Data Analysis and Data Mining

Contents

What will you learn in this course?

This course is about data analysis. What does it mean? Nowadays every computer system is connected to something that is able to store data (datacenter) and thanks to the increasing power of computers we are now able to extract new information from all this data. We will see many techniques to extract meaning from data. Learning from data is something that cannot be solved by a deterministic algorithm and so we have to rely on statistic. Statistic provides us a way to construct a model that works not always but very often. In order to do so, humans are working on make intelligent machines which are able to discover useful information from big data. One famous case is the Turing test. This test has been recently passed by IBM machine. Although many advanced researches in this field, we are far from a true intelligence, because intelligence to be so, needs imagination. Today a machine isn't able to invent something new, is just able to represent in an intelligent way something that already knows. A machine cannot demonstrate a new theorem, it can only proves one that is already known. Artificial intelligence is another way to see data analysis. We will start from very basic concepts in order to understand all intelligent things existing in the technological world around us.

Material

More information regarding the course can be found in [1, 4, 2, 3, 5, 7, 6, 8].

An Introduction to Data Mining

We are drowning in information and starving for knowledge. Rutherford D. Roger

Vast amounts of data are being generated in many fields, and the statistician's job is to make sense of it all: to extract important patterns and trends, and understand "what the data says". We call this *learning from data* or *data analysis*. Data mining is the study of collecting, cleaning, processing, analyzing, and gaining useful insights from data. The deluge of data is a direct result of advances in technology and the computerization of every aspect of modern life. It is, therefore, natural to examine whether one can extract concise and possibly actionable insights from the available data for application-specific goals. This is where the task of data mining comes in. The raw data may be arbitrary, unstructured, or even in a format that is not immediately suitable to be processed by an automated computer program to gain insights. To address this issue, data mining analysts use a pipeline of processing, where the raw data are collected, cleaned, and transformed into a standardized format. This pipeline of processing is conceptually similar to that of an actual mining process from a mineral ore to the refined end product. The term *"mining"* derives its roots from this analogy. Data mining applications are often closely connected to one of four "superproblems": association pattern mining, clustering, classification, and outlier detection. These problems are so important because they are used as building blocks in a majority of the applications in some indirect form or the other. In a typical scenario, we have an outcome measurement, usually quantitative or categorical, that we wish to predict based on a set of *features*. We have a *training set* of data, in which we observe the outcome and feature measurements for a set of objects. Using this data we build a prediction model, or *learner*, which will enable us to predict the outcome for new unseen objects. A good learner is one that accurately predicts such an outcome. It is called "supervised" because of the presence of the outcome variable to guide the learning process. In the unsupervised learning problem, we observe only the features and have no measurements of the outcome.

1 Data, Uncertainty and Learning Problems

Not everything that can be counted counts, and not everything that counts can be counted. Albert Einstein

1.1 Variable types and terminology.

Data is some information characterizing something else. Data format can be *structured* or *unstructured* . Structured data refers to information with a high degree of organization, such that inclusion in a relational database is seamless and readily searchable by simple.Unstructured data is essentially the opposite: it refers to information that doesn't reside in a traditional row-column database. Unstructured data files often include text and multimedia content. Examples include e-mail messages, word processing documents, videos, photos, audio files, presentations, webpages and many other kinds of business documents. Note that while these sorts of files may have an internal structure, they are still considered "unstructured" because the data they contain doesn't fit neatly in a database. The data may also have different types.The type may be *quantitative* (e.g., age), *categorical* (e.g., ethnicity), text, spatial, temporal, or graph-oriented. The type is a *quantitative* measurement, when some measurements are bigger than others, and measurements close in value are close in nature.The type is qualitative (or *categorical*) if it assumes values in a finite set G, also called *class*. There is no explicit ordering in the classes, and in fact often descriptive labels rather than numbers are used to denote the classes. Qualitative variables are typically represented numerically by codes. The easiest case is when there are only two classes or categories, such as success or failure. These are often represented by a single binary digit or bit as 0 or 1, or else by -1 and 1. For reasons that will become apparent, such numeric codes are sometimes referred to as *targets*. When there are more than two categories, several alternatives are available. The most useful and commonly used coding is via dummy variables. Here a K-level qualitative variable is represented by a vector of K binary variables or bits, only one of which is on at a time.
Let's assume we have to predict a temperature. If we make a mistake and we predict for example 5 degrees instead of 7 degrees, is not so relevant and we can measure how much our prevision is wrong because we have a reference scale for temperature.Now let's assume that we have three classes: killer, student and psychopathic. A categorical variable can assume one of this value. Is it a great mistake predict one category instead of another one? Yes, because classes are different things and cannot be compared. So in order to keep the right distance between classes we can represent this information with vectors. For example we represent student with 001, killer with 010 and psychopathic with 100. In this way there is the same distance between all classes.
For both types of outputs it makes sense to think of using the inputs to predict the output. This distinction in output type has led to a naming convention for the prediction tasks: *regression* when we predict quantitative outputs, and *classification* when we predict qualitative outputs.

1.1.1 Supervised/Unsupervised learning problems.

Prediction is very difficult, especially if it's about the future. Nils Bohr

The first step of data analysis is to have a data set, possibly very large. The second one is to make uniform this data because typically we have data in file ascii, or file excel, or relational database, or datawarehouse, or in non-relational db. Once we have data in the format we want, what we can do with those data?

- **Unsupervised Learning Problem:**
  In unsupervised learning, there is no outcome measure, and the goal is to describe the associations and patterns among a set of input measures.

- Data Clustering
  Given a data matrix D , partition its rows (records) into sets $C_1 \cdots C_k$, such that the rows (records) in each cluster are "similar" to one another. We have intentionally provided an informal definition here because clustering allows a wide variety of definitions of similarity, some of which are not cleanly defined in closed form by a similarity function. A clustering problem can often be defined as an optimization problem, in which the variables of the optimization problem represent cluster memberships of data points, and the objective function maximizes a concrete mathematical quantification of intragroup similarity in terms of these variables.
- Novelty detection
  Given a data matrix D, determine the rows of the data matrix that are very different from the remaining rows in the matrix. An outlier is a data point that is significantly different from the remaining data. Hawkins formally defined the concept of an outlier as follows: "An outlier is an observation that deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". Outliers are also referred to as abnormalities, discordants, deviants, or anomalies in the data mining and statistics literature.The outlier detection problem is related to the clustering problem by complementarity. This is because outliers correspond to dissimilar data points from the main groups in the data. On the other hand, the main groups in the data are clusters.

- **Supervised Learning Problem:**
In supervised learning, the goal is to predict the value of an outcome measure based on a number of input measures; Different main methods to solve supervised learning problems are classification, regression and forecast. In the classification method are considered categorical data and so there isn't an order while in the regression one exist an order and so we can speak about the concept of distance. Forecast is different from regression because in forecast there is the concept of time. Moreover forecast can be a short term prediction or long term prediction. Let's see in more detail supervised methods.

  - Data Classification
    Given an $n \times d$ training data matrix D (database D), and a class label value in $\{1 \cdots k\}$ associated with each of the n rows in D (records in D), create a training model M, which can be used to predict the class label of a d-dimensional record. Many data mining problems are directed toward a specialized goal that is sometimes represented by the value of a particular feature in the data. This particular feature is referred to as the class label. Therefore, such problems are supervised, where in the relationships of the remaining features in the data with respect to this special feature are learned. The data used to learn these relationships is referred to as the training data. The learned model may then be used to determine the estimated class labels for records, where the label is missing. The record whose class label is unknown is referred to as the test record.
  - Regression
    Let D be an $n \times d$ data matrix whose ith data point (row) is the d-dimensional input feature vector X_i, and the corresponding response variable is y_i. Let the n-dimensional column-vector of response variables be denoted by $y = (y_1, \cdots , y_n)^T$. In linear regression, the dependence of each response variable y_i on the corresponding independent variables X_i is modeled in the form of a linear relationship.
  - Forecast
    Forecasting is one of the most common applications of time series analysis. The prediction of future trends has applications in retail sales, economic indicators, weather forecasting, stock markets, and many other application scenarios. In this case, we have one or more series of data values, and it is desirable to predict the future values of the series using the history of previous values.

- **Semisupervised Learning Problem:**
In these cases, unlabeled examples are used to improve the effectiveness of classifiers. Although unlabeled data does not contain any information about the label distribution, it does contain a significant amount of information about the manifold and clustering structure of the underlying data. Because the classification problem is a supervised version of the clustering problem, this connection can be leveraged to improve the classification accuracy. The core idea is that in most real data sets, labels vary in a smooth way over dense regions of the data. The determination of dense regions in the data only requires unlabeled information. For example we have a matrix and for half samples we know the class and for the remaining samples I have to predict it.

  - Active Learning
    In real life, it is often expensive to acquire labels. In active learning, the user (or an oracle) is actively involved

in determining the most informative examples for which the labels need to be acquired. Typically, these are examples that provide the user the more accurate knowledge about the uncertain regions in the data, where the distribution of the class label is unknown. In active learning the machine starts to learn like humans. One examples is amazon's AI that learns through like or rank mechanism and on the base of this information is able to suggest to you correlated products.

– Reinforcement Learning
  Reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. Reinforcement learning is different from supervised learning. Supervised learning is learning from examples provided by a knowledgeable external supervisor. This is an important kind of learning, but alone it is not adequate for learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. One of the challenges that arise in reinforcement learning and not in other kinds of learning is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future. The agent must try a variety of actions and progressively favor those that appear to be best.

1.2 Statistic

Statistic is a way to measure the uncertainty of the world. Luca Oneto

1.2.1 Introduction

Let's start with an example to introduce the concept of mean and variance. The high of a person can be expressed by $h = 1.60 + -0.01$ at 95% m. What does it mean? It means that the high is 1 m and 60 cm with an uncertainty of 1 cm 95% of times, or in other words with a probability of 0, 95. In this case 1, 60 is the mean value (μ) and 0.01 is the variance or the standard deviation (σ) and 95% is the confidence often written as $(1 - \delta)$. So in this case $\delta = 0.05$ which is a very common value for the confidence.The mean value is defined by its mathematical formula, it is defined with the fact that I take a lot of measures, I sum them and I divide by the number of measurements. The only connection between mean and variance is that they are computed over the same set of samples, while the confidence can't be computed from the samples that I have. Note that the confidence is not completely independent from μ and σ. If σ is big then probably my measurement is correct and so δ is high. How to chose δ? I need some assumptions, for example it can be assumed that my sample are a good representation of the measurements that can be collected or it can be assumed that the samples have a Gaussian distribution. Now we are able to define mean.

1.2.2 Mean

The mean value is a random variable (every time I take a measure is a random variable). The true value according to the frequency theory is deterministic, in the sense that even if the measure is random, the value is deterministic. (Note that in the bayesian statistic is the opposite: the true value is not deterministic while the measure it is. In this course it will be used the frequency theory). In particular we have a random variable $x \in R$ and we know its distribution represented by the probability density function $p(x)$. Note that $p(x)$ is a deterministic variable and completely defines the random variable. The true mean value in the case $x \in R$ it can be computed as

$$\mu = \int_{-\infty}^{+\infty} t\, p(t)\, \mathrm{dt} \tag{1}$$

while, if $x \in$ finite set of values,

$$\mu = \sum_{t \in X} t\, p(t) \tag{2}$$

Another way to express the mean is to use the expected value. It means that you have to sample x from this distribution infinite number of times e than take the mean value. Obviously it's something that it can't be done because we cannot sample infinite number of times.

$$\mathbb{E}_x\{x\} \tag{3}$$

What it can be compute is the empirical average. Note that, in this formulation, the probability is encapsulated in the fact that the values of x that are more probable will appear more times in this sum respect to the value of x that have smaller probability.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{4}$$

Moreover, μ is the empirical unbiased estimator of the mean value. i.e is equal to the expected value respect $x_1, x_2, ..., x_n$ of $\bar{x}$.

$$\mu = \mathbb{E}_{x_1,...x_N}\{\bar{x}\} \tag{5}$$

It means that if we sample n samples, compute the average value, and repeat this procedure infinite number of times and finally take the average value of all these averages we obtain the true value of the mean. In particular it can be prove by just applying the definition:

$$\begin{aligned}\mu &= \mathbb{E}_{x_1,\cdots,x_N}\{\bar{x}\} \\ &= \mathbb{E}_{x_1,\cdots,x_N}\{\frac{1}{N}\sum_{i=1}^{N} x_i\} \\ &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{x_i}\{x_i\} \\ &= \frac{1}{N}\sum_{i=1}^{N} \mu \\ &= \mu\end{aligned} \tag{6}$$

The expected value of the empirical average is the true average. This is the definition of the empirical unbiased estimator of the mean value. It also known as *first degree estimator*.

1.2.3 Variance

The variance is the measure of the distance of the samples from its mean value. The true variance is defined as

$$\sigma^2 = \int_{-\infty}^{+\infty} (t-u)^2 \, p(t) \, dt \tag{7}$$

or in terms of the expected value:

$$\sigma^2 = \mathbb{E}_x\{(x-\mu)^2\} \tag{8}$$

The expression for the variance can be expanded:

$$\begin{aligned}\sigma^2 &= \mathbb{E}_x\{(x-\mu)^2\} \\ &= \mathbb{E}_x\{x^2 - 2x\mu + \mu^2\} \\ &= \mathbb{E}_x\{x^2\} - \mathbb{E}_x\{2x\mu\} + \mathbb{E}_x\{\mu^2\} \\ &= \mathbb{E}_x\{x^2\} - 2\mu\mathbb{E}_x\{x\} + \mu^2 \\ &= \mathbb{E}_x\{x^2\} - 2\mu^2 + \mu^2 \\ &= \mathbb{E}_x\{x^2\} - \mu^2\end{aligned} \tag{9}$$

As we seen before the above quantity can't be really compute. The empirical unbiased estimator of the variance, or *second degree estimator* is given by :

$$\bar{s} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{10}$$

It can be proved that

$$\mathbb{E}_{x_1,\cdots,x_N}\{\bar{s}\} = \frac{N-1}{N}\sigma^2 \tag{11}$$

Proof.

$$\mathbb{E}_{x_1,\cdots,x_N}\{\bar{s}\} = \mathbb{E}_{x_1,\cdots,x_N}\{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2\} \tag{12}$$

$$\begin{aligned}
&= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{x_i,\cdots,x_N}\{(x_i - \bar{x})^2\} \\
&= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{x_i,\cdots,x_N}\{\left(x_i - \frac{1}{N} \sum_{j=1}^{N} x_j \right)^2\} \\
&= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{x_i,\cdots,x_N}\{x_i^2 - 2\frac{1}{N} x_i \sum_{j=1}^{N} x_j + \frac{1}{N^2} \sum_{j=1}^{N}\sum_{k=1}^{N} x_j x_k\} \\
&= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{x_i,\cdots,x_N}\{x_i^2 - \left(2\frac{1}{N} x_i^2 + 2\frac{1}{N} x_i \sum_{\substack{j=1\\ j\neq i}}^{N} x_j \right) + \left(\frac{1}{N^2} \sum_{j=1}^{N} x_j^2 + \frac{1}{N^2} \sum_{j=1}^{N}\sum_{\substack{k=1\\ k\neq j}}^{N} x_j x_k \right)\} \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[\mathbb{E}_{\mathbf{x}}\{\frac{N-2}{N} x_i^2\} - \mathbb{E}_{\mathbf{x}}\{\frac{2}{N} \sum_{\substack{j=1\\ j\neq i}}^{N} x_j x_i\} + \mathbb{E}_{\mathbf{x}}\{\frac{1}{N^2} \sum_{j=1}^{N} x_j^2\} + \mathbb{E}_{\mathbf{x}}\{\frac{1}{N^2} \sum_{j=1}^{N}\sum_{\substack{k=1\\ k\neq j}}^{N} x_j x_k\} \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{N-2}{N} \mathbb{E}_{\mathbf{x}}\{x_i^2\} - \frac{2}{N} \sum_{\substack{j=1\\ j\neq i}}^{N} \mathbb{E}_{\mathbf{x}}\{x_j x_i\} + \frac{1}{N^2} \sum_{j=1}^{N} \mathbb{E}_{\mathbf{x}}\{x_j^2\} + \frac{1}{N^2} \sum_{j=1}^{N}\sum_{\substack{k=1\\ k\neq j}}^{N} \mathbb{E}_{\mathbf{x}}\{x_j x_k\} \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{N-2}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} - \frac{2}{N} \sum_{\substack{j=1\\ j\neq i}}^{N} \mathbb{E}_{\mathbf{x}}\{\mu^2\} + \frac{1}{N^2} \sum_{j=1}^{N} \mathbb{E}_{\mathbf{x}}\{x^2\} + \frac{1}{N^2} \sum_{j=1}^{N}\sum_{\substack{k=1\\ k\neq j}}^{N} \mathbb{E}_{\mathbf{x}}\{\mu^2\} \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{N-2}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} - \frac{2\,(N-1)}{N} \mu^2 + \frac{N}{N^2} \mathbb{E}_{\mathbf{x}}\{x^2\} + \frac{N\,(N-1)}{N^2} \mathbb{E}_{\mathbf{x}}\{\mu^2\} \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{N-2}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} - \frac{2\,(N-1)}{N} \mu^2 + \frac{1}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} + \frac{(N-1)}{N} \mathbb{E}_{\mathbf{x}}\{\mu^2\} \right] \\
&= \left[\frac{N-2}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} - \frac{2\,(N-1)}{N} \mu^2 + \frac{1}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} + \frac{(N-1)}{N} \mathbb{E}_{\mathbf{x}}\{\mu^2\} \right] \\
&= \frac{N-1}{N} \mathbb{E}_{\mathbf{x}}\{x^2\} - \frac{(N-1)}{N} \mu^2 \\
&= \frac{N-1}{N} \left[\mathbb{E}_{\mathbf{x}}\{x^2 - \mu^2\} \right] \\
&= \frac{N-1}{N} \sigma^2 \, . \qquad (13)
\end{aligned}$$

In order to obtain the unbiased estimator of variance, we have to divide $\bar{s}$ for the gain. This term guarantees that $\bar{s}$ will converge to μ^2 even if N is small.

$$\bar{s}_x = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 \qquad (14)$$

We can also prove that increasing the number of samples, the accuracy of the estimation will be more precise.

