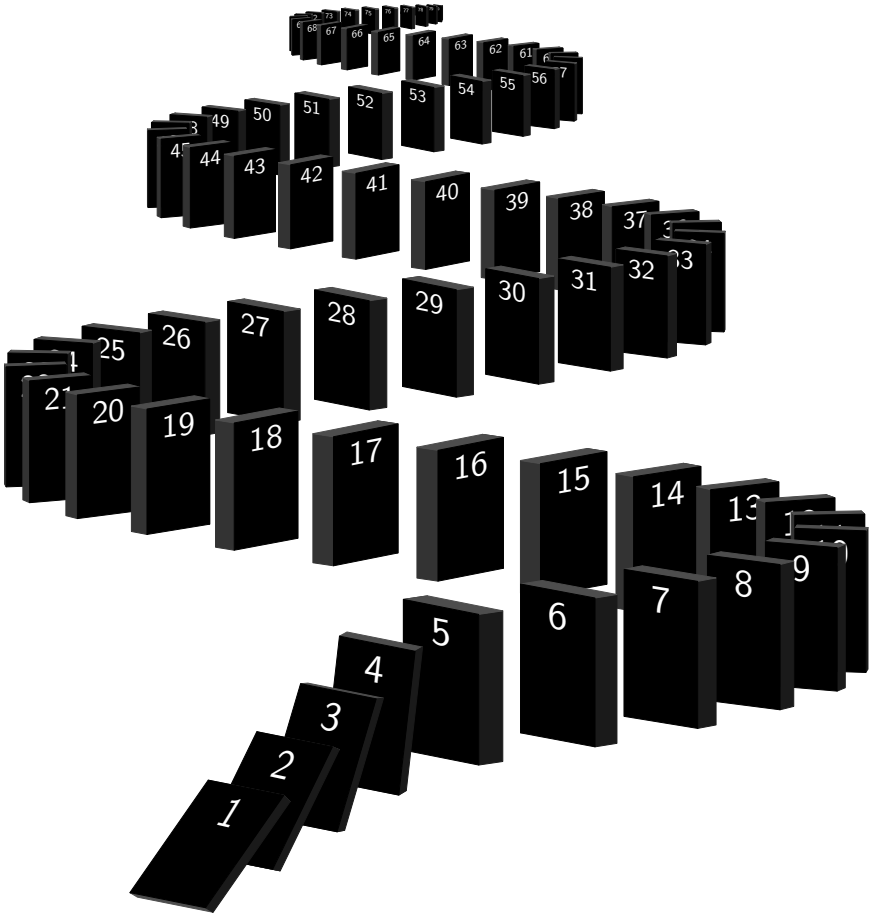
Data Mining

To find signals in data, we must learn to reduce the noise - not just the noise that resides in the data, but also the noise that resides in us. It is nearly impossible for noisy minds to perceive anything but noise in data.

STEPHEN FEW

Lucas Cabooter
Computer Science
Radboud University



CONTENTS

Page

Preface . . . 3
Chapter 1: Introduction to Data Mining . . . 4
1.1 What is Data Mining? . . . 4
1.2 Data Mining and Knowledge Discovery . . . 4
1.3 Data Mining Tasks . . . 5
Chapter 2: Data . . . 10
2.1 Attributes and Objects . . . 10
2.2 Types of Attributes . . . 10
2.3 Discrete, Continuous and Asymmetric Attributes . . . 12
2.4 Types of data sets . . . 13
2.4.1 Record data . . . 13
2.4.2 Graph data . . . 14
2.4.3 Ordered data . . . 15
2.5 Data quality and problems . . . 16
2.5.1 Measurement and Data Collection Errors . . . 16
2.5.2 Precision, Bias, and Accuracy . . . 16
2.5.3 Types of data quality problems . . . 17
2.6 Data Preprocessing . . . 18
2.6.1 Aggregation . . . 18
2.6.2 Sampling . . . 18
2.6.3 Dimensionality reduction . . . 19
2.6.4 Feature subset selection . . . 20
2.6.5 Feature creation . . . 21
2.6.6 Discretization and binarization . . . 21
2.6.7 Variable and attribute transformation . . . 23
2.7 Similarity and dissimilarity . . . 24
2.8 Similarity and Dissimilarity between simple attributes . . . 24
2.9 Dissimilarities between Data Objects . . . 25
2.9.1 Euclidean Distance . . . 25
2.9.2 Minkowski Distance . . . 25
2.9.3 Properties of a distance . . . 26
2.10 Similarity Measures for Binary Data . . . 27
2.10.1 Simple Matching Coefficient (SMC) and Jaccard coefficient . . . 27
2.10.2 Correlation . . . 28
2.10.3 Extended Jaccard Coefficient (Tanimoto Coefficient) . . . 28
2.10.4 Cosine Similarity . . . 28
Chapter 3: Data Exploration . . . 29
3.1 Summary Statistics . . . 29
3.1.1 Frequencies, Mode . . . 29
3.1.2 Percentiles . . . 29

3.1.3 Measures of Location: Mean and Median . . . 29
3.1.3 Measures of Spread: Range and Variance . . . 30
3.2 Visualization . . . 32
3.2.1 General Concepts of Visualization . . . 33
3.2.2 Visualization Techniques . . . 35
3.2.3 Visualizing Higher-Dimensional Data . . . 38
3.2.4 Do's and Don'ts of visualization . . . 39
Chapter 4: Classification: Basic Concepts, Decision Trees, and Model Evaluation . . . 39
4.1 Classification . . . 40
4.2 Solving a classification problem and the structure of a model . . . 41
4.3 Decision tree . . . 42
4.3.1 How does a decision tree work . . . 42
4.3.2 Building a decision tree . . . 43
4.3.3 Hunt's algorithm . . . 44
4.3.4 Splitting attributes based on their types . . . 46
4.3.5 Measures for selecting the best split . . . 48
Chapter 8: Cluster Analysis: Basic Concepts and Algorithms . . . 53
Appendix A: Linear Algebra . . . 54
A.1 Some theory about vectors and matrices . . . 54
A.1.1 Vectors spaces . . . 54
A.1.2 Basis vectors,linear combinations and span . . . 54
A.2 Working with matrices . . . 54
A.2.1 Dot product, transposing and orthogonality . . . 55
A.2.2 Length, angle and projection . . . 55
A.2.3 Matrix multiplication . . . 56
A.2.4 Inverse of a matrix . . . 56
A.2.5 Finding the Eigenvalues and Eigenvectors . . . 56
A.2.6 Diagonalizing a matrix . . . 57
A.2.7 Normalizing a vector . . . 58
A.2.8 Rank of a matrix . . . 58
Appendix B: Dimensionality Reduction . . . 59
B.1 Eigenvalue Decomposition . . . 59
B.2 Singular Value Decomposition . . . 60
B.3 Principal Component Analysis . . . 61
Appendix C: Probability and Statistics . . . 62
Appendix D: Regression . . . 63
Appendix E: Optimization . . . 64
E-Chapter I: Extra resources . . . 65
E-Chapter I.1: Books . . . 65
E-Chapter I.2: YouTube channels and Videos . . . 65
E-Chapter I.3: Articles and websites . . . 65
E-Chapter II: Undiscussed topics in Chapter 1-4 for the Midterm Exam . . . 66
E-Chapter II.1: Extra's from Chapter 5 . . . 66
E-Chapter II.1.1: Overfitting and Underfitting . . . 66
E-Chapter II.1.2: Post-pruning and Pre-pruning . . . 67
E-Chapter II.1.3: Evaluating the performance of a classifier . . . 68
E-Chapter II.1.4: Class imbalance metrics . . . 69
E-Chapter II.1.5: The Receiver Operating Characteristic Curve . . . 71

Preface

This document will contain the summary for the course Data Mining (NWI-IBI008). This document will contain a *detailed* summary of everything we have to know for mid-term (first 7 weeks) and the end-term (week 8 through 14). The summaries are based on the slides given in the lecture, and the book "Introduction to Data Mining" by Tan, Steinbach and Kumar.

One note is that this summary will be a *lot* more detailed than what is presented in the slides and during the lectures. So in theory you could say that this is more a summary of the book rather than that of the lectures, but the main reason for making the summary this way is that not every topic is as well explained in the lecture as it is in the book or somewhere else online. Hence, not everything in the summary is important to know, but it's always good to have read through it to know what certain things might mean.

Some things are more explained more thorough than others due to their importance. In sections 3.2.2 *Visualization Techniques* and 3.2.3 *Visualizing Higher-Dimensional Data* for instance, I won't go into too much detail and leave the more detailed research to the student themselves.

In this summary there will also be chapters on *Linear Algebra* (Appendix A), *Dimensionality Reduction* (Appendix B), *Probability and Statistics* (Appendix C), *Regression* (Appendix D), *Optimization* (Appendix E) and extra chapters (*E-Chapter* ...) at the end to help with the exams and to have an overview of the fundamentals discussed in this course.

Further resources used for reference within this document are:

Solution manual for the exercises in the book (instructor manual):
https://www-users.cse.umn.edu/~kumar001/dmbook/sol.pdf
PPT's about the chapters within the book:
https://www-users.cse.umn.edu/~kumar001/dmbook/index.php

And other online sources like YouTube and articles. One of the E-chapters will be dedicated to helpful resources about a multitude of topics. I had to research everything myself since I'm not a magician. so hopefully these resources can help.

Chapter 1: Introduction to Data Mining

1.1 What is Data Mining?

Data mining is the process of automatically discovering useful information in large data repositories. Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown. They also provide capabilities to predict the outcome of a future observation, such as predicting whether a newly arrived customer will spend more than $100 at a department store.
Looking up individual records using a DBMS is for instance not Data Mining, but **Information retrieval**.

1.2 Data Mining and Knowledge Discovery

Data mining is an integral part of **knowledge discovery in databases (KDD)**, which is the overall process of converting raw data into useful information (Figure 1).

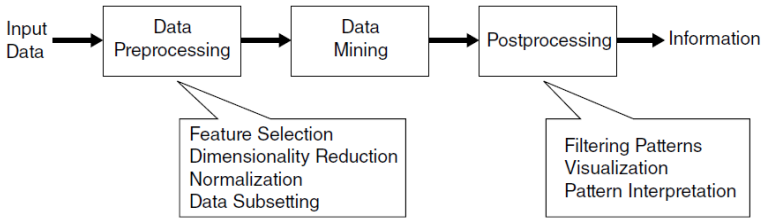


Figure 1: The Process of Knowledge Discovery in Databases (KDD)

Traditional techniques may be unsuitable due to data that is:
- Large-scale
- High dimensional
- Heterogeneous
- Complex
- Distributed

Other fields which make use of data science or which are closely related are for instance parallel computing and distributed computing.

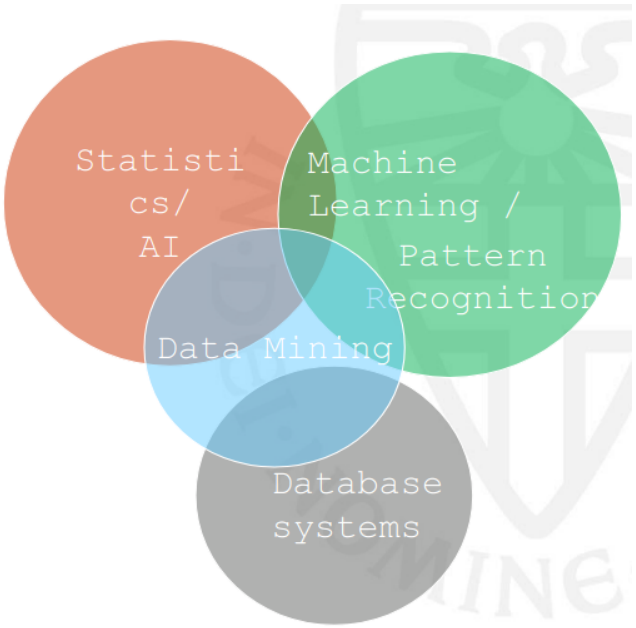


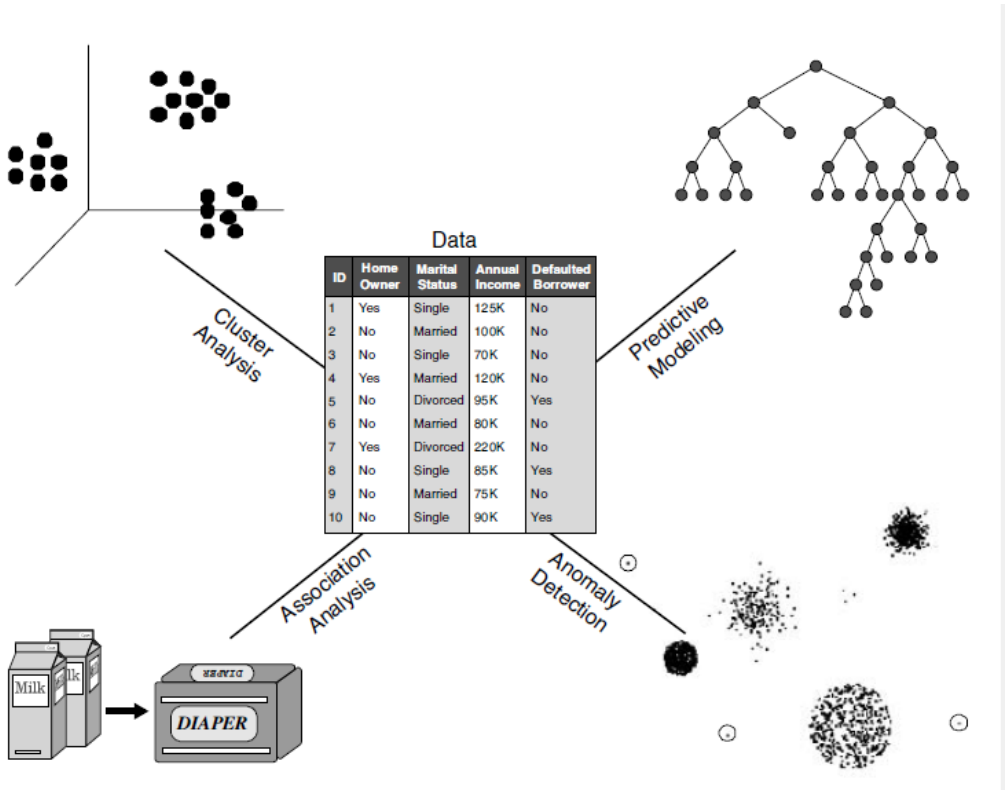
Figure 2: The overlap between fields

1.3 Data Mining Tasks

Data mining tasks are divided into two major categories:

1. Predictive tasks:
The objective of these tasks is to predict the value of a particular attribute based on the values of other attributes. The attribute to be predicted is commonly known as the target or dependent variable, while the attributes used for making the prediction are known as the explanatory or independent variables.
In short: use some variables to predict unknown or future values of other variables.

2. Descriptive tasks:
The objective is to derive patterns (correlations, trends, clusters, trajectories, and anomalies) that summarize the underlying relationships in data. Descriptive data mining tasks are often exploratory in nature and frequently require post-processing techniques to validate and explain the results.
In short: find human-interpretable patterns that describe the data.



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	80K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 3: Four of the core data mining tasks

Technique 1: Predictive modeling (Predictive)
Predictive modeling refers to the task of building a model for the target variable as a function of the explanatory variables.
There are two types of predictive modeling tasks:

1. Classification, which is used for discrete target variables.
    - Predicting whether a Web user will make a purchase at an online bookstore is a classification task because the target variable is binary-valued.
    - Classifying credit card transactions as legitimate or fraudulent. The transaction is either legitimate or fraudulent, meaning it's a non-continuous task (target variable is binary-valued).
    - Classifying land covers (water bodies, urban areas, forests, etc.) using satellite data. A land cover is one of the many types of terrain, meaning the target variable is discrete, meaning is a classification task.
2. Regression refers to the prediction of a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency. Or in short: regression is used for continuous target variables.
    - Predicting sales amounts of a new product based on advertising expenditure.
    - Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
    - Forecasting the future price of a stock (this is a regression task because price is a continuous-valued attribute).

The goal of both tasks is to learn a model that minimizes the error between the predicted and true values of the target variable. Predictive modeling in general can be used to identify customers that will respond to a marketing campaign, or if given a flower with a certain petal width and petal length, to decide which plant family it belongs to (see Figure 4).

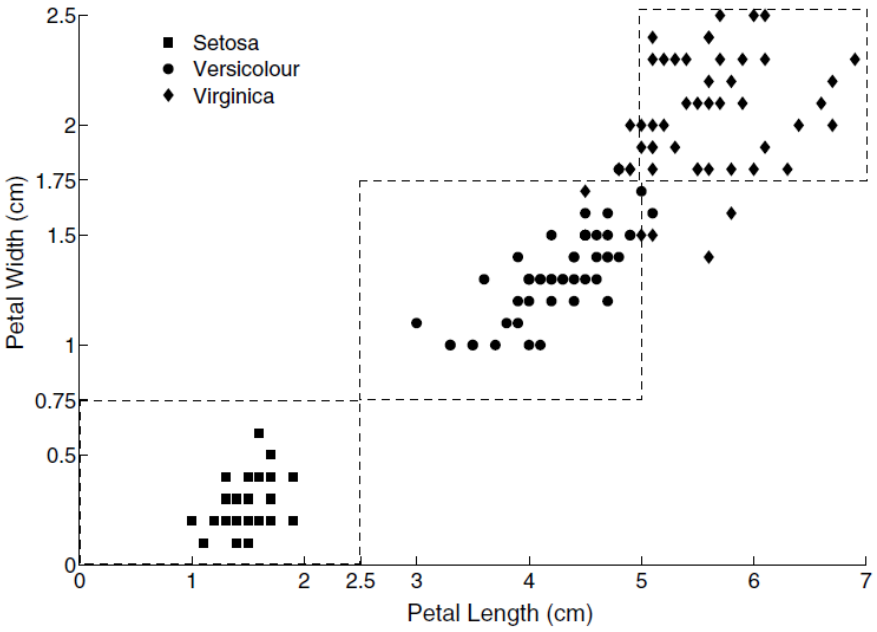


Figure 4: Predicting the type of flower based on plant families

Technique 2: Association Analysis (Descriptive)(Market Basked Analysis)

Association analysis is used to discover patterns that describe strongly associated features in the data.
So if we have a set of records, each of which contain some number of times from a given collection, we can produce dependency rules which will predict the occurrence of an item based on the occurrences of other items.
So for instance if we have the following information about transactions made in a supermarket:

Transaction ID	Items
1	{Bread, Butter, Diapers, Milk}
2	{Coffee, Sugar, Cookies, Salmon}
3	{Bread, Butter, Coffee, Diapers, Milk, Eggs}
4	{Bread, Butter, Salmon, Chicken}
5	{Eggs, Bread, Butter}
6	{Salmon, Diapers, Milk}
7	{Bread, Tea, Sugar, Eggs}
8	{Coffee, Sugar, Chicken, Eggs}
9	{Bread, Diapers, Milk, Salt}
10	{Tea, Eggs, Cookies, Diapers, Milk}

Figure 5: Table with transaction information (set of records)

If we apply association analysis we find out that in multiple cases, if diapers are purchased, milk also gets purchased. These patterns are typically represented in the form of implication rules or feature subsets (see Figure 6).

$$\{\text{Diapers}\} \longrightarrow \{\text{Milk}\}$$

Figure 6: Pattern in the form of an implication rule

Some other applications of association analysis are:

- *Market-basket analysis*: rules are used for sales promotion, shelf management, and inventory management. This application was also described above, meaning something like sales promotion can be applied to diapers and milk, meaning people will be more likely to keep buying it.
- *Telecommunication alarm diagnosis*: Rules are used to find combination of alarms that occur together frequently in the same time period. This can be used to determine what the best combination of alarms when there is country-wide danger.
- *Medical Informatics*: Rules are used to find combination of patient symptoms and test results associated with certain diseases (see Figure 7).

