, in real world data it's very rare to observe a -0.9235 correlation score

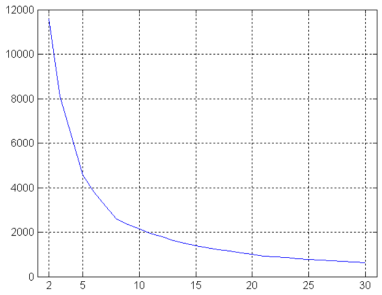


Figure 4.10: SSE of cluster found using K-means with various values of k

4.8.2.2 Cohesion and Separation

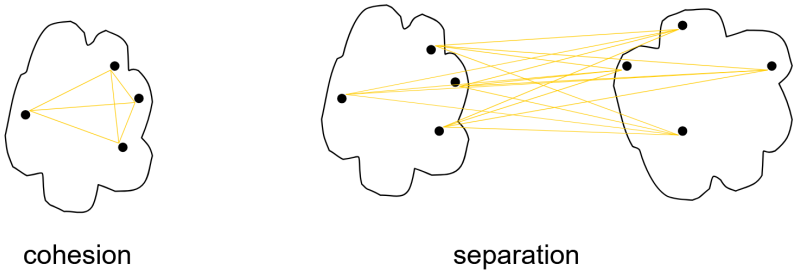


Figure 4.11: Cohesion and separation

A proximity graph based approach can also be used for cohesion and separation.

- Cluster **cohesion** is the sum of the weight of all links within a cluster, it measures how closely related are objects in a cluster. (SSE is a measure of cohesion)

$$SSE = WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

- Cluster **separation** is the sum of the weights between nodes in the cluster and nodes outside the cluster. Measures how distinct or well-separated a cluster is from other clusters. (Squared error between clusters is a measure of separation)

$$BSS = \sum_i |C_i|(m - m_i)^2 \qquad |C_i| \text{ size of cluster } C_i$$

Note that $BSS + WSS = TSS$ where TSS is the total sum of squares, and is constant and independent of the clustering.

4.8.2.3 Silhouette coefficient

Silhouette coefficient combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings.

Consider a point i in cluster A:

- ◇ Calculate a = average distance of i to the points in its cluster
- ◇ Calculate b = min (average distance of i to points in another cluster)
- ◇ The silhouette coefficient for a point is then given by

$$s = \frac{b - a}{\max(a, b)}$$

- ◇ Typically between 0 and 1. The closer to 1 the better.
- ◇ Can calculate the average silhouette coefficient for a cluster or a clustering.

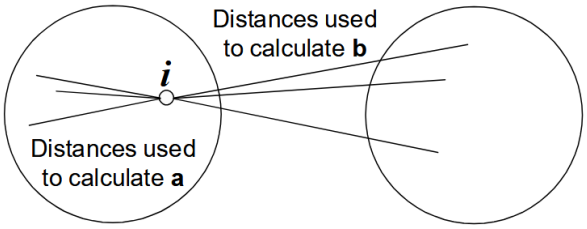


Figure 4.12: Silhouette coefficient for i

4.8.3 External measures

External measures compare the clustering results to an externally known class labels. Examples may be **entropy** and **purity**.

Table 4.2: K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

4.8.3.1 Entropy

Entropy measures how the various clusters are distributed with respect to the class labels.

For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the "probability" that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j.

Then using this class distribution, the entropy of *each* cluster j is calculated using the standard formula:

$$e_j = -\sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$$

where L is the number of classes.

The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e.:

$$e = \sum_{i=1}^{K} \frac{m_i}{m} e_j$$

where m_j is the size of cluster j, K is the number of clusters, and m is the total number of data points.

4.8.3.2 Purity

Using the terminology derived for entropy, the purity of cluster j, is given by:

$$purity_j = \max_i p_{ij}$$

and the overall purity of a clustering by:

$$purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$$

Chapter 5

Anomaly Detection

We may refer to anomalies in data also as *outliers.*

5.1 Outliers

- ◇ Inherently **fuzzy** - An instance has a degree of outlierness, which we can threshold to decide whether an instance is an outlier or not.
- ◇ **Data-dependent** - Outlier are exceptions to the data. But outliers themselves define the data...?
- ◇ **Not noise** - Noise is random, outliers are exceptional.
- ◇ **Mono/multi-dimensional** - An outlier can be so on one just one dimension, or on multiple.