$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{N}} \qquad (15)$$

Proof.

$$\begin{aligned}
\sigma_{\bar{x}}^2 &= \mathbb{E}_{\bar{x}}\{(\bar{x} - \mu)^2\} \qquad (16) \\
&= \mathbb{E}_{\bar{x}}\{\bar{x}^2\} - \mu^2 \\
&= \mathbb{E}_{x_i,\cdots,x_N}\{\left(\frac{1}{N} \sum_{i=1}^{N} x_i \right)^2\} - \mu^2
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{N^2}\mathbb{E}_{x_i,\cdots,x_N}\{\left(\sum_{i=1}^{N} x_i\right)^2\} - \mu^2 \\
&= \frac{1}{N^2}\,\mathbb{E}_{x_i,\cdots,x_N}\{\sum_{i=1}^{N}\sum_{k=1}^{N} x_i x_k\} - \mu^2 \\
&= \frac{1}{N^2}\,\mathbb{E}_{x_i,\cdots,x_N}\{\left(\sum_{i=1}^{N} x_i^2 + \sum_{j=1}^{N}\sum_{\substack{i=1\\ i\neq j}}^{N} x_j x_k\right)\} - \mu^2 \\
&= \frac{1}{N^2}\,\left(\sum_{i=1}^{N}\mathbb{E}_{x_i,\dots,x_N}\{x_i^2\} + \sum_{j=1}^{N}\sum_{\substack{i=1\\ i\neq j}}^{N}\mathbb{E}_{x_i,\cdots,x_N}\{x_j x_k\}\right)\} - \mu^2 \\
&= \frac{1}{N^2}\,\left(N\mathbb{E}_{x_i,\cdots,x_N}\{x^2\} + N\left(N-1\right)\mu^2\}\right) - \mu^2 \\
&= \frac{1}{N}\,\left(\mathbb{E}_{x_i,\cdots,x_N}\{x^2\} + \left(N-1\right)\mu^2\}\right) - \mu^2 \\
&= \frac{1}{N}\,\left(\mathbb{E}_{x_i,\cdots,x_N}\{x^2\} - \mu^2\}\right) \\
&= \frac{1}{N}\,\sigma_x^2
\end{aligned}
\tag{17}
$$

1.2.4 Confidence

Before continuing with definitions, just keep in mind that the confidence level describes the uncertainty associated with a sampling method. Suppose we used the same sampling method to select different samples and to compute a different interval estimate for each sample. Some interval estimates would include the true population parameter and some would not. A 95% confidence level means that 95% of the intervals would include the parameter. Pay attentions that this not means there is a 95% chance that the population mean falls between 100 and 200. This is incorrect. Like any population parameter, the population mean is a constant, not a random variable. It does not change. Now that the mean μ and standard deviation σ have been formally defined, is it possible to relate them in any way? At first glance, it may appear that the answer is no. For example, if the mean age of a certain population is known, that tells us nothing about the standard deviation σ. Likewise, knowing that the standard deviation of a population is, tells us absolutely nothing about the mean value. So it is perhaps somewhat surprising that there is in fact a relation of sorts between the two. While it may not be possible to derive one directly from the other, there is still something we can say, albeit very general. One of the most famous results in statistics asserts that almost all the values are near to the mean.

$$
\begin{aligned}
\sigma_{\bar{x}}^2 &= \int_{-\infty}^{+\infty} \left(\bar{x}-\mu\right)^2\, p\left(\bar{x}\right) d\bar{x} \\
&\geq \int_{|\bar{x}-\mu|\geq\varepsilon} \left(\bar{x}-\mu\right)^2\, p\left(\bar{x}\right) d\bar{x} \\
&\geq \int_{|\bar{x}-\mu|\geq\varepsilon} \varepsilon^2\, p\left(\bar{x}\right) d\bar{x} \\
&= \varepsilon^2 \int_{|\bar{x}-\mu|\geq\varepsilon} p\left(\bar{x}\right) d\bar{x} \\
&= \varepsilon^2\, \mathbb{P}\{|\bar{x}-\mu| \geq \varepsilon\}
\end{aligned}
\tag{18}
$$

This computation leads to the formulation of the *Chebyshev inequality*:

$$
\mathbb{P}\{|\bar{x}-\mu| \geq \varepsilon\} \leq \frac{\sigma_{\bar{x}}^2}{\varepsilon^2} = \frac{\sigma_x^2}{N\varepsilon^2} = \delta \tag{19}
$$

From the fact that if the probability that $a > b$ is δ and the probability that $a < b$ is $1-\delta$ the above inequality can be also expressed as:

$$
\mathbb{P}\{|\bar{x}-\mu| \leq \varepsilon\} \geq 1-\delta \tag{20}
$$

where ε is defined as follows:

$$
\varepsilon = \sqrt{\frac{\sigma_x^2}{N\delta}} \tag{21}
$$

In the end we can state that the confidence interval of the estimation is:

$$\mu \leq \bar{x} + \sqrt{\frac{\sigma_x^2}{N\delta}} \;\; with \; (1-\delta) \tag{22}$$

1.3 Regression

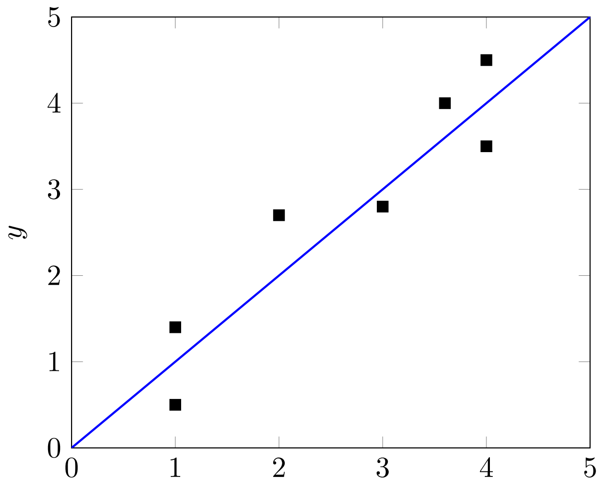


Figure 1: Linear regression

1.4 Regression function.

Let be $D_n = \{(x_1, y_1), ..., (x_1, y_1)\}$ the given data set and $y = f(x) + N$ the function which describe our model, where x is the independent variable which it can be measured while y is the dependent variable. The measurement is corrupt by a noise $N = G(0, \lambda^2)$ i.e the noise has a Gaussian distribution with zero mean and variance λ^2. The regression function $f(x)$ can be approximated in many ways, we have to choose the best one. For example, it can be approximated with a constant, with a straight line, with a polynomial or more in general with a function of degree p.

$$\begin{aligned}
f(x) &= c_0 \\
f(x) &= c_1 x + c_0 \\
f(x) &= c_2 x^2 + c_1 x + c_0 \\
&\vdots \\
f(x) &= \sum_{i=0}^{p} c_i x^i
\end{aligned} \tag{23}$$

1.5 Parameter estimation.

Here, $\underline{c} = (c_1 ... c_p)$ is a p-dimensional vector of coefficients that needs to be learned from the training data so as to minimize the least square error. The error is defined as the difference between the true function and the estimated one (we denote $f(x)$ with $\hat{y}$ to underline that is an estimation). The problem is to identify parameters in order to minimize this error. So the functional cost can be expressed by :

$$\min_f \int_{-\infty}^{+\infty} (y - f(x))^2 \, dx \tag{24}$$

but $f(x)$ is known only in few value of x, not for all. So this error can be compute empirically.

$$\min_f \sum_{i=1}^{N} (y_i - f(x))^2 \tag{25}$$

In this case, for example, if we assume $f(x) = c_0$, we have:

$$\min_f \sum_{i=1}^{N} (y_i - f(x))^2 = \min_{c_0} \sum_{i=1}^{N} (y_i - c_0)^2 = \min_{c_0} \|\underline{y} - c_0\underline{1}\|^2 \tag{26}$$

Here $\underline{y}$ is defined as $\underline{y} = [y_1, ..., y_N]^T$ and $\underline{1} = [1, ..., 1]^T$. The same procedure can be applied in the case of $f(x) = c_1x+c_0$.

$$\min_{c_0,c_1} \sum_{i=1}^{N} (y_i - c_1x - c_0)^2 = \min_{\underline{c}} \|\underline{y} - X^1\underline{c}^1\|^2 \tag{27}$$

Here $\underline{y}$ is defined as $\underline{y} = [y_1, ..., y_N]^T$ and $X^1 = [1|\underline{x}]$ and $\underline{x} = [x_1, ..., x_N]^T$ and $\underline{c}^1 = [c_0\ c_1]^T$. Note also that to obtain the mean error this equation has too be divided for the number of measures:

$$\min_{\underline{c}} \frac{1}{n} \|\underline{y} - X\underline{c}^2\| \tag{28}$$

This is the empirical unbiased estimated error. Then, in the general case with $f(x) = \sum_{i=0}^{p} c_i x^i$ the formulation of the problem is $\min_{\underline{c}} \|\underline{y} - X\underline{c}\|^2$. The minimum of this quantity is located where the gradient of $\underline{c}$ is equal to zero:

$$\nabla_{\underline{c}} \left(\|\underline{y} - X\underline{c}\|^2\right) = 0 \tag{29}$$

$$\nabla_{\underline{c}} \left(\left[\left(\underline{y} - X\underline{c}\right)^T \left(\underline{y} - X\underline{c}\right)\right]\right) = 0 \tag{30}$$

$$\nabla_{\underline{c}} \left(\underline{y}^T\underline{y} - \underline{c}^T X^T \underline{y} - \underline{y}^T X\underline{c} + \underline{c}^T X^T X\underline{c}\right) = 0 \tag{31}$$

$$\nabla_{\underline{c}} \left(-2\underline{c}^T X^T \underline{y} + \underline{c}^T X^T X\underline{c}\right) = 0 \tag{32}$$

$$-X^T\underline{y} + X^T X\underline{c} = 0 \tag{33}$$

$$\underline{c} = \left(X^T X\right)^+ X^T \underline{y} \tag{34}$$

The last equation, is known as "one-shot" solution and it is the vector of estimated parameters we were searching for. Note that even though the undetermined system of equations $X^T X\underline{c} = X^T\underline{y}$ has infinitely many solutions, the pseudo-inverse always provides a solution. In order to compute the inverse it must be assumed that X is symmetric and semi-positive define.
One of the most famous results in statistics asserts that the least squares estimates of the parameters has the smallest variance among all linear unbiased estimates. We will make this precise here, and also make clear that the restriction to unbiased estimates is not necessarily a wise one. This observation will lead us to consider biased estimates. The Gauss-Markov theorem implies that the least squares estimator has the smallest mean squared error of all linear estimators with no bias. However, there may well exist a biased estimator with smaller mean squared error. Such an estimator would trade a little bias for a larger reduction in variance. Biased estimates are commonly used. Any method that shrinks or sets to zero some of the least squares coefficients may result in a biased estimate. From a more pragmatic point of view, most models are distortions of the truth, and hence are biased; picking the right model amounts to creating the right balance between bias and variance. A more effective approach is to add a bias to the objective function:

$$\min_{\underline{c}} \|\underline{y} - X\underline{c}\|^2 + \lambda\|\underline{c}\|^2 \tag{35}$$

This is the empirical bias estimated error. To minimize this error, it can be used the same procedure as before putting the gradient to zero. In that case, the solution becomes $\underline{c} = \left(X^T X + \lambda I\right)^{-1} X^T\underline{y}$, where I is a $d \times d$ identity matrix. The matrix $\left(X^T X + \lambda I\right)$ can be shown to be always positive-definite and therefore invertible. Why in some case we need the bias? It is useful to counteract the unbiased value introduced minimizing the function. λ is a trade off between the accuracy (given by the unbiased term) and the complexity (given by biased term). In general, it can be seen through four different interpretation:

1. *Statistical interpretation.* We attempt to minimize the empirical unbiased estimator of the error , but minimizing it, we produce a bias that fight against the unbias and so what we have to minimize is a biased estimator.

2. *Regularization effect.* If λ is big we don't care about making mistakes (first term) but we just care about making simple function that means polynomial of small degree. If λ is too small we don't care about using a lot of coefficients so finding polynomial of high degree, we just care about making small error over data. In other words λ represents my trust over the quality of data: if the quality is good there is small noise respect to the function that I'm searching for, so I'm able to fit the data. Else if data is corrupted by noise I cannot trust of minimize the error, I can just trust of selecting simple function to not fit the noise.

3. *Philosophical interpretation.* Occam (or Ockham's) razor: the principle stated that "Among competing hypotheses, the one with the fewest assumptions should be selected". It can be interpreted as stating "If we have two theories which explains the same phenomena I have to select the simplest one".

4. *Filter.* We're trying to take data and to filter them, improving the matrix and its conditions. In particular, when λ is different from zero, for example if λ is a big number, $X^T X$ disappear so we have the inverse of identity matrix and the result depends just on the projection of y over X. In other terms, the inverse is close to zero, then, c is zero, so we are creating a function which is f(x)=0 which means that we don't care about fitting the data, but we just want to obtain a simple function. Since this sum produces a full rank matrix, it represents a filter.

2 Perceptron, Neural Networks and Kernel

Data beats emotions. Sean Rad

2.1 Introduction

Neural networks are a model of simulation of the human nervous system.The human nervous system is composed of cells, referred to as neurons. Biological neurons are connected to one another at contact points, which are referred to as synapses. Learning is performed in living organisms by changing the strength of synaptic connections between neurons. Typically, the strength of these connections change in response to external stimuli. Neural networks can be considered a simulation of this biological process. As in the case of biological networks, the individual nodes in artificial neural networks are referred to as neurons. These neurons are units of computation that receive input from some other neurons, make computations on these inputs, and feed them into yet other neurons. The computation function at a neuron is defined by the weights on the input connections to that neuron. This weight can be viewed as analogous to the strength of a synaptic connection. By changing these weights appropriately, the computation function can be learned, which is analogous to the learning of the synaptic strength in biological neural networks. The "external stimulus" in artificial neural networks for learning these weights is provided by the training data. The idea is to incrementally modify the weights whenever incorrect predictions are made by the current set of weights. The key to the effectiveness of the neural network is the architecture used to arrange the connections among nodes. A wide variety of architectures exist, starting from a simple single-layer perceptron to complex multilayer networks.

2.2 Geometry Notions

Before proceeding with advanced concept let us digress slightly and review some important concept of geometry and optimization theory.

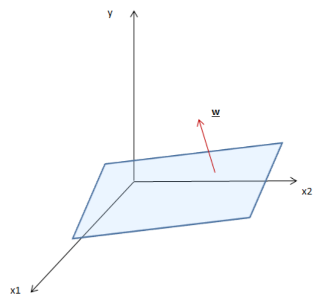


Figure 2: hyperplane passing through origin

2.2.1 Hyperplane in 2-dim passing through the origin.

A hyperplane passing through the origin is defined by the equation $f(x) = \underline{w}^T \underline{x}$, with $\underline{w} = [w_1\, w_2]^T$ and $\underline{x} = [x_1\, x_2]^T$. For any points in the hyperplane holds $\underline{w}^T \underline{x} = 0$.

2.2.2 Hyperplane in 2-dim.

A hyperplane is defined by the equation $f(x) = \underline{w}^T \underline{x} + b$. For any points in the hyperplane holds $\underline{w}^T \underline{x} + b = 0$.

2.2.3 Distance.

The distance of any point p to the given hyperplane can be computed in the following way. Suppose that p can be defined as the sum of two components.

Then,

$$\underline{p} = \underline{x}_p + d\frac{w}{\|w\|} \tag{36}$$

and multiplying all terms of the equation for the norm $\|w\| = \underline{w}^T\underline{w}$, we obtain the distance:

$$d = \frac{\underline{w}^T p + b}{\|w\|} \tag{37}$$

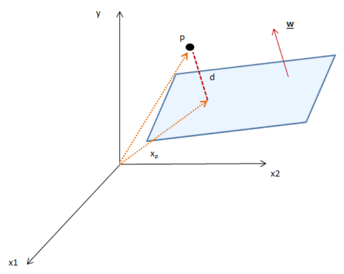


Figure 3: Distance of a point to hyperplane

2.2.4 Parallel Hyperplane.

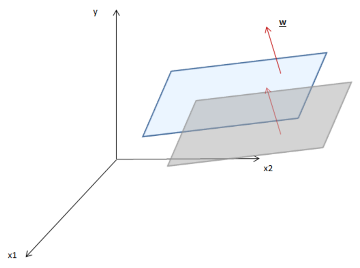


Figure 4: Parallel hyperplanes

Suppose two parallel hyperplanes are defined as $\pi_a : \underline{w}^T\underline{x} + b_a$ and $\pi_b : \underline{w}^T\underline{x} + b_b$ where $\underline{w}$ is the same vector (normal vector) (otherwise they are not parallel). Finally, the distance between the two hyperplane can be found just taking a point belonging to π_a and then computing the distance of this point to the other hyperplane π_b.

$$\begin{aligned} d &= \frac{\underline{w}^T p_a + b_b}{\|w\|} && (38) \\ &= \frac{\underline{w}^T \frac{-b_a}{\underline{w}^T} + b_b}{\|w\|} \\ &= \frac{|b_b - b_a|}{\|w\|} \\ &= \frac{\underline{w}^T \frac{-b_a}{\underline{w}^T} + b_b}{\|w\|} \\ &= \frac{|-b_a + b_b|}{\|w\|} && (39) \end{aligned}$$

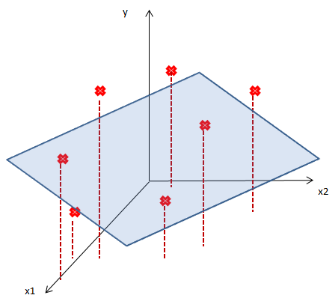


Figure 5: hyperplane best fitting some points

2.2.5 Least-Squares Fitting of a Hyperplane

Let's suppose to have some points (red cross) and to know the corresponding y value. In general is given a data set $D_n = \{(\underline{x}_1, y_1), \cdots, (\underline{x}_n, y_n)\}$. The problem is to find a hyperplane that best fit this points or in other terms is to identify the best $\underline{w}$ and b. This can be done by minimizing the least square error that in this particular case can be expressed like $\|\underline{y} - \left(\underline{w}^T\underline{x} + b\right)\|^2$. The value of y is known only in some points, so it can be computed only the empirical unbiased estimator of the error.

$$\min_{\underline{w},b} \frac{1}{n} \sum_{i=0}^{n} \|\underline{y}_i - \left(\underline{w}^T\underline{x}_i + b\right)\|^2 \tag{40}$$

This equation can be rewritten in a simpler form just introducing these vectors: $\underline{y} = [y_1, ..., y_n]^T$, $\underline{w} = [w_1, ..., w_n]^T$, $\underline{1} = [1, ..., 1]^T$, $\underline{X} = \left[\underline{x}_1^T, ..., \underline{x}_n^T\right]^T$ (note that X is a matrix).

$$\min_{\underline{w},b} \frac{1}{n} \sum_{i=0}^{n} \|\underline{y}_i - (X\underline{w} + \underline{1}b)\|^2 \tag{41}$$

From this point introducing one more vector $\underline{p} = [\underline{w}\, b]^T$ and a matrix $A = [X|\underline{1}]$ the above equation became:

$$\min_{\underline{p}} \frac{1}{n} \|\underline{y} - A\underline{p}\|^2 \tag{42}$$

We found the same equation at (28) in the previous chapter. To minimize this quantity we follow the same procedure seen before and according to this the vector of the parameters is:

$$\underline{p} = \left(A^T A\right)^+ A^T \underline{y} \tag{43}$$

As in the previous section, this case can be extended with a bias:

$$\min_{\underline{p}} \|\underline{y} - A\underline{p}\|^2 + \lambda\|\underline{p}\|^2 \tag{44}$$

and the solution is:

$$\underline{p} = \left(A^T A + \lambda I\right)^+ A^T \underline{y} \tag{45}$$

2.2.6 Optimization with constraint.

A general optimization problem is

$$\begin{aligned} &\min_{\underline{x}} f(x) \qquad (46)\\ &h_i(\underline{x}) = 0 \;\; i = 1, .., n\\ &g_j(\underline{x}) \geq 0 \;\; j = 1, .., p\\ &e_k(\underline{x}) \leq 0 \;\; k = 1, .., q \end{aligned} \tag{47}$$

Where $h_i(\underline{x}) = 0$ is an equality constraint, $g_j(\underline{x}) \geq 0$ is a inequality constraint as the last one. To solve the problem is not necessary to use all this constraint, but it can be shown that we obtain an equivalent formulation of the problem using only h_i and g_j or only g_j and e_k or just g_j or just e_k. To make an example the equality constraint can be expressed by:

$$h_i\left(\underline{x}\right) = 0 \longrightarrow \begin{cases} h\left(\underline{x}\right) \geq 0 \\ h\left(\underline{x}\right) \leq 0 \end{cases} \longrightarrow \begin{cases} h\left(\underline{x}\right) \geq 0 \\ -h\left(\underline{x}\right) \geq 0 \end{cases} \tag{48}$$

. If we have to keep separate two constraints we have to choose carefully which constraint is better to use: using an inequality constraint the dimension of the problem is the same, while adopting an equality constraint the dimension is reduced by one (solution lies in a straight line). The starting problem can be more generalized in the case $\underline{f}\left(\underline{x}\right)$ is a

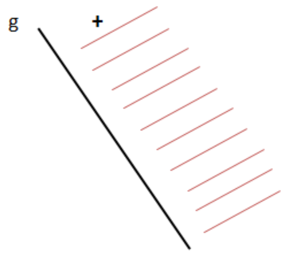


Figure 6: On the left the the dimension remain the same (half space) while on the right the solution belong to the line.

vector of functions $\left[f_1\left(\underline{x}\right), .., f_n\left(\underline{x}\right)\right]^T$. Then without loss of generality, we can consider the optimization problem:

$$\min_{\underline{x}} f\left(x\right) \tag{49}$$

$$h_i\left(\underline{x}\right) = 0 \;\; i = 1, \cdots, n$$
$$g_j\left(\underline{x}\right) \geq 0 \;\; j = 1, \cdots, p \tag{50}$$

. How it can be solved this problem? For sure not by hand. Since the dimension of the problem can be very high is not convenient at all compute all stationary points. A better approach is to use Newton's methods or Tangent's method or other algorithm known in mathematical programming. The problem of those methods is that they don't allow to impose constraints. So to solve our problem we have to find a way to make an equivalent formulation of the problem hiding explicit constraint in the problem itself. In Lagrange multiplier form the problem can be formulated as a weighted sum:

$$\min_{\underline{x}} f\left(\underline{x}\right) - \sum_{i=1}^{n} \alpha_i h_i\left(\underline{x}\right) - \sum_{j=1}^{p} \beta_j g_j\left(\underline{x}\right) \tag{51}$$

In order to keep the equality I need also to re-write the old constraints:

$$h_i\left(\underline{x}\right) = 0 \;\; i = 1, .., n \tag{52}$$
$$g_j\left(\underline{x}\right) \geq 0 \;\; j = 1, .., p \tag{53}$$
$$\beta_j g_j\left(\underline{x}\right) = \underline{0} \tag{54}$$
$$\beta \geq 0 \tag{55}$$

The last constraint $\beta \geq 0$ allow us to state that the function always decrease. At this point one can ask himself what is the real meaning of α and β. α and β can be seen as a *strength that force the constraints to be satisfied.* They are also known as KKT multipliers. If we are under strong duality assumptions, the dual formulation of the problem is

$$\max_{\alpha,\beta} \min_{\underline{x}} f\left(\underline{x}\right) - \sum_{i=1}^{n} \alpha_i h_i\left(\underline{x}\right) - \sum_{j=1}^{p} \beta_j g_j\left(\underline{x}\right) \tag{56}$$

$$h_i\left(\underline{x}\right) = 0 \;\; i = 1, .., n$$
$$g_j\left(\underline{x}\right) \geq 0 \;\; j = 1, .., p$$
$$\beta_j g_j\left(\underline{x}\right) = \underline{0}$$
$$\beta \geq 0 \tag{57}$$

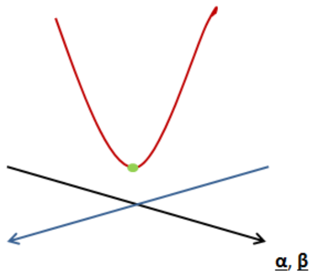


Figure 7: How the function changes respect to α and β.