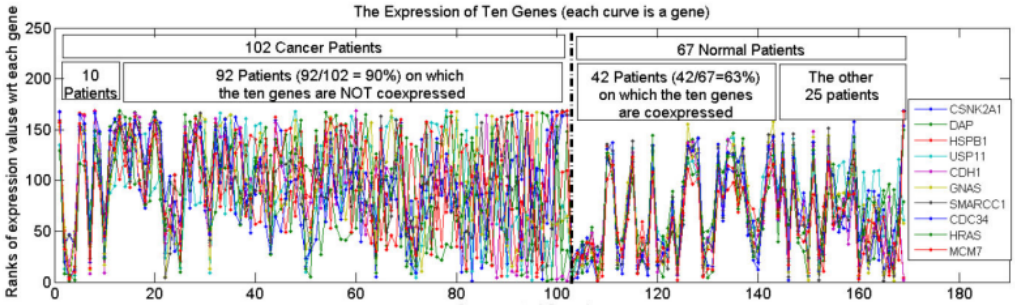


Figure 7: An example of a subspace differential coexpression pattern from a lung cancer dataset

Technique 3: Cluster Analysis (Descriptive)

Cluster analysis seeks to find groups of closely related observations so that observations that belong to the same cluster are more similar to each other than observations that belong to other clusters. Clustering is mainly used to reduce/compress the size of large data sets.

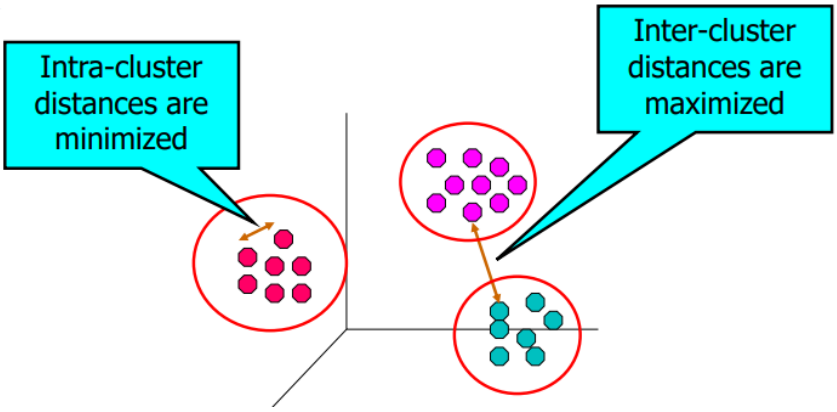


Figure 8: Intra-clusters and inter-clusters

Intra-cluster distances is the distance between members of a cluster. This metric gives a sense of how well the distance measure was able to bring the items together. A good distance measure will return a small distance between objects that are similar and produce clusters that are tighter, and that are therefore more reliably discriminated from one another.
Inter-cluster distances is the distance between two different cluster sets and thus the distance is bigger than that of a intra-cluster.

Applications of cluster analysis are:

- *Market segmentation:* subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix (group a set of related customers).
- *Document Clustering*: find groups of documents that are similar to each other based on the important terms appearing in them. The collection of news articles shown in Figure 8 can be grouped based on their respective topics. Each article is represented as a set of word-frequency pairs (w, c), where w is a word and c is the number of times the word appears in the article. There are two natural clusters in the data set. The first cluster consists of the first four articles, which correspond to news about the economy, while the second cluster contains the last four articles, which correspond to news about health care. A good clustering algorithm should be able to identify these two clusters based on the similarity between words that appear in the articles.

Article	Words
1	dollar: 1, industry: 4, country: 2, loan: 3, deal: 2, government: 2
2	machinery: 2, labor: 3, market: 4, industry: 2, work: 3, country: 1
3	job: 5, inflation: 3, rise: 2, jobless: 2, market: 3, country: 2, index: 3
4	domestic: 3, forecast: 2, gain: 1, market: 2, sale: 3, price: 2
5	patient: 4, symptom: 2, drug: 3, health: 2, clinic: 2, doctor: 2
6	pharmaceutical: 2, company: 3, drug: 2, vaccine: 1, flu: 3
7	death: 2, cancer: 4, drug: 3, public: 4, health: 3, director: 2
8	medical: 2, cost: 3, increase: 2, patient: 2, health: 3, care: 1

Figure 9: Collection of news articles

Technique 4: Anomaly detection (Predictive)

Anomaly detection is the task of identifying observations whose characteristics are significantly different from the rest of the data. Such observations are known as anomalies or outliers. The goal of an anomaly detection algorithm is to discover the real anomalies and avoid falsely labeling normal objects as anomalous. In other words, a good anomaly detector must have a high detection rate and a low false alarm rate. Applications of anomaly detection include the detection of fraud, network intrusions, unusual patterns of disease, and ecosystem disturbances.

Chapter 2: Data

2.1 Attributes and Objects

Data is a collection of data objects and their attributes (as seen in Figure 10).
An *attribute* is a property or characteristic of an object, for instance:

- Eye color, temperature (the columns)
- An attribute is also known as a variable, field, dimension, or feature.

A collection of attributes describe an *object* (the information in the columns).
An object is also known as record, point, case, entity, or sample.

Attribute values are numbers or symbols assigned to an attribute for a particular object.
There is a distinction between attributes and attribute values, since the same attribute can be mapped to different attribute values (e.g. height can be measure in feet or meters). Different attributes can also be mapped to the same set of values (e.g. attribute values for ID and age are integers).

2.2 Types of Attributes

There are 4 different types of attributes:

- Nominal For instance: ID numbers, eye color, zip codes.
  Distinctness: $=, \neq$
- Ordinal For instance: ranking (e.g. on a scale of 1-10), grades, height tall, medium, short.
  Order: $<, >$
- Interval For instance: calendar dates, temperatures in Celsius or Fahrenheit.
  Addition: $+, -$
- Ratio For instance, length, counts, elapsed time, temperature in Kelvin.
  Multiplication: $*, /$

An overview of the types of attributes is given in Figure 11. The important thing to take away is that Ratio can perform all properties/operations, Interval those of itself, Ordinal and Nominal, but not those of Ratio.

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 10: Attribute and Object example

Attribute Type		Description	Examples	Operations
Categorical (Qualitative)	Nominal	The values of a nominal attribute are just different names; i.e., nominal values provide only enough information to distinguish one object from another. $(=, \neq)$	zip codes, employee ID numbers, eye color, gender	mode, entropy, contingency correlation, χ^2 test
	Ordinal	The values of an ordinal attribute provide enough information to order objects. $(<, >)$	hardness of minerals, $\{good, better, best\}$, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric (Quantitative)	Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. $(+, -)$	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, electrical current	geometric mean, harmonic mean, percent variation

Figure 11: Attribute overview

Nominal and ordinal attributes are collectively referred to as categorical or qualitative attributes. As the name suggests, *qualitative attributes*, such as employee ID, lack most of the properties of numbers. Even if they are represented by numbers, i.e., integers, they should be treated more like symbols. The remaining two types of attributes, interval and ratio, are collectively referred to as *quantitative* or *numeric attributes*. Quantitative attributes are represented by numbers and have most of the properties of numbers. Note that quantitative attributes can be integer-valued or continuous.
The types of attributes can also be described in terms of transformations that do not change the meaning of an attribute. These transformations are also known as permissible transformations (described in Figure 12).

Attribute Type		Transformation	Comment
Categorical (Qualitative)	Nominal	Any one-to-one mapping, e.g., a permutation of values	If all employee ID numbers are reassigned, it will not make any difference.
	Ordinal	An order-preserving change of values, i.e., $new\_value = f(old\_value)$, where f is a monotonic function.	An attribute encompassing the notion of good, better, best can be represented equally well by the values $\{1, 2, 3\}$ or by $\{0.5, 1, 10\}$.
Numeric (Quantitative)	Interval	$new\_value = a * old\_value + b$, a and b constants.	The Fahrenheit and Celsius temperature scales differ in the location of their zero value and the size of a degree (unit).
	Ratio	$new\_value = a * old\_value$	Length can be measured in meters or feet.

Figure 12: Permissible Transformations

For example, the meaning of a length attribute is unchanged if it is measured in meters instead of feet. The statistical operations that make sense for a particular type of attribute are those that will yield the same results when the attribute is transformed using a transformation that preserves the attribute's meaning. To illustrate, the average length of a set of objects is different when measured in meters rather than in feet, but both averages represent the same length. Figure 12 shows the permissible (meaning-preserving) transformations for the four attribute types of Figure 11.

Temperature provides a good illustration of some of the concepts that have been described. First, temperature can be either an interval or a ratio attribute, depending on its measurement scale. When measured on the Kelvin scale, a temperature of 2° is, in a physically meaningful way, twice that of a temperature of 1°. This is not true when temperature is measured on either the Celsius or Fahrenheit scales, because, physically, a temperature of 1 ° Fahrenheit (Celsius) is not much different than a temperature of 2° Fahrenheit (or Celsius). The problem is that the zero points of the Fahrenheit and Celsius scales are, in a physical sense, arbitrary, and therefore, the ratio of two Celsius or Fahrenheit temperatures is not physically meaningful.
Basically what it means is that Ratio scales are like Interval scales except they have true zero points. Celsius and Fahrenheit can be negative, meaning they **don't have** a *true zero point*, while Kelvin can't be negative, meaning it **has** a *true zero point*.
A good article that explains when age is a rational attribute and when it is an ordinal attribute: Is age interval, ratio or ordinal?.

2.3 Discrete, Continuous and Asymmetric Attributes

Discrete attributes

A discrete attribute has a finite or countably infinite set of values.
Examples are ZIP codes, counts, or the set of words in a collection of documents. These attributes are often represented as integer variables and note that *binary attributes* are a special case of discrete attributes (0 or 1).

Continuous attributes

A continuous attribute has a real number as its attribute value.
Examples are temperature, height, or weight. Practically, real values can only be measured and represented using a finite number of digits. Continuous attributes are typically represented as floatingpoint (e.g. 7.10) variables.

Asymmetric attributes

For asymmetric attributes, only presence — *a non-zero attribute value* — is regarded as important.
Consider a data set where each object is a student and each attribute records whether or not a student took a particular course at a university. For a specific student, an attribute has a value of 1 if the student took the course associated with that attribute and a value of 0 otherwise. Because students take only a small fraction of all available courses, most of the values in such a data set would be 0. Therefore, it is more meaningful and more efficient to focus on the non-zero values.
It is also possible to have discrete or continuous asymmetric features. For instance, if the number of credits associated with each course is recorded, then the resulting data set will consist of asymmetric discrete or continuous attributes.

2.4 Types of data sets

There are a few main types of data sets discussed in the book. There are more types, but the following ones are the main ones/the most important ones.

2.4.1 Record data

Data that consists of a collection of records, each of which consists of a fixed set of attributes (e.g. Figure 10).

- Data Matrix
  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 a data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute.

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Figure 13: Data Matrix

- Document Data
  Each document becomes a 'term' vector. Each term is a component (attribute) of the vector. The value of each component is the number of times the corresponding term occurs in the document.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

Figure 14: Document Data

- Transaction Data
  A special type of data, where each transaction involves a set of items.
  For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items. You can represent transaction data as record data.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Figure 15: Transaction Data

2.4.2 Graph data

We have two types of graphs depending on their use case:
(1) The graph captures relationships among data objects.
(2) The data objects themselves are represented as graphs.

- Data with Relationships among Objects (World Wide Web)
  The relationships among objects frequently convey important information. In such cases, the data is often represented as a graph. In particular, the data objects are mapped to nodes of the graph, while the relationships among objects are captured by the links between objects and link properties, such as direction and weight. An example is the links between web pages on the WWW (Figure 16).

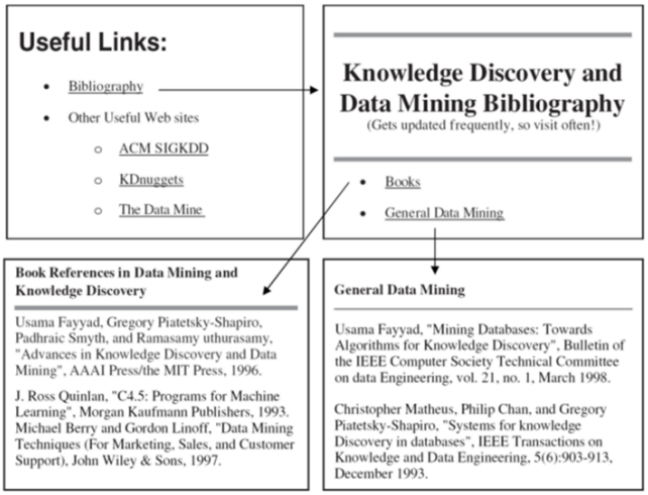


Figure 16: Data with relationships among objects

- Data with Objects That Are Graphs (Molecular Structures)
  If objects have structure, that is, the objects contain sub-objects that have relationships, then such objects are frequently represented as graphs. For example, the structure of chemical compounds can be represented by a graph, where the nodes are atoms and the links between nodes are chemical bonds (Figure 17).

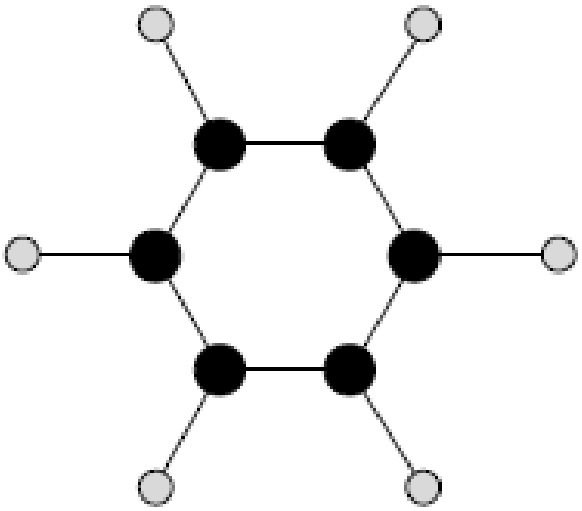


Figure 17: Data with objects that are graphs (Benzene molecule)

2.4.3 Ordered data

For some types of data, the attributes have relationships that involve order in time or space. Here we discuss the main types of ordered data.

- Spatial data
  Some objects have spatial attributes, such as positions or areas, as well as other types of attributes (Figure 18). An example of spatial data is weather data (precipitation, temperature, pressure) that is collected for a variety of geographical locations. An important aspect of spatial data is spatial auto-correlation. I.e., objects that are physically close tend to be similar in other ways as well. Thus, two points on the Earth that are close to each other usually have similar values for temperature and rainfall.

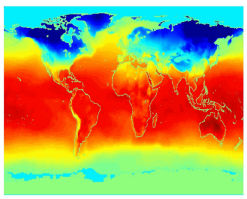


Figure 18: Average temperature in the world in January of land and ocean

- Sequential Data/Temporal Data
  Sequential data, also referred to as temporal data, can be thought of as an extension of record data, where each record has a time associated with it. Consider a retail transaction data set that also stores the time at which the transaction took place.

Time	Customer	Items Purchased
t1	C1	A, B
t2	C3	A, C
t2	C1	C, D
t3	C2	A, D
t4	C2	E
t5	C1	A, E

Customer	Time and Items Purchased
C1	(t1: A,B) (t2:C,D) (t5:A,E)
C2	(t3: A, D) (t4: E)
C3	(t2: A, C)

Figure 19: Sequential transaction data

  Time series data is a special type of sequential data in which each record is a time series, i.e., a series of measurements taken over time.

- Genetic Sequence Data
  Sequence data consists of a data set that is a sequence of individual entities, such as a sequence of words or letters. It is quite similar to sequential data, except that there are no time stamps; instead, there are positions in an ordered sequence. For example, the genetic information of plants and animals can be represented in the form of sequences of nucleotides that are known as genes. Many of the problems associated with genetic sequence data involve predicting similarities in the structure and function of genes from similarities in nucleotide sequences.

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

Figure 20: Genetic sequence data

2.5 Data quality and problems

2.5.1 Measurement and Data Collection Errors

The term measurement error refers to any problem resulting from the measurement process. A common problem is that the value recorded differs from the true value to some extent. For continuous attributes, the numerical difference of the measured and true value is called the error. The term *data collection error* refers to errors such as omitting data objects or attribute values, or inappropriately including a data object. For example, a study of animals of a certain species might include animals of a related species that are similar in appearance to the species of interest. Both measurement errors and data collection errors can be either systematic or random.

2.5.2 Precision, Bias, and Accuracy

In statistics and experimental science, the quality of the measurement process and the resulting data are measured by precision and bias.
We provide the following definitions and we assume that we make repeated measurements of the same underlying quantity and use this set of values to calculate a mean (average) value that serves as our estimate of the true value.

Precision: The closeness of repeated measurements (of the same quantity) to one another.

Bias: A systematic variation of measurements from the quantity being measured.
Precision is often measured by the standard deviation of a set of values, while bias is measured by taking the difference between the mean of the set of values and the known value of the quantity being measured. Bias can only be determined for objects whose measured quantity is known by means external to the current situation.

Suppose that we have a standard laboratory weight with a mass of 1g and want to assess the precision and bias of our new laboratory scale. We weigh the mass five times, and obtain the following five values: 1.015, 0.990, 1.013, 1.001, 0.986. The mean of these values is 1.001, and hence, the bias is 0.001. The precision, as measured by the standard deviation, is 0.013.

It is common to use the more general term, accuracy, to refer to the degree of measurement error in data.

Accuracy: The closeness of measurements to the true value of the quantity being measured.

Accuracy depends on precision and bias, but since it is a general concept, there is no specific formula for accuracy in terms of these two quantities. One important aspect of accuracy is the use of significant digits. The goal is to use only as many digits to represent the result of a measurement or calculation as are justified by the precision of the data. For example, if the length of an object is measured with a meter stick whose smallest markings are millimeters, then we should only record the length of data to the nearest millimeter. The precision of such a measurement would be ± 0.5mm.

2.5.3 Types of data quality problems

There are a few main data quality problems which will be discussed here:

- **Noise**
  Noise in data means that for objects, noise is an *extraneous object* (existing on or coming from the outside). For attributes, noise refers to the modification of original values. For instance the distortion of a person's voice when talking on a poor phone or "snow" on a television screen.

