# Navigation

# First Application of Data Analytics

## Dr. John Snow's Map of the 1854 London Cholera Outbreak

Source: http://www.udel.edu/johnmack/frec480/cholera

# Use Cases of Data Analytics - Examples

# Fundamental Skills of Data Analytics

# The Data Analytics Process

Source : http://www.datasciencecentral.com/profiles/blogs/data-science-simplified-principles-and-process

# Need for Knowledge about the Algorithms

The distance between using Excel and VBA for modeling in credit scoring, for example, and using machine learning algorithms and R or Python to enhance the results, is not that great, compared to the distance between someone running a packaged algorithm they don’t really understand and someone who understands the mathematical and statistical operations within an algorithm and can optimize or adapt it as needed – and do so in the

# Statistics vs. Machine Learning

**Statistics** about finding valid conclusions about the underlying applied theory, and on the interpretation of parameters in their models. It insists on proper and rigorous methodology, and is comfortable with making and noting assumptions. It cares about how the data was collected and the resulting properties of the estimator or experiment (e.g. p-value). The focus is on hypothesis testing.

**Machine Learning (ML)** aims to derive practice-relevant

# Statistical Regression vs. Machine Learning Algorithms

# Explanation vs. Prediction (I)

Question 1: I have a headache. If I take an aspirin now, will it go away?

Question 2: I had a headache, but it passed. Was it because I took an aspirin two hours ago? Had I not taken such an aspirin, would I still have a headache?

# Explanation vs. Prediction (II)

## Explanation :

- Explanation is about understanding relationships and why certain things happen.
- It requires an understanding of cause and effect.
- Tests of causal hypotheses are fundamental.
- Measures of significance are central.
- A good explanatory model may also have predictive power.

## Prediction :

# Correlation

Source: Organic Trade Association, 2011 Organic lndustry Survey, U.p. Department of Education, Office of Special Education Programs, Data Analysis System (DANS)

Organic food sales and the rate of autism seem to have a very strong correlation, but no one is suggesting that one causes the other!

# Correlation vs. Causality (I)

**Correlation: Two data series behave “similar”**

**Causality: Principle of Cause and Effect**

# Correlation vs. Causality (II)

# Correlation vs. Causality (III)

**But:**

**Sometimes it is better to know/predict something even if we**

# Statistical Estimation

Source: http://www.dxbydt.com/the-size-of-your-sample

# Navigation

# Definitions

A **method** is a composition of formalized principles that form the basis for a stringent calculation process.

An **algorithm** is a procedure or set of steps or rules to accomplish a task. It is usually the implementation of a method. Algorithms are used to build models.

In the given context, a **model** is the description of the relationship between variables. It is used to create output data from given input data, for example to make predictions.

# Traditional Analytics Process

# Example Regression - Fitting the model

# Example Regression - Testing the model

# Data Errors and their Consequences

# Modern Analytics Process

# Best Fit vs. Best Generalization

# Over- and Underfitting

Due to the problem of overfitting, the main goal is to maximize the prediction quality and not to fit the data that is used for the model estimation as well as possible. This is equivalent to minimizing the risk that the model will have weak predictive ability.

# The Bias-Variance Tradeoff

The prediction error is influenced by three components:

$$\text{Error} = \text{Bias} + \text{Variance} + \text{Noise}$$

Bias is the inability of the used method to learn the relevant relations between the inputs and the outputs. It reflects the method quality, e.g. if a method only produces linear models.

Variance is represents the deviation

# Summarizing: Statistics vs. Data Analytics

# Which Method should I choose?

The choice of the method of data analysis depends on the one hand on the scope of application, but on the other hand on the interrelationships of the data to be analyzed.

In the Big Data area, data spaces are often highly-dimensional, making it difficult to visualize the interrelationships.

For this reason, the choice of the method can often not be made ex ante. In these cases, different methods are

# Linear World

# Quadratic World

# Nonlinear World (Type 1)

# Nonlinear World (Type 2)

# Nonlinear World (Type 3)

# Nonlinear World (Type 4)

# The Data Analytics Process - Technical View

Source: http://blogs.msdn.microsoft.com/martinkearn/2016/03/01/machine-learning-is-for-muggles-too/

# Data Preparation and Enrichment

**The data collection and preparation phase is the most labor-intensive one, consuming on average between 60-80% of a data scientist's time. It's critical therefore to select a tool that can automate or at least speed the workflows associated with data preparation.**

# Data Cleaning

## 1. Proof of correctness of the data

- examine for irregular outliers (e.g. Age=236)
- examine for typographical errors (e.g. Frankfrut)
- examine for different writing styles (e.g. behavior/behaviour)
- —

## 2. Handling missing values

# Missing Values Strategies

# Sampling

A population can be defined as including all people or items with the characteristic one wishes to understand.