The first two constraints are the primary conditions (constraint of the primal problem), the third is a complementary slackness condition and the last one is the condition over Lagrange multiplier (dual feasibility). KKT conditions guarantees to reach a minimum. Setting the gradient to zero $\nabla_{\underline{x}} = 0$, if the Hessian matrix is positive define, we find a minimum in

$$x^* = \underline{KKT}\left(\underline{\alpha}, \underline{\beta}\right) \tag{58}$$

Without making other assumptions the above equation can have infinite solutions. This problem is in fact an NP problem (non polynomial). Recall that it exist three main type of problem, see Fig. 8. *Polynomial problem* (P) it need a polynomial time to solve it so it can be solve by a computer, *Non Polynomial problem* (NP), the time grows faster than a polynomial(i.e. exponential function), Not Computable problem (NC), I don't know how many time it is necessary to solve it. NP problem deal with smart problems such as deep learning. The things that we have to do here are NP problems. But the time grows too fast so we 'll solve something which is easier as P problems. Is it always better to deal with P or NP problems? No. In practice we prefer to solve NP problem, because the computation time depends on the constant in front of the speed of the grow. Sometimes we will prefer to define the learning problem or the minimization problem as a non polynomial problem because of this characteristic. One simple hypothesis to solve

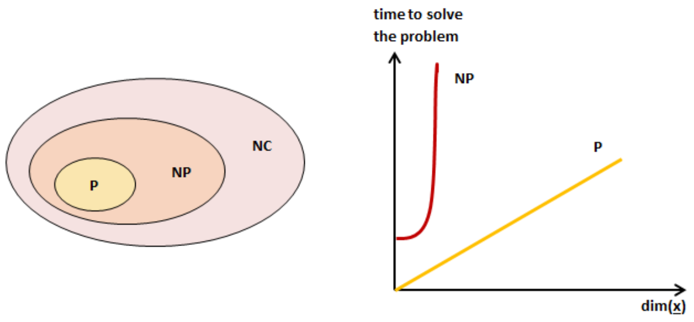


Figure 8: On the left the polynomial hierarchy. On the right the graphics shows how computational time increase with higher complexity.

this problem making it a P problem, is to assume that the function is convex and the constraint produce a convex space. In this way the problem is easy to solve, because if the domain is convex, thinking to a ball, wherever I put the ball, it

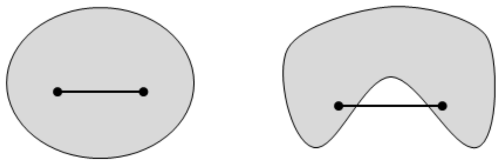


Figure 9: On the left a convex domain: the black line segment joining two points lies completely within the set. On the right non convex set: part of the segment joining the points lies outside of the set.

will reach the real minimum. So we can formally stated that if $f\left(\underline{x}\right)$ is convex and constraints give origin to a convex domain the solution of $x^* = \underline{KKT}\left(\underline{\alpha}, \underline{\beta}\right)$ is unique. To summarize what we have discover:

1. Under strong duality assumptions solving the minimization problem (primal) is equivalent to solve the maximization on the dual formulation.

2. If g,h,f are convex functions respect to α and β, this problem is linear in α and β and the dual one is concave (equivalent to say that is convex with a minus before)

3. Constraints with x disappear because they are already satisfied. The only remaining constraint is $\beta \geq 0$.

2.2.7 Example1

Let's suppose to have a a data set $D_n = \{\underline{x}_1, .., \underline{x}_n\} \in R^d$. Assume that this points have a spherical distribution (the cloud of points it can be approximates with a sphere), and that there is no noise so data well represent the population. In general to define a sphere we need three points, then in our case we need $d+1$ samples. The problem formulation then, is

$$\min_{a,R} \ R \tag{59}$$
$$\|\underline{x}_i - \underline{a}\|^2 \leq R^2 \ \ \forall i \in 1, \cdots, n \tag{60}$$

With the only purpose to simplify computation we can minimize R^2 instead of R, so the objective function became

$$\min_{a,R^2} \ R^2 \tag{61}$$
$$\|\underline{x}_i - \underline{a}\|^2 \leq R^2 \ \ \forall i \in 1, \cdots, n \tag{62}$$

This problem is not trivial to solve because the domain is an intersection of n hyperspheres. But since both the objective function and constraints are convex, the resulting domain is convex itself (the intersection of any collection of convex sets is convex). Then we can solve this problem using Lagrange multipliers in order to obtain constraints easier to be satisfied even if the objective function becomes more complex. The Lagrange (primal) function to be minimized is

$$L_p\left(\underline{a}, R^2, \underline{\alpha}\right) = R^2 - \sum_{i=1}^{n} \alpha_i \left[R^2 - \|\underline{x}_i - \underline{a}\|^2\right] \tag{63}$$

Setting the derivatives to zero, we obtain:

$$\frac{\partial L_p}{\partial R^2} = 1 - \sum_{i=1}^{n} \alpha_i = 0 \implies \sum_{i=1}^{n} \alpha_i = 1 \tag{64}$$

$$\begin{aligned}
\frac{\partial L_p}{\partial \underline{a}} &= \frac{\partial \left[R^2 - \sum_{i=1}^{n} \alpha_i \left[R^2 - \|\underline{x}_i - \underline{a}\|^2\right]\right]}{\partial \underline{a}} & (65)\\
&= \frac{\partial \left[R^2 - \sum_{i=1}^{n} \alpha_i \left[R^2 - x_i^T x_i + 2a^T x_i - a^T a\right]\right]}{\partial \underline{a}} & (66)\\
&= \sum_{i=1}^{n} \alpha_i \left(-2x_i + 2a\right) = 0 & (67)\\
&\implies \sum_{i=1}^{n} \alpha_i x_i = a & (68)
\end{aligned}$$

In addition the solution must satisfy the Karush-Kuhn-Tucker conditions and

$$\alpha_i \left[R^2 - \|\underline{x}_i - \underline{a}\|^2\right] = 0 \ \forall i \tag{69}$$

$$\alpha_i \geq 0 \tag{70}$$

and substituting these in the primal function we obtain the dual formulation

$$\begin{aligned}
L_d\left(\underline{\alpha}\right) &= R^2 - \sum_{i=1}^{n} \alpha_i R^2 - \sum_{i=1}^{n} \alpha_i \left[-\|\underline{x}_i - \underline{a}\|^2\right] & (71)\\
&= \sum_{i=1}^{n} \alpha_i \left[\|\underline{x}_i - \underline{a}\|^2\right] & (72)
\end{aligned}$$

$$= \sum_{i=1}^{n} \alpha_i x_i^T x_i - 2\sum_{i=1}^{n} \alpha_i x_i^T a + \sum_{i=1}^{n} \alpha_i a^T a \tag{73}$$

$$= \sum_{i=1}^{n} \alpha_i x_i^T x_i - 2\sum_{i=1}^{n} \alpha_i x_i^T \sum_{j=1}^{n} \alpha_j x_j + \sum_{i=1}^{n} \alpha_i x_i^T \sum_{j=1}^{n} \alpha_j x_j \tag{74}$$

$$= \sum_{i=1}^{n} \alpha_i x_i^T x_i - \sum_{i=1}^{n} \alpha_i x_i^T \sum_{j=1}^{n} \alpha_j x_j \tag{75}$$

$$\tag{76}$$

The solution is obtained by maximizing LD

$$\max_{\underline{\alpha}} \; -\sum_{i=1}^{n} \alpha_i x_i^T \sum_{j=1}^{n} \alpha_j x_j + \sum_{i=1}^{n} \alpha_i x_i^T x_i \tag{77}$$

To get a minimization problem, we have just to invert the sign

$$\begin{aligned} &\min_{\underline{\alpha}} \; \sum_{i=1}^{n} \alpha_i x_i^T \sum_{j=1}^{n} \alpha_j x_j - \sum_{i=1}^{n} \alpha_i x_i^T x_i \\ &\alpha_i \geq 0 \\ &\sum_{i=1}^{n} \alpha_i = 1 \end{aligned} \tag{78}$$

Since $H = x_i^T x_i \geq 0$ the solution α^* is unique, there is only one minimum. Let's check if α^* is the optimal solution. At the optimal all constraints have to be satisfied from the solution. In particular if we take a point inside the sphere then $\|\underline{x}_i - \underline{a}\|^2 \leq R^2$ and so $\alpha_i = 0$. If, instead, $\alpha_i \geq 0$ to satisfy the constraint we must have $R^2 - \|\underline{x}_i - \underline{a}\|^2 = 0$ that means that the point is on the sphere. In this case α^* is a sparse vector with many zeros (compression) In the matrix formulation our problem can be re-written as

$$\begin{aligned} &\min_{\underline{\alpha}} \; \underline{\alpha}^T X X^T \underline{\alpha} - \underline{\alpha}^T diag\left(X X^T\right) \\ &\underline{\alpha} \geq 0 \\ &\underline{1}^T \underline{\alpha} = \underline{1} \end{aligned} \tag{79}$$

This problem can be solved by software programs like Matlab, Cplex and others. In general we are able to solve quadratic and linear programming problems as

$$\begin{aligned} &\min_{\underline{\alpha}} \; \frac{1}{2}\underline{\alpha}^T Q \underline{\alpha} + \underline{r}^T \underline{\alpha} \\ &B\underline{\alpha} \leq \underline{c} \\ &A\underline{\alpha} = \underline{b} \\ &\underline{L} \leq \underline{\alpha} \leq \underline{U} \end{aligned} \tag{80}$$

$$\begin{aligned} &\min_{\underline{\alpha}} \; \underline{r}^T \underline{\alpha} \\ &B\underline{\alpha} \leq \underline{c} \\ &A\underline{\alpha} = \underline{b} \\ &\underline{L} \leq \underline{\alpha} \leq \underline{U} \end{aligned} \tag{81}$$

2.2.8 Example2

Now let us make an example solvable by hand. We have two points: -1 and 1 so the data set is $D_n = \{-1, 1\}$. We refer to the matrix formulation seen before obtaining

$$\min_{\underline{\alpha}} \; \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^T \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} - \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{82}$$

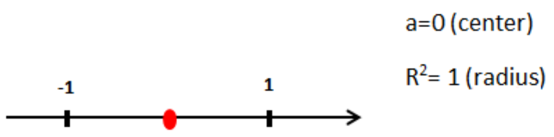


Figure 10: We want to find the optimal value of a and R.

$$\begin{aligned} \alpha_1, \alpha_2 \geq 0 \\ \alpha_1 + \alpha_2 = 1 \end{aligned} \tag{83}$$

Combining constraints $\alpha_2 = 1 - \alpha_1$ we note that only one constraint remains, it will be the domain

$$0 \leq \alpha_2 \leq 1 \tag{84}$$

Computing the objective function we get

$$\min_{\underline{\alpha}} \ \alpha_1 (\alpha_1 - \alpha_2) + \alpha_2 (\alpha_2 - \alpha_1) - \alpha_1 + \alpha_2 \tag{85}$$

$$\min_{\underline{\alpha}} \ \alpha_1\alpha_1 - \alpha_1\alpha_2 - \alpha_1\alpha_2 + \alpha_2\alpha_2 - \alpha_1 + \alpha_2 \tag{86}$$

Substituting in this $\alpha_2 = 1 - \alpha_1$ we obtain

$$\min_{\underline{\alpha}} \ \alpha_1\alpha_1 - 2\alpha_1 (1 - \alpha_1) + (1 - \alpha_1)(1 - \alpha_1) - \alpha_1 + (1 - \alpha_1) \tag{87}$$

Then, finally what we have to minimize is

$$\min_{\underline{\alpha}} \ 4\alpha_1^2 - 4\alpha_1 \tag{88}$$

From the picture we can easily state that $\alpha_1 = \frac{1}{2}$ and then also $\alpha_2 = \frac{1}{2}$. This solution satisfies the previous constraint.

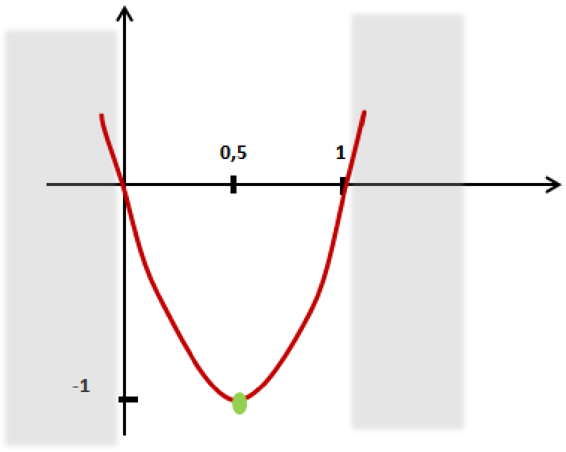


Figure 11: The equation plot is a parabola. The green point is the minimum. According to that we learn that α_1 is equal to $\frac{1}{2}$.

Now we have to find the center

$$a = \sum_{i=1}^{N} \alpha_i x_i = \frac{1}{2} - \frac{1}{2} = 0 \tag{89}$$

and the radius

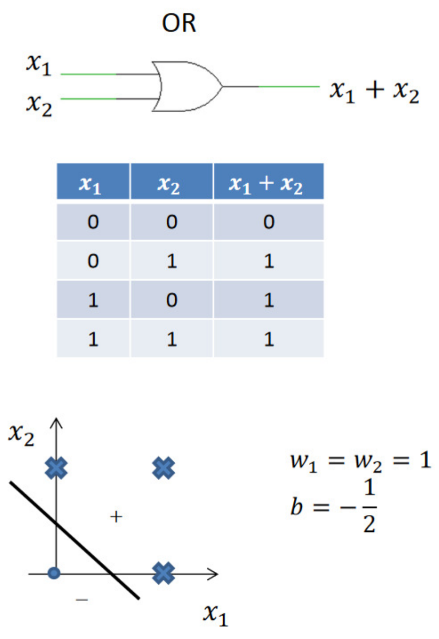
$$R = \|\alpha_i - a\|^2 = \alpha_1 + \alpha_2 = \frac{1}{2} + \frac{1}{2} = 1 \tag{90}$$

2.3 Perceptron

Essentially, all models are wrong, but some are useful. George E. P. Box

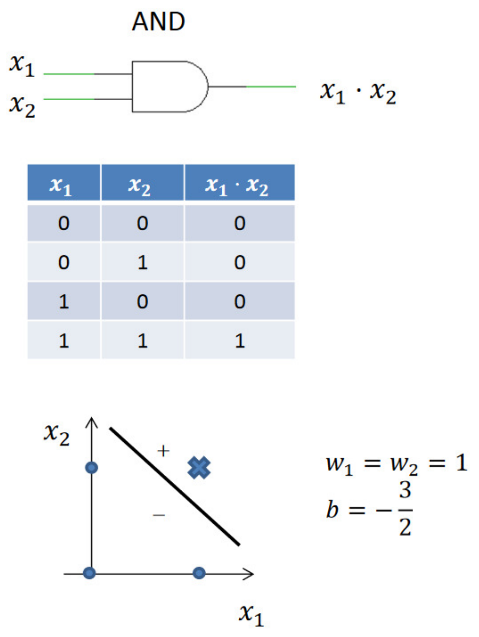
2.3.1 Introduction

The perceptron algorithm was invented in 1957 by Frank Rosenblatt (even if Turing proposed a similar concept in 1948). The perceptron was intended to be a machine, rather than a program. This machine was designed for image recognition. Although the perceptron initially seemed promising, it was quickly proved that perceptrons could not be trained to recognize many classes of patterns. Single layer perceptrons are only capable of learning linearly separable patterns; in 1969 a famous book entitled *Perceptrons* by Marvin Minsky and Seymour Papert showed that it was impossible for these classes of network to learn an XOR function. This caused the field of neural network research to stagnate for many years, before it was recognized that a feed-forward neural network with two or more layers (also called a multilayer perceptron) had far greater processing power than perceptrons with one layer (also called a single layer perceptron). To go deeper in the details, it can be shown that a perceptron is a universal logic gate i.e. it can realize any logic network (as any logic network of n variables can be realized with 2 ANDs and 1 OR). If we have a perceptron with both boolean input and output then we can use it to represent all boolean functions, and we can also



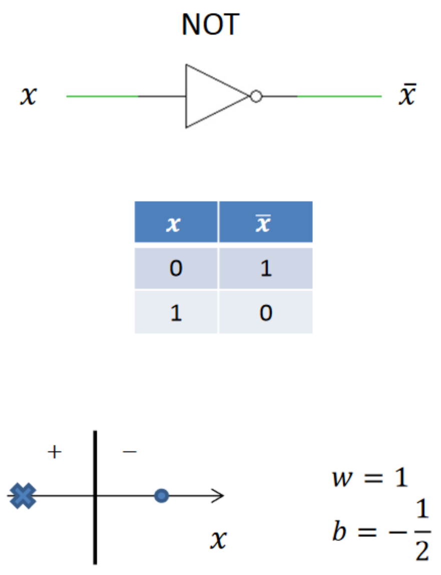
x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Figure 12: Or



x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Figure 13: And



x	$\bar{x}$
0	1
1	0

Figure 14: Not

express complex boolean functions as $f(x_1, x_2, x_3) = \bar{x}_1 x_2 + x_2 \bar{x}_3$, as we can see in figure 15 and fig. 16 . In order to

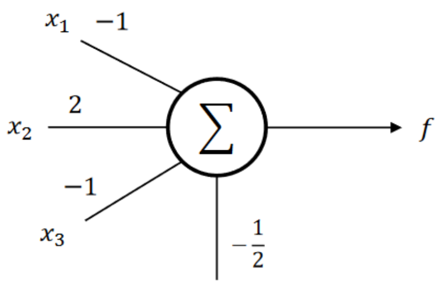
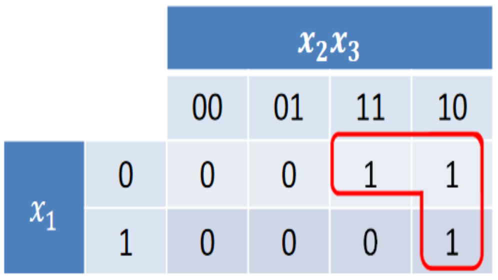


Figure 15: LTG



		x_2x_3			
		00	01	11	10
x_1	0	0	0	1	1
	1	0	0	0	1

Figure 16: LTG

represent XOR function we need quadratic perceptrons (eclipse). If a quadratic classifier is not enough we can use a Polynomial Threshold Gate, as big as the order of the function we have to deal with according to this two theorems: **Theorem 1**: *Any boolean function of n variables can be implemented by a PTG(r) with $r \leq n$.* **Theorem 2**: *Any PTG(r) can be implemented by $\sum_{i=1}^{r} \binom{n}{i} - n$ AND and one LTG.*

2.3.2 The perceptron: a linear classifier

The most basic architecture of a neural network is referred to as the perceptron. An example of the perceptron architecture is illustrated in Fig. 17 The perceptron contains two layers of nodes, which correspond to the input nodes,

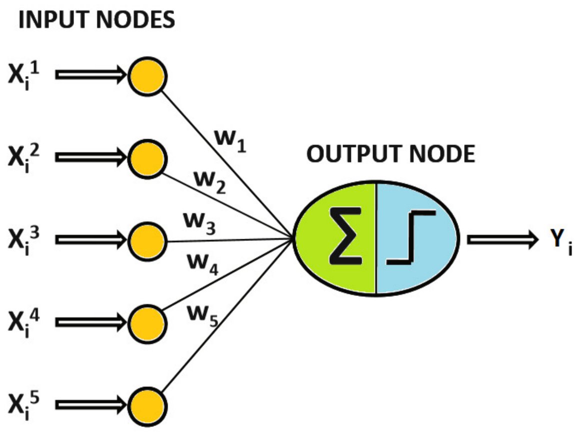


Figure 17: Perceptron: one layer neural network.

and a single output node. The number of input nodes is exactly equal to the dimensionality d of the underlying data. In the basic perceptron model, the output node is the only node that performs a mathematical function on its inputs. For simplicity of further discussion, it will be assumed that all input variables are numerical. Furthermore, it will be assumed that the classification problem contains two possible values for the class label, drawn from $\{-1, +1\}$. As discussed earlier, each input node is connected by a weighted connection to the output node. These weights define a function from the values transmitted by the input nodes to a binary value drawn from $\{-1, +1\}$. This value can be interpreted as the perceptron?s prediction of the class variable of the test instance fed to the input nodes. Just as learning is performed in biological systems by modifying synaptic strengths, the learning in a perceptron is performed by modifying the weights of the links connecting the input nodes to the output node whenever the predicted label does not match the true label. The function learned by the perceptron is referred to as the *activation function*, which is a signed linear function. Let $W = (w_1 \cdots w_d)$ be the weights for the connections of d different inputs to the output neuron for a data record of dimensionality d. In addition, a bias b is associated with the activation function. The output $y_i \in \{-1, +1\}$ for the feature set $(x_i^1 \cdots x_i^d)$ of the ith data record X_i, is as follows:

$$y_i = sign\{\sum_{i=1}^{d} w_i x_i + b\} = \underline{wx} + b \tag{91}$$

The value y_i represents the prediction of the perceptron for the class variable of $\underline{x}$. The goal in neural network algorithms is to learn the vector of weights W and bias b, so that y_i approximates the true class variable as closely as possible. The perceptron simply divide the space of inputs in two semi spaces as depicted in fig 18. The equation, in the case we are