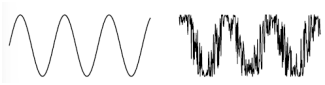


Figure 21: A function with and without noise

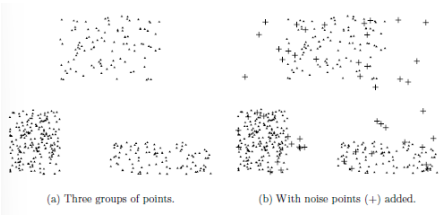


Figure 22: Noise in a spatial context

  Data errors may be the result of a more deterministic phenomenon, such as a streak in the same place on a set of photographs. Such deterministic distortions of the data are often referred to as artifacts.

- **Outliers**
  Outliers are data object with characteristics that are considerably different than most of the other objects in the data set (outliers can be useful unline noise).
  - **Case 1:** Outliers are noise that interferes with data analysis
  - **Case 2:** Outliers are the goal of our analysis (e.g. credit card fraud, intrusion detection (anomaly detection)).

- **Missing values**
  Reasons for missing values in a data set can be:
  - Information is not collected (e.g. people decline to give their age and weight).
  - Attributes may not be applicable to all cases (e.g. annual income for children).

  The way to handle these missing values can be done by:
  - Eliminate data objects or attributes.
  - Estimate missing values (e.g. time series of temperature).
  - Ignore the missing value during analysis.

- **Duplicate data**
  Data set may include data objects that are duplicates, or almost duplicates of one another. This is a major issue when merging data from heterogeneous sources, since then there is a very high chance some data is duplicated, and thus not useful. For example when a person has multiple email addresses.
  Sometimes you don't want to clean out duplicate data though, for instance medical records are good to keep, even if they are the same a year later.

- **Inconsistent values**
  Data can contain inconsistent values. Consider an address field, where both a zip code and city are listed, but the specified zip code area is not contained in that city. It may be that the individual entering this information transposed two digits, or perhaps a digit was misread when the information was scanned. Once an inconsistency has been detected, it is sometimes possible to correct the data. A product code may have "check" digits, or it may be possible to double-check a product code against a list of known product codes, and then correct the code if it is incorrect, but close to a known code. The correction of an inconsistency requires additional or redundant information.

2.6 Data Preprocessing

Data preprocessing is a broad area and consists of a number of different strategies and techniques that are interrelated in complex ways.
Roughly speaking, these items fall into two categories: selecting data objects and attributes for the analysis or creating/changing the attributes. In both cases the goal is to improve the data mining analysis with respect to time, cost, and quality. Details are provided in the following sections.

- **2.6.1 Aggregation**
  Aggregation is combining two or more attributes (or objects) into a single attribute (or object).
  The purpose of this is to reduce the number of attributes or objects and to make the data more "stable"- aggregated data tends to have less variability.
  An example could be to replace all the transactions of a single store with a single store-wide transaction.

- **2.6.2 Sampling**
  Sampling is a commonly used approach for selecting a subset of the data objects to be analyzed. In statistics, it has long been used for both the preliminary investigation of the data and the final data analysis. In some cases, using a *sampling algorithm* can reduce the data size to the point where a better, but more expensive algorithm can be used. So basically, sampling is the main technique employed for data reduction. *Sampling* is typically used in data mining because processing the entire set of data of interest is too expensive or time consuming.
  In turn, **a sample is representative** if it has approximately the same property (of interest) as the original set of data. If the mean (average) of the data objects is the property of interest, then a sample is representative if it has a mean that is close to that of the original data.

  There are a few main types of sampling:

  - 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 and 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.
  - Progressive sampling
    The proper sample size can be difficult to determine, so adaptive or progressive sampling schemes are sometimes used. These approaches start with a small sample, and then increase the sample size until a sample of sufficient size has been obtained. While this technique eliminates the need to determine the correct sample size initially, it requires that there be a way to evaluate the sample to judge if it is large enough.

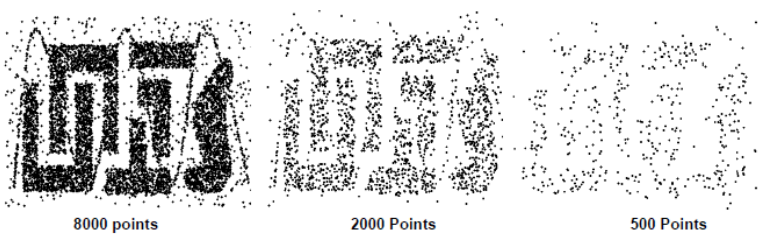


Figure 23: How visibility of a pattern changes depending on the sample size

- **2.6.3 Dimensionality reduction**
  The term dimensionality reduction is often reserved for those techniques that reduce the dimensionality of a data set by creating new attributes that are a combination of the old attributes. The reduction of dimensionality by selecting new attributes that are a subset of the old is known as *feature subset selection* or *feature selection* (both discussed later on).
  The benefits of dimensionality reduction are:

  – The first benefit is that many data mining algorithms work better if the dimensionality, the number of attributes in the data, is lower. This is partly because dimensionality reduction can eliminate irrelevant features and reduce noise and partly because of the curse of dimensionality (also explained later).
  – The second benefit is that a reduction of dimensionality can lead to a more understandable model because the model may involve fewer attributes. Also, dimensionality reduction may allow the data to be more easily visualized.

The curse of dimensionality refers to the phenomenon that many types of data analysis become significantly harder as the dimensionality of the data increases. Specifically, as dimensionality increases, the data becomes increasingly sparse in the space that it occupies.

Some of the most common approaches for dimensionality reduction, particularly for continuous data, use techniques from linear algebra to project the data from a high-dimensional space into a lower-dimensional space. Principal Components Analysis (PCA) is a linear algebra technique for continuous attributes that finds new attributes (principal components) that (1) are linear combinations of the original attributes, (2) are orthogonal (perpendicular) to each other, and (3) *capture the maximum amount of variation in the data* (the mathematics and idea behind PCA will be discussed in Chapter 3: Linear Algebra).

Singular Value Decomposition (SVD) is a linear algebra technique that is related to PCA and is also commonly used for dimensionality reduction (the mathematics and idea behind SVD will be discussed in Chapter 3: Linear Algebra).

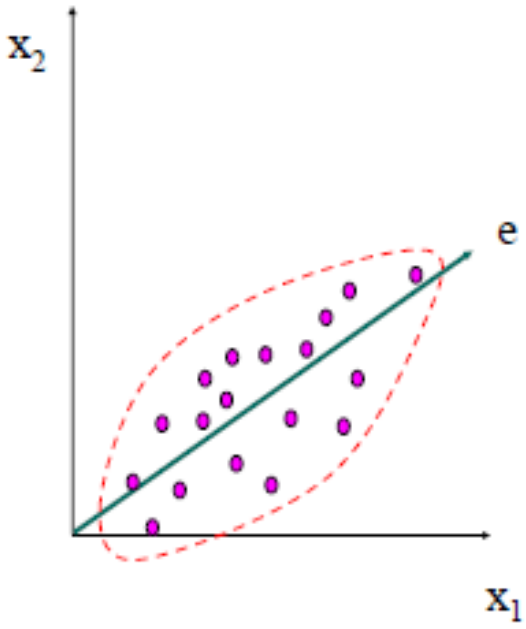


Figure 24: A visualization of PCA

- **2.6.4 Feature subset selection**
  Another way to reduce the dimensionality is to use only a subset of the features. While it might seem that such an approach would lose information, this is not the case if redundant and irrelevant features are present. Redundant features duplicate much or all of the information contained in one or more other attributes. For example, the purchase price of a product and the amount of sales tax paid contain much of the same information. Irrelevant features contain almost no useful information for the data mining task at hand. For instance, students' ID numbers are irrelevant to the task of predicting students' grade point averages. Redundant and irrelevant features can reduce classification accuracy and the quality of the clusters that are found.

  The number of subsets involving n attributes is 2^n, such an approach is impractical in most situations and alternative strategies are needed. There are three standard approaches to feature selection: *embedded*, *filter*, and *wrapper*.

  - Embedded approaches
    Feature selection occurs naturally as part of the data mining algorithm. Specifically, during the operation of the data mining algorithm, the algorithm itself decides which attributes to use and which to ignore.
  - Filter approaches
    Features are selected before the data mining algorithm is run, using some approach that is independent of the data mining task. For example, we might select sets of attributes whose pairwise correlation is as low as possible.
  - Wrapper approaches
    These methods use the target data mining algorithm as a black box to find the best subset of attributes, in a way similar to that of the ideal algorithm described above, but typically without enumerating all possible subsets.

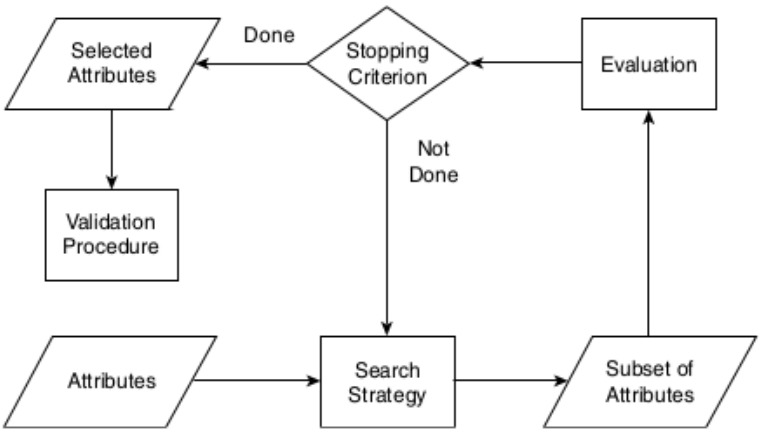


Figure 25: Flowchart of a feature subset selection process

- **2.6.5 Feature creation**
  It is frequently possible to create, from the original attributes, a new set of attributes that captures the important information in a data set much more effectively. Furthermore, the number of new attributes can be smaller than the number of original attributes, allowing us to reap all the previously described benefits of dimensionality reduction. For this there are three general methodologies: *feature extraction, mapping the data to a new space, feature construction.*

  – Feature extraction
    The creation of a new set of features from the original raw data is known as **feature extraction**. An example is to consider a set of photographs, where each photograph is to be classified according to whether or not it contains a human face. The raw data is a set of pixels, and as such, is not suitable for many types of classification algorithms. However, if the data is processed to provide higher-level features, such as the presence or absence of certain types of edges and areas that are highly correlated with the presence of human faces, then a much broader set of classification techniques can be applied to this problem.

  – Mapping the data to a new space
    A totally different view of the data can reveal important and interesting features. Consider, for example, time series data, which often contains periodic patterns. If there is only a single periodic pattern and not much noise, then the pattern is easily detected. If, on the other hand, there are a number of periodic patterns and a significant amount of noise is present, then these patterns are hard to detect. Such patterns can, nonetheless, often be detected by applying a Fourier transform (example on page 58 of the book or this video) to the time series in order to change to a representation in which frequency information is explicit.

  – Feature construction
    Sometimes the features in the original data sets have the necessary information, but it is not in a form suitable for the data mining algorithm. In this situation, one or more new features constructed out of the original features can be more useful than the original features (Video about feature construction).

- **2.6.6 Discretization and binarization**
  If we want to transform both continuous and discrete attributes into one or morebinary attributes, we can binarization. For association problems, it is therefore necessary to introduce one binary attribute for each categorical value, as in Figure 26.

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

Figure 26: Conversion of a categorical attribute into five asymmetric binary attributes (hence there only being 0 and 1's)

A simple technique to binarize a categorical attribute is the following: If there are m categorical values, then uniquely assign each original value to an integer in the interval $[0, m-1]$. If the attribute is ordinal, then order must be maintained by the assignment.

Another example is the following Figure 27, where the attributes x_2 and x_3 are correlated because information about the *good* value is encoded using both attributes. If the number of resulting attributes is too large, then there are techniques discussed later which can be used to reduce the number of categorical values before binarization.

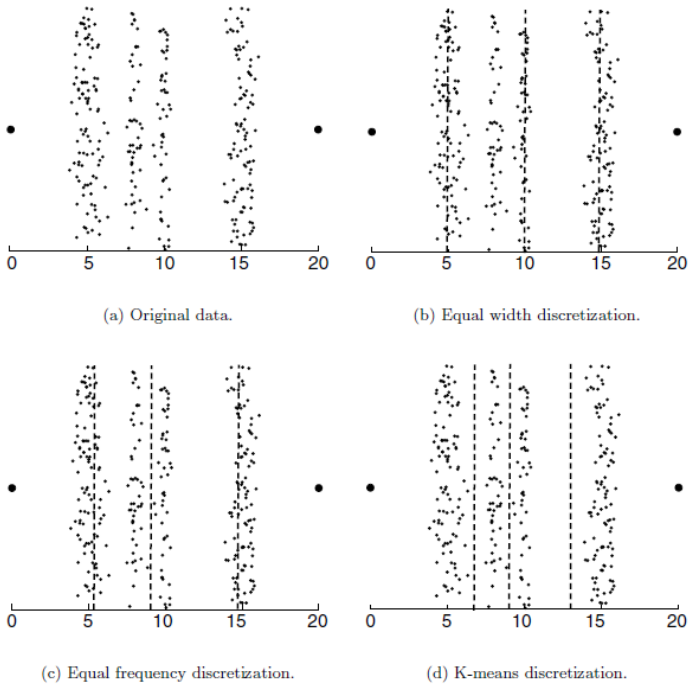
Categorical Value	Integer Value	x_1	x_2	x_3
awful	0	0	0	0
poor	1	0	0	1
OK	2	0	1	0
good	3	0	1	1
great	4	1	0	0

Figure 27: Conversion of a categorical attribute to three binary attributes

Discretization is the process of converting a continuous attribute into an ordinal attribute. Discretization can be used in either an *unsupervised* and *supervised* setting.

Unsupervised Discretization is a method for classification when the class information is not used. If class information is not used, then relatively simple approaches are common. For instance, the **equal width** approach divides the range of the attribute into a user-specified number of intervals each having the same width. Such an approach can be badly affected by outliers, and for that reason, an **equal frequency** (**equal depth**) approach, which tries to put the same number of objects into each interval, is often preferred. As another example of unsupervised discretization, a clustering method, such as **K-means** can also be used. Finally, visually inspecting the data can sometimes be an effective approach.

Figure 28 shows data points belonging to four different groups, along with two outliers—the large dots on either end. The techniques of the previous paragraph were applied to discretize the x values of these data points into four categorical values. (Points in the data set have a random y component to make it easy to see how many points are in each group.) Visually inspecting the data works quite well, but is not automatic, and thus, we focus on the other three approaches. The split points produced by the techniques equal width, equal frequency, and K-means are shown in Figures (b), (c), (d), respectively. The split points are represented as dashed lines. If we measure the performance of a discretization technique by the extent to which different objects in different groups are assigned the same categorical value, then K-means performs best, followed by equal frequency, and finally, equal width.



(a) Original data. (b) Equal width discretization.

(c) Equal frequency discretization. (d) K-means discretization.

Figure 28: Different discretization techniques

Supervised Discretization is a bit different to unsupervised discretization. Here we take an the approach to place the splits in a way that maximizes the purity of the intervals. In practice, however, such an approach requires potentially arbitrary decisions about the purity of an interval and the minimum size of an interval. To overcome such concerns, some statistically based approaches start with each attribute value as a separate interval and create larger intervals by merging adjacent intervals that are similar according to a statistical test. Entropy- based approaches are one of the most promising approaches to discretization, and a simple approach based on entropy will be presented.
There is quite some explanation behind this method, but I will leave this explanation to be done by the following video: https://www.futurelearn.com/info/courses/more-data-mining-with-weka/0/steps/29114

- **2.6.7 Variable and attribute transformation**
  A variable transformation refers to a transformation that is applied to all the values of a variable. In other words, for each object, the transformation is applied to the value of the variable for that object.
  We have two types of variable transformations, namely simple functional transformations and normalization (discussed in Chapter 3: Linear Algebra).

  For **simple functions**, we have for instance the transformations applied to mathematical functions: $x^k, log(x), \sqrt{x}, \frac{1}{x}, sin(x)$ and $|x|$.

  In statistics, the transformations *sqrt*, *log*, and $\frac{1}{x}$ are mostly used.

  Suppose the variable of interest is the number of data bytes in a session, and the number of bytes ranges from 1 to 1 billion. This is a huge range, and it may be advantageous to compress it by using a $log10$ transformation. In this case, sessions that transferred 108 and 109 bytes would be more similar to each other than sessions that transferred 10 and 1000 bytes ($98 = 1$ versus $3 - 1 = 2$). For some applications, such as network intrusion detection, this may be what is desired, since the first two sessions most likely represent transfers of large files, while the latter two sessions could be two quite distinct types of sessions.

2.7 Similarity and dissimilarity

Similarity is the numerical measure between a range of [0,1] of how *alike* two data objects are. The higher the value is, the more alike the objects are.
If the objects have a similarity of 1, they are *completely similar* and if they have a similarity of 0 they are complete dissimilar.

Dissimilarity is the numerical between a range of (mostly) [0,1] of how *different* two data objects are. The higher the value is, the more different the objects are. The upper limit can vary, but most often if the objects have a dissimilarity of 1, they are *completely different* and if the they have a dissimilarity value of 0, they are complete similar.

There is a clear correlation between the two terms, since that *similarity* = 1− *dissimilarity* and *dissimilarity* = 1− *similarity*.
In the more general case, the transformation of similarities to the interval $[0, 1]$ is given by the expression $s' = \frac{(s-min\_s)}{(max\_s-min\_s)}$, where max_s and min_s are the maximum and minimum similarity values, respectively.
Likewise, dissimilarity measures with a finite range can be mapped to the interval $[0, 1]$ by using the formula $d' = \frac{(d-min\_d)}{(max\_d-min\_d)}$.

2.8 Similarity and Dissimilarity between simple attributes

There are a few ways to compute the similarity and dissimilarity for the different types of (simple) attributes:

- **Nominal attributes** p and q:

$$d = \begin{cases} 0, & p = q \\ 1, & p \neq q \end{cases} \qquad s = 1 - d$$

  The way it works is for nominal attributes is that they are either completely similar or completely dissimilar. This means that if they are similar, the dissimilarity d is 0 if the objects are the same and 1 if they are different. This means that the similarity $s = 1 - d$ (since nominal attributes only cover distinctness $=, \neq$).

- **Ordinal attributes** map n distinct values to integers from 0 to n - 1:

$$d = \frac{|p - q|}{n - 1} \qquad s = 1 - d$$

  Consider an attribute that measures the quality of a product, e.g., a candy bar, on the scale {*poor, fair, OK, good, wonderful*}. It would seem reasonable that a product, $P1$, which is rated *wonderful*, would be closer to a product $P2$, which is rated *good*, than it would be to a product $P3$, which is rated *OK*.
  To make this observation quantitative, the values of the ordinal attribute are often mapped to successive integers, beginning at 0 or 1, e.g., {*poor = 0, fair = 1, OK = 2, good = 3, wonderful = 4*}.
  Then, $d(P1, P2) = 3 - 2 = 1$ or, if we want the dissimilarity to fall between 0 and 1, $d(P1, P2) = \frac{3-2}{4} = 0.25$.
  A similarity for ordinal attributes can then be defined as $s = 1 - d$.
  This definition of similarity (dissimilarity) for an ordinal attribute can look a bit weird since it assumes equal intervals, and this is not so (e.g. realistically *fair* is not 1 more than *poor*). Otherwise, we would have an interval or ratio attribute. As said, you can't really compare *wonderful* to *good*, but in practice our options are limited, and in the absence of more information, this is the standard approach for defining proximity between ordinal attributes.