Sampling is about to find a representative subset of that population.

Data represents the traces of the real-world processes, and exactly which traces we gather are decided by our sampling method.

There are two sources of randomness and uncertainty:

1. the randomness and uncertainty underlying the process

# Sampling in Times of Big Data

**Question:**

**Is there any need for sampling in times of Big Data?**

**Why not “N=ALL”?**

**Answer:**

**Data is not objective! Data does not speak for itself.**

**Data is just a quantitative echo of the events of our society.**

**Examples:**

# Reasons for Sampling

- The volume of data is too large to capture and process
- Design the analytics process using a subset of the data for performance reasons. Later use the complete data set.
- The data set doesn’t perfectly represent the target population.
- The data set is imbalanced.

# Popular Methods of Sampling (I)

# Popular Methods of Sampling (II)

# Systematic Sampling

# Random Sampling

# Proportional Sampling

# Downsampling

# SMOTE

SMOTE (Synthetic Minority Oversampling Technique) is an oversampling technique where the synthetic samples are generated for the minority class.

At first the total number of oversampling observations N is set up. Usually, it is selected such that the resulting class distribution is 1:1. Now, the iteration starts by

# Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process.

A feature (variable, attribute) is depicted by a column in a dataset. Considering a generic two-dimensional dataset, each observation is depicted by a row and each feature by a column, which will have a specific value for an observation:

# Variants of Feature Engineering

## 1. Transformation

- convert features (e.g., birth date → age)
- build lag structures (e.g., time-lags)
- normalization / standardization / scaling

## 2. Type Conversion

- if numerical type is needed, transform categorical into numerical data using dummy features
- if categorical type is needed or more informative, discretize numerical features (e.g., income → poor / rich classes)

## 3. Feature Combination

# Scaling

Most datasets contain features highly varying in magnitudes, units and range.

Most machine learning algorithms have problems with this because they use distance measures or calculate gradients. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes and gradients may end up taking a long time or are not accurately calculable.

To overcome this effect, we scale the features to bring them to the same level of magnitudes. The two most discussed scaling methods are Normalization and Standardization.

# Type Conversion (Encoding)

Many machine learning algorithms cannot work with categorical data directly. To convert categorical data to numbers, there exist two variants:

Label encoding refers to transforming the word labels into numerical form so that the algorithms can understand how to operate on them. Every categorical value is assigned to one numerical value, e.g. young -> 1, middle_age -> 2, old -> 3. This only works in specific situations where you have somewhat continuous-like data, e.g. if the categorical feature is ordinal.

One hot encoding is a representation of a categorical variable as binary vectors. Every categorical value is assigned to an artificial

# Example of Feature Engineering (I)

Data sets often contain date/time features. These features are rarely useful in their original form because they only contain ongoing values. However, they can be useful for extracting cyclical factors, such as weekly or seasonal effects. Suppose, we are given a data “flight date time vs status”. Then, given the date-time data, we have to predict the status of the flight.

But the status of the flight may depend on the hour of the day, not on the date-time. To analyze this, we will create the new feature ” Hour_Of_Day”. Using the “Hour_Of_Day” feature, the machine will learn better as this feature is directly related to the status of the

# Example of Feature Engineering (II)

Suppose we are given the latitude, longitude and other data with the objective to predict the target feature “ Price_Of_House “. Latitude and longitude are not of use in this context if they are alone. So, we will combine the latitude and the longitude to make one feature.

In other cases, it might be appropriate to transform latitude and longitude into categories which reflect regions, for example

# Example of Feature Engineering (III)

Suppose we are given a feature “ Marital_Status ” and other data with the objective to classify customers into “Creditworthy” and “ Not_Creditworthy “. In the data set the martial status has many different values, for example

- single living alone
- single living with his parents
- married living together
- married living separately
- divorced
- divorced but living together
- registered partnerships
- living in marriage-like community

# Partitioning the Data

The partitioning of the data in Training and Test Data has the aim to proof if the analytical results can be generalized. The analysis (e.g. the development of a classifier) is carried out on the basis of training data. Subsequently, the results are applied to the test data. If the results are significantly worse than the training data, the model is not generalizable, which is called overfitting.

The partitioning of the data in training and test data can be carried out in the following ways:

# Applying Training and Test Data

Source: http://www.cs.kent.edu/~jin/BigData/Lecture10-ML-Classification.pptx

# Partitioning

# Exploratory Data Analysis

In Exploratory Data Analysis (EDA), there is no hypothesis and there is no model.

People are not very good at looking at a column of numbers or a whole data table and then determining important characteristics of the data. EDA techniques have been devised as an aid in this situation.