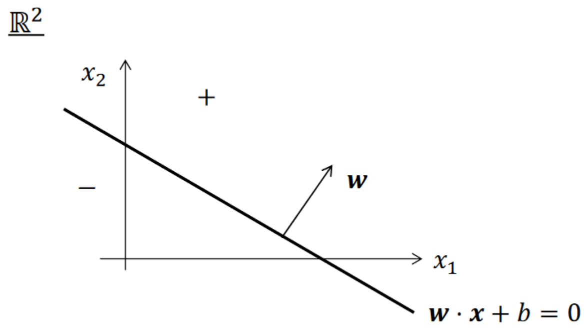

$\underline{\mathbb{R}^2}$
x_2
+
−
$\boldsymbol{w}$
x_1
$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

Figure 18: The perceptron divide the space of input

in $\Re^2$ can be expressed as a linear combinations,

$$w_1 x_1 + w_2 x_2 + b\} = \underline{wx} + b \tag{92}$$

2.3.3 Perception Learning

Let assume to have as input $\{\underline{x}_i, y_i\}_{i=1}^n$ and as output $y_i = \{+1, -1\}$. The sigmoid function is $f(\underline{x}) = sign(\underline{wx})$. Now we have to find the vector of weights $\underline{w}$. The perceptron learning rule is made up of the so called "Hebbian learning" in which simultaneous activation of cells leads to pronounced increases in synaptic strength between those cells (to continue biological metaphor of neurons made in the introduction of this section). In general is given by,

$$\begin{cases} \underline{w}_1 \text{random} \\ \underline{w}_{k+1} = \underline{w}_k + \rho \left(y_k - f\left(\underline{x}_k \right) \right) \underline{x}_k \end{cases} \tag{93}$$

Here, $\rho > 0$ and $k \rightarrow (k-1)\, mod n + 1$. To simplify the problem, let's assume to have $\rho = \frac{1}{2}$, then we have

$$\begin{cases} \underline{w}_1 \text{random} \\ \underline{w}_{k+1} = \underline{w}_k + \begin{cases} y_k \underline{x}_k \text{ if } y_k - f\left(\underline{x}_k \right) \leq 0 \\ 0 \text{ otherwise} \end{cases} \end{cases} \tag{94}$$

It can be shown that the perceptron learning algorithm converges in a finite number of steps (if n is finite and data is linearly separable).The following theorem gives a bound on the learning error of the perceptron algorithm, when it is run as an online algorithm that performs an update each time it gets an example wrong. (Novikoff 1962). Hp : If $\underline{w}^*$ is the solution then $y_f f_k^* > 0 \ \forall k$. So, we have,

$$\underline{w}_{k+1} - \alpha \underline{w}^* = \underline{w}_k - \alpha \underline{w}^* + y_k \underline{x}_k \quad \forall \alpha > 0 \tag{95}$$

If we compute the euclidean norm,

$$\begin{aligned} \|\underline{w}_{k+1} - \alpha \underline{w}^*\|^2 &= \|\underline{w}_k - \alpha \underline{w}^*\|^2 + \|y_k \underline{x}_k\|^2 + 2 y_k \underline{x}_k \left(\underline{w}_k - \alpha \underline{w}^* \right) & (96) \\ &= \|\underline{w}_k - \alpha \underline{w}^*\|^2 + \|\underline{x}_k\|^2 + 2 y_k \underline{x}_k \underline{w}_k - 2 y_k \alpha_k \underline{x}_k \underline{w}^* & (97) \\ &\leq \|\underline{w}_k - \alpha \underline{w}^*\|^2 + \|\underline{x}_k\|^2 - 2 y_k \alpha_k \underline{x}_k \underline{w}^* & (98) \\ &= \|\underline{w}_k - \alpha \underline{w}^*\|^2 + R^2 - 2\alpha\gamma & (99) \end{aligned}$$

This is valid $\forall \alpha > 0$ then we choose $\alpha = \frac{R^2}{\gamma}$, then substituting it in the previous equation, we obtain,

$$\|\underline{w}_{k+1} - \alpha \underline{w}^*\|^2 \leq \|\underline{w}_k - \alpha \underline{w}^*\|^2 - R^2 \leq \|\underline{w}_1 - \alpha \underline{w}^*\|^2 - kR^2 \tag{100}$$

But, since $\|\underline{w}_{k+1} - \alpha \underline{w}^*\|^2 \geq 0$ then k is finite and it is given by,

$$k \leq \frac{\|\underline{w}_1 - \alpha \underline{w}^*\|^2}{R^2} = \frac{\|\gamma \underline{w}_1 - R^2 \underline{w}^*\|^2}{\gamma^2 R^2} \tag{101}$$

So the algorithm converges in $k < \infty$ steps. Please note, we have used the trick inside the Fig. (19) and Fig. (20) to major the equation above. Let's now make some observation. The first one is that if the starting weight $\underline{w}_1$ is null, then $k \leq \frac{R^2}{\gamma} \|\underline{w}^*\|^2$, so the algorithm take more steps to converge. If $\gamma \leq 0$ this algorithm does not converge. A solution to this problem is given by the "*Pocket algorithm*", which we do not describe here. The above algorithm can considers one sample at the time and so it is called "*learning by pattern*" or all the samples at the time and so we speak about "*learning by batch*" or "*by epoch*". Under convergence constraints, the algorithm converges to one of the ∞ solutions. All these algorithms can be seen also as optimization algorithm. In this sense, if the cost/ error function is $E\left(\underline{w} \right) = -\sum_{i=1}^n y_i \underline{x}_i \underline{w}$ then the learning rule can be expressed as,

$$\underline{w}_{k+1} = \underline{w}_k + \rho \sum_{i=1}^n y_i \underline{x}_i \underline{w} \tag{102}$$

This is obtained by $\underline{w}_{k+1} = \underline{w}_k - \rho \nabla E\left(\underline{w} \right)$ computed in $\underline{w} = \underline{w}_k$ which is the gradient descendent method. The equation (102) is also known as *Batch Perceptron Learning Rule.*

2.4 Neural Networks

Neural network diagrams like Figure 21 are sometimes drawn with an additional bias unit feeding into every unit in the hidden and output layers. The units in the middle of the network, computing the derived features h (green), are called hidden units because the values h are not directly observed. In general there can be more than one hidden layer. The neural network is then a standard linear model using these transformations as inputs.

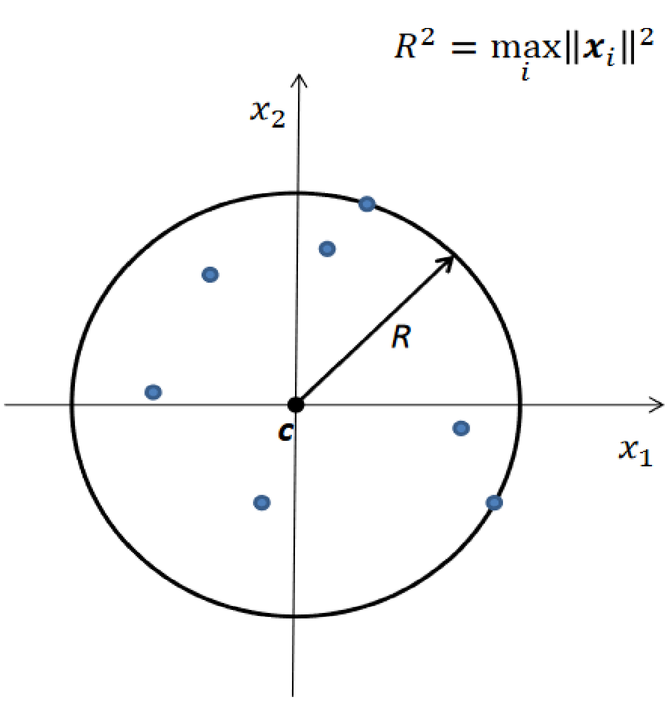


Figure 19: Maximization of the distance

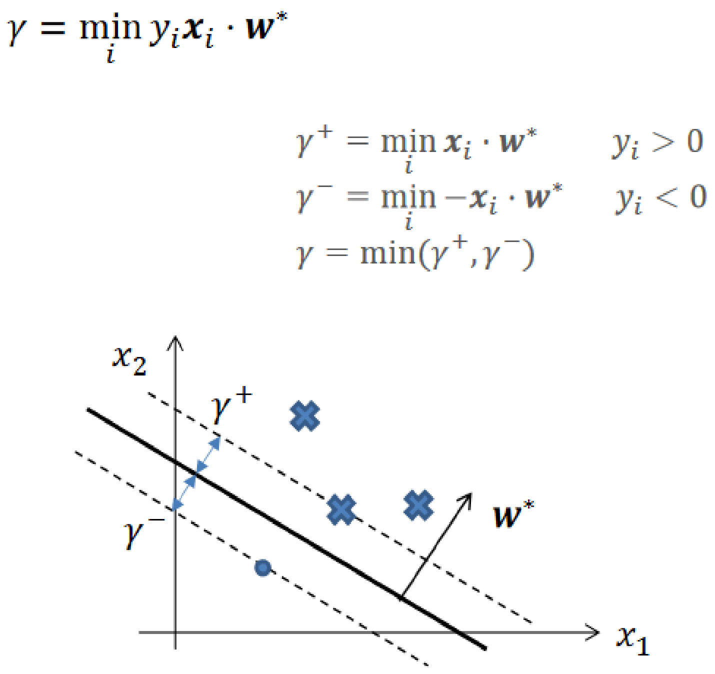

$\gamma = \min_i y_i \boldsymbol{x}_i \cdot \boldsymbol{w}^*$
$\gamma^+ = \min_i \boldsymbol{x}_i \cdot \boldsymbol{w}^* \quad y_i > 0$
$\gamma^- = \min_i -\boldsymbol{x}_i \cdot \boldsymbol{w}^* \quad y_i < 0$
$\gamma = \min(\gamma^+, \gamma^-)$
x_2
γ^+
γ^-
$\boldsymbol{w}^*$
x_1

Figure 20: Margin minimization

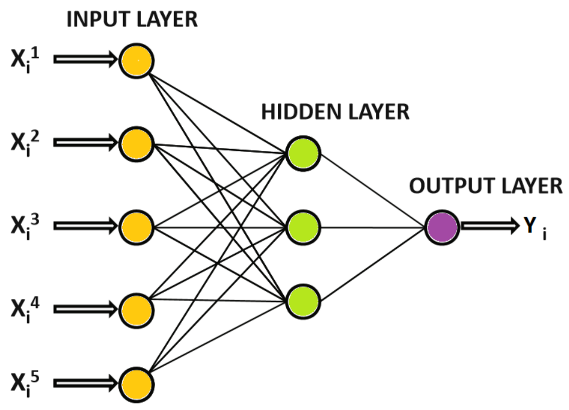


Figure 21: Deep Neural Network

2.4.1 Linear perceptron: Adaline

What happens if we build a multi-layer ADALINE?

$$f\left(\underline{x}\right) = \underline{wh}\left(\underline{x}\right) = \underline{w}\begin{bmatrix} h_1\left(\underline{x}\right) \\ h_n\left(\underline{x}\right)\end{bmatrix} = \underline{w}\begin{bmatrix} \underline{v}_1\underline{x} \\ \underline{v}_n\underline{x}\end{bmatrix} = \sum_i w_i\underline{v}_i\underline{x} = \underline{ux} \tag{103}$$

So we can state that a multi-layer ADALINE with weights $\underline{w}$, $\underline{v}_i$ is equivalent to a single-layer ADALINE with weight $\underline{u}$.

2.4.2 Non Linear perceptron

As we said before in this section, we can manage this problem as a minimization problem. The problem here is that we can't use as a sigmoid the sign function because it is not derivable. The idea is to use a similar function, like hyperbolic tan which can be derived. In this case we have $f\left(\underline{x}\right) = tanh\left(\underline{wx}\right)$ where the activation function is $tanh\left(x\right) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. If we derive it, we obtain $tanh'\left(x\right) = 1 - tanh^2\left(x\right)$. In non linear case then the learning method is the following. Due to the fact that tanh is differentiable we can use delta rule. The cost function is $E\left(\underline{w}\right) = \frac{1}{2}\left(y - f\right)$ so we can compute weights as,

$$\underline{w}_{k+1} = \underline{w}_k - \eta\nabla E\left(\underline{w}\right) \tag{104}$$

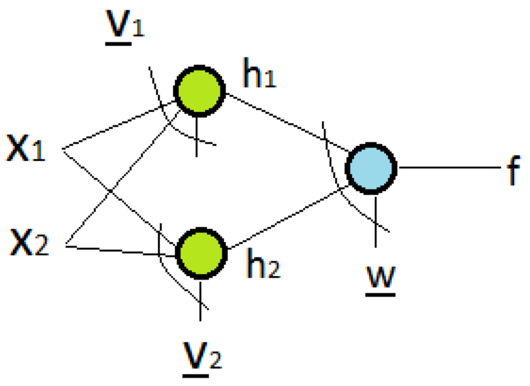


Figure 22: Single layer perceptron

where,

$$\nabla E(\underline{w}) = \frac{1}{2} 2 (y - f) \left[-f'(\underline{w}\underline{x}) \right] \underline{x} = -(y - f) f'(\underline{w}\underline{x}) \underline{x} = \delta \underline{x} \tag{105}$$

So we obtain the so called delta rule,

$$\underline{w}_{k+1} = \underline{w}_k - \eta \delta_k \underline{x}_k \tag{106}$$

In 1986 , David Rumelhart, proposed an algorithm for learning in non linear multi layer perceptrons, known as *Error Back Propagation*. We can compute the output as,

$$f(\underline{x}) = f\left(\sum_j w_j h_j \right) = f\left(\sum_j w_j h\left(\sum_i v_{ij} x_i \right) \right) \tag{107}$$

If the cost function is $E(\underline{w}, \underline{x}) = \frac{1}{2}(y - f)^2$ and according to the delta rule for output layer,

$$\underline{w}_{k+1} = \underline{w}_k - \eta \delta_k \underline{x}_k \tag{108}$$

the learning rule for hidden layers is given by,

$$\underline{v}_j^{k+1} = \underline{v}_j^k - \eta \frac{\partial E}{\partial \underline{v}_j} |_{\underline{v}_j = \underline{v}_j^k} \tag{109}$$

If we expand the partial derivative term using the *chain rule* we obtain,

$$\begin{aligned} \frac{\partial E}{\partial \underline{v}_{ij}} &= \frac{\partial E}{\partial h_j} \frac{\partial h_j}{\partial v_{ij}} & (110) \\ &= \frac{1}{2} 2 (y - f) \left(-f' \right) w_j \frac{\partial h_j}{\partial v_{ij}} \\ &= -(y - f) f' w_j h_j' x_i \\ &= -\delta w_j h_j' x_i \\ &= -\delta_j x_i & (111) \end{aligned}$$

According to this equation, the delta rule for hidden layer can be expressed as,

$$\underline{v}_j^{(k+1)} = \underline{v}_j^{(k)} - \eta \delta_j^{(k)} \underline{x}^{(k)} \tag{112}$$

Since δ_j of hidden layers is computed from δ of output layer as,

$$\delta_j^{(k)} = \delta w_j^{(k)} h_j^{'(k)} \tag{113}$$

we refer to this as *back-propagation* of error. Now , let's see some useful properties of multi layer perceptron.

2.4.3 Approximation properties

Let assume we have a sigmoidal function $\varphi(x)$ with $f(\underline{x}, \underline{w}, \underline{v}_i, b) = \sum_i^N w_i \varphi(\underline{v}_i \underline{x} + b)$. It can be shown that if $g : X^n \to \Re$ with $X^n \in \Re^n$ is continuous and $\epsilon > 0$ then $\exists (\underline{w}^*, \underline{v}_i^*, b^*, N)$ such that $|g - f| \leq \epsilon \; \forall \epsilon > 0$. In other words, we are able to approximate f with a finite sum $g(x)$ with an accuracy of ϵ. This theorem can be extended to the first derivative. If $g^{'}$ is the first derivative of g then $|g^{'} - f^{'}| \leq \epsilon \; \forall \epsilon > 0$. It means that $f^{'}$ approximates the derivative of $g^{'}$ and g can be discontinuous. The interest reader can go through the proofs of this two theorems reading the following papers: *Approximation by Superpositions of a Sigmoidal Function** by G. Cybenkot (1989) and *Approximation capabilities of Multilayer Feedforward Networks* by K. Hornik (1991).

2.4.4 Classification properties

Let be $\{\underline{x}, y_i\}_{i=1}^n$ the datatset, and let assume to have two possible classes,

$$\underline{x}_i \in \begin{cases} C_1 \text{ iff } y_i = +1 \\ C_2 \text{ iff } y_i = -1 \end{cases} \tag{114}$$

We can define the Bayesian error as $g(x) = P(C_1/\underline{x}) - P(C_2/\underline{x})$ and use it as discriminating function $\hat{y}_i = sign(g(\underline{x}_i))$. Here $P(C_i/\underline{x})$ is the probability that x belong to class C_i. It can be shown that perceptron is the best approximation of the Bayes discriminating function. In other words, back-propagation finds a minimum mean squared-error approximation to the Bayes optimal discriminant function. In fact, the minimum squared error is in this case,

$$\epsilon^2(\underline{w}) = \int (f(\underline{x}, \underline{w}) - g(\underline{x}))^2 p(\underline{x}) d\underline{x} \tag{115}$$

where f represents the neural network and g the optimal discriminating function. The empirical error (the sample data error function) is given by,

$$E_s(\underline{w}) = \sum_{i \in C_1} (f(\underline{x}_i, \underline{w}) - 1)^2 + \sum_{i \in C_2} (f(\underline{x}_i, \underline{w}) + 1)^2 \tag{116}$$

while the true error (average error)is,

$$E(\underline{w}) = \lim_{n \to \infty} \frac{1}{n} E_s(\underline{w}) \tag{117}$$

where n is the total number of samples for both classes. The theorem proposed by D.W. Ruck, in “the multilayer perceptron as an approximation to a Bayes optimal discriminating function” (1990), state that to minimize ϵ^2 is equal to minimize $E(\underline{w})$,

$$\arg\min_{\underline{w}} E(\underline{w}) = \arg\min_{\underline{w}} \epsilon^2 \tag{118}$$

Now we can rewrite the expression for $E(\underline{w})$ using the number of samples in each class,

$$\begin{aligned} E(\underline{w}) &= \lim_{n \to \infty} \left[\frac{n_1}{n} \frac{1}{n_1} \sum_{i \in C_1} (f(\underline{x}, \underline{w}) - 1)^2 + \frac{n_2}{n} \frac{1}{n_2} \sum_{i \in C_2} (f(\underline{x}, \underline{w}) + 1)^2 \right] \qquad (119) \\ &= \lim_{n \to \infty} \frac{n_1}{n} \lim_{n \to \infty} \frac{1}{n_1} \sum_{i \in C_1} (f(\underline{x}, \underline{w}) - 1)^2 + \lim_{n \to \infty} \frac{n_2}{n} \lim_{n \to \infty} \frac{1}{n_2} \sum_{i \in C_2} (f(\underline{x}, \underline{w}) + 1)^2 \\ &= p(C_1) \int (f(\underline{x}, \underline{w}) - 1)^2 p(\underline{x}|C_1) d\underline{x} + p(C_2) \int (f(\underline{x}, \underline{w}) + 1)^2 p(\underline{x}|C_2) d\underline{x} \\ &= \int \{ [f^2(\underline{x}, \underline{w}) + 1] [p(\underline{x}|C_1) p(C_1) + p(\underline{x}|C_2) p(C_2)] + \\ &- 2f(\underline{x}, \underline{w}) [p(\underline{x}|C_1) p(C_1) - p(\underline{x}|C_2) p(C_2)] \} d\underline{x} \qquad (120) \end{aligned}$$

Using the Bayes Formula, we can simplify the above equation and introduce the optimal discriminating function g,

$$\begin{aligned} E(\underline{w}) &= \int \{ [f^2(\underline{x}_i, \underline{w}) + 1] [1 - 2f(\underline{x}, \underline{w}) (p(C_1|\underline{x}) p(\underline{x}) - p(C_2|\underline{x}) p(\underline{x}))] \} d\underline{x} \qquad (121) \\ &= \int \left[(f^2(\underline{x}, \underline{w}) + 1) p(\underline{x}) - 2f(\underline{x}, \underline{w}) g(\underline{x}) p(\underline{x}) \right] d\underline{x} \\ &= \int \left[f^2(\underline{x}, \underline{w}) p(\underline{x}) + p(\underline{x}) - 2f(\underline{x}, \underline{w}) g(\underline{x}) p(\underline{x}) \right] d\underline{x} \end{aligned}$$

$$\begin{aligned}
&= \int \left[\left(f^2(\underline{x},\underline{w}) - 2f(\underline{x},\underline{w})\, g(\underline{x}) + g^2(\underline{x}) \right) p(\underline{x}) + p(\underline{x}) - 2g^2(\underline{x})\, p(\underline{x}) \right] d\underline{x} \\
&= \int \left[f(\underline{x},\underline{w}) - g(\underline{x}) \right]^2 p(\underline{x})\, d\underline{x} + \int p(\underline{x})\, d\underline{x} - \int g^2(\underline{x})\, p(\underline{x})\, d\underline{x} \\
&= \epsilon^2(\underline{w}) + 1 + \text{const} \qquad (122)
\end{aligned}$$

Note, however, that the accuracy of this approximation is limited. If n is small the function approximating the bayes optimal discriminant function will not provide a good match.