- **Interval or ratio attributes**:

$$d = |p - q| \qquad s = \frac{1}{1+d}$$

For interval or ratio attributes, the natural measure of dissimilarity between two objects is the absolute difference of their values. For example, we might compare our current weight and our weight a year ago by saying "I am ten pounds heavier." In cases such as these, the dissimilarities typically range from 0 to ∞, rather than from 0 to 1. The similarity of interval or ratio attributes is typically expressed by transforming a similarity into a dissimilarity.

2.9 Dissimilarities between Data Objects

2.9.1 Euclidean Distance

The Euclidean Distance, or ED for short in this summary, is basically the Pythagorean theorem from high school.

The official formula for ED is as follows:

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

where n is the number of dimensions and x_k and y_k, the k^{th} attributes of x and y.

For instance, in a two-dimensional space we have the coordinates $P_1 = (0, 2)$ and $P_2 = (3, 1)$, then $d(P_1, P_3) = \sqrt{(0-3)^2 + (2-1)^2} = \sqrt{9+1} = \sqrt{10}$.

For instance, in a three-dimensional space we have the coordinates $P_3 = (2, 0, 3)$ and $P_4 = (1, 4, 2)$, then $d(P_3, P_4) = \sqrt{(2-1)^2 + (0-4)^2 + (3-2)^2} = \sqrt{1+16+1} = \sqrt{18} = 3\sqrt{2}$

2.9.2 Minkowski Distance

The Euclidean distance measure given in 2.7.3 is generalized by the Minkowski distance metric given by the following formula:

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

where r is a parameter. In the slides and book the following three examples are explained:

- $r = 1$. City block, Manhattan distance, (L_1 norm). Known as the **Hamming distance**, which is the number of bits that are between to object that have only binary attributes.

  For instance, if we use the same coordinates we used for P_1, P_2, P_3 and P_4 we get the following distances: $d(P_1, P_2) = |0-3| + |2-1| = 4$ and $d(P_3, P_4) = |2-1| + |0-4| + |3-2| = 6$.
- $r = 2$. Euclidean distance (L_2 norm).
- $r = \infty$. Supremum (L_{max} or L_∞). This is the maximum distance between any attribute of the objects.

  More formally, L_∞ is represented by the formula

$$d(x, y) = lim_{r \to \infty} (\sum_{k=1}^{n} |x_k - y_k|^r)^{\frac{1}{r}}$$

For instance, if we again use the same coordinates we used for P_1, P_2, P_3 and P_4 we get the following distances:

$d(P_1, P_2) = |0-3| + |2-1| = 3$, because we here look what the maximum distance is between one of the coordinates. Here the distance between the x-values (3) > y-values (1), meaning that is the maximum distance and hence seen as the distance between the two points.

$d(P_3, P_4) = |2-1| + |0-4| + |3-2| = 4$. Here the distance between the y-values is higher than that of the x-values and z-values, meaning the distance between the points will get the value of the distance between the y-values of the points.

2.9.3 Properties of a distance

Distances have some well-known properties and if a distance satisfies the following properties, it can be called a metric:

- **Positive definiteness:** $d(p,q) \geq 0$ for all p and q (distance between points can't be negative) and $d(p,q) = 0$ iff $p = q$.
- **Symmetry:** $d(p,q) = d(q,p)$ for all p and q.
- **Triangle inequality:** $d(p,r) \leq d(p,q) + d(q,r)$ for all p, q and r.

2.10 Similarity Measures for Binary Data

Similarity measures between objects that contain only binary attributes are called similarity coefficients, and typically have values between 0 and 1. A value of 1 indicates that the two objects are completely similar, while a value of 0 indicates that the objects are not at all similar (like it was discussed in *2.7 Similarity and dissimilarity*). Let x and y be two objects that consist of n binary attributes. The comparison of two such objects, or two binary vectors, leads to the following four quantities (frequencies):
f_{00} = the number of attributes where x is 0 and y is 0
f_{01} = the number of attributes where x is 0 and y is 1
f_{10} = the number of attributes where x is 1 and y is 0
f_{11} = the number of attributes where x is 1 and y is 1

In the lecture slides, the frequencies f are denoted as M, but in this case I follow the notation of the book rather than that of the slides.

2.10.1 Simple Matching Coefficient (SMC) and Jaccard coefficient

- Simple Matching Coefficient (SMC) is used to count both presences and absences equally.

  SMC is defined as follows:

$$SMC = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{f_{11} + f_{00}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

  For instance, the SMC could be used to find students who had answered questions similarly on a test that consisted only of true/false questions.

  If we have the two vectors:
  p = [1 0 0 0 0 0 0 0 0 0] and
  q = [0 0 0 0 0 0 1 0 0 1]
  then our $SMC = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{7}{10} = 0.7$.

- The Jaccard Coefficient (J) is similar to that of the SMC, but with Jacard you only look at the objects consisting of asymmetric binary attributes (attributes that exist with a (1) value). This means that J is defined as follows:

$$J = \frac{\text{number of matching presences}}{\text{number of attributes not involved in 00 matches}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

  With the two same vectors as before:
  p = [1 0 0 0 0 0 0 0 0 0] and
  q = [0 0 0 0 0 0 1 0 0 1]
  then our $J = \frac{\text{number of matching presences}}{\text{number of attributes not involved in 00 matches}} = \frac{f_{11}}{f_{01}+f_{10}+f_{11}} = \frac{0}{3} = 0$

2.10.2 Correlation

The correlation between two data objects that have binary or continuous variables is a measure of the linear relationship between the attributes of the objects. More precisely, **Pearson's correlation** coefficient between two data objects, x and y, is defined by the following equation:

$$\text{corr}(\mathbf{x},\mathbf{y}) = \frac{\text{covariance}(\mathbf{x},\mathbf{y})}{\text{standard deviation}(\mathbf{x}) * \text{standard deviation}(\mathbf{y})} = \frac{s_{xy}}{s_x \cdot s_y}$$

where

$$\text{covariance}(\mathbf{x},\mathbf{y}) = s_{xy} = \frac{1}{n-1}\sum_{k=1}^{n}(x^k - \overline{x})(y_k - \overline{y})$$

$$\text{standard deviaton}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(x^k - \overline{x})^2}$$

$$\text{standard deviaton}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(y^k - \overline{y})^2}$$

Correlation is always in the range -1 to 1. A correlation of 1 -(1) means that x and y have a perfect positive (negative) linear relationship.

2.10.3 Extended Jaccard Coefficient (Tanimoto Coefficient)

The extended Jaccard coefficient can be used for document data and that reduces to the Jaccard coefficient in the case of binary attributes. The extended Jaccard coefficient is also known as the Tanimoto coefficient.

It is defined as

$$EJ(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{||x||^2 + ||y||^2 - x\cdot y}$$

2.10.4 Cosine Similarity

Cosine similarity is specifically used for documents vectors where s is defined as follows:

$$s(p,q) = \frac{p\cdot q}{||p||\cdot||q||}$$

where the inner product is

$$p\cdot q = \sum_{k=1}^{n} p_k q_k$$

and the vector length is

$$||p|| = \sqrt{p\cdot p}$$

For instance if:
p = [3 2 0 5 0 0 0 2 0 0]
q = [1 0 0 0 0 0 0 1 0 2]
Inner product $p\cdot q = 3\cdot 1 + 2\cdot 0 + 0\cdot 0 + 5\cdot 0 + 0\cdot 0 + 0\cdot 0 + 0\cdot 0 + 2\cdot 1 + 0\cdot 0 + 0\cdot 2 = 5$.
Vector lengths $||p|| = \sqrt{3^2 + 2^2 + 5^2 + 2^2} = \sqrt{42}$ and $||q|| = \sqrt{1^2 + 1^2 + 2^2} = \sqrt{6}$.
So, our cosine similarity will be $s(p,q) = \frac{p\cdot q}{||p||\cdot||q||} = \frac{5}{\sqrt{252}} = \frac{5}{6\sqrt{7}}$

Chapter 3: Data Exploration

Data Exploration is a preliminary investigation of the data in order to better understand its specific characteristics. Data exploration can aid in selecting the appropriate preprocessing and data analysis techniques. It can even address some of the questions typically answered by data mining. For example, patterns can sometimes be found by visually inspecting the data. Also, some of the techniques used in data exploration, such as visualization, can be used to understand and interpret data mining results.

This chapter will cover three topics:

- Summary statistics: such as the mean and standard deviation of a set of values
- Visualization: such as histograms and scatter plots (as practiced with in the assignments)
- On-Line Analytical Processing (OLAP): a set of techniques for exploring multidimensional arrays of values and how to create a summary of data tables from a multidimensional data array (not discussed in this summary).

3.1 Summary Statistics

Summary statistics are quantities, such as the mean and standard deviation, that capture various characteristics of a potentially large set of values with a single number or a small set of numbers. Examples could be the average household income or the fraction of people doing a masters degree.

3.1.1 Frequencies, Mode

Given a categorical attribute x, which can take values $v_1, ..., v_i, ...v_k$ and a set of m objects, the frequency of a value v_i is defined as

$$\text{frequency}(v_i) = \frac{\text{number of objects with attribute value } v_i}{m}.$$

Here, the mode of a categorical attribute (e.g. type of classes in high school) is the value that has the highest frequency.
Categorical attributes often, but not always, have a small number of values, and consequently, the mode and frequencies of these values can be interesting and useful. For *continuous data*, the mode, as currently defined, is often not useful because a single value may not occur more than once (and thus the frequency doesn't have any use).

3.1.2 Percentiles

For ordered data, it is more useful to consider the percentiles of a set of values.
In particular, given an ordinal or continuous attribute x and a number p between 0 and 100, the p^{th} percentile x_p is a value of x such that p% of the observed values of x are less than x_p.
For instance, the 50^{th} percentile is the value $x_{50\%}$ such that 50% of all values of x are less than $x_{50\%}$.
The median is $x_{50\%}$ and the quartiles are $x_{25\%}$ and $x_{75\%}$ (often seen in boxplots).

3.1.3 Measures of Location: Mean and Median

For continuous data, two of the most widely used summary statistics are the mean (gemiddelde) and median (mediaan, middelste), which are measures of the location of a set of values.

Where the mean and the median are computed as the following:

$$Mean(x) = \frac{1}{n}\sum_{k=1}^{n} x^k \quad \text{(very sensitive to outliers)}$$

$$Median(x) = \left\{ \begin{array}{ll} x_{r+1}, & n = 2r+1 \quad \text{(odd)} \\ \frac{1}{2}(x_{(r)} + x_{(r+1)}), & n = 2r \quad \text{(even)} \end{array} \right\}$$

3.1.4 Measures of Spread: Range and Variance

Next to the Mean and Median, we also have other summary statistics for continuous data, but these measure the dispersion or spread of a set of values. This can help to see if the values are spread out of if they are more centered together.
We have five measures:

- **Range**
  This is the simplest measure of spread, which, given an attribute set of m values $\{x_1, \ldots, x_m\}$, is defined as

  $$\text{range}(x) = \max(x) - \min(x) = x_{(m)} - x_{(1)}$$

  So, if we have a set of values $m = \{1,3,7,2,8,32,72,2,4\}$, then our range is $72 - 1 = 71$.
- **Variance**
  Although the range identifies the maximum spread, it can be misleading if most of the values are concentrated in a narrow band of values, but there are also a relatively small number of more extreme values.
  Hence, the variance is preferred as a measure of spread. The variance is defined as:

  $$\text{variance}(x) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \overline{x})^2$$

  In this formula x_k is the value of each individual attribute, $\overline{x}$ is the mean of all of the attributes in the data set and n is the number of elements in the data set.
  So, if we have the set $\{46,69,32,60,52,41\}$, these individual values are our x_k values.
  $\overline{x} = \frac{46+69+32+60+52+41}{6} = 50$, and thus
  $\sum_{k=1}^{n}(x_k - \overline{x})^2 = (46-50)^2 + (69-50)^2 + (32-50)^2 + (60-50)^2 + (52-50)^2 + (41-50)^2 = 886$.
  So,

  $$\text{variance}(x) = \frac{886}{6-1} = \frac{886}{5} = 177,2$$
- **Standard deviation**
  The standard deviation (std) is simply the square root of the variance, meaning if we take the values from the variance above we get:

  $$\text{standard deviation}(x) = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(x_k - \overline{x})^2} = \sqrt{177.2} \approx 13.3$$

  The mean can be distorted by outliers, and since the variance is computed using the mean, it is also sensitive to outliers. Indeed, the variance is particularly sensitive to outliers since it uses the squared difference between the mean and other values. As a result, more robust estimates of the spread of a set of values are often used (explained below).
- **Absolute Average Deviation (AAD)**
  The AAD is defined as the following:

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

For the example, I take the same set of values used for the example of *variance*.
This means we get

$$\text{AAD}(x) = \frac{1}{6}(|46-50|+|69-50|+|32-50|+|60-50|+|52-50|+|41-50|) = \frac{4+19+18+10+2+9}{6} = 10.33\ldots$$

Due to the definition of the AAD, the results will be closely familiar to those of the standard deviation.

– **Median Absolute Deviation (MAD)**
The MAD is defined as the following:

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

We again use the same set as before.
We first find the median of the whole set, which is $\frac{32+60}{2} = 46$. We then subtract each value from the set with this value. This creates a new set, which we sort, and then take the median of all this new set, which is our MAD.

$$(|46-46|,|69-46|,|32-46|,|60-46|,|52-46|,|41-46|) = (0,23,14,14,6,5)$$

We now sort the set with the lowest value left and the highest to the right, and then take the median of that.

$$\text{MAD}(x) = \text{median}\,(0,5,6,14,14,23) = \frac{14+6}{2} = 10$$

– **Interquartile Range (IQR)**
The IQR is used to build box plots, simple graphical representations of a probability distribution.
The IQR is defined as the following:

$$\text{interquartile range}(x) = x_{75\%} - x_{25\%}$$

If we take the set of values, then our median are the values 32 and 60, meaning our first quartile are the values 46 and 69 and our third quartile the values 52 and 41. We take the median of both quartiles and subtract quartile 3 from quartile 1. This means we get ($x_{75\%} = \frac{52+41}{2} = 46.5$) - ($x_{25\%} = \frac{46+69}{2} = 57.5$) = -11

3.2 Visualization

Visualization is the conversion of data into a visual or tabular format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported.

Visualization of data is one of the most powerful and appealing techniques for data exploration.

- Humans have a well developed ability to analyze large amounts of information that is presented visually.
- Can detect general patterns and trends.
- Can detect outliers and unusual patterns.

The overriding motivation for using visualization is that people can quickly absorb large amounts of visual information and find patterns in it. For instance Figure 30 shows the Sea Surface Temperature (SST) in degrees Celsius for July, 1982. This picture summarizes the information from approximately 250,000 numbers and is readily interpreted in a few seconds.

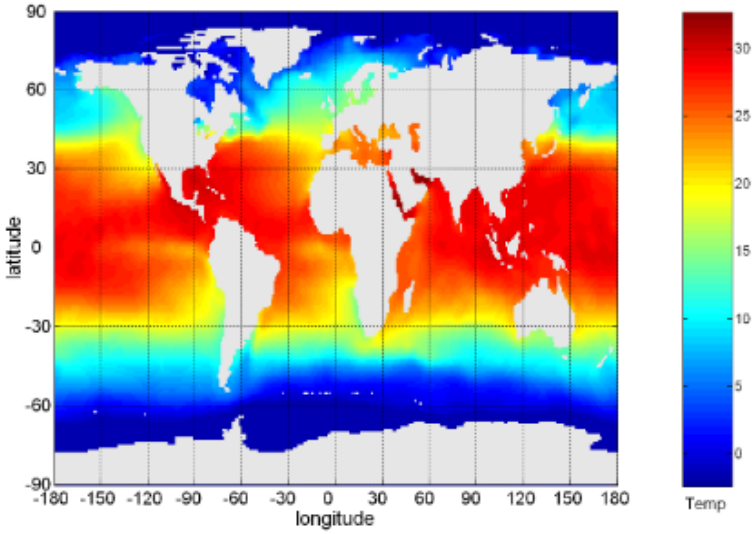


Figure 29: SST for July, 1982

3.2.1 General Concepts of Visualization

The part will discuss some general concepts related to visualization, in particular, general approaches for visualizing the data and its attributes.

- **Representation: Mapping Data to Graphical Elements**
  Representation refers to the mapping of information to a visual format. Data objects, their attributes, and the relationships among data objects are translated into graphical elements such as points, lines, shapes, and colors.

  Objects are usually represented in three possible ways:

  1. If only a single categorical attribute of the object is being considered, then objects are often lumped into categories based on the value of that attribute, and these categories are displayed as an entry in a table or an area on a screen. An example could be a bar chart, since each single value in the graph gets its own 'bar'.
  2. If an object has multiple attributes, then the object can be displayed as a row (or column) of a table or as a line on a graph.
  3. If an object is interpreted as a point in two- or three-dimensional space, where graphically, the point might be represented by a geometric figure, such as a circle, cross, or box.

  For attributes, the representation depends on the type the attribute is (nominal, ordinal, or continuous). Ordinal and continuous attributes can be mapped to continuous, ordered graphical features such as location along the x, y, or z axes, intensity, color, or size (diameter, width, height, etc.).

  For categorical attributes, each category can be mapped to a distinct position, color, shape, orientation, embellishment, or column in a table. However, for nominal attributes, whose values are unordered, care should be taken when using graphical features, such as color and position that have an inherent ordering associated with their values. In other words, the graphical elements used to represent the ordinal values often have an order, but ordinal values do not.

- **Arrangement**
  As discussed earlier, the proper choice of visual representation of objects and attributes is essential for good visualization.
  The arrangement of items within the visual display is also crucial.

  This is best shown with an example:

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	0	1	0	1	1	0
4	1	0	1	0	0	1
5	0	1	0	1	1	0
6	1	0	1	0	0	1
7	0	1	0	1	1	0
8	1	0	1	0	0	1
9	0	1	0	1	1	0

	6	1	3	2	5	4
4	1	1	1	0	0	0
2	1	1	1	0	0	0
6	1	1	1	0	0	0
8	1	1	1	0	0	0
5	0	0	0	1	1	1
3	0	0	0	1	1	1
9	0	0	0	1	1	1
1	0	0	0	1	1	1
7	0	0	0	1	1	1

Figure 30: An ordered and and unordered table

Figure 31 shows two tables where the left table has $0's$ and $1's$ all mixed up. Yes it is readable, but having the $0's$ and $1's$ sorted would help with understanding the data better (so having the right table). More formally, you would say that the binary attributes are permuted so that the relationships of the rows and columns are clear.

- **Selection**

  Selection refers to the elimination or the de-emphasis of certain objects and attributes. Specifically, while data objects that only have a few dimensions can often be mapped to a two- or three-dimensional graphical representation in a straightforward way, there is no completely satisfactory and general approach to represent data with many attributes.