Outliers are something that is either **unusual** or **extreme**, or both.
Outliers are, by nature, defined in terms of other instances. Whatever approach we use to detect them, we should take into account that they influence it as well.

5.2 Outlier Detection Algorithms

Algorithms used to detect outliers usually involve two key steps:

1. **Grading** - Define a grading function $\tilde{o}$ that assigns to each instance x a degree of outlierness/anomaly $\tilde{o}(x)$.
2. **Thresholding** - Decide on a threshold $\hat{o}$ such that instances with $\tilde{o}(x) > \hat{o}$ are considered outliers.

We can categorize outlier detection algorithms depending on the *axis* on which they operate:

- ◇ **Locality** - is the outlier detection performed in a local neighborhood (local) or considering the entire dataset (global)?
- ◇ **Sensitivity** - is the outlier detection sensitive to the presence of other outliers (sensitive) or not (robust)?
- ◇ **Interpretability** - can we interpret/explain why an instance is considered an outlier (interpretable) or not (black-box)?

5.2.1 Distributions

Locality	**Global/Local**
Sensitive	**Robust/Sensitive**
Interpretable	**Black-box/Interpretable**

In case of a normal distribution $\mathcal{N}(\mu, \sigma)$, we can define outliers as instances that are more than k standard deviations away from the mean μ (see Figure 5.2).
This approach is **global**, **robust** and **interpretable**.

Locality	**Global**
Sensitive	Outliers influence the distribution but may be removed by Grubb's test
Interpretable	**Black-box**, no clear explanation for outlierness, simply, there are not many similar instances.

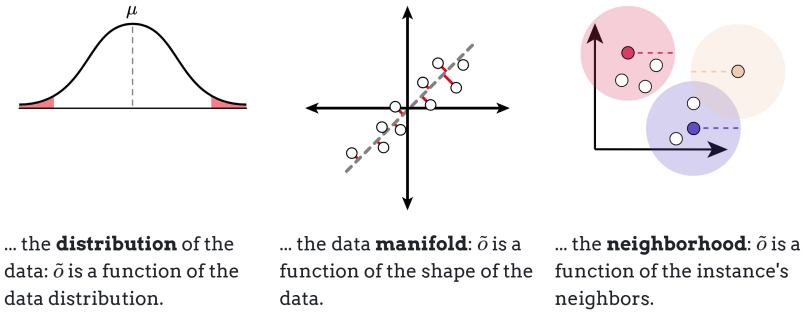


Figure 5.1: Distribution, manifold, neighborhood

The degree of anomaly $\tilde{o}(x)$ can be defined as:

$$\text{z-score} = \tilde{o}(x) = \frac{|x - \mu|}{\sigma}$$

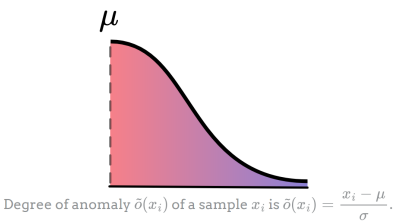


Figure 5.2: For a normal distribution $\mathcal{N}(\mu, \sigma)$, we can define outliers as instances that are more than k standard deviations away from the mean μ.

5.2.1.1 Grubbs test

$z - scores$ generate sample-dependent outlier degrees $\tilde{o}(x_1), \tilde{o}(x_2), \ldots, \tilde{o}(x_n)$, but does not tackle the **+1 problem**.

The +1 problem is the problem of deciding whether the most extreme instance in a dataset is an outlier or not.
Grubb's test iterates over detected outliers, removing one layer of outliers at a time, until no more outliers are found.

Algorithm 5 Grubb's test for outliers

1: Find current outlier set $\hat{X}$
2: If $\hat{X} = \emptyset$, stop
3: $X = X \setminus \hat{X}$ - remove outliers from dataset
4: Go to step 1

Data may vary *locally*: subsets of the data each follow a different distribution.
Assumption: there exists a partition of the data, each block distributed according to a Normal distribution.
Thus we could use multiple models to model the data, and define outliers as instances that are outliers with respect to their local distribution.

One of k models $M_{\theta_0}, M_{\theta_1}, \ldots, M_{\theta_{k-1}}$ is sampled, each with a sampling probability m_i. Different distributions sample different regions of the density.