2.5 Kernel

It is a tale Told by an idiot, full of sound and fury, Signifying nothing. Macbeth, Act V, Scene V

2.5.1 Notion 1

Recalling geometry notions seen in chapter 2 at paragraph *Least-Square Fitting of a Hyperplane.* The solution of the minimization problem to fit an hyperplane passing through the origin is

$$\underline{w} = \left(X^T X + \lambda I \right)^+ X^T \underline{y} \qquad (123)$$

Now we want to know the cost we pay to compute $\underline{w}$. Due to the fact that $dim\left(X^T X + \lambda I\right) = d \times d$ and $dim\left(X^T \underline{y}\right) = d \times 1$ the dimension of $\underline{w}$ is $d \times 1$. The major part of the computational cost is the inversion of the matrix. The simplest way to solve the inversion is to diagonalize the matrix but this way has an high degree of complexity. The complexity in the linear case depends on the dimensionality of data (we have to invert an $d \times d$ matrix) and not on the number of data. So the problem is that with this method we are able to manage only linear separator. Now we have to understand how is made the solution of the minimization problem,

$$\min_{\underline{w}} \sum_{i=1}^{n} \left(\underline{w}^T \underline{x}_i - \underline{y}_i \right)^2 + \lambda \|\underline{w}\|^2 \qquad (124)$$

How we can express $\underline{w}$ in an other form? It can be expressed as a linear combination or a weighted sum of the points in the training set.

$$\underline{w} = \sum_{i=1}^{n} \alpha_i \underline{x}_i \qquad (125)$$

The above equation means that $\underline{w}$ can only assumes values deriving from this computation. So to solve the minimization problem $\underline{w}$ must be a subspace defined by x_i. This is known as the "*Representer theorem*". It can be shown that the statement is correct by reducing to absurd. To do this, let's introduce $\underline{w}^\perp$ which is a vector that doesn't lie on the hyperplane, in other words it is not contained in the subspace of $\underline{x}$, but it is perpendicular to it. So we have,

$$\underline{w}^* = \sum_{i=1}^{n} \alpha_i \underline{x}_i + \underline{w}^\perp \qquad (126)$$

Now if we substitute $\underline{w}^*$ in the equation (124),

$$\sum_{i=1}^{n} \left(\underline{w}^{T*} \underline{x}_i - \underline{y}_i \right)^2 + \lambda \|\underline{w}^*\|^2 \qquad (127)$$

in which,

$$\underline{w}^{T*} \underline{x}_i = \sum_{j=1}^{n} \alpha_i \underline{x}_j^T \underline{x}_i + \underline{w}^{T\perp} \underline{x}_i \qquad (128)$$

Note that the term $\underline{w}^{T\perp} \underline{x}_i$ is equal to zero because is perpendicular to the space generated by $\underline{\text{x}}$ and so this quantity does not influences the initial computation. If instead we look at the norm,

$$\|\underline{w}\| = \left\| \sum_{i=1}^{n} \alpha_i \underline{x}_i + \underline{w}^\perp \right\| \qquad (129)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \underline{x}_i^T \underline{x}_j - 2\sum_{i=1}^{n} \alpha_i \underline{x}_i^T \underline{w}^{\perp} + \underline{w}^{\perp T} \underline{w}^{\perp} \tag{130}$$

$$= \left\| \sum_{i=1}^{n} \alpha_i \underline{x}_i \right\|^2 + \left\| \underline{w}^{\perp} \right\|^2 \tag{131}$$

It is trivial to say that this term influences the computation. If we put to zero the quantity $\underline{w}^{\perp}$ we obtain something minor with respect to $\underline{w}^*$ and so we have shown that $\underline{w}^*$ isn't the real minimum since the minimum would be unique having a convex problem. As a consequence we can state that $\underline{w}$ can only be expressed as $\underline{w} = \sum_{i=1}^{n} \alpha_i \underline{x}_i$. In the matrix form is given by,

$$\underline{w} = \sum_{i=1}^{n} \alpha_i \underline{x}_i = X^T \underline{\alpha} \tag{132}$$

where $dim\left(X^T\right) = d \times n$ and $dim\left(\underline{\alpha}\right) = n \times 1$. This time we have to minimize with respect to $\underline{\alpha}$,

$$\min_{\underline{\alpha}} \left\| X X^T \underline{\alpha} - \underline{y} \right\|^2 + \lambda \underline{\alpha}^T X X^T \underline{\alpha} \tag{133}$$

which can be rewrite as,

$$\min_{\underline{\alpha}} \left\| Q \underline{\alpha} - \underline{y} \right\|^2 + \lambda \underline{\alpha}^T Q \underline{\alpha} \tag{134}$$

Here $Q = XX^T$ and the element Q_{ij} is the i-th row of x multiply by the j-th column of x^T, i.e. $Q_{ij} = \underline{x}_i^T \underline{x}_j$. Due to the fact that $Q = Q^T$ i.e. Q is symmetric and positive define so it is invertible, we can go on with the computation of the gradient and put it to zero,

$$\underline{\alpha}^T QQ\underline{\alpha} - 2\underline{\alpha}^T Q\underline{y} + \underline{y}^T \underline{y} + \lambda \underline{\alpha}^T Q \underline{\alpha} = \underline{0} \tag{135}$$

then

$$\underline{\alpha} = \left(Q + \lambda I\right)^{+} \underline{y} \tag{136}$$

where $dim\left(\underline{\alpha}\right) = n \times 1$ according to the fact that $dim\left(\left(Q + \lambda I\right)^{+}\right) = n \times n$ and $dim\left(\underline{y}\right) = n \times 1$.

2.5.2 Notion 2

We transform the original input data into a higher dimensional space using a nonlinear mapping. Several common nonlinear mappings can be used in this step, as we will describe further below. Once the data have been transformed into the new higher space, the second step searches for a linear separating hyperplane in the new space. We again end up with a quadratic optimization problem that we are able to solve. The basic idea is that if we can't separate samples in a low-dimensional space using a hyperplane, then mapping everything to a higher dimensional space we can separate them. To do this transformation we introduce vectorial function $\phi : \Re^d \rightarrow \Re^D$. In this way the dimension of the problem

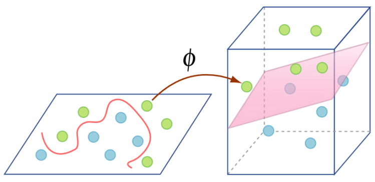


Figure 23: Mapping to a higher dimensional space using ϕ

will be D instead of d. This means that we can increase the dimensionality as we want. In this new space we want to approximate our points with an hyperplane so we have to compute,

$$\underline{w}\underline{\phi}\left(\underline{x}\right) = \sum_{i=1}^{n} \alpha_i \underline{\phi}^T \left(\underline{x}_i\right) \underline{\phi}\left(\underline{x}_j\right) \tag{137}$$

Whit this equation we can compute the value of y relative to a new point. But how it can be expressed the product $\underline{\phi}^T\left(\underline{x}_i\right)\underline{\phi}\left(\underline{x}_j\right)$? This is especially expensive. Hence, the dot product computation required is very heavy and costly. We can use another math trick. Instead of computing the dot product on the transformed data tuples, it turns out that it is mathematically equivalent to instead apply a kernel function, $K(X_i, X_j)$, to the original input data. That is, $K(X_i, X_j) = f(X_i)f(X_j)$. In other words, everywhere that $f(X_i)f(X_j)$ appears in the training algorithm, we can replace it with $K(X_i, X_j)$. In this way, all calculations are made in the original input space, which is of potentially much lower dimensionality. We can safely avoid the mapping. It turns out that we don't even have to know what the mapping is. After applying this trick, we can then proceed to find a maximal separating hyperplane. All of this things can be done according to the *Mercer's theorem* which state that exist some scalar function on vectorial variables called kernel such that,

$$k\left(\underline{x}_i, \underline{x}_j\right) = \underline{\phi}^T\left(\underline{x}_i\right)\underline{\phi}\left(\underline{x}_j\right) \tag{138}$$

An admissible kernel function is *Gaussian radial basis function kernel*:

$$K(x_i, x_j) = e^{-\gamma\left\|\underline{x}_i - \underline{x}_j\right\|^2} \tag{139}$$

In this way we can use the kernel instead of compute the inner product and we can compute the values in the space generated from $\underline{x}$ avoiding the course of dimensionality. Let's see if we find a way to compute ϕ after the projection in the new space. A trivial case is consider a space of dimension one, so kernel is given by,

$$K(u, v) = e^{-\gamma(u-v)^2} = e^{-\gamma u^2}e^{-\gamma v^2}e^{2\gamma uv} \tag{140}$$

if we expand this with Taylor's series we obtain,

$$e^{-\gamma x} = \sum_{i=0}^{n}\frac{1}{i!}f^i\left(x\right)x^i = 1 + x + \frac{1}{2}x^2 + \cdots = \sum_{i=0}^{\infty}\frac{1}{i!}x^i \tag{141}$$

and in our example,

$$e^{-\gamma u^2}e^{-\gamma v^2}e^{2\gamma uv} = \underline{\phi}^T\left(u\right)\underline{\phi}\left(v\right) \tag{142}$$

The Gaussian kernel makes a projection from a mono dimensional space to an infinite dimensional space where we can

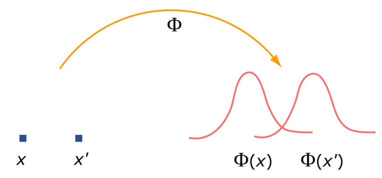


Figure 24: vectorial function $\phi : \Re^d \rightarrow \Re^D$

fit an infinite number of data. It is also called kernel with infinite approximation power for this reason. The product

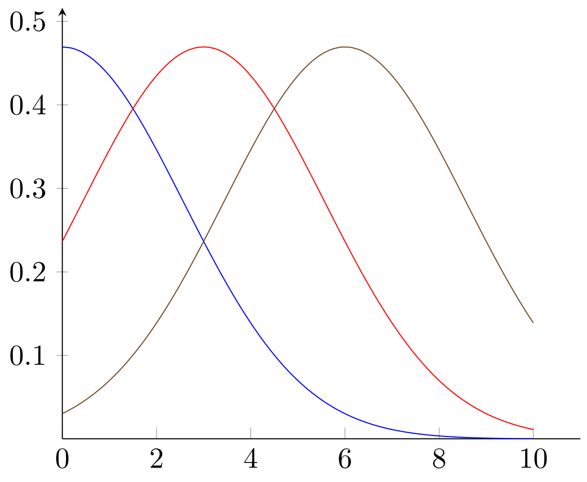


Figure 25: For γ small the bell goes left (blue line); For γ big the bell goes right (olive line)

$\phi\underline{w}$ gives origin to figure 25. When γ is small the bell goes left meaning that the exponential part is stronger then the polynomial one. If instead γ is big the bell goes right. If γ is very small, we are considering the first terms of Taylor expansion and so we have a simple separator (linear) else if γ is very big the resulting form is complex. Then γ can be seen as a regulator of the non linearity of the function, for small values of γ we obtain hyperplanes, otherwise we obtain complex separators. Then the bias term $\lambda\left\|\underline{w}\right\|^2$ can be seen as a filter, where λ shows the amplitude of the filter (low-pass filter, high-pass filter). It penalize $\underline{w}$ big, which is what we want since we are minimizing the function.

2.6 SVM

2.6.1 Introduction

In this section, we study Support Vector Machines, a method for the classification of both linear and nonlinear data. A support vector machine (or SVM) is an algorithm that works as follows. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors (training tuples) and margins (defined by the support vectors). We will go into these new concepts further below. The first paper on support vector machines was presented in 1992 by Vladimir Vapnik. SVMs can be used for prediction as well as classification as we will seen later.

2.6.2 Support Vector Machines

Let's first look at the simplest case, a two-class problem, where the classes are linearly separable. Let the data set D be given as $(X_1, y_1), (X_2, y_2), ...$ and $X \in \Re^{n\times d}$,where X_i is the set of training tuples with associated class labels, y_i. Each y_i can take one of two values, either +1 or -1 (i.e.,$y_i \in \{+1, -1\}$), and $y = f(\underline{x})$ which approximates the relation between x and y. If we plot the product $y\hat{f}$ i.e. the product between the true y and the estimated value f, we are interested in obtaining the correct tag during classification, so the error function is given by the so called hard loss (Fig. 26). When the signs are the same we have right prediction, else we have some error. The problem of this function is that it can't be easily minimized (NP problem). In order to minimize it it should be convex. Mathematicians to solve this problem apply minimum square method. In this way we obtain a parabola $(y - f)^2$, This parabola is an approximation

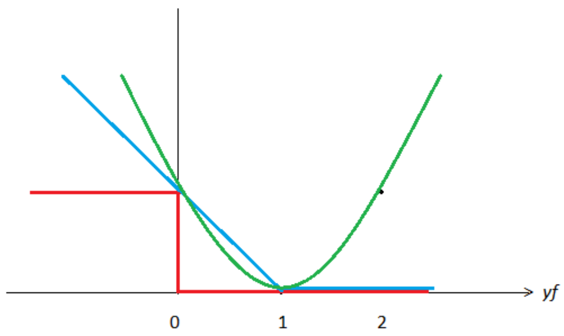


Figure 26: Hard loss in red, Hinge loss in blue, parabola in green is an approximation of hard loss function

of the hard loss, and it is good for $yf < 1$ but it is very bad for $yf > 1$. In 1979 Vapnik finds that the best convex approximation for a loss is a straight line known as hinge loss (Fig. 26). Given this function we want $yf > 1$ for all points,

$$\begin{cases} y_i f(x_i) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \tag{143}$$

and we have to minimize the error function $\min \frac{1}{n}\sum_{i=0}^{n} \xi_i$. Note that if we are looking for a linear separator then $f(x) = \underline{w}^T\underline{x} + b$ else $f(x) = \underline{w}^T\underline{\phi}(x) + b$ for non linear case. This is the first possible interpretation of svm algorithm, since we have to minimize the hinge loss function. Another way to see it, is the optimization problem of determining these hyperplanes with the notion of margin. Intuitively, a maximum margin hyperplane is one that cleanly separates the two classes, and for which a large region (or margin) exists on each side of the boundary with no training data points in it. If we add the regularization term to the error function, we have $\min \frac{1}{n}\sum_{i=0}^{n} \xi_i + \lambda \|\underline{w}\|^2$ in order to limit the non linearity of the function, we can state that the margin is $\frac{2}{\|\underline{w}\|}$. Looking at the figure we want $yf > 1$, which means that points have to be within the respective support hyperplane. Since we have to minimize $\|\underline{w}\|^2$ is equivalent to maximize the distance between the two support hyperplanes and so the margin. This leads to the second interpretation that is the maximum margin hyperplane search. From a statistical point of view it corresponds to minimize the unbiased estimator error plus a bias term. Now we can rewrite the error function as,

$$\min_{\underline{w},b,\underline{\phi}} \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{n} \xi_i \tag{144}$$

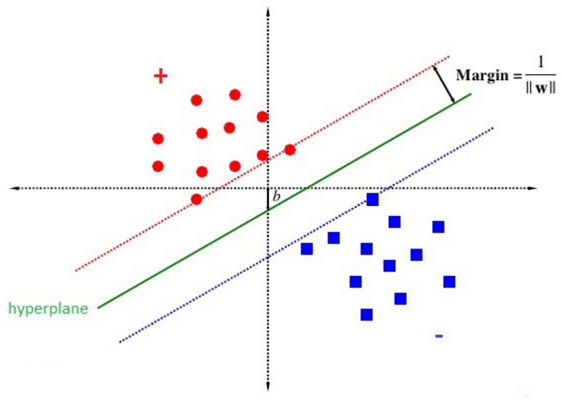


Figure 27: max margin hyperplane

$$y_i\left(\underline{w}^T\phi\left(x_i\right)+b\right)\geq 1-\xi_i \qquad \text{constraint 1} \tag{145}$$
$$\xi_i \geq 0 \qquad \text{constraint 2} \tag{146}$$

Here C is a regularization term, in the sense that if C goes to infinite, the error is small but the solution is very complex, in the other case if $C = 0$ then the error doesn't cost so much and we only have to minimize $\underline{w}$. Now, we have define the problem, so to solve it we can apply Lagrangian relaxation. Primal problem is given by,

$$\mathcal{L}_p\left(\underline{w}, b, \underline{\phi}, \underline{\alpha}, \underline{\beta}\right) =$$
$$\frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\underline{w}^T\phi\left(x_i\right)+b\right)-1+\xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i \tag{147}$$
$$y_i\left(\underline{w}^T\phi\left(x_i\right)+b\right)\geq 1-\xi_i \qquad \text{primal constraint} \tag{148}$$
$$\xi_i \geq 0 \qquad \text{primal constraint} \tag{149}$$
$$\alpha_i \geq 0 \qquad \text{inequality constraint} \tag{150}$$
$$\beta_i \geq 0 \qquad \text{inequality constraint} \tag{151}$$
$$\alpha_i\left[y_i\left(\underline{w}^T\phi\left(x_i\right)+b\right)-1+\xi_i\right]=0\ \forall i \qquad \text{compl. constraint} \tag{152}$$
$$\beta_i\xi_i = 0\forall i \qquad \text{complementary constraint} \tag{153}$$
$$\frac{\partial\mathcal{L}_p}{\partial\underline{w}} = 0 \qquad \text{KKT} \tag{154}$$
$$\frac{\partial\mathcal{L}_p}{\partial b} = 0 \qquad \text{KKT} \tag{155}$$
$$\frac{\partial\mathcal{L}_p}{\partial\underline{\xi}} = 0 \qquad \text{KKT} \tag{156}$$
$$\tag{157}$$

The original problem is convex, since w^2 is a paraboloid , the error function is convex for construction, and constraints give origin to a convex space. So after this check, we can go on and compute KKT conditions,

$$\frac{\partial\mathcal{L}_p}{\partial\underline{w}} = \underline{w} - \sum_{i=1}^{n}\alpha_i y_i\phi\left(x_i\right) = \underline{0} \qquad \text{KKT} \tag{158}$$
$$\frac{\partial\mathcal{L}_p}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0 \qquad \text{KKT} \tag{159}$$
$$\frac{\partial\mathcal{L}_p}{\partial\underline{\xi}} = C - \alpha_i - \beta_i = 0 \qquad \text{KKT} \tag{160}$$
$$\tag{161}$$

From the last constraint we have $\alpha_i \leq C$. Then using KKT conditions we obtain the dual formulation of the problem. Since the problem satisfies strong duality condition, is given by,

$$\min_{\underline{\alpha}}\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k\left(x_i, x_j\right) - \sum_{i=1}^{n}\alpha_i \tag{162}$$

$$0 \leq \alpha_i \leq C$$
$$-\sum_{i=1}^{n} \alpha_i y_i = 0$$

Once found α_i, we have to compute $f\left(\underline{x}\right)$ that using the first KKT condition, is

$$f\left(\underline{x}\right) = \underline{w}^T \underline{\phi}\left(x\right) + b = \sum_{i=1}^{n} \alpha_i y_i k\left(x_i, x_j\right) + b \tag{163}$$

Now b is the only unknown quantity. To find it we have to analyze $\underline{\alpha}$ and to study the solution for $0 \leq \alpha_i \leq C$.

- If $\xi_i \geq 0$, it means I'm considering the point in the wrong side of the hyperplane, then $\beta_i = 0$, then $\alpha_i = C$ so $y_i f_i = 1 - \xi_i$. Points in the wrong side have $\alpha_i = C$. In this case we know nothing on b.
- If $y_i f_i > 1$, it means I'm considering the point in the right side of the support hyperplane, then $\alpha_i = 0$, then $\beta_i = C$, then $\xi_i = 0$. Even in this case we know nothing on b.
- If $0 \leq \alpha_i \leq C$ from the last KKT condition we have $0 \leq \beta_i \leq C$, then $\xi_=0$ since $\beta_i \neq 0$. But if $\alpha_i \neq 0$ then $y_i\left(\underline{w}^T \phi\left(x_i\right) + b\right) = 1$. From this one we are able to compute b.

Due to this fact, we can state that svm algorithm make a sort of compression. We said that all points within the support hyperplane are classified in the correct way and they have $\alpha_i = 0$. If the model is good then many points will be rightly classified and then the vector alpha will be sparse. Points corresponding to $\alpha = 0$ are called *0 support vector*, Points with $0 \leq \alpha_i \leq C$ are known as *true support vector*, and those with $\alpha = C$ are the so called critical points or *support vector*. Keep in mind that the solution depends on critical points, which are noise. It learns from mistakes as humans do. Note also that we have to tune hyper parameters C and γ. If we are using a radial basis kernel with a small value of γ we obtain linear solutions, else we have non linear solutions. Moreover, if we use a small value of C then the solution will be smooth (it means that $\underline{w}$ will be small), while if C is big we only care about making a small error. The model shown in this section is also known as "KRLS" (Kernel Regularization Least Square). The solution is given by the pseudo-inverse and with no further assumptions, y could be either a classification problem with $y \in \{+1, -1\}$ or a regression problem with $y \in \Re$. Intuitively, it seems that KRLS, is not suited to classification problems and in particular for multiclass case, but according to *"no free lunch" theorem* which *states that any two optimization algorithms are equivalent when their performance is averaged across all possible problems.*, KRLS is used even in these cases. Using kernel we shown that we can solve problem projecting data in a higher dimensional space and to regularize non linearity with a bias term. SVM is the same concept but instead to use least square model it used hinge loss function. SVM limit s that it works only with binary problem. How we can solve multiclass problem with $y \in \{1, \cdots, c\}$ or regression problem , $y \in \Re$? Several strategies are possible to convert biclass classifiers into multilabeled classifiers. In the following discussion, it will be assumed that the number of classes is denoted by c.