  This means that we may have to choose a subset of attributes which represent the data the best. Ways of doing this can be dimensionality reduction (like PCA which was discussed in Chapter 4) or pair attributes together (also a dimensionality reduction technique).

  Selection may also involve choosing a subset of objects. This can be because the region of the screen can only show so many points (you can't have an infinitely big plot). You then zoom in on a particular region of the data or by taking a sample of the data points.

3.2.2 Visualization Techniques

There are many types of ways to visualize data. This section will be more of an overview of each type rather than an explanation of each type. For more information about each technique it's best to consult the book (Chapter *3.3.4*).

The most common ones, for visualizing a small number of attributes, are the following:

- **Histograms**
  A histogram is a plot that displays the distribution of values for attributes by dividing the possible values into bins and showing the number of objects that fall into each bin (the range).
  * For categorical data, each value is a bin.
  * For continuous data, the range of values is divided into bins, typically, but not always, of equal width.
  * The shape of the histogram depends on the number of bins.

  An example is the following where we have 10 and 20 bins respectively:

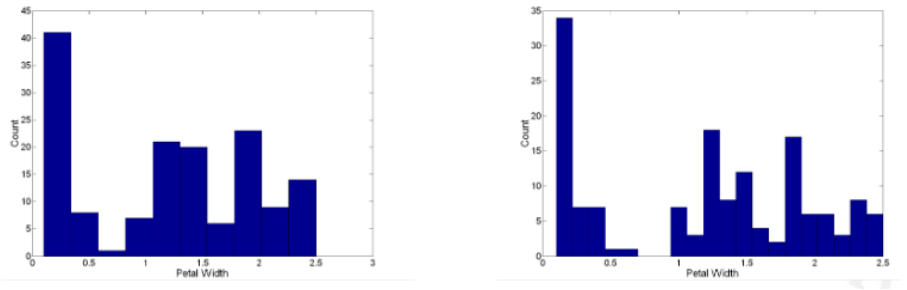


Figure 31: Histograms with 10 and 20 bins

- **Box plots**
  Box plots are another method for showing the distribution of the values of a single numerical attribute. Figure 32 shows a labeled box plot for sepal length. The lower and upper ends of the box indicate the 25^{th} and 75^{th} percentiles (computed by the *IQR*), respectively, while the line inside the box indicates the value of the 50^{th} percentile. The top and bottom lines of the tails indicate the 10^{th} and 90^{th} percentiles. Outliers are shown by " + " marks (previous exam question). Box plots are relatively compact, and thus, many of them can be shown on the same plot. Simplified versions of the box plot, which take less space, can also be used.

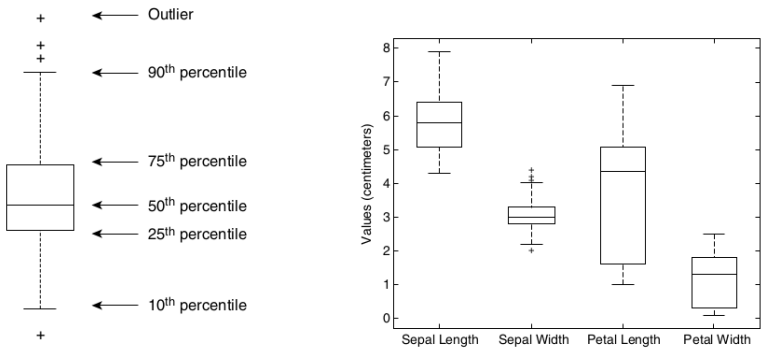


Figure 32: Description of box plot and the box plot for the Iris attributes

- **Scatter plots**
  Each data object is plotted as a point in the plane using the values of the two attributes as x and y coordinates. It is assumed that the attributes are either integer- or real-valued.
  * Two-dimensional scatter plots are most common, but there are three-dimensional scatter plots.
  * Often additional attributes can be displayed by using the size, shape, and color of the markers that represent the object.
  * Arrays of scatter plots can compactly summarize the relationships of several pairs of attributes.

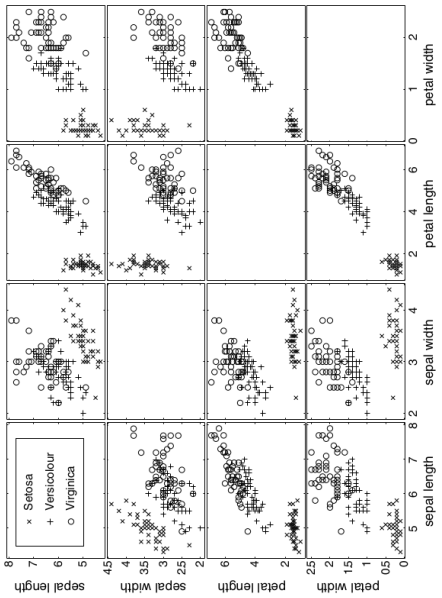


Figure 33: Matrix of scatter plots

Other types of charts and plots are a **Pie chart Percentile Plots** and **Cumulative Distribution Functions**.

We also have techniques of visualizing *spatio-temporal data*:

- **Contour Plots**
  * Useful when continuous attribute is measured on a spatial grid
  * They partition the plane into regions of similar values
  * The contour lines that form the boundaries of these regions connect points with equal values
  * The most common example is contour maps of elevation
  * Can also display temperature, rainfall, air pressure etc.

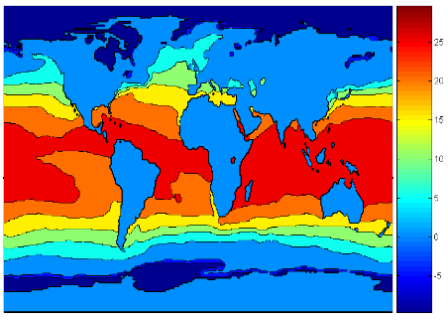


Figure 34: Contour plot: SST December 1998

– **Surface Plots**

Like contour plots, surface plots use two attributes for the x and y coordinates. The third attribute is used to indicate the height above the plane defined by the first two attributes. While such graphs can be useful, they require that a value of the third attribute be defined for all combinations of values for the first two attributes, at least over some range.

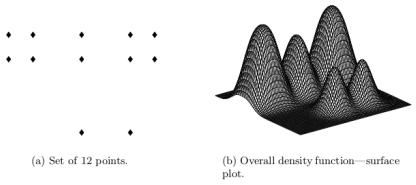


Figure 3.20. Density of a set of 12 points.

Figure 35: Density of 12 points and the corresponding surface plot

– **Vector field plots**

In some data, a characteristic may have both a magnitude and a direction associated with it. For example, consider the flow of a substance or the change of density with location. In these situations, it can be useful to have a plot that displays both direction and magnitude. This type of plot is known as a vector plot.

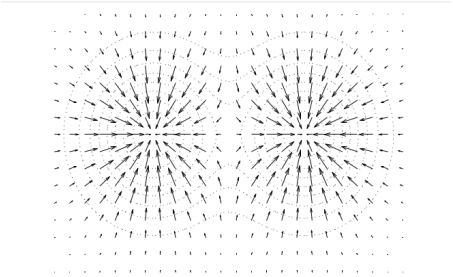


Figure 36: Vector plot of the gradient (change) in density for the bottom two density peaks of Figure 41

– **Lower-Dimensional Slices**

Consider a spatio-temporal data set that records some quantity, such as temperature or pressure, at various locations over time. Such a data set has four dimensions and cannot be easily displayed by the types of plots described so far. However, separate “slices” of the data can be displayed by showing a set of plots, one for each month. By examining the change in a particular area from one month to another, it is possible to notice changes that occur, including those that may be due to seasonal factors.

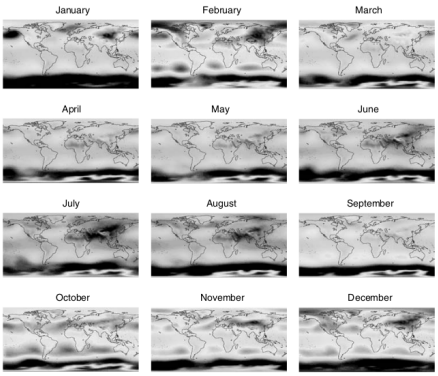


Figure 3.22. Monthly plots of sea level pressure over the 12 months of 1982.

Figure 37: Monthly plots of sea level pressure in 1982

3.2.3 Visualizing Higher-Dimensional Data

- **Matrix**
  - Can plot the data matrix
  - This can be useful when objects are sorted according to class
  - Typically, the attributes are normalized to prevent one attribute from dominating the plot (outliers)
  - Plots of similarity or distance matrices can also be useful for visualizing the relationships between objects

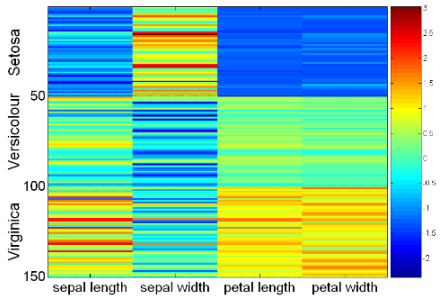


Figure 38: Plot of the Iris dataset with standardized columns to have a mean of 1 and a std of 0

- **Parallel Coordinates**
  - Used to plot the attribute values of high-dimensional data
  - Instead of using perpendicular axes, use a set of parallel axes
  - The attribute values of each object are plotted as a point on each corresponding coordinate axis and the points are connected by a line
  - Thus, each object is represented as a line and the lines representing a distinct class of objects group together, at least for some attributes
  - Ordering of attributes is important in seeing such groupings

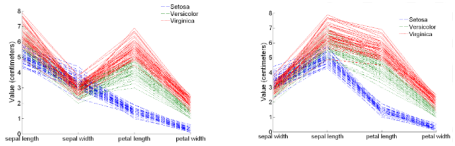


Figure 39: Parallel coordinates plots for the Iris Data set

- **Star Coordinates/plots**
  Star plots take a similar approach to parallel coordinates, but the axes radiate from a central point. The line connecting the values of an object is a polygon.

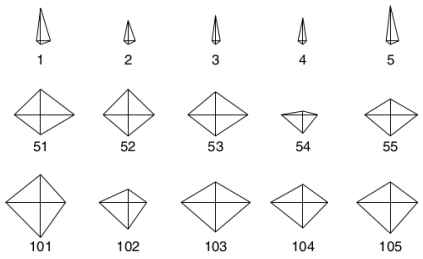


Figure 40: Plot of 15 Iris flowers using star coordinates

- **Chernoff Faces**
  - This approach associates each attribute with a characteristic of a face
  - The value of each attribute determine the appearance of the corresponding facial characteristic and each object becomes a seperate face.
  - It does rely on the human's ability to distinguish faces

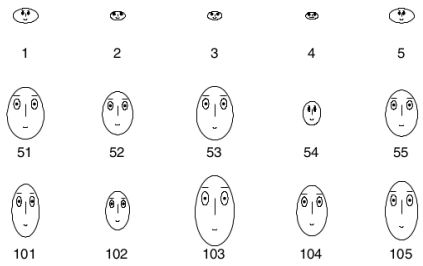


Figure 41: A plot of 15 Iris flowers using Chernoff faces

3.2.4 Do's and Don'ts of visualization

There are a few do's and don'ts regarding visualization. These are also known as the ACCENT Principles.

Apprehension
Ability to correctly perceive relations among variables. Does the graph maximize apprehension of the relations among variables?

Clarity
Ability to visually distinguish all the elements of a graph. Are the most important elements or relations visually most prominent?

Consistency
Ability to interpret a graph based on similarity to previous graphs. Are the elements, symbol shapes, and colors consistent with their use in previous graphs?

Efficiency
Ability to portray a possibly complex relation in as simple a way as possible. Are the elements of the graph economically used? Is the graph easy to interpret?

Necessity
The need for the graph, and the graphical elements. Is the graph a more useful way to represent the data than alternatives (table, text)? Are all the graph elements necessary to convey the relations?

Truthfulness
Ability to determine the true value represented by any graphical element by its magnitude relative to the implicit or explicit scale. Are the graph elements accurately positioned and scaled?

There is a chance that on the exam a question will be there about these principles, so know them well!

We also have Tufte's Guidleines, which can be used to achieve graphical excellence:

- Graphical excellence is the well-designed presentation of interesting data, a matter of *substance*, of *statistics*, and of *design*.
- Graphical excellence consists of complex ideas communicated with *clarity*, *precision*, and *efficiency*.
- Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
- Graphical excellence is nearly always multivariate.
- Graphical excellence requires telling the truth about the data.

Chapter 4: Classification: Basic Concepts, Decision Trees, and Model Evaluation

This chapter introduces the basic concepts of classification, describes some of the key issues such as model overfitting, and presents methods for evaluating and comparing the performance of a classification technique.

4.1 Classification

Classification is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications.

We also have a more detailed definition of *classification*:

- Given a collection of records (training set):
  Each record contains a set of attributes, one of the attributes is the class.
- Find a model for the class attribute as a function (*target function*) of the values of other attributes (as seen in figure 43).
- The goal is to assign previously unseen record to a class as accurately as possible.
  We can use a test set to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Figure 42: Classification as the task of mapping an input attribute set x into its class label y

The target function is also known as a **classification model**. A classification model can be used for the following things:

- **Descriptive modelling**
  A classification model can serve as an explanatory tool to distinguish between objects of different classes.
  Examples of this could be
  - Classifying credit card transactions as legitimate or fraudulent
  - Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil.
- **Predictive modelling**
  A classification model can also be used to predict the class label of unknown records. A classification model can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record.
  Examples of this could be
  - Predicting tumor cells as benign or malignant
  - Predicting to what type of species a newly discovered animal belongs

Classification techniques are most suited for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories (e.g., to classify a person as a member of high-, medium-, or low-income group) because they do not consider the implicit order among the categories.

4.2 Solving a classification problem and the structure of a model

A classification technique (or classifier) is a systematic approach to building classification models from an input data set. Examples include decision tree classifiers, rule-based classifiers, neural networks, support vector machines, and naïve Bayes classifiers. Each technique employs a *learning algorithm* to identify a model that best fits the relationship between the attribute set and class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records.

Figure 44 shows the general approach for building a classification model, as was described in section *6.1*.

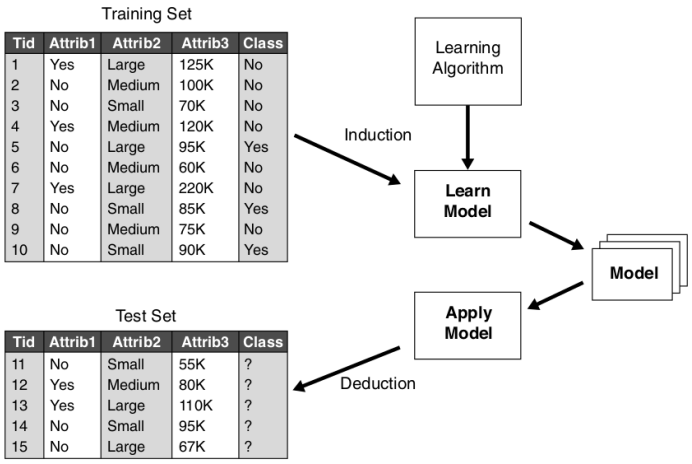


Figure 43: General approach for building a classification model

Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. These counts are tabulated in a table known as a confusion matrix.

For binary attributes, the *confusion matrix* looks like the following:

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

Figure 44: Confusion matrix for binary attributes in a 2-class problem

For instance, f_{01} is the number of records from class 0 incorrectly predicted as class 1. Based on the entries in the confusion matrix, the total number of correct predictions made by the model is $(f_{11} + f_{00})$ and the total number of incorrect predictions is $(f_{10} + f_{01})$.

Summarizing the information we gain from the confusion matrix can be more conveniently done by assigning a performance metric such as accuracy and subsequently the error rate:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{01} + f_{10}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

4.3 Decision tree

The classification technique which is explained in the lectures is the decision trees. This technique is widely usable and has come by in different courses (especially for CS students) and in different ways. This section will explain how it works, so the basic information about the tree, how to build one, splitting the attributes of tree (and in which best way). There is quite a lot to decision trees, and hence it's definitely good to watch a video about it and the application of it to data analysis and processing (will be referenced in *E-Chapter I: Extra resources*.

4.3.1 How does a decision tree work

I will use the example of the book (about mammal classification) as the example, hence that will be what you'll see in the figures etc.

In our decision, we have three types of nodes:

- A **root node** that has no incoming edges and zero or more outgoing edges.
- **Internal nodes**, each of which has exactly one incoming edge and two or more outgoing edges.
- **Leaf** or **terminal nodes**, each of which has exactly one incoming edge and no outgoing edges.

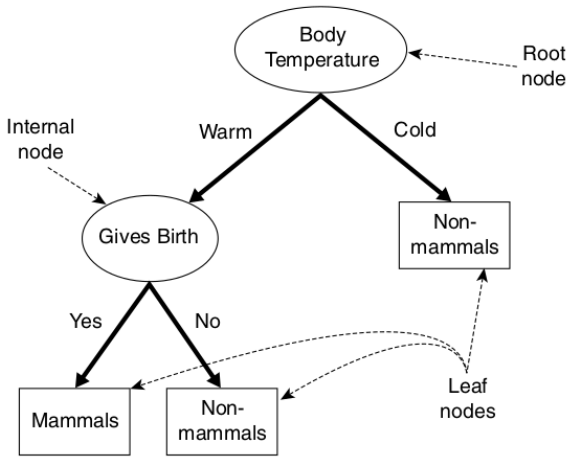


Figure 45: A decision tree for the mammal classification problem

In a decision tree, each leaf node is assigned a class label (Non-mammals, Mammals). The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics.

Classifying a test record is straightforward once a decision tree has been constructed. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. This will lead us either to another internal node, for which a new test condition is applied, or to a leaf node.

4.3.2 Building a decision tree

If we apply test data to this decision tree, we can classify a new object.

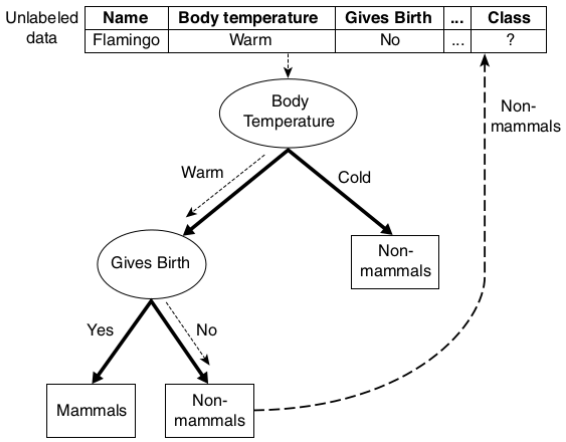


Figure 46: Classifying a new object

These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data.

The algorithm for this which is discussed in the lecture is *Hunt's algorithm* (explained in the next section).

4.3.3 Hunt's algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. For the explanation of the algorithm the data of the home owner, income etc. will be used.

Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, \ldots, y_c\}$ be the class labels.

Step 1 (base case):
If all the records in D_t belong to the same class y_t , then t is a leaf node labeled as y_t.

Step 2 (inductive case):
If D_t contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 47: The training set for predicting potential borrowers

To show how the algorithm works, we will use the above data set to determine whether a specific borrower will default on loan payments.