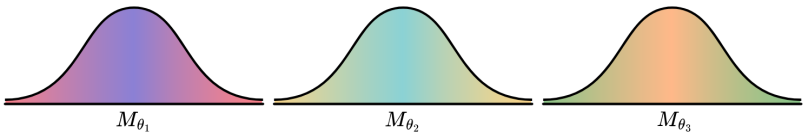


Figure 5.2: A mixture of Normals $M_{\theta_0}, M_{\theta_1}, M_{\theta_2}$; each sampled with probability m_0, m_1, m_2, respectively.

Locality	Local
Sensitive	Outliers influence the distribution, may be unstable, but may be removed by Grubb's test (?)
Interpretable	**Black-box**, no clear explanation for outlierness, simply, there are not many similar instances.

5.2.2 Thresholding

Grubb's test

Choosing a threshold $\hat{o}$ is arbitrary, but there are algorithms such as Grubb's test which define their own thresholding mechanism. In Grubb's test, the threshold is defined as:

$$\hat{o} = n\frac{\sum(x_i - \bar{X})^4}{(\sum(x_i - \bar{X})^2)^2}$$

z-scores assume a Normal distribution, but often this is not the case. Yet, we can still identify tails of a distribution, and in turn, anomalies.

- ◇ Markov inequality - for a variable (distribution)X with positive values and threshold β, it holds

  $$P(X \geq \beta) \leq \frac{\mathbb{E}(X)}{\beta}$$

  Thus, given an estimate of the variable's expected value, we can retrieve the inverse of an image of its cumulative distribution
- ◇ Chebyshev inequality - for a variable (distribution) X with mean μ $(= \mathbb{E}(X))$ and standard deviation σ, and threshold β, it holds

  $$P(|X - \mathbb{E}(X)| > \beta) \leq \frac{\sigma^2}{\beta^2}$$

  That is, the probability of deviation from the mean is inversely proportional to the deviation

5.2.3 Manifold

Distributional approaches define the density, but do not describe the data itself. $\tilde{o}$ is defined in terms of the manifold: does the given instance lie in the manifold? Just like the distributional approach, we must assume the manifold family. To preserve the interpretability of our results, we initially stick to linear manifolds.

We can define the anomaly degree $\tilde{o}(x)$ as the distance of x from the manifold.
We can use PCA to find the linear manifold that best fits the data.
PCA finds the directions of maximum variance in the data, and uses them as a new basis for the data.

A matrix A spans a linear space, thus every vector b in its spanned space is defined as a linear combination of $A : b = Ax$. For non fullrank matrices A, such a solution x may not exist. Thus, we need to project on the data manifold.

Least Squares

Least Squares is a method to find the best fitting solution to an overdetermined system of equations $Ax = b$ (more equations than unknowns).
The best fitting solution is the one that minimizes the residual sum of squares (RSS):

$$RSS = ||Ax - b||_2^2$$

Least squares assumes a linear manifold, and squared norm as distance metric.

The instability of least squares is due to the data collinearity. A possible solution: de-correlate the data! PCA does exactly that.

Some mathy examples are displayed in the lecture slides…

Manifold-based algorithms are as flexible as the defined manifold. Like with mixture models, neighbor-based approaches reintroduce locality: outliers are defined in function of their neighbors:

Locality	**Global**
Sensitive	Strongly influenced by outliers
Interpretable	Partial: which instances have lower degrees? What even is a "low" degree?

Table 5.1: Least squares

- ◇ **Connectivity** - An outlier is defined in terms of the connectivity to its neighbors
- ◇ **Concentration** - An outlier is defined in terms of its neighbor concentration

Each instance has a posting list of neighbors, from the closest to the farthest: the lower the aggregated position in other lists, the higher the connectivity degree.

- ◇ Posting position defines connectivity: it is not density
- ◇ Connectivity is asymmetric: I may be your closest instance, you may not be mine

5.2.3.1 Grading neighbors connectivity

Posting matrices are often used as a base on which to measure different indices of connectivity, e.g.,

- ◇ **hub** - instance x_i is at least the t^{th} neighbor of at least k instances.
  Definition used by ODIN: given a posting matrix A,x_i is a *hub* if it appears at least k times in the first t columns of A. Hence,x_i is an outlier if the opposite is true:

$$\tilde{o}(x_i) = \begin{cases} 1 & if \quad |i|i \in A_{\neq i,\leq t}| < k \\ 0 & otherwise \end{cases}$$