2.6.3 Multiclass case

What is multiclass classification? Each training point belongs to one of C different classes. The goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs. The idea here, is to reduce multiclass problem to many biclass problem that we are able to solve. The are three main methods to do that: (Please, refer to this paper to get more information about this topic:
`http://www.jmlr.org/papers/volume5/rifkin04a/rifkin04a.pdf`)
One versus All (**OVA**): In this approach, c different biclass classification problems are created, such that one problem corresponds to each class. In the ith problem, the ith class is considered the set of positive examples, whereas all the remaining examples are considered negative examples. The binary classifier is applied to each of these training data sets. This creates a total of c models. If the positive class is predicted in the ith problem, then the ith class is rewarded with a vote. Otherwise, each of the remaining classes is rewarded with a vote. The class with the largest number of votes is predicted as the relevant one. Let us make an example (refer to figure 28). The given dataset is X and the points are labeled in three classes. Then we create three sub-problems. In the first we put class one against remaining classes, filling with 1 the respective positions and with -1 the others cells. We have to do this for each class. The problem is that we have to solve c problems of size n , where n is the dimension of the training set; moreover the problem is not balanced, since we have $\frac{1}{c}$ in 1, and remaining $\frac{c-1}{c}$ in minus one. It doesn't support well increasing of classes, so it can be fine for problem with few classes.

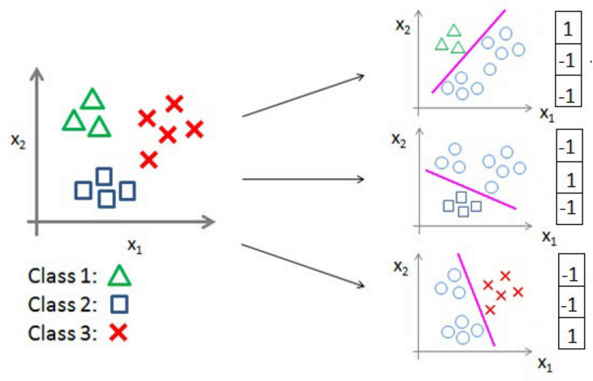
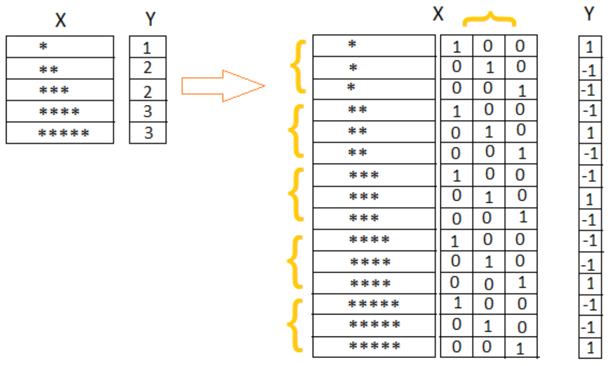


Figure 28: Ova classification

All versus All (**AVA**) : In this strategy, a training data set is constructed for each of the $\binom{c}{2}$ pairs of classes. The algorithm is applied to each training data set. This results in a total of $c\frac{(c-1)}{2}$ models. For each model, the prediction provides a vote to the winner. The class label with the most votes is declared as the winner in the end. Note that in this case if each model refers to different tags, we are not able to choose, and in this case we compute the distance to the hyperplane for each function and we choose the one which maximize it. At first sight, it seems that this approach is computationally more expensive, because it requires us to train $c\frac{(c-1)}{2}$ classifiers, rather than training c classifiers, as in the one-versus-all approach. However, the computational cost is ameliorated by the smaller size of the training data in the all-versus-all approach. Specifically, the training data size in the latter case is approximately $\frac{2n}{c}$ of the training data size used in the one-versus-all approach on the average. Viewed naively, AVA seems faster and more efficient in terms of memory. It requires $O(N^2)$ classifiers instead of $O(N)$, but each classifier is on average much smaller. If the time to build a classifier is superlinear in the number of data points, AVA is a better choice. We have to pay attention using AVA because in the case that for some classes there are few available patterns the classification can be misleading. Anyway with SVMs, AVA's probably best. However, if you can solve one RLS problem over your entire data set using a matrix factorization, you get multiclass classification essentially for free. So with RLS, OVA's a great choice.



X	Y
*	1
**	2
***	2
****	3
*****	3

X				Y
*	1	0	0	1
*	0	1	0	-1
*	0	0	1	-1
**	1	0	0	-1
**	0	1	0	1
**	0	0	1	-1
***	1	0	0	-1
***	0	1	0	1
***	0	0	1	-1
****	1	0	0	-1
****	0	1	0	-1
****	0	0	1	1
*****	1	0	0	-1
*****	0	1	0	-1
*****	0	0	1	1

Figure 29: Given a dataset X and y, we have to create a matrix in which each sample is repeated c times. For each group of repeated samples we concatenate a diagonal matrix $c \times c$. The resulting y will be a column vector which maps the samples with the proper class assigning 1 for correct tag and -1 for the wrong ones.

Augmented Binary (**AB**) is one of the methods generating a multicategory classifier in one step, without dividing the multiclass problem into binary sub-problems. This approach can be useful when the number of samples is very low. We transform the multiclass classification problem into a binary one, by replicating each pattern c times and augmenting each replica with a vector of size c. After transforming the dataset, we can solve the usual SVM optimization problem. Keep in mind that in this case the size of the problem is nc and is not balanced, but is easy to implement. In order to perform the classification of a new pattern, we obviously need to apply the same procedure described above, by replicating the pattern and augmenting it. Then, each augmented pattern is fed to the SVM and the decision is taken according to a Winner-Takes-All (WTA) arbiter. See at Fig. 29. Just to be sure we have well understand how the things work. Consider the figure. We have to classify three new data points. We can say that we are most sure about the classification of the green point, but we would know also the accuracy of this prediction, i.e a probability $prob = p$. But in general we can go further and say that we have $prob = p + \epsilon$ at $(1 - \delta)\,\%$. If the probability is under a certain threshold, as in the case of the orange point, we are recommended to add a class called “rejected” which will include this type of points and so we don’t have to make a decision. This can be useful as well as we are studying, for example, a medical case. We can model the probability with a sigmoid

$$p(x) = \frac{1}{1 + e^{-\gamma f(x)}} \tag{164}$$

Here γ is an indicator of how the probability goes to zero and the value of $f(x)$ gives us the probability: if $f(x)$ is strongly positive then $p = 1$, if is negative ($-\infty$) then $P = 0$, and in other case $p = 0.5$. How we compute γ? It is set in order to maximize the probability on data. If we have many errors then we will use a small γ. The fastest way to optimize γ in the case of one variable is to apply a brute force method.

2.7 SVR

The interested reader can find more details in this paper:
`http://alex.smola.org/papers/2004/SmoSch04.pdf`
In this section we show how SVMs can be adapted for regression with a quantitative response, in ways that inherit some of the properties of the SVM classifier. In regression problems what is important is $y - f$, i.e the difference between the true value and the predicted one. If we plot the error function computed with minimum square method as in figure 31 we would pay a quadratic cost. We do not want to pay a quadratic cost, so we want an error function which ignores data

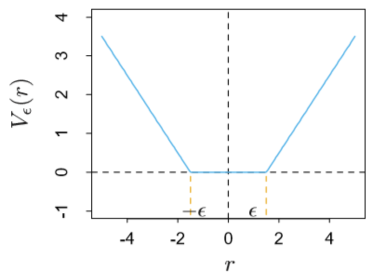


Figure 30: ϵ-insensitive error function

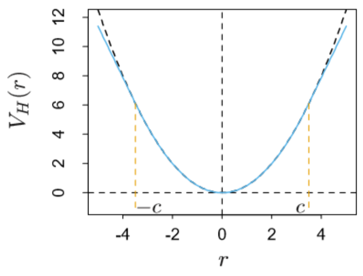


Figure 31: Least square error function

point with a small error and which not penalizes to much if the error is too big. Then we can use the “ϵ-insensitive” error measure, ignoring errors of size less than ϵ as in figure 30. Once have chosen the loss function we can proceed with the regression problem. We have to minimize,

$$\min_{w,b} \frac{1}{2} \|\underline{w}\|^2 + C \sum_{i=1}^{n} (\hat{\varepsilon}_i + \breve{\varepsilon}_i) \tag{165}$$

$$y_i - \left(\underline{w}^T \underline{\phi}(\underline{x}_i)\right) \leq \epsilon + \hat{\varepsilon}_i \tag{166}$$

$$y_i - \left(\underline{w}^T \underline{\phi}(\underline{x}_i)\right) \geq -\epsilon - \breve{\varepsilon}_i \tag{167}$$

$$\hat{\varepsilon}_i, \breve{\varepsilon}_i \geq 0 \tag{168}$$

The primal problem is given by,

$$\begin{aligned}\mathcal{L}_p =& \frac{1}{2} \|\underline{w}\|^2 + C \sum_{i=1}^{n} (\hat{\varepsilon}_i + \breve{\varepsilon}_i) - \sum_{i=1}^{n} \hat{\alpha}_i \left[-y_i + \left(\underline{w}^T \underline{\phi}(\underline{x}_i)\right) + \epsilon + \hat{\varepsilon}_i\right] + \\ & - \sum_{i=1}^{n} \breve{\alpha}_i \left[y_i - \left(\underline{w}^T \underline{\phi}(\underline{x}_i)\right) + \epsilon + \breve{\varepsilon}_i\right] - \sum_{i=1}^{n} (\hat{\mu}_i \hat{\varepsilon}_i + \breve{\mu}_i \breve{\varepsilon}_i)\end{aligned} \tag{169}$$

where the usual constraints are required. We set to zero the partial derivatives of the Lagrangian and we obtain the following:

$$\frac{\partial \mathcal{L}_p}{\partial \underline{w}} = 0 \rightarrow \underline{w} = \sum_{i=1}^{n} (\hat{\alpha}_i - \breve{\alpha}_i) \underline{\phi}(x_i) \tag{170}$$

$$\frac{\partial \mathcal{L}_p}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} (\hat{\alpha}_i - \breve{\alpha}_i) = 0 \tag{171}$$

$$\frac{\partial \mathcal{L}_p}{\partial \hat{\varepsilon}_i} = 0 \to C - \hat{\alpha}_i - \hat{\mu}_i = 0 \tag{172}$$
$$\frac{\partial \mathcal{L}_p}{\partial \breve{\varepsilon}_i} = 0 \to C - \breve{\alpha}_i - \breve{\mu}_i = 0 \tag{173}$$

Substituting these conditions in the primal problem, we obtain the dual formulation for SVR:

$$\mathcal{L}_d = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{\alpha}_i - \breve{\alpha}_i)(\hat{\alpha}_j - \breve{\alpha}_j) K\left(\underline{x}_i - \underline{x}_j\right) + \sum_{i=1}^{n}(\hat{\alpha}_i - \breve{\alpha}_i) y_i - \sum_{i=1}^{n}(\hat{\alpha}_i + \breve{\alpha}_i)\varepsilon \tag{174}$$

Under strong duality we can convert this maximum problem into a minimum problem,

$$\min_{\hat{\alpha},\breve{\alpha}} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{\alpha}_i - \breve{\alpha}_i)(\hat{\alpha}_j - \breve{\alpha}_j) K\left(\underline{x}_i - \underline{x_j}\right) - \sum_{i=1}^{n}(\hat{\alpha}_i - \breve{\alpha}_i) y_i + \sum_{i=1}^{n}(\hat{\alpha}_i + \breve{\alpha}_i)\varepsilon \tag{175}$$
$$\sum_{i=1}^{n}(\hat{\alpha}_i - \breve{\alpha}_i) = 0$$
$$0 \leq \hat{\alpha}_i \leq C$$
$$0 \leq \breve{\alpha}_i \leq C$$

If we want the matrix formulation is given by,

$$\min_{\underline{\hat{\alpha}},\underline{\breve{\alpha}}} \begin{bmatrix}\underline{\hat{\alpha}}\\ \underline{\breve{\alpha}}\end{bmatrix}^T \begin{bmatrix} Q & -Q \\ -Q & Q\end{bmatrix} \begin{bmatrix}\underline{\hat{\alpha}}\\ \underline{\breve{\alpha}}\end{bmatrix} + \begin{bmatrix}\epsilon - y_i\\ \epsilon + y_i\end{bmatrix}^T \begin{bmatrix}\underline{\hat{\alpha}}\\ \underline{\breve{\alpha}}\end{bmatrix} \tag{176}$$

$$\begin{bmatrix}\underline{1}\\ -\underline{1}\end{bmatrix}^T \begin{bmatrix}\underline{\hat{\alpha}}\\ \underline{\breve{\alpha}}\end{bmatrix} = 0 \tag{177}$$
$$0 \leq \begin{bmatrix}\underline{\hat{\alpha}}\\ \underline{\breve{\alpha}}\end{bmatrix} \leq C \tag{178}$$

In this case, taking into account the ϵ-insensitivity, the maximum rank of Q is n. So the problem is not well conditioned in the sense that if we have a not full ranked matrix we can't measure the space directly. Then we have computational problems since to reach the minimum it can take a very long time. Instead, if we have $\epsilon = 0$ and $\underline{\beta} = \underline{\hat{\alpha}} - \underline{\breve{\alpha}}$ the problem is now,

$$\min_{\underline{\beta}} \frac{1}{2}\underline{\beta}^T Q \underline{\beta} - \underline{y}^T \underline{\beta} \tag{179}$$
$$\underline{1}^T \underline{\beta} = 0$$
$$-\underline{c} \leq \underline{\beta} \leq \underline{c} \tag{180}$$

Here, Q is full rank and the size of the problem is n. Now let's take look to the solution:

- If $\underline{\hat{\alpha}}, \underline{\breve{\alpha}} = 0$ then the solution lies inside the ϵ-insensitive tube and so it is considered to be correctly predicted;
- If $0 \leq \underline{\hat{\alpha}}, \underline{\breve{\alpha}} \leq C$ then the solution lies on the discontinuity point which means that it is a True Support Vector and can be used for computing the bias b;
- If $\underline{\hat{\alpha}}, \underline{\breve{\alpha}} = C$ then it lies outside the ϵ-tube and, then, is mispredicted.

Also in this case we make compression and we learn from errors.

3 Data Preprocessing and Exploratory Data Analysis

Success depends upon previous preparation, and without such preparation there is sure to be failure. Confucius

3.1 Introduction

The data preprocessing phase is perhaps the most crucial one in the data mining process. Yet, it is rarely explored to the extent that it deserves because most of the focus is on the analytical aspects. The data preparation phase is a multistage process that comprises several individual steps as follows:

1. *Feature extraction and portability*: The raw data is often in a form that is not suitable for processing. In some cases where the data is obtained from multiple sources, it needs to be integrated into a single database for processing. In addition, some algorithms may work only with a specific data type, whereas the data may contain heterogeneous types. In such cases, data type portability becomes important where attributes of one type are transformed to another. This results in a more homogeneous data set that can be processed by existing algorithms.

2. *Data cleaning*: In the data cleaning phase, missing, erroneous, and inconsistent entries are removed from the data. In addition, some missing entries may also be estimated.

3. *Data reduction, selection, and transformation*: In this phase, the size of the data is reduced through data subset selection, feature subset selection, or data transformation. The gains obtained in this phase are twofold. First, when the size of the data is reduced, the algorithms are generally more efficient. Second, if irrelevant features or irrelevant records are removed, the quality of the data mining process is improved.

In particular, the data cleaning process is important because of the errors associated with the data collection process. There are several important aspects of data cleaning:

1. Handling missing entries: Many entries in the data may remain unspecified because of weaknesses in data collection or the inherent nature of the data. Such missing entries may need to be estimated.

2. Handling incorrect entries: In cases where the same information is available from multiple sources, inconsistencies may be detected. Such inconsistencies can be removed as a part of the analytical process. Another method for detecting the incorrect entries is to use domain-specific knowledge about what is already known about the data. More generally, data points that are inconsistent with the remaining data distribution are often noisy. Such data points are referred to as outliers. It is, however, dangerous to assume that such data points are always caused by errors. For example, a record representing credit card fraud is likely to be inconsistent with respect to the patterns in most of the (normal) data but should not be removed as "incorrect" data.

3. Scaling and normalization: The data may often be expressed in very different scales. This may result in some features being inadvertently weighted too much so that the other features are implicitly ignored. Therefore, it is important to normalize the different features.

The following sections will discuss each of these aspects of data cleaning.

3.2 Data Preprocessing

3.2.1 Missing Data.

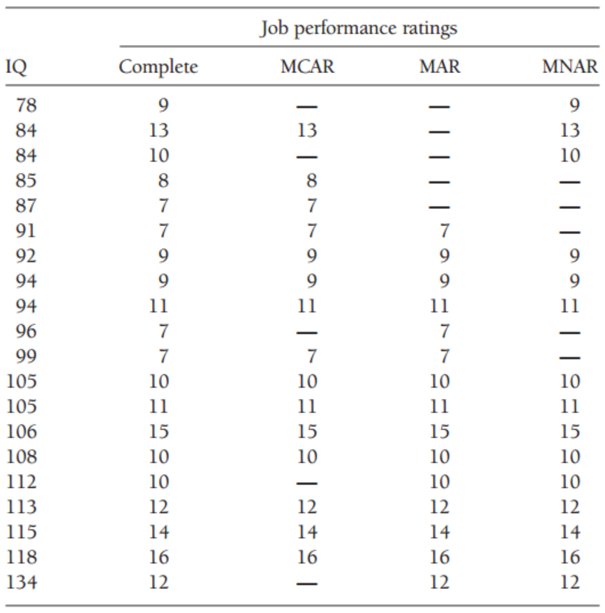
It is quite common to have observations with missing values for one or more input features. The usual approach is to impute the missing values in some way. However, the first issue in dealing with the problem is determining whether the missing data mechanism has distorted the observed data. Roughly speaking, data are missing at random if the mechanism resulting in its omission is independent of its (unobserved) value.

- Missing at Random (MAR) :The probability of missing data on a variable X is related to some other measured variable(s) but not on the value of X.

- Missing Completely at Random (MCAR): The probability of missing data on a variable X is unrelated to all measured variables (including itself).

- Missing Not at Random (MNAR):The probability of missing data is related to the value of the variable X.

Let's see an example (Fig.32) where IQ column is the X value, Complete column is Y and we want to know the correlation between them. In the case of MCAR and MAR we can infer or extract the missing value from data we have, while in the last case is not possible to make any assumptions. Assuming the features are missing completely at random or are missing at random, there are a number of ways of proceeding:



	Job performance ratings			
IQ	Complete	MCAR	MAR	MNAR
78	9	—	—	9
84	13	13	—	13
84	10	—	—	10
85	8	8	—	—
87	7	7	—	—
91	7	7	7	—
92	9	9	9	9
94	9	9	9	9
94	11	11	11	11
96	7	—	7	—
99	7	7	7	—
105	10	10	10	10
105	11	11	11	11
106	15	15	15	15
108	10	10	10	10
112	10	—	10	10
113	12	12	12	12
115	14	14	14	14
118	16	16	16	16
134	12	—	12	12

Figure 32: Job Performance Ratings with MCAR, MAR, and MNAR Missing Values.

- Rely on the learning algorithm to deal with missing values (e.g. decision tree).
- *Deletion*: Discard observations with any missing values (so remove the entire sample).
- *Imputation*: Impute all missing values before training.

Note that approach (2) can be used if the relative amount of missing data is small, but otherwise should be avoided. For most learning methods, the imputation approach (3) is necessary. The simplest tactic is to impute the missing value with the mean or median of the non missing values for that feature. We have n samples, m missing samples. Let's define $r = n - m$ i.e. the number of available samples. If we define the set of available index as $s_r = \{i | 1 \leq i \leq n, Y_i \text{ is available}\}$ and the set of missing index as $s_m = \{i | 1 \leq i \leq n, Y_i \text{ is missing}\}$ then we can construct a set as big as the set of missing values $\{Y_i^*\}, i \in s_m$ and now we are able to compute the mean value on the available data $\bar{Y}_r = \frac{1}{r} \sum_{i \in s_r} Y_i$ and also the mean value on the missing data $\bar{Y}_m^* = \frac{1}{m} \sum_{i \in s_m} Y_i^*$. The different assignment to $\tilde{Y}_i$ leads to different imputation methods:

- Mean imputation: the missing fields are filled with the mean value computed over the available data $\tilde{Y}_i = \bar{Y}_r$. It can be shown that this method doesn't work properly.
- Random imputation: the missing fields are filled with one of the value contained in the dataset $\tilde{Y}_i = Y_i^*$. This method is statistically correct and the estimation is quite reliable.
- Adjusted random imputation: This method is an improvement of the previous one because it introduces a bias term which can be seen as a correction factor and it makes the estimation more reliable. $\tilde{Y}_i = \bar{Y}_r + \left(Y_i^* - \bar{Y}_m^*\right)$

However,with this estimation we do not gain in information. Filling-in null fields simply allow us to make known algorithms run on that dataset. After imputation, missing values are typically treated as if they were actually observed. This ignores the uncertainty due to the imputation, which will itself introduce additional uncertainty into estimates and predictions from the response model. One can measure this additional uncertainty by doing multiple imputations and hence creating many different training sets.