Step 1:
We check whether all records in D_t belong to the same class y_t. In our case the class "Defaulted Borrower" has the values *Yes* and *No*. The records y_5, y_8 and y_{10} have as a value *Yes* and the rest *No*, meaning not all records belong to the same class. So, we move on to the next step.

Step 2:
There are records belonging to more than one class, so we apply an attribute test condition to partition the records into smaller subsets.
So, we look at the class attribute and see most people don't borrow any money, meaning we get the tree that looks like Figure 48(a).

The tree, however, needs to be refined since the root node contains records from both classes. The records are subsequently divided into smaller subsets based on the outcomes of the *Home Owner* test condition, as shown in Figure 48(b) (so we are redoing step 2).

From the training set given in Figure 47, notice that all borrowers who are home owners successfully repaid their loans. The left child of the root is therefore a leaf node labeled Defaulted = No (see Figure 4.7(b)). For the right child, we need to continue applying the recursive step of Hunt's algorithm until all the records belong to the same class. The trees resulting from each recursive step are shown in Figures 48(c) and (d).

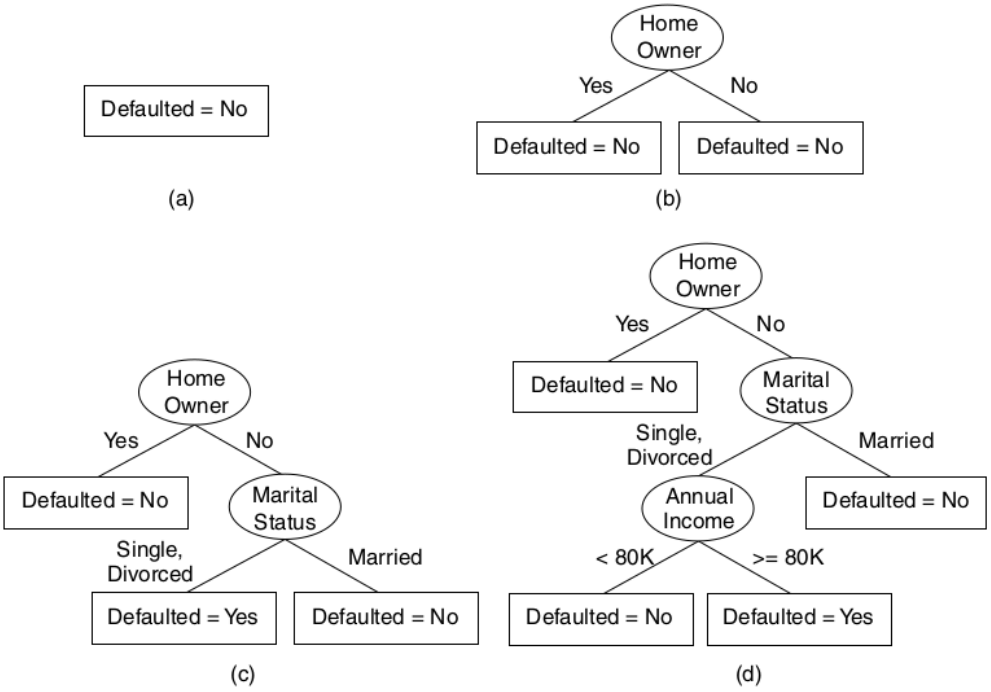


Figure 48: Visualization of Hunt's algorithm

Hunt's algorithm will work if every combination of attribute values is present in the training data and each combination has a unique class label. These assumptions are too stringent for use in most practical situations. Additional conditions are needed to handle the following cases:

1. It is possible for some of the child nodes created in Step 2 to be empty; i.e., there are no records associated with these nodes. This can happen if none of the training records have the combination of attribute values associated with such nodes. In this case the node is declared a leaf node with the same class label as the majority class of training records associated with its parent node.
2. In Step 2, if all the records associated with D_t have identical attribute values (except for the class label), then it is not possible to split these records any further. In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

There are also two design issues regarding decision tree induction (so issues with Hunt's algorithm):

1. **How should the training records be split?**
   Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets. To implement this step, the algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.
2. **How should the splitting procedure stop?**
   A stopping condition is needed to terminate the tree-growing process. A possible strategy is to continue expanding a node until either all the records belong to the same class or all the records have identical attribute values. Although both conditions are sufficient to stop any decision tree induction algorithm, other criteria can be imposed to allow the tree-growing procedure to terminate earlier.

4.3.4 Splitting attributes based on their types

Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types.

Splitting based on Nominal Attributes
We have a Multi-way split: use as many partitions as distinct values:

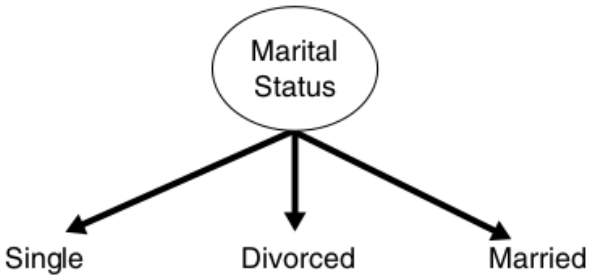


Figure 49: Multi-way split

We have a Binary split: divides values into two subsets; find coherent partitioning:

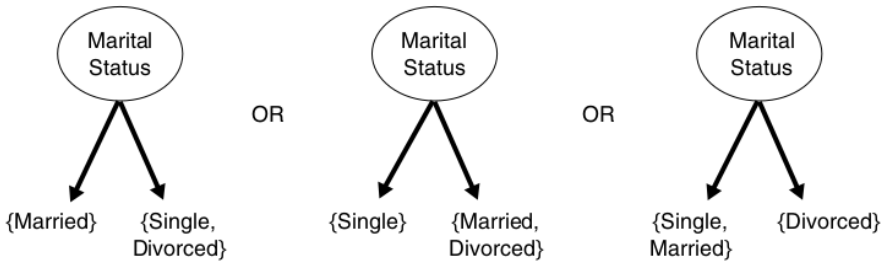


Figure 50: Binary split (by grouping attribute values, multiple possible splits)

Splitting based on Ordinal Attributes
We have a Multi-way split: use as many partitions as distinct values (same as with Nominal Attributes).

We have a Binary split: divides values into two subsets; find coherent partitioning:

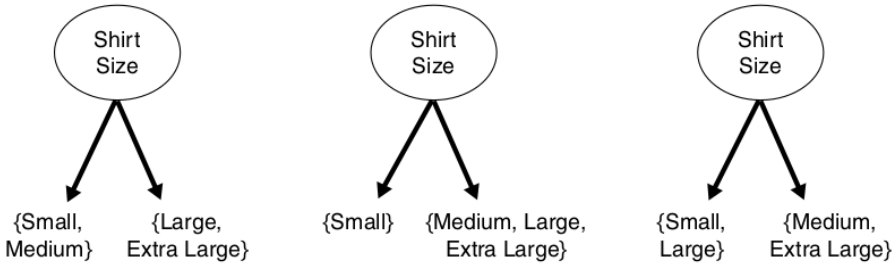


Figure 51: Different ways to split groups of ordinal values

Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the attribute values. Figure 51 illustrates various ways of splitting training records based on the `Shirt Size` attribute.
The groupings shown in Figures 51(a) and (b) preserve the order among the attribute values, whereas the grouping shown in Figure 51(c) violates this property because it combines the attribute values `Small` and `Large` into the same partition while `Medium` and `Extra Large` are combined into another partition (meaning the order is not correct anymore).

Splitting based on Continuous Attributes

There are different ways of splitting continuous attributes:

Discretization to form an ordinal categorical attribute (Figure 52(a)).

Binary Decision: $(A < v)$ or $(A \leq v)$ (Figure 52(b)).

We consider all possible splits and we find the best cut, which sadly can be computationally intensive.

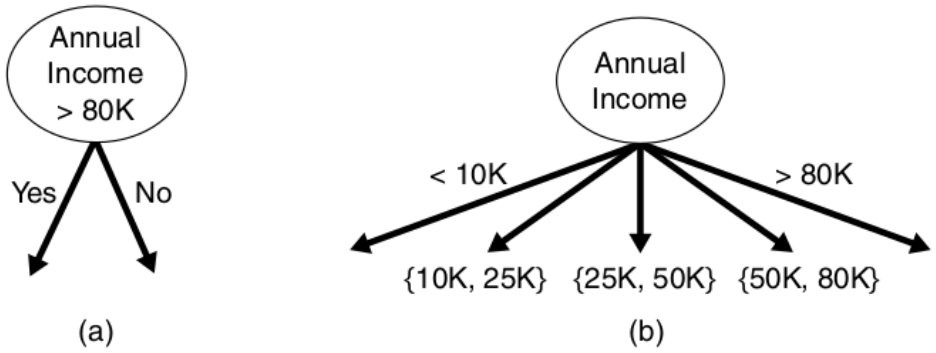


Figure 52: Different ways to split groups of ordinal values

Since we are dealing with mathematics and data, there are ways to determine what the best splits/cuts are. So how to find them will be discussed in the next section.

4.3.5 Measures for selecting the best split

There are many measures that can be used to determine the best way to split the records. These measures are defined in terms of the class distribution of the records before and after splitting.

If we for instance take the following splits:

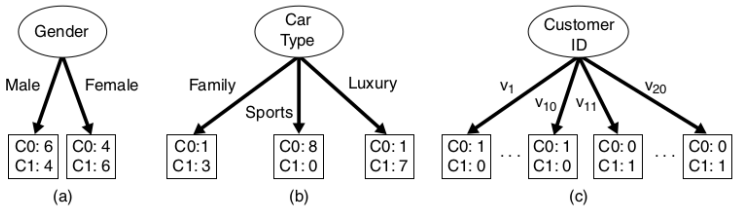


Figure 53: Multiway and binary splits

Then the class distribution before splitting is (0.5, 0.5) because there are an equal number of records from each class.
If we split the data using the `Gender` attribute, then the class distributions of the child nodes are $(0.6, 0.4)$ and $(0.4, 0.6)$, respectively.
Although the classes are no longer evenly distributed, the child nodes still contain records from both classes.
Splitting on the second attribute, `Car Type`, will result in purer partitions.
The measures developed for selecting the best split are often based on the degree of impurity of the child nodes.
The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution $(0, 1)$ has zero impurity, whereas a node with uniform class distribution $(0.5, 0.5)$ has the highest impurity.

In the lecture, the following three measures of node impurity are considered:

- **GINI Index**
  - GINI Index for a given node t:
    $$GINI(t) = 1 - \sum_c [p(c|t)]^2$$
    where $p(c|t)$ is the relative frequency of class c at node t.
  - Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying highest impurity.
  - Minimum (0) when all records belong to one class, implying highest purity.
- **Entropy**
  - Entropy at a given node t:
    $$Entropy(t) = -\sum p(c|t) log p(c|t)$$
    where $p(c|t)$ is the relative frequency of class c at node t.
  - Measures homogeneity of a node
  - Maximum $(\log n_c)$ when records are equally distributed among all classes implying highest impurity.
  - Minimum (0) when all records belong to one class, implying highest purity.
- **Classification error**
  - Classification error at a node t:
    $$Error(t) = 1 - \max_c [p(c|t)]$$
  - Measures misclassification error made by a node
  - Maximum $(1 - \frac{1}{n_c}$ when records are equally distributed among all classes, implying highest impurity
  - Minimum (0) when all records belong to one class, implying highest purity

Examples of each measure:

Node N_1	Count
Class=0	0
Class=1	6

$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$
$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
$\text{Error} = 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$
$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$
$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$

Figure 54: Examples of each impurity measure

The preceding examples, along with Figure 55, illustrate the consistency among different impurity measures. Based on these calculations, node N_1 has the lowest impurity value, followed by N_2 and N_3. Despite their consistency, the attribute chosen as the test condition may vary depending on the choice of impurity measure.

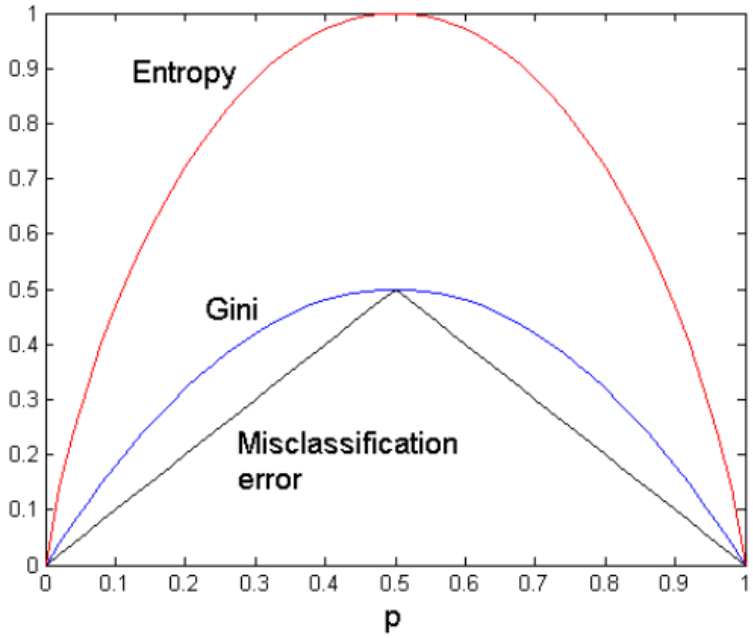


Figure 55: Comparison among the impurity measures for binary classification problems

To determine how well a test condition performs, we need to compare the degree of impurity of the parent node (before splitting) with the degree of impurity of the child nodes (after splitting). The larger their difference, the better the test condition. The gain, Δ, is a criterion that can be used to determine the goodness of a split:

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

where $I(\cdot)$ is the impurity measure of a given node, N is the total number of records at the parent node, k is the number of attribute values, and $N(v_j)$ is the number records associated with the child node, v_j. Decision tree induction algorithms often choose a test condition that maximizes the gain Δ. Since $I(\text{parent})$ is the same for all test conditions, maximizing the gain is equivalent to minimizing the weighted average impurity measures of the child nodes.

With the information from above we can answer the following question:

When training a decision tree, we use the classification error as impurity measure $I(t)$ given by

$$I(t) = 1 - \max_c [p(c|t)]$$

We will consider classification of wine into *Red* and *White* wine. At a potential split we have:

- Before the split: 5 Red and 10 White
- After the split:
  - 3 Red and 2 White in the left node
  - 2 Red and 8 White in the right node

Compute the gain Δ *after splitting.*

Remember that

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

where N is the total number of data objects at the parent node, k is the number of child nodes and $N(v_j)$ is the number of data objects associated with the child node v_j.

We first compute the I(parent), then v_{left} and v_{right} and then Δ.

$I(\text{parent})\ 1 - max[\frac{5}{15} \vee \frac{10}{15}] = 1 - \frac{10}{15} = \frac{5}{15} = \frac{1}{3}$
$I(v_{left}) = 1 - max[\frac{3}{5} \vee \frac{2}{5}] = 1 - \frac{3}{5} = \frac{2}{5}$
$I(v_{right}) = 1 - max[\frac{2}{10} \vee \frac{8}{10}] = 1 - \frac{8}{10} = \frac{2}{10} = \frac{1}{5}$
$\Delta = \frac{1}{3} - \frac{5}{15} \cdot \frac{2}{15} - \frac{10}{15} \cdot \frac{1}{5} = \frac{1}{3} - \frac{10}{75} - \frac{10}{75} = \frac{25}{75} - \frac{20}{75} = \frac{5}{75} = \frac{1}{15}$

So, our total gain from doing a split is $\frac{1}{15}$.

The gain after splitting is also possible to be calculated for the entropy, which is called Information Gain and is denoted as $GAIN_{split}$ or Δ_{info}

Quality of Entropy: Information Gain

- Information Gain:
$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$
  where parent node p is split into k partitions and n_i is the number of record in partition i
- Measures reduction in entropy achieved when using *split* to partition the values of attribute a
- Choose the split with maximum GAIN
- Disadvantage: tends to prefer splits that result in large number of partitions, each being small but pure

Splitting of Binary Attributes

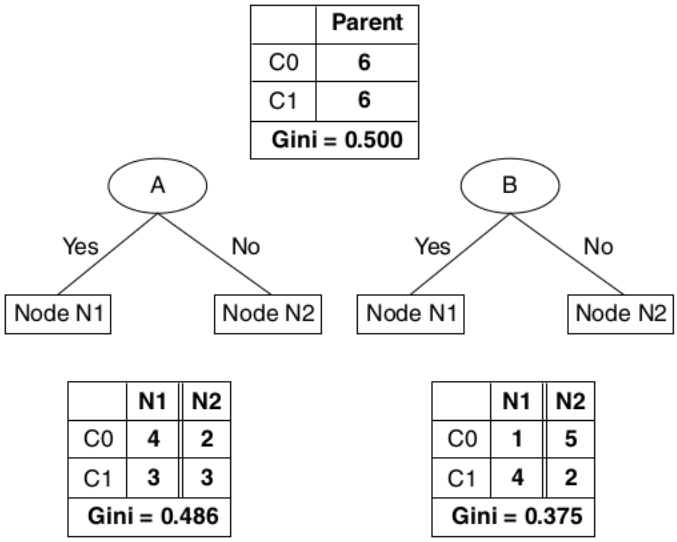


Figure 56: Splitting binary attributes

If we look at the diagram in Figure 56 and suppose there are two ways to split the data into smaller subsets. Before splitting, the Gini index is 0.5 since there are an equal number of records from both classes (you can calculate this yourself, is a good exercise ;D).

If attribute A is chosen to split the data, the Gini index for node N_1 is 0.4898, and for node N_2, it is 0.480. The weighted average of the Gini index for the descendent nodes is $(7/12) \cdot 0.4898 + (5/12) \cdot 0.480 = 0.486$. Similarly, we can show that the weighted average of the Gini index for attribute B is 0.375. Since the subsets for attribute B have a smaller Gini index, it is preferred over attribute A.

Splitting of Nominal Attributes

As previously noted, a nominal attribute can produce either binary or multi-way splits, as shown in Figure 57. The computation of the Gini index for a binary split is similar to that shown for determining binary attributes. For the first binary grouping of the `Car Type` attribute, the Gini index of {Sports, Luxury} is 0.4922 and the Gini index of {Family} is 0.3750. The weighted average Gini index for the grouping is equal to $\frac{16}{20} \cdot 0.4922 + \frac{4}{20} \cdot 0.3750 = 0.468$. Similarly, for the second binary grouping of {Sports} and {Family, Luxury}, the weighted average Gini index is 0.167. The second grouping has a lower Gini index because its corresponding subsets are much purer.