- ◇ **popularity** - instance x_i is on average the t^{th} neighbor of at least k instances
  Given a posting matrix A, x_i is an outlier if, on average, is not less than the $t-th$ neighbor of other instances:

$$\hat{o}(x_i) = \frac{\sum_{l=0,l\neq i}^{n-1} \sum_{j=0}^{n-1} \mathbb{1}\{a_{l,j} = x_i\} l}{n-1} > t.$$

  where $\mathbb{1}\{a_{l,j} = x_i\}$ is an indicator function that equals 1 if $a_{l,j} = x_i$ and 0 otherwise, representing the position in the posting list.
- ◇ **ostracism** - instance x_i is at worst t^{th} neighbor of other k instances

Connectivity and concentration can be approximated through similar structures: we go from *postings* matrix to distance matrix! To ease notation, we use a row-sorted distance matrix A_γ, so that row i holds increasing distances from instance x_i.

$$A = \begin{bmatrix} 0 & 2.28 & 0.16 & 0.21 \\ 2.21 & 0 & 1.21 & 3.91 \\ 0.16 & 1.21 & 0 & 0.76 \\ 0.21 & 3.91 & 0.76 & 0 \end{bmatrix}$$

$$A_\gamma = \begin{bmatrix} 0.16 & 0.21 & 2.28 \\ 1.21 & 2.28 & 3.91 \\ 0.16 & 0.76 & 1.21 \\ 0.21 & 0.76 & 3.91 \end{bmatrix}$$

A distance matrix A (top), and its row-sorted version A_γ (bottom). First column of 0s trimmed from A_γ.

5.2.4 Reach

An instance x has reach $\gamma^k(x)$ if the $k-th$ nearest neighbor is at distance γ^k, and average reach $\bar{\gamma}^k(x)$ if the average of $\{\gamma^1, \ldots, \gamma^k\}$ is $\bar{\gamma}^k(x)$.

Our row-sorted distance matrix A_γ is the *reach* matrix of the data! Indeed, A_γ defines both reach and average reach.

The reach of instance x_i is encoded in the row-sorted distance matrix A_γ as follows:

$$A_\gamma = \begin{bmatrix} \gamma^1(x_1) & \gamma^2(x_1) & \gamma^3(x_1) \\ \gamma^1(x_2) & \gamma^2(x_2) & \gamma^3(x_2) \\ \gamma^1(x_3) & \gamma^2(x_3) & \gamma^3(x_3) \end{bmatrix}$$

where $\gamma^j(x_i)$ denotes the j-th nearest neighbor of instance x_i.

For example, consider the following transformation:

$$A_\gamma \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \bar{\gamma}^1(x_1) & \bar{\gamma}^2(x_1) & \bar{\gamma}^3(x_1) \\ \bar{\gamma}^1(x_2) & \bar{\gamma}^2(x_2) & \bar{\gamma}^3(x_2) \\ \bar{\gamma}^1(x_3) & \bar{\gamma}^2(x_3) & \bar{\gamma}^3(x_3) \end{bmatrix}$$

A_γ explicitly encodes reach (A_γ itself) and average reach.

5.2.4.1 Reach ratio factor

Assumption: Inliers have lower reach than their neighbors. We formalize this in a reach (ratio factor:

$$\tilde{o}^k_{i,j} = \frac{\bar{\gamma}^k(x_i)}{\bar{\gamma}^k(x_j)}$$

which 1 is for pairs x_i, x_j with equal k-neighbors concentration, and > 1 for instances with different concentrations, x_i laying in a sparser area of the space.

Local outlier factor generalizes outlier factor by averaging the outlier factor over the neighbors of an instance:

$$\tilde{o}(x_i) = \sum_{x_j \in neigh(x_i)} \tilde{o}^k_{i,j}$$

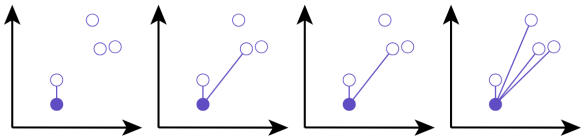


Figure 5.3: Neighbors at different k values. Local outlier factor respects the postings matrix, as it creates *clusters* of neighbors.