Reasons for EDA:

- gain intuition about the data
- make comparisons between distributions
- sanity checking (making sure the data is on the scale you

# Univariate Non-Graphical EDA

Non-graphical exploratory data analysis is the first step when beginning to analyze the data. This preliminary data analysis step focuses on four points:

- measures of central tendency, i.e. mean and median. The median, known as 50th percentile, is more resistant to outliers.
- measures of spread, i.e. variance, standard deviation, and interquartile range
- the shape of the distribution
- the existence of outliers

# Tests on Outliers

Outlier are data objects, which are clearly different from the others.

Usually, the detection of outliers is an unsupervised process, because they are not known before analyses.

In the case of numerical attributes the Interquartil Range can be used. Here, an outlier is defined if the attribute lies outside the interval

Usually, k has a value between 1.5 and 3. The bigger k, the more different the values must be to be classified as outliers.

# Handling Outliers

- Outlier have to be eliminated if they
  - 1. would bias the analysis, e.g. if 9 persons have an age between 20 and 30 and the 10th person is 80 years old.
  - 2. are erogenous data, e.g. as a result of input errors or a defect sensor.
- It is not always acceptable to drop an observation just because it is an outlier. They can be legitimate observations and are sometimes interesting ones. It’s important to investigate the nature of the outlier before deciding.
- In those cases where you shouldn’t drop the outlier, one option

# Univariate Graphical EDA

Non-graphical and graphical EDA methods complement each other, they have the same focus. While the non-graphical methods are quantitative and objective, they do not give a full picture of the data. The distribution of a variable tells us what values the variable takes and how often each value occurs.

Types of displays:

for numerical variables: Histograms, Boxplots, Quantile-normal plots, …

for categorical variables: Pie charts, Bar graphs, …



# Multivariate Non-Graphical EDA

**Multivariate non-graphical EDA techniques generally show the relationship between two or more variables in the form of either cross-tabulation for categorical variables or correlation statistics for numerical variables.**

# Multivariate Graphical EDA

**Multivariate graphical EDA techniques are scatterplots for numerical variables, Barcharts for categorical variables, or Boxplots for mixed types.**

# Touring Diagram

# Categories in Machine Learning

# Supervised Learning

# Unsupervised Learning

# Supervised and Unsupervised Learning

# Use Cases Quiz

# Reinforcement Learning

The solution to many of the problems in our lives cannot be automated. This is not because current computers are too slow, but simply because it is too difficult for humans to determine what the program should do.

Supervised learning is a general method for training an approximator. However, supervised learning requires sample input-output pairs from the domain to be learned.

For example, we might not know the best way to program a computer to recognize an infrared picture of a tank, but we do have a large collection of infrared pictures, and we do know

# Reinforcement Learning

**The agent learns how to achieve a given goal by trial-and-error interactions with its environment by maximizing a reward.**

# AlphaGo

Go is one of the hardest games in the world for AI because of the huge number of different game scenarios and moves. The number of potential legal board positions is greater than the number of atoms in the universe.

The core of AlphaGo is a deep neural network. It was initially trained to learn playing by using a database of around 30 million recorded historical moves

# Libratus

An artificial intelligence called Libratus has beaten four of the world's best poker players in a grueling 20-day tournament in January 2017.

Poker is more difficult because it's a game with imperfect information. With chess and Go, each player can see the entire board, but with poker, players don't get to see each other's hands. Furthermore, the AI is required to bluff and correctly interpret misleading information in order to win.

"We didn't tell Libratus how to play poker. We gave it the rules of poker and said 'learn on your own'." The AI started playing randomly but over the course of playing trillions of hands was able

# Types of Artificial Intelligence

**Discriminative AI** is designed to differentiate and classify input, but not to create new content. Examples include image or speech recognition, credit scoring or stock price prediction.

**Generative AI** is able to generate new content based on existing information and user specifications. This includes texts, images, videos, program code, etc. The generated content can often hardly be distinguished from human-generated content. As things stand at

# ChatGPT

ChatGPT is a generative AI that produces human-like text and communicates with humans.

The “GPT” in ChatGPT comes from the language model of the same name, which was extended for ChatGPT with various components for communication and quality assurance.