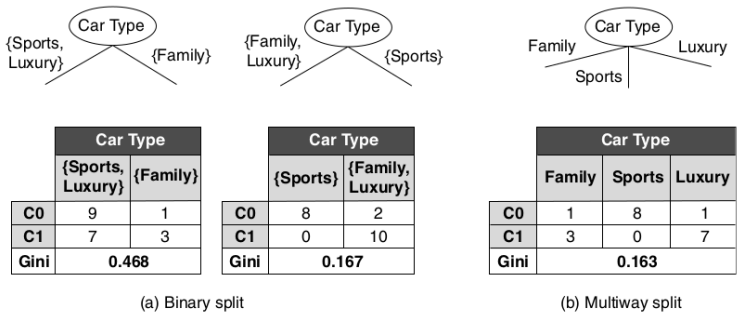


Figure 57: Splitting nominal attributes

For the multiway split, the Gini index is computed for every attribute value.
Since Gini({Family}) = 0.375, Gini({Sports}) = 0, and Gini({Luxury}) = 0.219, the overall Gini index for the multiway split is equal to $\frac{4}{20} \cdot 0.375 + \frac{8}{20} \cdot 0 + \frac{8}{20} \cdot 0.219 = 0.163$.
The multiway split has a smaller Gini index compared to both two-way splits. This result is not surprising because the two-way split actually merges some of the outcomes of a multiway split, and thus, results in less pure subsets.

Gain Ratio

Impurity measures such as entropy and Gini index tend to favor attributes that have a large number of distinct values (e.g. Figure 48).
Comparing the first test condition, `Gender`, with the second, `Car Type`, it is easy to see that `Car Type` seems to provide a better way of splitting the data since it produces purer descendent nodes.
However, if we compare both conditions with `Customer ID`, the latter appears to produce purer partitions.
Yet Customer ID is not a predictive attribute because its value is unique for each record. Even in a less extreme situation, a test condition that results in a large number of outcomes may not be desirable because the number of records associated with each partition is too small to enable us to make any reliable predictions.

There are two strategies for overcoming this problem.
The *first strategy* is to restrict the test conditions to binary splits only. This strategy is employed by decision tree algorithms such as CART.
Another strategy is to modify the splitting criterion to take into account the number of outcomes produced by the attribute test condition. For example, in the C4.5 decision tree algorithm, a splitting criterion known as gain ratio is used to determine the goodness of a split. This criterion is defined as follows:

$$\text{Gain ratio} = \frac{\Delta_{\text{info}}}{\text{Split Info}}$$

where

$$\text{Split info} = -\sum_{i=1}^{k} P(v_i) log_2 P(v_i)$$

and k is the number of splits.
For example, if each attribute value has the same number of records, then $\forall i : P(v_i) = \frac{1}{k}$ and the split information would be equal to $log_2(k)$. This example suggests that if an attribute produces a large number of splits, its split information will also be large, which in turn reduces its gain ratio.

This *Split Info* is also called the Adjusted Information Gain, since it is designed to overcome the disadvantage of Information Gain.

For the GINI-Index, we can also compute the quality of the split that we have made:

Quality of GINI:

- Used in decision tree algorithms CART, SLIQ, SPRINT
- When a node p is split into k partitions (children nodes), the quality of the split is computed as

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

  where n_i = number of records at child node i and n = number of records at node p

Chapter 8: Cluster Analysis: Basic Concepts and Algorithms

Appendix A: Linear Algebra

This section of the summary will consist of some basic techniques that have to be known to be able to successfully traverse the Data Mining course. These techniques are mostly based off of matrices, matrix multiplication and finding eigenvalues and its eigenvectors.

A.1 Some theory about vectors and matrices

A.1.1 Vectors spaces

In this summary, a vector space will be denoted as "V".
The vectors in the vector space V are called the *elements* of the vector space.
Elements within the vector space abide by the "properties of vectors":

- $\vec{a} + \vec{b} = \vec{b} + \vec{a}$
- $\vec{a} + \vec{0} = \vec{a}$
- $\vec{a} + (\vec{b} + \vec{c}) = (\vec{a} + \vec{b}) + \vec{c}$
- $\vec{a} + (-\vec{a}) = \vec{0}$
- $c(\vec{a} + \vec{b}) = c\vec{a} + c\vec{b}$

V is a collection of elements that can be:
1) added together in any combination
2) multiplied by scalars in any combination

Closure properties of vectors:
1) given $\vec{a} \in V$ and scalar c, then $c\vec{a} \in V$
2) given $\vec{a} \in V$ and $\vec{b} \in V$, then $\vec{a} + \vec{b} \in V$
These closure properties both need to be met to determine whether V is a vector space.
For (1), add any number times the vector, if it still is in the set of reals, requirement (1) is met. The same works for (2), only then with adding two vectors.

Vector spaces can also be made of functions for the set of linear polynomials of the form $ax + b$
1) $c(ax + b) = (ca)x + (bc)$
2) $(a_1x + b_1) + (a_2x + b_2) = (a_1 + a_2)x + (b_1 + b_2)$
All are real numbers, meaning they are closer under the properties, thus able to create a vector space.

A.1.2 Basis vectors, linear combinations and span

In algebra, *basis vectors* are also known as unit vectors (vectors with length 1), or in this summary denoted as:

- $\hat{i}$, in the x-direction
- $\hat{j}$, in the y-direction

Thus, instead of thinking as a vector just being length x, it is a vector of length 1, multiplied by a scalar of length x.
For instance imagine the coordinate $\begin{bmatrix} 3 \\ -2 \end{bmatrix}$.
Instead of just saying $\begin{bmatrix} 3 \\ -2 \end{bmatrix}$, we say its $3\hat{i}$ and $-2\hat{j}$, meaning that this coordinate can be described as the sum of the two vector, written as $3\hat{i} + +(-2)\hat{j}$.

We can call the possibility of combining unit vectors a linear combination.
So in general, *Linear combination* is of the form: $\vec{v}$ and $\vec{w}$: $a\vec{v} + b\vec{w}$ or $a\hat{i} + b\hat{j}$.

The span of $\hat{i}$ and $\hat{j}$ is the set of all linear combinations where $a\hat{i} + b\hat{j}$ holds and where a and b are real numbers.

A.2 Working with matrices

A.2.1 Dot product, transposing and orthogonality

The dot product or scalar product is an algebraic operation that takes two equal-length sequences of numbers (usually coordinate vectors), and returns a single number, meaning that a dot product is different from *matrix multiplication.*
The dot product, following what CS students learn in the course Matrix Multiplication, is written as $< v, w >$.
Thus for example, if $\vec{v}: \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\vec{w}: \begin{bmatrix} -3 \\ 4 \end{bmatrix}$
Then $< v, w >= 2 \cdot (-3) + 1 \cdot 4 = -2$
Or if $\vec{x}: \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix}$ and $\vec{y}: \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}$
Then $< x, y >= (-2) \cdot 2 + 1 \cdot 1 + 3 \cdot (-1) = -6$

Transposing a matrix basically means that the rows of a matrix become columns and columns become rows.

For example $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, then $A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

Two vectors are orthogonal if their dot product is equal to 0.
For instance, if $\vec{a}: \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$ and $\vec{b}: \begin{bmatrix} 3 \\ -2 \\ -1 \end{bmatrix}$
Then $< a, b >= 1 \cdot 3 + 2 \cdot (-2) + (-1) \cdot (-1) = 0$.

A.2.2 Length, angle and projection

The length of a vector is calculated as the following: $||u|| = \sqrt{u \cdot u} = \sqrt{u_1^2 + u_1^2 + \cdots + u_n^2}$

For example. if $\vec{u} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$
Then $||u|| = \sqrt{2^2 + 1^2 + 3^2} = \sqrt{14}$

The angle between two vectors is calculated as the following: $cos(\theta) = \frac{<u,v>}{||u|| \cdot ||v||}$
This means we need to find the dot product of the two vectors and the individual lengths of both vectors.

If we take $\vec{v} = (1, 1)$ and $\vec{w} = (-1, -1)$
Then $< v, w >= 1 \cdot -1 + 1 \cdot -1 = -1 - 1 = -2$
$||v|| = \sqrt{1^2 + 1^2} = \sqrt{2}$ and $||w|| = \sqrt{(-1)^2 + (-1)^2} = \sqrt{2}$
So, $cos(\theta) = \frac{-2}{\sqrt{2} \cdot \sqrt{2} = \frac{-2}{2}} = -1$, thus $\theta = \pi$

The projection of a vector a on (or onto) a nonzero vector b, is the orthogonal projection of a onto a straight line parallel to b.

The projection of u on v is calculated as the following: $\frac{<u \cdot v>}{||v||^2} \cdot v$
If $\overrightarrow{u} = (1,0)$ and $\overrightarrow{v} = (2,1)$
Then $< u, v >= 1 \cdot 2 + - \cdot 1 = 2$
$||v||^2 = (\sqrt{2^2 + 1^2})^2 = (\sqrt{5})^2 = 5$

$u_v = \frac{<u \cdot v>}{||v||^2} \cdot v = \frac{2}{5} \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{4}{5} \\ \frac{2}{5} \end{bmatrix}$

Note: if you would project v on u, the projected vector is not the same as the vector when projecting u on v.

A.2.3 Matrix multiplication

Matrix multiplication can be only performed if the number of rows of matrix A is equal to the number of columns of vector B. The basic technique is to multiply each value in the first row of matrix A with the value of the first column in matrix B and add up these values.

Lets say that

$$C = \begin{bmatrix} -1 & 2 & 3 \\ 2 & -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 4 \\ 0 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 \cdot 5 + 2 \cdot 0 + 3 \cdot -1 & -1 \cdot 4 + 2 \cdot 2 + 3 \cdot 1 \\ 2 \cdot 5 + -2 \cdot 0 + 1 \cdot -1 & 2 \cdot 4 + -2 \cdot 2 + 1 \cdot 1 \end{bmatrix} = \begin{bmatrix} -5 + 0 - 3 & -4 + 4 + 3 \\ 10 + 0 - 1 & 8 - 4 + 1 \end{bmatrix} = \begin{bmatrix} -8 & 3 \\ 9 & 5 \end{bmatrix}$$

A.2.4 Inverse of a matrix

The inverse of A is A_{-1} iff $AA^{-1} = A^{-1}A = I$

The general computation for the inverse is as follows: $\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{1}{ad-bc} \cdot \begin{bmatrix} -d & -b \\ -c & a \end{bmatrix}$ where $ad - bc$ is the Determinant or just "D".

For example $E = \begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}$

Then $E^-1 = \begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}^{-1} = \frac{1}{4 \cdot 6 - 7 \cdot 2} \cdot \begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \frac{1}{10} \cdot \begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$

If $A \cdot A^{-1} = I$ and $A^{-1} \cdot A = I$, then we know that A^{-1} is the inverse of A.

$$\begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix} \cdot \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix} = \begin{bmatrix} 4 \cdot 0.6 + 7 \cdot -0.2 & 4 \cdot -0.7 + 7 \cdot 0.4 \\ 2 \cdot 0.6 + 6 \cdot -0.2 & 2 \cdot -0.7 + 6 \cdot 0.4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

A.2.5 Finding the eigenvalues and eigenvectors

If we have a square n x n matrix A:
Then $A\overrightarrow{x} = \lambda \overrightarrow{x}$ where $\overrightarrow{x}$ is the eigenvector(s) and λ the eigenvalue(s).
Each eigenvalue is associated with a specific eigenvector. This means that for every eigenvalue you find, you have to find its associated eigenvector which can lead to a lot of work.

The goal is to ultimately find the eigenvectors with the formula $(A - \lambda I)\overrightarrow{x} = 0$, but to find those we first need to find the values for λ.

Since we don't know $\vec{x}$ yet, we use $|A-\lambda I|=0$, to find the eigenvalues.

If we take $A=\begin{bmatrix}1 & 1\\ 4 & 1\end{bmatrix}$

Then we get $\begin{bmatrix}1 & 1\\ 4 & 1\end{bmatrix}-\lambda\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 1\\ 4 & 1\end{bmatrix}-\begin{bmatrix}\lambda & 0\\ 0 & \lambda\end{bmatrix}=\begin{bmatrix}1-\lambda & 1\\ 4 & 1-\lambda\end{bmatrix}$

Then we use the determinant $ad-bc$:

$det(A)=ad-bc=((1-\lambda)\cdot(1-\lambda))-4\cdot 1=\lambda^2-2\lambda+1-4=\lambda^2-2\lambda-3$

$\lambda^2-2\lambda-3=0$

$(\lambda+1)(\lambda-3)=0$

$\lambda=-1\vee\lambda=3\rightarrow$ these are the eigenvalues

Then we plug in the eigenvalues in the original formula $(A-\lambda I)\vec{x}=0$ to find the eigenvectors.

For $\lambda=-1$ we get $(A+I)\vec{x}=0$. We then want to solve for $A+I=0$ which gives $\begin{bmatrix}1 & 1\\ 4 & 1\end{bmatrix}+\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}=\begin{bmatrix}2 & 1\\ 4 & 2\end{bmatrix}$

We then perform Gauss-elimination to make it easier to find values for the eigenvector. How to perform Gauss-elimination won't be shown here, but after seeing the example it shouldn't be too hard to remember how it was done.

$$\begin{bmatrix}2 & 1\\ 4 & 2\end{bmatrix}\xrightarrow{R_2:=R_2-2R_1}\begin{bmatrix}2 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=(A+I)\vec{x}$$

So, this gives us the equation $2x_1+x_2=0$

$x_2=-2x_1$

So, if for instance $x_1=1$,, then $x_2=-2, so\ \vec{x}=\begin{bmatrix}1\\ -2\end{bmatrix}$, which is one of the possible eigenvectors when the value for $\lambda=-1$.

The computation for when $\lambda=3$ won't be worked out here, but the steps that have to be taken to find the eigenvector is the same as it was for $\lambda=-1$.

A.2.6 Diagonalizing a matrix

Diagonalizing a matrix is the last step that we need before we can properly move on to the next section, meaning this part is also the one that involves all what is discussed before.

To diaganolize a matrix we:

1. Compute each eigenvalue $\lambda_1,\lambda_2,\ldots,\lambda_n$ by solving the characteristic polynomial.
2. For each eigenvalue, compute the associated eigenvector.
3. Write down A as the product of three matrices:
   $A=T_{B\rightarrow S}\cdot D\cdot T_{S\rightarrow B}$
   where:
   - $T_{B\rightarrow S}$ has the eigenvectors $v_1,\ldots,v_n$ (in order) as its columns.
   - D has the eigenvalues down its diagonal and 0's everywhere else.
   - $T_{S\rightarrow B}$ is the inverse of $T_{B\rightarrow S}$.

For an example of diagonalization I'm going to use the matrix and eigenvectors found under the *Finding the eigenvalues and eigenvectors* section.

Thus, we have the eigenvector $\begin{bmatrix} 1 \\ -2 \end{bmatrix}$ for the value $\lambda = -1$ and the eigenvector $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ for the value $\lambda = 3$.

Matrix $D = \begin{bmatrix} -1 & 0 \\ 0 & 3 \end{bmatrix}$

Matrix $T_{B\to S} = \begin{bmatrix} 1 & 1 \\ -2 & 2 \end{bmatrix}$

Matrix $T_{S\to B} = (T_{B\to S})^{-1} = \begin{bmatrix} 1 & 1 \\ -2 & 2 \end{bmatrix}^{-1} = \frac{1}{1\cdot 2-(-2)\cdot 1} \cdot \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} = \frac{1}{4} \cdot \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix}$

So, $A = \begin{bmatrix} 1 & 1 \\ -2 & 2 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} 1\cdot -1+1\cdot 0 & 1\cdot 0+1\cdot 3 \\ -2\cdot -1+2\cdot 0 & -2\cdot 0+2\cdot 3 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} -1 & 3 \\ 2 & 6 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix} =$

$\begin{bmatrix} -1\cdot\frac{1}{2}+3\cdot\frac{1}{2} & -1\cdot -\frac{1}{4}+3\cdot\frac{1}{4} \\ 2\cdot\frac{1}{2}+6\cdot\frac{1}{2} & 2\cdot -\frac{1}{4}+6\cdot\frac{1}{4} \end{bmatrix} =$

$\begin{bmatrix} 1 & 1 \\ 5 & 1 \end{bmatrix} \rightarrow$ the diagonalized matrix

A.2.7 Normalizing a vector

You may come across something called 'normalizing a vector'. What this means is that the magnitude of the vector is changed to that it is easier to work with (to a certain extend).

The way we do this is by dividing each element in the vector by the sum of squares of all the elements in the vector.

For instance if we have $\vec{x} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ 2 \end{bmatrix}$, then $SS(x)$ (sum of all squares) $= 1^2 + 4^2 + 3^2 + 2^2 = 30$, thus $\vec{x}$ will be normalized to 30.

We then get the normalized vector $\overrightarrow{x_{norm}} = \begin{bmatrix} \frac{1}{\sqrt{30}} \\ \frac{4}{\sqrt{30}} \\ \frac{3}{\sqrt{30}} \\ \frac{2}{\sqrt{30}} \end{bmatrix}$

A.2.8 Rank of a matrix

The rank of a matrix is the maximal number of linearly independent columns of such matrix. This, in turn, is identical to the dimension of the vector space spanned by its rows.

The *column rank* of A is the dimension of the column space of A, while the *row rank* of A is the dimension of the row space of A.

The following matrix: $\begin{bmatrix} 1 & 0 & 1 \\ -2 & -3 & 1 \\ 3 & 3 & 0 \end{bmatrix}$

has rank 2: the first two columns are linearly independent, so the rank is at least 2, but since the third is a linear combination of the first two (the second subtracted from the first), the three columns are linearly dependent so the rank must be less than 3.

Appendix B: Dimensionality Reduction

This chapter will cover Principal Component Analysis (PCA), Singular Value Decomposition (SVD) and other reduction techniques like Factor Analysis, locally linear embedding (LLE) and others. Namely *PCA* and *SVD* are very important techniques to understand and to be able to work with them, since they will be guaranteed to appear on the exam.

Before going through this chapter its important to understand everything which is talked about in Chapter 3 Linear Algebra, since many techniques of that are used here.

We begin with a discussion of Principal Components Analysis (PCA) and Singular Value Decomposition (SVD). These methods are described in some detail since they are among the most commonly used approaches and we can build on the discussion of linear algebra in Chapter 3. However, there are many other approaches that are also employed for dimensionality reduction, and thus, we provide a quick overview of several other techniques.

B.1 Eigenvalue Decomposition

Matrix operations such as transformations or multiplications are computationally expensive. In applications such as machine learning, you often have thousands or millions of dimensions. Imagine you have to perform matrix transformations repeatedly on matrices in millions of dimensions. Even the best computers quickly reach their limits. But as we discussed before, operations are much simpler on diagonal matrices. So if we can decompose a matrix into a diagonal form before we apply any kind of costly operation, it makes our lives, as well as the lives of our computers, much easier. This is where we use Eigenvalue Decomposition for.
Know that Eigenvalue Decomposition can only be applied if the matrix is square!

The following video is a good introduction to EVD and is best to watch before reading any further: https://www.youtube.com/watch?v=KTKAp9Q3yWg&list=PL4HCKYuyhl5g7SfJExtQ9w80O5FOTl0wR&index=2&t=2s.

So to perform Eigenvalue Decomposition we use the set up which you use to diagonalize a matrix ($A = T_{B\to S} \cdot D \cdot T_{S\to B}$), but the difference between diagonalizing a matrix and decomposing it, is that EVD is used when we want to factor the matrix n times.

So, instead of having to multiply one matrix lets say 8 times, you only have to factor the matrix that is in the middle (the D matrix) consisting of the eigenvalues. This means that the computation times gets incredibly reduced the more a matrix has to be factored.