Connectivity outlier factor is based on the idea that outliers are poorly connected to their neighbors. It is defined as:

$$\tilde{o}(x_i) = \sum_{x_j \in connect\_neigh(x_i)} \tilde{o}^k_{i,j}$$

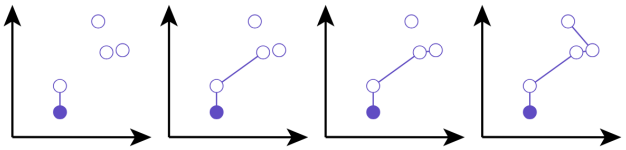


Figure 5.4: Neighbors with different k values. Connectivity Outlier Factor does not respects the posting matrix. Rather, it creates *chains* of neighbors.

k-NN outlier factor (KOF) replaces the average reach at k (denoted with $\bar{\gamma}^k$) with the maximum reach at k (denoted with $\hat{\gamma}^k$):

$$\tilde{o}(x_i) = \gamma^k(x_i)$$

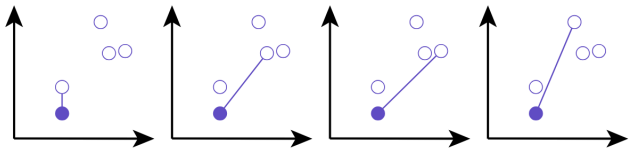


Figure 5.5: Neighbors at different k values

5.2.5 Concentration

We compute concentration on a two-radii approach:

- ◇ **concentration radius** ε: determines the hyperspheres $B(x_i, \varepsilon)$ estimating concentration $c^{\varepsilon}(x_i)$ of x_i within a radius ε
- ◇ **neighborhood radius** δ: proportional to ε, determines the neighborhood B_i of x_i as the instances laying within $B(x_i, \delta)$

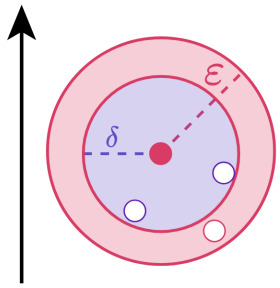


Figure 5.6: The two radii ε, δ: the former is used to estimate *concentration*, the latter to choose which neighbors to compare concentration against. **Note:** δ may also be larger than ε!

Locality	**Local**
Sensitive	Choice of neighborhood, connectivity parameter
Interpretable	Partial: can inspect what instances lead to different reaches

Table 5.2: Grading connectivity factors

5.2.6 Neighborhoods

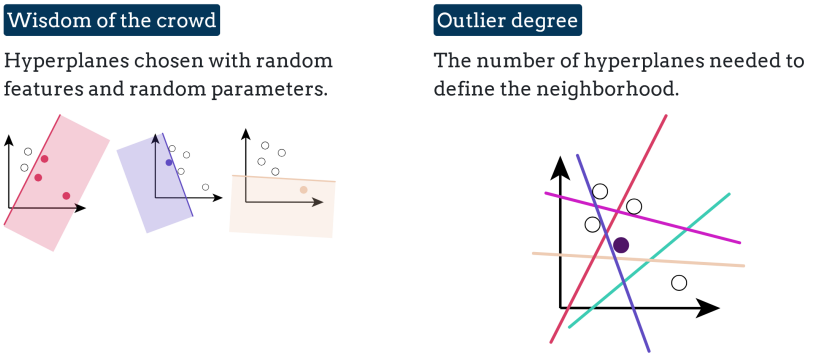


Figure 5.3: Faster and easier to compute hyperplane-based neighborhoods.

An **isolation tree** t is a random tree which randomly partitions the space into a set of blocks.

- ◇ Splits are sampled randomly
- ◇ Tree grows up to a predefined height, or until all leaves contain one instance

Outlier degree $\tilde{o}^t(x_i) = \frac{path(x_i,t)}{c}$, where c defines the average path length in the tree.

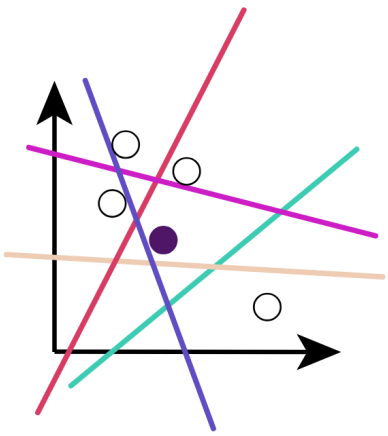


Figure 5.4: Isolation tree

An isolation forest T is comprised of several isolation trees, further sampling the hyperplane space. Outlier degree $\tilde{o}(x_i) = 2 - \frac{\sum_{t \in T} path(x_i,t)}{|T|c}$, where $|T|$ is the number of trees in the forest.

Locality	**Global** and **local**
Sensitive	Dataset noise can be interpreted as outlier
Interpretable	One of the most interpretable. Splits are induced by the tree, if the tree is univariate.

Table 5.3: Grading Isolation Forests