3.2.2 Data reduction.

This can be done in two main ways: a row reduction which can lead to a dataset size reduction and a columns reduction which can leads to a dimensionality reduction.

3.2.3 Dataset size reduction

The goal of this one in to reduce the number of samples. The first method it can be a *random sampling* but it doesn't maintain the original data distribution. So it can be done a *stratified sampling* which preserves a certain distribution. A sophisticate way to do this is the *Homogeneous multi-partitioning.* The complete algorithm is given in fig. 1. Alternating between the assigned parts walking along the chain, enforces data from different parts to be closer to each others with higher probability than a random assignment. It requires to provide a distance measure d between the data points, the number N_p of parts in the partition, and to select a seed point as the starting point of the chain. We call this algorithm the Homogeneous Multi Partition (HMP) generator. Starting from the seed, data are connected step by step along a chain, appending to its end, the nearest datum not yet in. Then it assigns each datum to one of the N_p parts cycling the parts along the chain. The sets S_E and S_U contain explored and unexplored points, respectively.

Algorithm 1 Homogeneous Multi Partition algorithm

```
1: INPUTS : $S = (x_1, \cdots .x_n)$ data set to partition,
2: $N_p$ number of parts,
3: $d$ distance function,
4: $Seed$ index of the seed
5: OUTPUTS : $P_{1,\cdots,N_p}$ parts of the partition
6: procedure HMP
7:     $Rank \leftarrow 1$;
8:     $P_1 \leftarrow Seed$
9:     $P_{2,\cdots,N_p} \leftarrow 0$;
10:    $S_E \leftarrow Seed$
11:    $S_U \leftarrow \{1, \cdots, |S|\} \backslash Seed$
12:    $Cur \leftarrow Seed$
13:    while $S_U \neq 0$ do
14:        $Cur \leftarrow \arg\min_{i \in S_U} (d(x_i, x_Curr))$
15:        $S_E \leftarrow S_E \cup Cur$;
16:        $S_U \leftarrow S_U \backslash Cur$;
17:        $Rank \leftarrow Rank + 1$;
18:        $Num \leftarrow 1 + ((Rank - 1)\, modN_p)$;
19:        $P_Num \leftarrow P_Num \cup Cur$
    endwhile
```

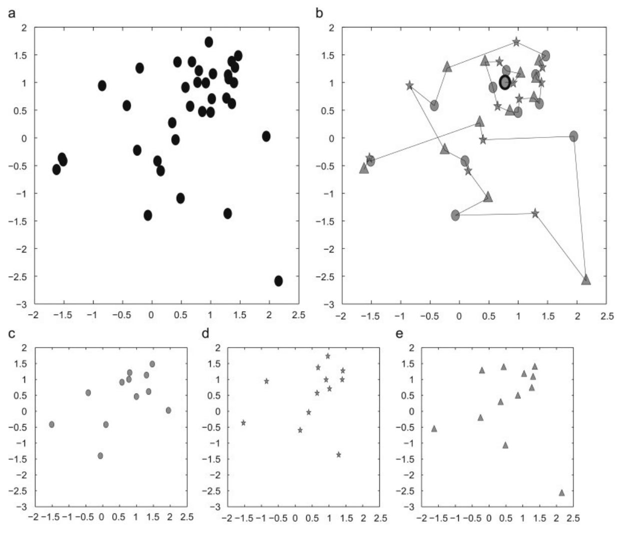


Figure 33: HMP algorithm on a two-dimensional set of points. Homogeneous Multi Partition algorithm: (a) Example on two-dimensional data with Np=3 and d the Euclidean distance. (b) The chain starts at the seed point (bold circle) arbitrarily selected in the dense area. The parts labels alternate along the chain. (c,d,e) The three parts obtained are homogeneous.

3.2.4 Dimensionality reduction

Here the goal is to reduce the number of features or in other words is to extract the more relevant features. This is very useful to not meet with the course of dimensionality, which means that there are too data and algorithms can't manage them. The first approach is *dimensionality reduction*: the data is "projected" on a low-dimensional manifold.

$$x_1, \cdots, x_d \longrightarrow x_1, \cdots, x_q \quad q \ll d \tag{181}$$

Take into account that in this way the real meaning of variables is completely lost. In order to do dimensionality reduction there are linear and non linear methods. Linear methods

- *Principal Component Analysis* For example if data have in some sense a linear distribution, performing an axes rotation we can reduce the dimensionality of the problem redefining the problem equations with respect to the new axis.

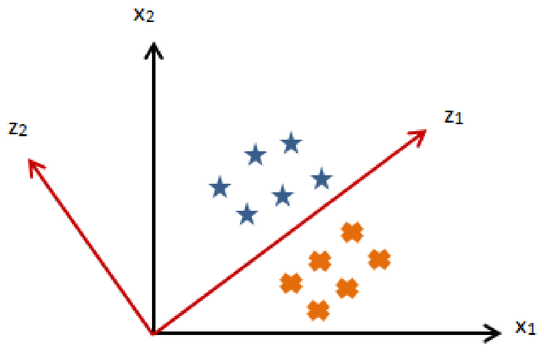


Figure 34: After axis rotation we can use only z_2 to discriminate points.

Non linear models

- *Distance preservation*: Euclidean distances (Metric Multidimensional scaling, Sammon's nonlinear mapping, Curvilinear Component Analysis)
- *Distance preservation*: graph distances (Isomap, Geodesic nonlinear mapping, Curvilinear Distance Analysis)
- *Distance preservation*: other distances (Kernel PCA,Semidefinite embedding)
- *Topology preservation* (Kohonen's Self-Organizing Maps, Generative Topographic Mapping, Locally Linear Embedding, Laplacian Eigenmaps, Isotop)
- *Information preservation* (Auto Associative Neural Network)

The second approach is the *Feature selection*: a subset of features is selected.There are three general approaches

- *Filters* (Correlation, Mutual Information) : we can compute the correlation between columns and discard those which have a low correlation
- *Wrappers* Build a model and evaluate it by adding (forward) or removing (backward) features step-by-step. The search space of different subsets of features need to be explored to determine the optimum combination of features.
- *Embedded*: the algorithm returns a sparse model and automatically chooses significant features.

3.2.5 Data Preprocessing.

We need to normalize data in order to make them numerically treatable. For numerical variables we can normalize in known ranges e.g the uniform range $[0, 1]$.

$$X' = \frac{X - MIN_X}{MAX_X - MIN_X} \tag{182}$$

or we can may be interested in the range $[-1, 1]$:

$$X' = 2\frac{X - MIN_X}{MAX_X - MIN_X} \tag{183}$$

The problem of this one is that it deletes also outliers and in some case we need them to discover non frequent patterns and so this method can't be applied. A less disruptive way to normalize data is to apply *Z-score* method:

$$X' = \frac{X - \bar{X}}{\sigma_X} \tag{184}$$

For example the candidate for dropping is the variable with the smallest Z-score, or if we have a Gaussian distribution we can decide to drop the tail value according to the z-score.

3.3 Exploratory Data Analysis

The first important thing to do with data is to observe them. If we limited the analysis to statistical value , this can lead us to wrong conclusions. If we look at Fig.35 in which are represented four datasets and we compute statistical computation, we obtain the same value, and so the same regression model even if , if we look at the real data distribution we discover that are all different (Fig.36). To make a detailed analysis, Pearson's correlation coefficient can be computed

Anscombe's quartet

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Figure 35: Four different dataset.

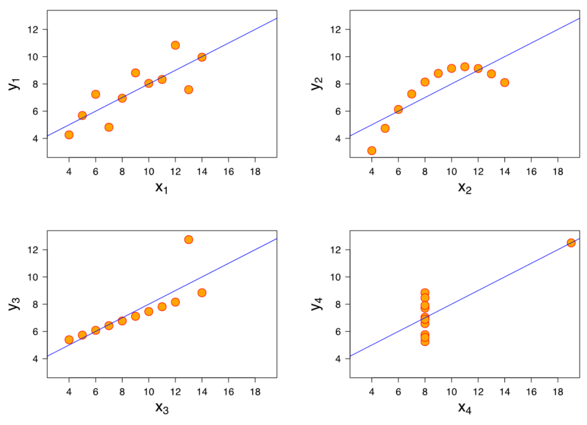


Figure 36: The four dataset have different distribution, even if they have the same statistic values.

over data. It is a measure of how well they are related (Fig. 37). Note that variables can also be non linearly correlated. According to Pearson's coefficient if two variables are independent the coefficient would be zero ($r = 0$) but the viceversa

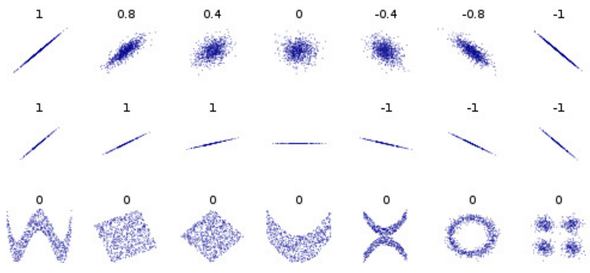


Figure 37: Pearson's correlation coefficient

doesn't hold, which means that a coefficient equal to zero it tells us almost nothing on the independence of the two variables. We have to compute the coefficient for all couple of variables and so we obtain a scatterplot matrix (Fig.38).

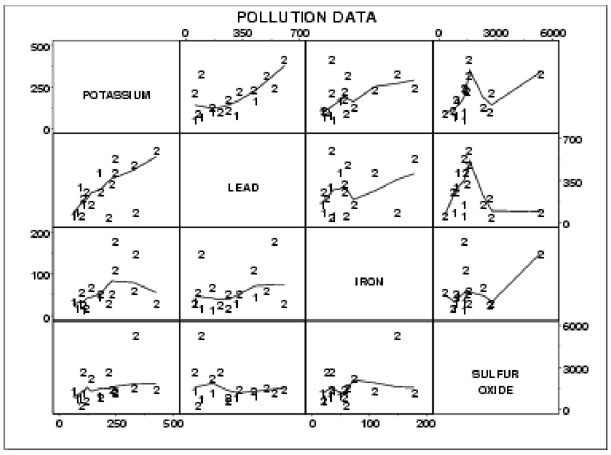


Figure 38: Scatterplot matrix

4 Association Mining

In regione caecorum rex est luscus. Desiderius Erasmus Roterodamus

4.1 Introduction

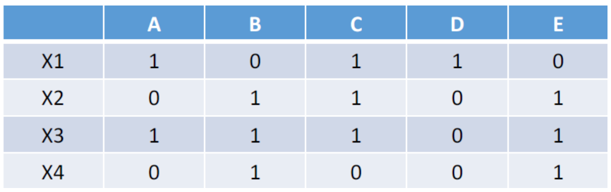
The classical problem of association pattern mining is defined in the context of supermarket data containing sets of items bought by customers, which are referred to as transactions. The goal is to determine associations between groups of items bought by customers, which can intuitively be viewed as k-way correlations between items. The most popular model for association pattern mining uses the frequencies of sets of items as the quantification of the level of association. The discovered sets of items are referred to as large itemsets, frequent itemsets, or frequent patterns. Because the frequent pattern mining problem was originally proposed in the context of market basket data, a significant amount of terminology used to describe both the data (e.g., transactions) and the output (e.g., itemsets) is borrowed from the supermarket analogy. Frequent itemsets can be used to generate association rules of the form X $\Longrightarrow$ Y , where X and Y are sets of items. A famous example of an association rule, which has now become part of the data mining folklore, is Beer $\Longrightarrow$Diapers. This rule suggests that buying beer makes it more likely that diapers will also be bought. Thus, there is a certain directionality to the implication that is quantified as a conditional probability. Association rules are particularly useful for a variety of target market applications.

4.2 Association rules

Consider two sets of items A and B. The confidence of the rule $A \Longrightarrow B$ is defined as the fraction of transactions containing A, which also contain B. In other words, the confidence is obtained by dividing the support of the pattern A$\Longrightarrow$B with the support of pattern A. A combination of support and confidence is used to define association rules. Note that correlation does not imply causality and support is symmetric while confidence is not. Given an association rule $A, B, C \Longrightarrow D$ we says that a customer who bought A,B,C will also by D. Denoting the left side as LHS and the right side as RHS we can define support as $Pr\left(LHS, RHS\right)$ i.e the joint probability of LHS and RHS or in other words the probability of event LHS **and** event RHS occurring. Moreover, we can define the confidence as the $Pr\left(LHS|RHS\right)$ i.e. the conditional probability that is the probability of event RHS occurring, given that event LHS occurs. Definition (Association Rules) Let A and B be two sets of items. The rule $A \implies B$ is said to be valid at support level s and confidence level c, if the following two conditions are satisfied:

1. The support of the item set A is at least s.
2. The confidence of $A \implies B$ is at least c.

Let us now go through an example to understand this basic knowledge: Looking at the table in Fig.39 and analyzing



	A	B	C	D	E
X1	1	0	1	1	0
X2	0	1	1	0	1
X3	1	1	1	0	1
X4	0	1	0	0	1

Figure 39: Association Rules

the rule $A \implies C$ we can state that the support is 50% because A and C both occurs only in transaction x_1 and in x_3 and the confidence is 100% because when A occurs then also C occurs. If we consider $C \implies A$ then the support is the same while the confidence is now 66.7% while there is a case in which C occurs but A does not.

4.3 Apriori algorithm

Apriori is a classic algorithm for learning association rules. Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent itemsets in a database (is said to be frequent an item which have a support greater than ϵ).
2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

Finding all frequent itemsets in a database is difficult since it involves searching all possible itemsets(item combinations). The set of possible itemsets is the power set over I and has size $2^n - 1$. Although the size of the powerset grows exponentially in the number of items n in I, efficient search is possible using the *downward closure property* of support which guarantees that for a frequent itemset, all its subsets are also frequent and thus for a non frequent itemset, all its supersets must also be non frequent. Exploiting this property, efficient algorithms can find all frequent itemsets. Apriori property: all nonempty subsets of a frequent itemset must also be frequent. The Apriori algorithm generates candidates with smaller length k first and counts their supports before generating candidates of length $(k+1)$. The resulting frequent k-itemsets are used to restrict the number of $(k+1)$-candidates with the downward closure property. Candidate generation and support counting of patterns with increasing length is interleaved in Apriori. In other words Apriori uses a “bottom up” approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against data. The algorithm terminates when no further successful extensions are found. Apriori uses breadth-first search and a tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length $k-1$. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates. The overall algorithm is illustrated in Fig. (2). A-priori iteratively finds frequent itemsets with cardinality from 1 to k. It maintains two lists , one is C_k candidate itemsets of size k and the other is L_k candidate itemsets of size k Let’s look at a concrete example. Consider a retailer of breakfast cereals which surveys 5000 students on the activities they engage in the morning. The data shows that 3000 students play basketball, 3750 eat cereal, and 2000 students both play basketball and eat cereal. For a minimum support of 40%, and minimum confidence of 60%, we find the following association rule: play basketball $\implies$ eat cereal. The association rule is misleading because the overall percentage of students eating cereal is 75% which is even larger than 60%. Thus, playing basketball and eating cereals are negatively associated: being involved in one decreases the chances of being involved in the other. Consider the following association: play basketball $\implies$ (not) eat cereal. Although this rule has both lower support and lower confidence than the rule implying positive association, it is far more accurate. Thus, if we set the support and confidence sufficiently low, two contradictory

Algorithm 3 The heart of the algorithm is an iterative loop that generates $(k+1)$-candidates from frequent k-patterns for successively higher values of k and counts them. The three main operations of the algorithm are candidate generation, pruning, and support counting.

```
    procedure Apriori
 2:     $L_1 = \{\text{frequent items}\}$;
        for $(k = 1; L_k \text{not empty}; k++)$ do
 4:         $C_{k+1} = join(L_k, L_k)$
            for $(i = 1; i < n; i++)$ do
 6:             Increment counts of all candidates in $C_{k+1}$ containing $X_i$
                $L_{k+1} = C_{k+1}$with support greater than threshold
 8:     $returnL_1, \cdots, L_{k-1}$
```

rules would be generated; on the other hand if we set the parameters sufficiently high only the inaccurate rule would be generated. In other words, no combination of support and confidence can generate purely the correct association. Introducing the concept of *interest*, the estimation becomes more accurate:

$$Interest = \frac{P(A,B)}{P(A)P(B)} = \frac{P(A|B)}{P(A)} \tag{185}$$

and referring at the previous example we obtain:

$$\frac{P(C|B)}{P(C)} = \frac{\frac{2000}{3000}}{\frac{3750}{5000}} \approx 0.89 \tag{186}$$

$$\frac{P(C|notB)}{P(C)} = \frac{\frac{1750}{2000}}{\frac{3750}{5000}} \approx 1.16 \tag{187}$$

So the joint probability has to be normalized with respect to the events probability. It is wrong to only rely on support index.

4.4 Naive Bayes

Probabilistic classifiers construct a model that quantifies the relationship between the feature variables and the target (class) variable as a probability. There are many ways in which such a modeling can be performed. *Bayes classifier*: The Bayes rule is used to model the probability of each value of the target variable for a given set of feature variables. *Logistic regression*: The target variable is assumed to be drawn from a Bernoulli distribution whose mean is defined by a parametrized logit function on the feature variables. Thus, the probability distribution of the class variable is a parametrized function of the feature variables. This is in contrast to the Bayes model that assumes a specific generative model of the feature distribution of each class. The first type of classifier is referred to as a generative classifier, whereas the second is referred to as a discriminative classifier. In the following, both classifiers will be studied in detail.

4.4.1 Naive Bayes Classifier

The Bayes classifier is based on the Bayes theorem for conditional probabilities. This theorem quantifies the conditional probability of a random variable (class variable), given known observations about the value of another set of random variables (feature variables). For ease in discussion, it will be assumed that all feature variables are categorical. Let C be the random variable representing the class variable $y_i \in [C_1, \cdots, C_k]$ of an unseen test instance $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with d-dimensional feature values $dim(\mathbf{x}_i) = d$. The goal is to estimate the conditional probability $p(c|\mathbf{x}) = p(c|x_1, \cdots, x_d)$. This is difficult to estimate directly from the training data. Then, by using Bayes theorem, the following equivalence can be inferred:

$$p(c|\mathbf{x}) = \frac{p(c)\,p(\mathbf{x}|c)}{p(\mathbf{x})} \propto p(c)\,p(\mathbf{x}|c) \tag{188}$$

The second relationship above is based on the fact that the denominator of the first relationship is independent of the class. Therefore, it suffices to only compute the numerator to determine the class with the maximum conditional probability. The value of $P(C = c)$ is the *prior* probability of the class identifier c and can be estimated as the fraction of the training data points belonging to class c,

$$p(c_i) = \frac{card(c_i) + 1}{n + k} \tag{189}$$

The key usefulness of the Bayes rule is that the terms on the right-hand side can now be effectively approximated from the training data with the use of a naive Bayes approximation. The naive Bayes approximation assumes that the values on the different attributes $x_1 \cdots x_d$ are independent of one another conditional on the class variable. This implies,

$$p(c)\, p(\mathbf{x}|c) = p(c) \prod_{j=1}^{d} p(x_j|c) \tag{190}$$

Note that each term $p(x_j|c)$ is easier to estimate from the training data because enough training examples will exist in the former case to provide a robust estimate. Specifically, the maximum likelihood estimate is,

$$c^* = argmax_i p(c_i) \prod_{j=1}^{d} p(x_j|c_i) \tag{191}$$

4.5 Logistic Regression

While the Bayes classifier assumes a specific form of the feature probability distribution for each class, logistic regression directly models the class-membership probabilities in terms of the feature variables with a discriminative function. In the simplest form of logistic regression, it is assumed that the class variable is binary. Logistic regression can be viewed as either a probabilistic classifier or a linear classifier. In linear classifiers a linear hyperplane is used to separate the two classes.

$$\ln \frac{\pi}{1-\pi} = \mathbf{w}^T\mathbf{x} + b = f(\mathbf{x}) \tag{192}$$

The left side is know as *logit*, while the right side represents linear discriminating function. Moreover the *sigmoid* is defined as

$$\pi = \frac{1}{1+e^{-f(\mathbf{x})}} \tag{193}$$

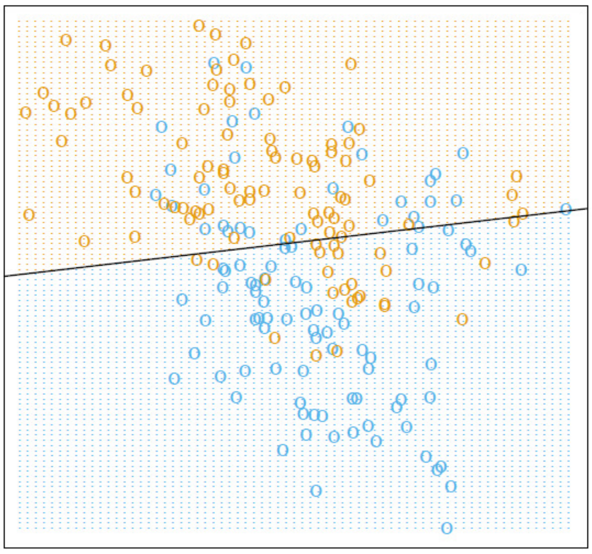


Figure 40: A classification example in two dimensions. The classes are coded as a binary variable.