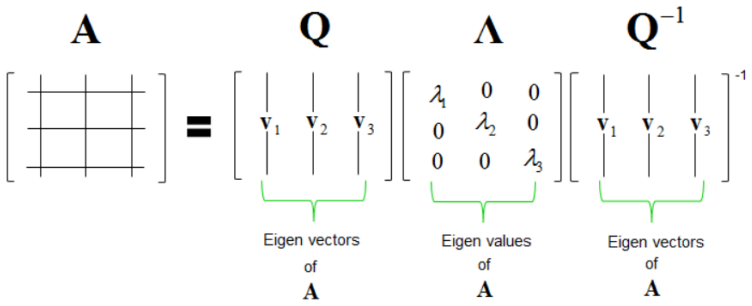


Figure 58: Defined equation for EVD and Diagonalization

So, in short: if you know how to diagonalize a matrix, then performing Eigenvalue Decomposition is a piece of cake.

B.2 Singular Value Decomposition

Singular Value Decomposition is similar to Eigenvalue Decomposition, only that SVD is not limited to just square matrices, which Eigenvalue Decomposition is. SVD is used for dimensionality reduction and to reduce the size of the matrix, since small singular values will be put to zero.

The reason for this is because when a matrix is very large, values that are very small don't have much added value to ultimately representing the data, hence they get put to 0, so that they don't clutter the data which is more valuable/usable.

The easiest way to understand SVD is to watch a few videos of the series about SVD from Steve Brunton: https://www.youtube.com/watch?v=gXbThCXjZFM&list=PLMrJAkhIeNNSVjnsviglFoY2nXildDCcv&index=1, since understanding the topic is as important as it is to know how to compute the SVD.

For a clear example you can watch the following video from Bernard Meulenbroek (with a strong Dutch accent): https://www.youtube.com/watch?v=HeGdlgB8450.

B.3 Principal Component Analysis

Principal Component Analysis is used for three main reasons:

- Dimensionality reduction
- Improvement of data visualization
- Feature extraction (creating a new attribute out of other attributes)

A nice overview video is the following:

https://www.youtube.com/watch?v=pmG4K79DUoI

And for the mathematics behind it:

https://www.youtube.com/watch?v=dhK8nbtii6I

A great article about PCA with an example:

https://towardsdatascience.com/
the-mathematics-behind-principal-component-analysis-fff2d7f4b643

Principal component analysis is a method that rotates the dataset in a way such that the rotated features are statistically uncorrelated. This rotation is often followed by selecting only a subset of the new features, according to how important they are to explaining the data.

The algorithm PCA uses proceeds by first finding the direction of maximum variance (often labeled as *component 1*, as is seen in the following figure:

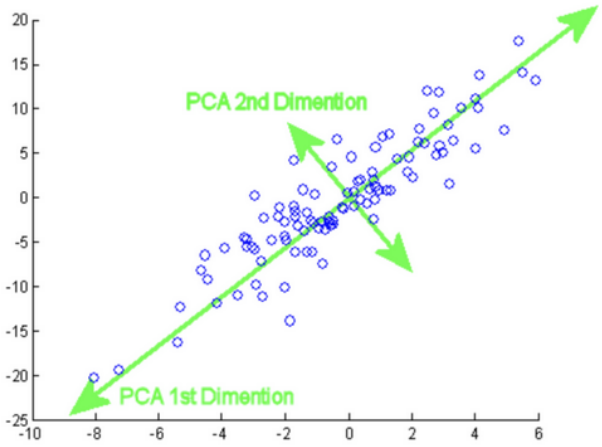


Figure 59: Two components showing that the most variance is that of the first component

This is the direction along which the features are most correlated with each other. Then, the algorithm finds the direction that contains the most information while being orthogonal to the first direction. The component is often pictured as an arrow, but it doesn't really matter where the head or the tail is. They are called *principal components*, since they are the main directions of variance in the data.

We can use PCA for dimensionality reduction by retaining only some of the principal components (in figure 59 it is 2). Since the first component contains the most interesting and important data, it would be most logical, if we only want one feature, to remove the second component.

Appendix C: Probability and Statistics

COMING SOON ...

Appendix D: Regression

COMING SOON ...

Appendix E: Optimization

COMING SOON ...

E-Chapter I: Extra resources

In this section I will provide some resources that I have used myself which I haven't mentioned in the sections themselves.

E-Chapter I.1: Books

Here I will list the books that I have either used while making the summary or the books that are interesting to read through in case you are interested in going more in-depth with machine learning, statistics or deep learning.

- Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow by Aurélien Géron.
  For the fundamentals of machine learning and to see how different classifiers are used.
  https://www.knowledgeisle.com/wp-content/uploads/2019/12/2-Aur%C3%A9lien-G%C3%A9ron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow_-Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-O%E2%80%99Reilly-Media-2019.pdf
- Mathematical Statistics with Applications.
  For matrix algebra and general statistical mathematics.
  https://webéducation.com/wp-content/uploads/2018/02/Dennis-D.-Wackerly-William-Mendenhall-Richard-L.-Scheaffer-Mathematical-Statistics-with-Applications-Cengage-Learning-2008.pdf
- Data Science from Scratch.
  To get started with Python and to see some of the fundamentals in action.
  https://www.m-fozouni.ir/wp-content/uploads/2020/08/Joel_Grus_Data_Science_from_Scratch_First_Princ.pdf
- Introduction to Machine Learning with Python.
  No easy link to be found.

E-Chapter I.2: YouTube channels and Video's

Here I will link the Youtube channels and possibly specific videos which have helped me understand all the theory. This list will be updated over time since new topics will be explained by different people.

- Daniel Bourke:
  Daniel has several (long) videos where he goes in-depth in deep learning and machine learning. In these videos he explains the topics but also codes with them in Python.
  https://www.youtube.com/channel/UCr8O8l5cCX85Oem1d18EezQ
- StatQuest:
  As will be mentioned in *E-Chapter II Undiscussed topics in Chapters 1-4 for the Midterm exam*, the StatQuest channel has great videos about machine learning and about many of the topics that we need to know for the Data Mining course (and he sings songs in the beginning which is kinda cool).
  https://www.youtube.com/c/joshstarmer

E-Chapter I.3: Articles and websites

In this section I will link articles and websites that have helped me understand the theory or websites with some exercises to try.

- For practicing with the similarity measures.
  https://csucidatamining.weebly.com/assign-3.html

E-Chapter II: Undiscussed topics in Chapters 1-4 for the Midterm Exam

E-Chapter II.1: Extra's from Chapter 5

For the midterm exam there are a few topics discussed in the lecture which belong to Chapter 5 of the book. Once we reach Chapter 5 the following discussed part will be missing since it's discussed here.

E-Chapter II.1.1: Overfitting and Underfitting

This section will incorporate section *4.4 Model Overfitting* of the book, since it makes more sense to include it here then in Chapter 4 of the summary.

The errors committed by a classification model are generally divided into two types: *training errors* and *generalization errors*.
Definition:
Taining error, also known as **resubstitution error** or **apparent error**, is the number of misclassification errors committed on training records.
Generalization errors are the expected errors of the model on previously unseen records.

In other words, a good model must have low training error as well as low generalization error (the optimal situation). This is important because a model that fits the training data too well can have a poorer generalization error than a model with a higher training error, this then is called *model overfitting*.

Definition:
Overfitting: If a decision tree is too large. So, if we build a model which is too complex for the amount of information we have (making it too specific, complex etc.). Overfitting occurs when you fit a model that works well on the training set but is not able to generalize to new data.

Underfitting occurs because the model has yet to learn the true structure of the data. As a result, it performs poorly on both the training and the test sets. As the number of nodes in the decision tree increases, the tree will have fewer training and test errors.

Definition:
Underfitting would occur if the model is too simple, meaning the model will even do bad on the training set (so, overfitting is in theory better than underfitting, since it can at least do well on the training set).
Potential causes of model overfitting:

- Overfitting due to presence of noise
- Overfitting due to lack of representative samples

More definitions:
Tree pruning: reducing the size of the decision tree (discussed in the next section).

Data fragmentation: if, at the leaf nodes, the number of records may be too small to make a statistically significant decision about the class representation of the nodes. The way to solve this is to disallow splitting when the number of records fall below a certain threshold.

Oblique decision trees recursively divide the feature space by using splits based on linear combinations of attributes. Compared to their unvaried counterparts, which only use a single attribute per split, they are often smaller and more accurate.

E-Chapter II.1.2: Post-pruning and Pre-pruning

Having a reliable estimate of generalization error allows the learning algorithm to search for an accurate model without overfitting the training data. This section presents two strategies for avoiding model overfitting in the context of decision tree induction.

Pre-pruning (Early Stopping Rule):

In this approach, the tree-growing algorithm is halted before generating a fully grown tree that perfectly fits the entire training data. To do this, a more restrictive stopping condition must be used; e.g., stop expanding a leaf node when the observed gain in impurity measure (or improvement in the estimated generalization error) falls below a certain threshold.

The advantage of this approach is that it avoids generating overly complex subtrees that overfit the training data. Nevertheless, it is difficult to choose the right threshold for early termination. Too high of a threshold will result in underfitted models, while a threshold that is set too low may not be sufficient to overcome the model overfitting problem.

Post-pruning: In this approach, the decision tree is initially grown to its maximum size. This is followed by a tree-pruning step, which proceeds to trim the fully grown tree in a bottom-up fashion. Trimming can be done by replacing a subtree with

1. A new leaf node whose class label is determined from the majority class of records affiliated with the subtree
2. The most frequently used branch of the subtree.

The tree-pruning step terminates when no further improvement is observed.

Post-pruning tends to give better results than pre-pruning because it makes pruning decisions based on a fully grown tree, unlike pre-pruning, which can suffer from premature termination of the tree-growing process. However, for post-pruning, the additional computations needed to grow the full tree may be wasted when the subtree is pruned.

E-Chapter II.1.3: Evaluating the performance of a classifier

The estimated error helps the learning algorithm to do model selection; i.e., to find a model of the right complexity that is not susceptible to overfitting. Once the model has been constructed, it can be applied to the test set to predict the class labels of previously unseen records.
It is often useful to measure the performance of the model on the test set because such a measure provides an unbiased estimate of its generalization error. The accuracy or error rate computed from the test set can also be used to compare the relative performance of different classifiers on the same domain.
This section reviews some of the methods commonly used to evaluate the performance of a classifier.

So, we have three possible methods:

- **Holdout Method**
  In the holdout method, the original data with labeled examples is partitioned into two disjoint sets, called the training and the test sets, respectively.
  A classification model is then induced from the training set and its performance is evaluated on the test set. The proportion of data reserved for training and for testing is typically at the discretion of the analysts (e.g., 50-50 or two-thirds for training and one-third for testing). The accuracy of the classifier can be estimated based on the accuracy of the induced model on the test set.

  The holdout method has several well-known limitations.
  First, fewer labeled examples are available for training because some of the records are withheld for testing. As a result, the induced model may not be as good as when all the labeled examples are used for training.
  Second, the model may be highly dependent on the composition of the training and test sets. The smaller the training set size, the larger the variance of the model. On the other hand, if the training set is too large, then the estimated accuracy computed from the smaller test set is less reliable. Such an estimate is said to have a wide confidence interval.
  The holdout method can be repeated several times to improve the estimation of a classifier's performance. This approach is known as random subsampling.

- **Cross-Validation**
  In this approach, each record is used the same number of times for training and exactly once for testing. To illustrate this method, suppose we partition the data into two equal-sized subsets.
  First, we choose one of the subsets for training and the other for testing. We then swap the roles of the subsets so that the previous training set becomes the test set and vice versa. This approach is called a two fold cross-validation. The total error is obtained by summing up the errors for both runs. In this example, each record is used exactly once for training and once for testing.
  The k-fold cross-validation method generalizes this approach by segmenting the data into k equal-sized partitions.
  During each run, one of the partitions is chosen for testing, while the rest of them are used for training. This procedure is repeated k times so that each partition is used for testing exactly once. Again, the total error is found by summing up the errors for all k runs.

  A good video about cross-validation is made by StatQuest (BAM!) which helped me understand the approach better: https://www.youtube.com/watch?v=fSytzGwwBVw

- **Bootstrap**
  The methods presented so far assume that the training records are sampled without replacement. As a result, there are no duplicate records in the training and test sets. In the bootstrap approach, the training records are sampled with replacement; i.e., a record already chosen for training is put back into the original pool of records so that it is equally likely to be redrawn.

If the original data has N records, it can be shown that, on average, a bootstrap sample of size N contains about 63.2% of the records in the original data. This approximation follows from the fact that the probability a record is chosen by a bootstrap sample is $1-(1-\frac{1}{N})^N$. When N is sufficiently large, the probability asymptotically approaches $1-e^{-1}=0.632$. Records that are not included in the bootstrap sample become part of the test set.

Just as for cross-validation, StatQuest has made a video about boostrapping which is very helpful (Double BAM!): https://www.youtube.com/watch?v=Xz0x-8-cgaQ

E-Chapter II.1.4: Class imbalance metrics

Since the accuracy measure treats every class as equally important, it may not be suitable for analyzing imbalanced data sets, where the rare class is considered more interesting than the majority class. For binary classification, the rare class is often denoted as the positive class, while the majority class denoted as the negative class.
But, we do have a confusion matrix for a binary classification problem where the classes are not equally important. The matrix summarizes the number of instances predicted correctly or incorrectly by a classification model:

		Predicted Class	
		+	−
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	−	f_{-+} (FP)	f_{--} (TN)

Figure 60: A confusion matrix with unequal classes

The following terminology is often used when referring to the counts tabulated in a confusion matrix:

- True positive (TP) or f_{++} , which corresponds to the number of positive examples correctly predicted by the classification model.
- False negative (FN) or f_{+-} , which corresponds to the number of positive examples wrongly predicted as negative by the classification model.
- False positive (FP) or f_{-+} , which corresponds to the number of negative examples wrongly predicted as positive by the classification model.
- True negative (TN) or f_{--} , which corresponds to the number of negative examples correctly predicted by the classification model.

The counts in a confusion matrix can also be expressed in terms of percentages (**Important for the exam!**).
True positive rate (TPR) or **sensitivity**:

$$TPR = \frac{TP}{TP+FN}$$

True negative rate (TNR) or **specificity**:

$$TNR = \frac{TN}{TN+FP}$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP+TN}$$

False negative rate (FNR):

$$FNR = \frac{FN}{FN+TP}$$

Precision p:

$$p = \frac{TP}{TP + FP}$$

Recall r:

$$r = \frac{TP}{TP + FN}$$

Precision determines the fraction of records that actually turns out to be positive in the group the classifier has declared as a positive class. The higher the precision is, the lower the number of false positive errors committed by the classifier. Recall measures the fraction of positive examples correctly predicted by the classifier. Classifiers with large recall have very few positive examples misclassified as the negative class. In fact, the value of recall is equivalent to the true positive rate.

Precision and recall can be summarized into another metric known as the F_1 measure.
F_1 **measurement**:

$$F_1 = \frac{2rp}{r + p}$$

We also have the accuracy (correctly predicted instances) and error rate (incorrectly predicted instances).
Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Error rate:

$$\text{Error rate} = \frac{FP + FN}{TP + FP + TN + FN}$$

Example of a confusion matrix with the named metrics:

Model M_1		Predicted Class	
		Class +	Class -
Actual Class	Class +	150	40
	Class -	60	250

Figure 61: Example confusion matrix

$TPR = \frac{TP}{TP+FN} = \frac{150}{150+250} = \frac{150}{400} = \frac{3}{8} = 37,5\%$

$TNR = \frac{TN}{TN+FP} = \frac{250}{250+60} = \frac{250}{310} = \frac{25}{31} \approx 80\%$

$FPR = \frac{FP}{FP+TN} = \frac{60}{60+250} = \frac{60}{310} = \frac{6}{31}$

$FNR = \frac{FN}{FN+TP} = \frac{40}{40+150} = \frac{40}{190} = \frac{4}{19}$

$p = \frac{TP}{TP+FP} = \frac{150}{150+60} = \frac{50}{70} = \frac{5}{7}$

$r = \frac{TP}{TP+FN} = \frac{3}{8} = \text{TPR}$

$F_1\text{measure} = \frac{2rp}{r+p} = \frac{2 \cdot \frac{3}{8} \cdot \frac{5}{7}}{\frac{3}{8} + \frac{5}{7}} = \frac{2 \cdot \frac{15}{56}}{\frac{21}{56} + \frac{40}{56} = \frac{30}{56}{\frac{61}{56}}} \approx 49\%$

$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} = \frac{150+250}{150+250+40+60} = \frac{400}{500} = \frac{4}{5} = 0.8 = 80\%$

$\text{Error rate} = \frac{FP+FN}{TP+FP+TN+FN} = \frac{40+60}{40+60+150+250} = \frac{100}{500} = \frac{1}{5} = 0.2 = 20\%$

E-Chapter II.1.5: The Receiver Operating Characteristic Curve

A receiver operating characteristic (ROC) curve is a graphical approach for displaying the tradeoff between true positive rate and false positive rate of a classifier. In an ROC curve, the true positive rate (TPR) is plotted along the y axis and the false positive rate (FPR) is shown on the x axis (it's the combination of multiple confusion matrices). Each point along the curve corresponds to one of the models induced by the classifier.

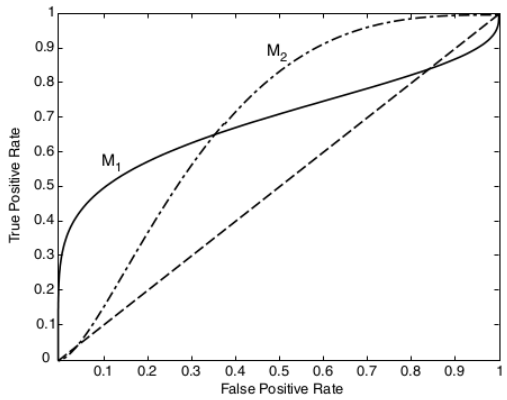


Figure 62: A confusion matrix with unequal classes

There are several critical points along an ROC curve that have well-known interpretations: $(TPR = 0, FPR = 0)$: Model predicts every instance to be a negative class.
$(TPR = 1, FPR = 1)$: Model predicts every instance to be a positive class.
$(TPR = 1, FPR = 0)$: The ideal model.

A good classification model should be located as close as possible to the upper left corner of the diagram, while a model that makes random guesses should reside along the main diagonal, connecting the points $(TPR = 0, FPR = 0)$ and $(TPR = 1, FPR = 1)$.

The area under the ROC curve (**AUC**) provides another approach for evaluating which model is better on average. If the model is perfect, then its area under the ROC curve would equal 1. If the model simply performs random guessing, then its area under the ROC curve would equal 0.5 ($1 \cdot 1 \cdot \frac{1}{2}$ or the area of a triangle with sides of length 1). A model that is strictly better than another would have a larger area under the ROC curve. This means, that if the ROC curve goes from the bottom left to the top left to the top right, we'd have the best possible curve.

Just as a few sections back, StatQuest has made a very good video explaining ROC curves and how to generate one (Triple BAM!): https://www.youtube.com/watch?v=4jRBRDbJemM