5 Decision Tree

In order to be an immaculate member of a flock of sheep, one must, above all, be a sheep oneself. Albert Einstein

5.1 Introduction

Algorithms we used until now can be seen as a black box model, in which given an input **x** we obtain **y** as output, knowing nothing about the model. If it is true that we can compute the prediction error, we have to take into account that in same cases it is not enough. If we think about a model which have to predict train delays, doesn't matter the reason of the resulting prediction error, but if we consider a medical case, the reason of the error is extremely relevant. This is the main reason for the slow adoption of machine learning system in medicine. Doctors prefers a least accurate model that explain why it reaches that solution, instead of a black box model even if it is much more accurate. As engineers we have to find an interpretable model. Trees. At each step we know exactly why we have choose one path and why we reach one particular leaf. We have also to choose the right sequence of splitting variables (features in our case) and when to cut the tree to not overfit data. In general, with trees is possible to fit any dataset, making the tree deeper. But in this way we are overfitting data and so we have to regularize by cutting the tree at one level. In order to construct the tree we have to choose at each node which feature to test. To make this decision we need to recall the concept of information gain, which is itself a measure of entropy.

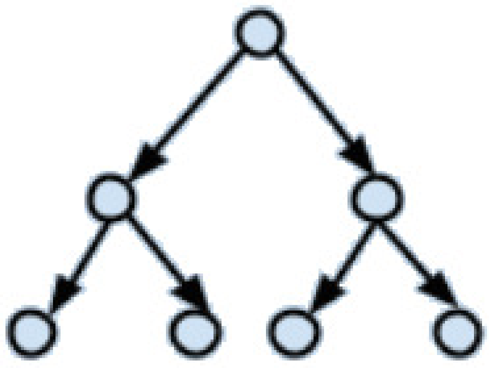


Figure 41: Decision tree.

5.2 Entropy

In the general case, we have the follow definition: *Given an arbitrary categorization, C into categories $c_1, \cdots, c-n$, and a set of examples, S, for which the proportion of examples in c_i is p_i, then the entropy of S is*:

$$\mathit{Entropy}(\mathrm{S}) = -p_i \ln_2(p_i) \tag{194}$$

To understand this concept let us make a step back in the past and show why bit is the minimum quantum of information needed to maximize the information itself. Imagine having a box containing two balls, white and black. So we have “p” as the probability to pick up the black ball, and “1-p” to pick up the white one. According to the entropy definition we gave above , in this case the entropy is,

$$H = -p \ln_2(p) - (1-p)\, p \ln_2(1-p) \tag{195}$$

Computing the first and second derivative , we obtain the following plot, in which is evident that for $p = \frac{1}{2}$ we have the maximum value of entropy. So, we can state that entropy is a measure of information. Entropy is what we need, because of the $-p \ln_2(p)$ construction: when p gets close to zero, then the entropy is nearly zero. Remembering that entropy measures the disorder of data, this result is good, as it reflects our need to reward categories with few classes in. Similarly, if p gets close to 1 , then the $\ln_2(p)$ part gets very close to zero, so the overall value gets close to zero. Hence we see that both when the category is nearly or empty at all, the score for the category gets close to zero, which is what we expected.

5.3 Information Gain

If we pass to a configuration (A) in which we have two black balls and two white balls in a box, so with a certain level of entropy, to another one in which we have a box with only white box (B1) and a box with only black balls (B2) the entropy goes to zero. We call this entropy gain. The general definition is, The information gain of attribute A, relative to a collection of examples, S, is calculated as:

$$Gain\left(\mathrm{S},\mathrm{A}\right) = Entropy\left(\mathrm{S}\right) - \sum_{v \in Values(A)} \frac{|S_v|}{\|S\|} Entropy\left(\mathrm{S}_v\right) \tag{196}$$

While in our example,

$$G = H\left(\mathrm{A}\right) - \frac{n_{B1}}{n_A} H\left(\mathrm{B1}\right) - \frac{n_{B2}}{n_A} H\left(\mathrm{B2}\right) \tag{197}$$

Then, we have to choose features and splitting points which maximize the information gain. To choose the feature we have to use brute force and trying them out. Finding the bounds, instead , cost in the worst case n^2 , while in the best $n \ln(n)$, because we have to sort data.

5.4 Random forest

Random forests are defined as an ensemble of decision trees, in which randomness has explicitly been inserted into the model building process of each decision tree.In the random forest approach, deep trees are grown without pruning. Each tree is grown on a bootstrapped sample of the training data to increase the variance reduction.

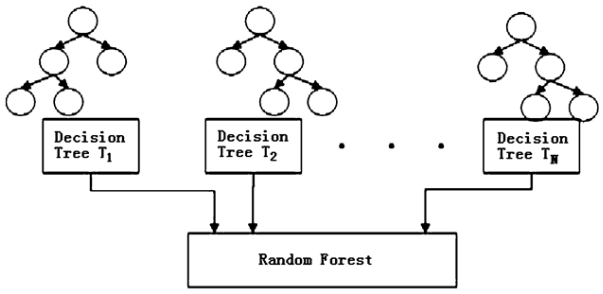


Figure 42: Random Forest.

5.4.1 Bootstrapped sampling approach

bootstrapping, is an approach that attempts to reduce the variance of the prediction. It is based on the idea that if the variance of a prediction is σ^2, then the variance of the average of k independent and identically distributed predictions is reduced to $\frac{\sigma^2}{k}$. Given sufficiently independent predictors, such an approach of averaging will reduce the variance significantly. Data points are sampled uniformly from the original data with replacement. This sampling approach is referred to as bootstrapping and is also used for model evaluation. A sample of approximately the same size as the original data is drawn. This sample may contain duplicates, and typically contains approximately $1-(1-1/n)n \approx 1-1/e$ fraction of the distinct data points in the original data.Here, e represents the base of the natural logarithm.This result is easy to show because the probability of a data point not being included in the sample is given by $(1-1/n)n$.A total of k different bootstrapped samples, each of size n, are drawn independently, and the classifier is trained on each of them. For a given test instance, the predicted class label is reported by the ensemble as the dominant vote of the different classifiers.

6 K-means

Never categorize yourself, society does that to you, don't do it to yourself. Jason Priestley

6.1 Introduction

Clustering is a method of unsupervised learning, and a common technique for statistical data analysis. Many applications require the partitioning of data points into intuitively similar groups. The partitioning of a large number of data points into a smaller number of groups helps greatly in summarizing the data and understanding it for a variety of data mining applications. An informal and intuitive definition of clustering is as follows: *Given a set of data points, partition them into groups containing very similar data points.* This is a very rough and intuitive definition because it does not state much about the different ways in which the problem can be formulated, such as the number of groups, or the objective criteria for similarity. A wide variety of models have been developed for cluster analysis. These different models may work better in different scenarios and data types. A problem, which is encountered by many clustering algorithms, is that many features may be noisy or uninformative for cluster analysis. Such features need to be removed from the analysis early in the clustering process. This problem is referred to as feature selection. The key models differ primarily in terms of how similarity is defined within the groups of data. In some cases, similarity is defined explicitly with an appropriate distance measure, whereas in other cases, it is defined implicitly with a probabilistic mixture model or a density-based model. Fundamental to all clustering techniques is the choice of distance or dissimilarity measure between two objects. We first discuss distance measures before describing algorithms for clustering.

6.2 Distance measure

The first problem to deal with is to determine how to compute the similarity of two samples. There are many distance functions. The general formulation is given by p-norm ,

$$\|\underline{x}\|_p^p = \sqrt[p]{|x_1|^p + \cdots + |x_d|^p} \tag{198}$$

From the above equation follows different norm functions.

- If $p = 0$ it means we are considering values in which only one feature is not null (blue star in the figure 43).
- For $p = 1$ We have the Manhattan distance (1-norm) (taxicab geometry) computed as $d\left(\underline{p},\underline{q}\right) = \left\|\underline{p} - \underline{q}\right\|_1 = \sum |p_i - q_i|$ where $\underline{p}, \underline{q}$ are vectors. It means we are considering directions in which at least one feature is null.
- For $p = 2$ we obtain the Euclidean distance (also called 2-norm distance).
- For $p = \infty$ we have the maximum norm (infinity norm).

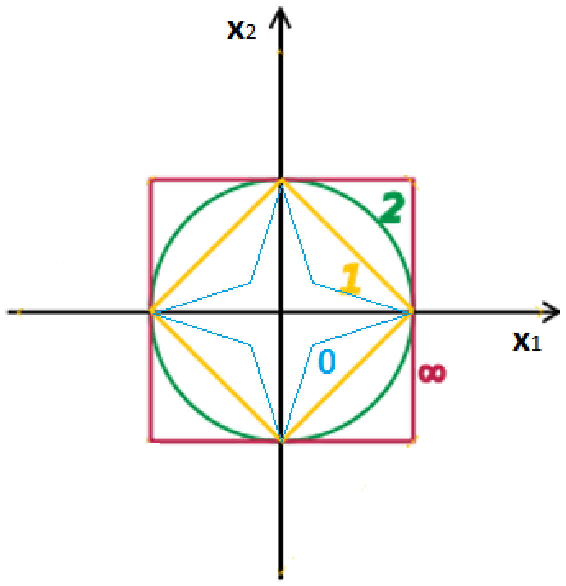


Figure 43: Norm: Infinity norm in red. 2-norm in green. 1-norm in yellow. 0-norm in blue.

Note that selecting the norm function we are also selecting the relevant features. But the feature selection depends itself on the distance measure we have adopted. So in non super-visioned learning problem we can't understand what model is good from data, but the choice is lead from the knowledge of the problem itself. For example, if we have to cluster city pubs then we will use Manhattan norm, if we have to cluster people according to their height we will use Euclidean norm, and so on... In non supervised learning then the knowledge of the problem is crucial. Assuming to have chosen the best function, the best cluster configuration will be the one which minimizes the cost function.

6.3 K-means

The first clustering algorithm was introduced by IBM. It starts from a random point and then it goes to the nearest point until it reaches all points. The second step is to segment the path, deleting the longest edge. The problem of this algorithm is that is not robust to the noise. Introducing some hypothesis we obtain algorithms, like k-means, that don't fit the noise. K-means algorithm is not a clustering method, but it is a vector quantization algorithm. The strong assumption is that groups are separable by linear classifiers. Then points can be clustered within convex margin and this is why we said it is a vector quantization method: each of k points represent one group, like in fig. 44 red point quantizes all other points of the group. Another possible interpretation is given by Jungian archetypes. Carl Jung

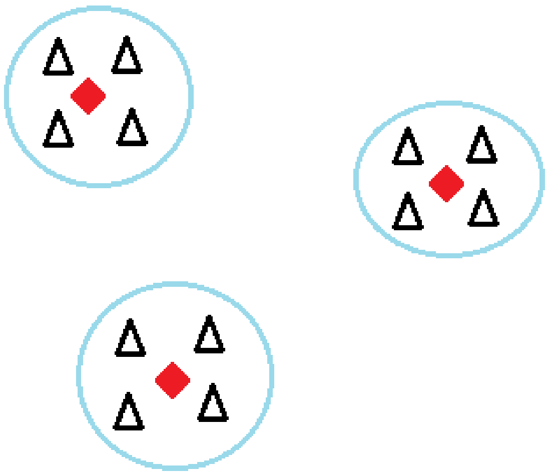


Figure 44: Data quantization.

understood archetypes as universal, archaic patterns and images that derive from the collective unconscious and are the psychic counterpart of instinct. It is a sort of categorization at high level. In our case the archetype is not strictly a real point of the dataset, but it could be a new point representing the others. To sum up, to apply k-means algorithm we have to make two assumptions: to know k and to have a convex space. The main problem is how we decide k. Once k is defined we have to find k points minimizing the distance of each point from its centroids. So we have to do two minimization. The second problem is that this cost function is not convex. So in this sense, this algorithm is heuristic, since it would reach different local minimum. This is why, typically, we have to run it several times, obtaining different solutions, due to the fact that the starting point is random. This method is known in literature as multi-start. One possible way to obtain a solvable problem is to use the *divide and conquer* technique. First we set the centroids and then, keeping them fixed, we look for the best clusters. We can iterate this two steps until the centroids don't change anymore and clusters are well defined. It can be shown that this method converge to a local minimum. Let's see in more details how k-means algorithm works. The k-means algorithm assigns each point to the cluster whose center (also called centroid) is nearest. The algorithm steps are the following:

1. Set the number of clusters, k.
2. Randomly generate k clusters and determine the cluster centers, or directly generate k random points as cluster centers.
3. Assign each point to the nearest cluster center, where "nearest" is defined with respect to one of the distance measures discussed above.
4. Recompute the new cluster centers.
5. Repeat step 3 and step 4 until some convergence criterion is met (usually that the assignment hasn't changed).

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignment. Another disadvantage is that we don't know k and this problem is not yet have a solution. The correct choice of k is often ambiguous, with interpretations depending on the shape and scale of the distribution of points in a data set and the desired clustering resolution of the user. In addition, increasing k without penalty will always reduce the amount of error in the resulting clustering, to the extreme case of zero error if each data point is considered its own cluster (i.e., when k equals the number of data points, n). Intuitively then, the optimal choice of k will strike a balance between maximum compression of data using a single cluster, and maximum accuracy by assigning each data point to its own cluster. If an appropriate value of k is not apparent from prior knowledge of the properties of the data set, it must be chosen somehow. We can think to choose the number of clusters by visually inspecting the data points, but we will soon realize that there is a lot of ambiguity in this process for all except the simplest cases. This is not always bad, because we are doing unsupervised learning and there's some inherent subjectivity in the labeling process. Here, having previous experience with that particular problem or something similar will help us to choose the right value. If we want some hint about the number of clusters we should use, we can apply the so called **Elbow method**: one should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the intraclass distance against the number of clusters, the first clusters will add much information (distance reduction), but at some point the marginal gain will drop, giving an angle in the graph. The right k is in that point. Take into account that the Elbow method is an heuristic and, as such, it may or may not work well in some particular case. Sometimes, there are more than one elbow, or no elbow at all. In those situations we usually end up calculating the best k by evaluating how well k-means performs in the context of the particular clustering problem we are trying to solve.

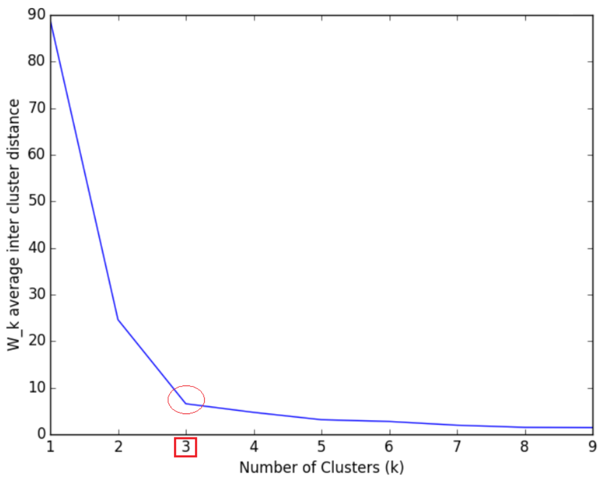


Figure 45: Elbow method.

6.4 Kernel K-means

The k-means algorithm can be extended to discovering clusters of arbitrary shape with the use of a method known as the kernel trick. The basic idea is to implicitly transform data so that arbitrarily shaped clusters map to Euclidean clusters in the new space. The main problem with the kernel k-means algorithm is that the complexity of computing the kernel matrix alone is quadratically related to the number of data points. Let's put this concept in a more rigorous formulation. Let $X = \{x_1, \cdots, x_n\}$ the data and $k \leq n$ the number of clusters. Then we denote: I_j, $j \in 1, \cdots, k$ the set of indexes that belong to the class j. Then the center of the different clusters are: $c_j = X_i \in I_j x_i |I_j|$, $j \in \{1, \cdots, k\}$ where $|I_j|$ is the cardinality (number of element). The last thing that we have to compute is the distance between a new point x and the center c_j : $D(c_j, x) = \|c_j - x\|^2$. Note that this algorithm do not solve the problems we have in the previous case, but on the contrary we are increasing the complexity since we have more local minimums than before.

6.5 Spectral clustering

Traditional clustering methods like K-means use a spherical or elliptical metric to group data points. Hence they will not work well when the clusters are non-convex, such as the concentric circles. Spectral clustering is a generalization of standard clustering methods, and is designed for these situations. The starting point is a $N \times N$ matrix of pairwise similarities between all observation pairs. We represent the observations in an undirected similarity graph $G =< V, E >$. The N vertices v represent the observations, and pairs of vertices are connected by an edge if their similarity is positive (or exceeds some threshold). Clustering is now rephrased as a graph-partition problem, where we identify connected components with clusters. We wish to partition the graph, such that edges between different groups have low weight, and within a group have high weight. The idea in spectral clustering is to construct similarity graphs that represent the local neighborhood relationships between observations. Instead of working in the time domain, we will work in the frequency domain. Here one uses the top eigenvectors of a matrix derived from the distance between points. We analyze the algorithm that uses the k eigenvectors simultaneously and gives condition in witch algorithm can be expected to do well. Given a set of points $S = s_1, \cdots, s_n \in \Re^d$ that we want to cluster into k subset:

1. Form the affinity matrix $A \in \Re^{n\times n}$ defined by:

$$A_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \quad \text{if} \quad i \neq j \tag{199}$$

$$A_{ij} = 0 \quad \text{if} \quad i == j \tag{200}$$

2. Define D to be the diagonal matrix whose (i,i)-element is the sum of the A's i-th rows and construct the matrix $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
3. Find the k $(x_1, \cdots, x_k)$ largest eigenvector of L (chosen to be orthogonal in case of repeated eigenvalues) and form the matrix $X = [x_1, \cdots, x_k] \in \Re^{n\times k}$ by stacking the eigenvector in columns.
4. Form the matrix Y from X be re-normalizing each of X-s row to have unit length:

$$Y_{ij} = X_{ij} / \left(\sum_j X_{ij}^2\right)^{\frac{1}{2}} \tag{201}$$

5. Treating each row of Y ad a point in $\Re^k$, cluster them with k-means or any other algorithm.
6. Finally assign the original point to the cluster j if and only if row i of the matrix Y was assigned to cluster j.

At first sight, this algorithm seems to make little sense. Since we run k-means why we do not just apply k-means directly on data? Look at the Figure 46 . The natural cluster in $\Re^2$ do not correspond to convex region and then k-means find the unsatisfactory clustering of Figure 47. But when we map the point to $\Re^k$ they form a tight cluster of Figure 48 for which our methods obtains the good clustering of Figure 46.

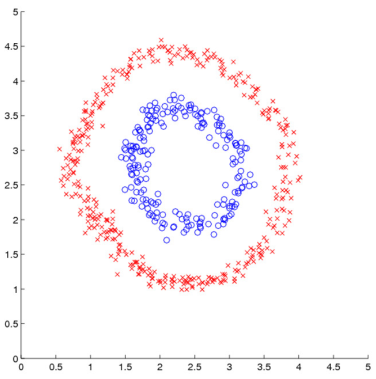


Figure 46: Non convex region

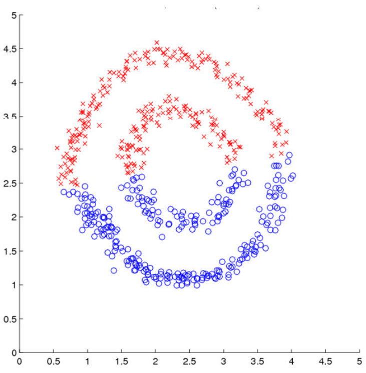


Figure 47: k-means clustering

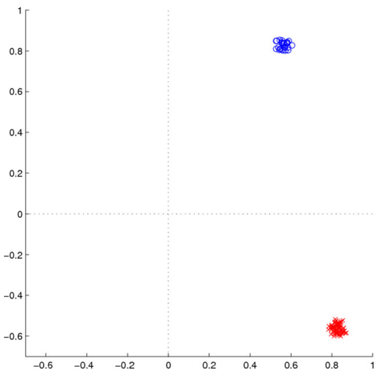


Figure 48: Point mapped in $\Re^k$

References

[1] C. D. Aggarwal. *Data Mining - The textbook.* Springer, 2015.

[2] M. R. Berthold, C. Borgelt, F. Hoppner, and F. Klawonn. *Guide to Intelligent Data Analysis.* Springer, 2010.

[3] C. K. Enders. *Applied Missing Data Analysis.* The Guilford Press, 2010.

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009.

[5] N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A Classification Perspective.* Cambridge University Pres, 2011.

[6] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. 1995.

[7] NIST. *Engineering Statistics Handbook.* NIST, 2006.

[8] L. Oneto. Machine learning algorithms for data mining. 2016.