GPT is based on a huge neural network that essentially represents the language model. While

# ChatGPT - Approach

**ChatGPT generates its response word by word via a sequence of probabilities, with each new word depending on the previous ones.**

**The most probable word is not always selected; instead, randomization takes place. This means that different variants can be created for the same task.**

# ChatGPT - Semantic Spaces (I)

# ChatGPT - Semantic Spaces (II)

# ChatGPT - Evaluation Component

# Navigation

# Introductory Example

**Credit-Scoring is a typical example for a classification problem. A bank wants to determine the creditworthiness of a customer.**

**Assume you have the age, income, and a creditworthiness category of “yes” or “no” for a bunch of people and you want to use the age and income to predict the creditworthiness for a new person.**

**You can plot people as points on the plane and label people with an empty circle if they have low credit ratings.**

# 4 Predictive Analytics

## 4.1 Subject of Predictive Analytics

## 4.2 The Analytics Process

## 4.3 Data Preparation

## 4.4 Methods, Algorithms and Applications

### 4.4.1 Classification

#### 4.4.1.1 K-Nearest Neighbors

#### 4.4.1.2 Evaluating the Quality of Classification

#### 4.4.1.3 Decision Tree Approaches

#### 4.4.1.4 Logistic Regression

#### 4.4.1.5 Neural Networks

# k-Nearest Neighbors

- k-Nearest Neighbors (k-NN) is an algorithm that can be used when you have a bunch of objects that have been classified or labeled in some way, and other similar objects that have not gotten classified or labeled yet, and you want a way to automatically label them.
- The intuition behind k-NN is to consider the most similar other items defined in terms of their attributes, look at their labels, and give the unassigned item the majority vote. If there’s a tie, you randomly select

# Measuring Similarity

# Unnormalized vs. Normalized

# Example (I)

# Example (II)

# Example (III)

## 3. Choose the k nearest neighbors

| Customer | Age | Monthly Income | Monthly Costs | Creditworthy | Distance |
|---|---|---|---|---|---|
| A | 0.0000 | 0.0303 | 0.0400 | yes | 0.4347 |
| C | 0.1714 | 0.3333 | 0.3600 | yes | 0.1726 |
| E | 0.3143 | 0.1818 | 0.2000 | no | 0.2010 |
| F | 0.4286 | 0.3939 | 0.6000 | no | 0.4482 |
| G | 0.4857 | 0.2121 | 0.1200 | yes | 0.3090 |
| X | 0.2286 | 0.3636 | 0.2000 | ? | |

# Creation and Use of Models

# Calculating Accuracies

# Determining Parameter k

1. Split the original labeled dataset into training and test data.
2. Pick an evaluation metric. Misclassification rate or accuracy are good ones.
3. Run k-NN a few times, changing k and checking the evaluation measure.
4. Optimize k by picking the one with the best evaluation measure.

k Accuracy

# Navigation

# Evaluating the quality of Classification (I)

True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), are the four different possible outcomes of a single prediction for a two-class case. A false positive is when the outcome is incorrectly classified as “yes”, when it is in fact “no”. A false negative is when the outcome is incorrectly classified as negative, when it is in fact positive. True positives and true negatives are obviously correct classifications.

# Evaluating the quality of Classification (II)

Test metrics are used to assess how accurately the model predicts the known values:

Most classification algorithms pursue to minimize the misclassification rate. They implicitly assume that all misclassification errors cost equally. In many real-world applications, this assumption is not true. Cost-sensitive learning takes costs, such as the misclassification cost, into consideration. Using costs, the error rate can be calculated via:

# Evaluating the quality of Classification (III)

Misclassification rate and accuracy can be misleading, for example in the case of imbalanced samples. Extreme case:

For problems like, this additional measures are required to evaluate a classifier.

Sensitivity (true positive rate, recall) measures the proportion of positives that are correctly identified as such. Specificity (true negative rate) measures the proportion of negatives that are correctly identified as such.

Using both measures, we can compute the Balanced Accuracy

# Problem of Imbalancing and Accuracy

Assume the following case: A credit card company wants to create a fraud detection system to include it into their transactional systems. The outcomes should be “Accept” (Y) and “Reject” (N). Because fraud rarely occurs, the data set consists of 320 observations for Y and 139 for N. They are partitioned into training and test set. Finally, the model is trained and tested.

Because of the majority of the Y class, the training process concentrates on these cases because their correct classification promises the highest accuracy

# Evaluating the quality of Classification (IV)

**Precision** measures the proportion of predicted positives who are true positives. A precision of 0.5 means that whenever the model classifies a positive, there is a 50% chance of it really being a positive.The higher the precision the smaller the number of false positives.

**Recall** measures the percentage of positives the model is able to catch. It is defined as the number of true

# Evaluating the quality of Classification (V)

The F1 Score can be interpreted as the weighted average of both precision and recall. The main idea of the F1 Score is to strike a balance between both precision and recall and measure it in a single metric.

A F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

It is commonly used in cases of high class imbalance.

# Creation and Use of Models

# Navigation

# Which one is better?

# Introductory Example

# Decision Trees (I)

Decision trees belong to the hierarchical methods of classification. They analyze step-by-step (recursive partitioning).

A decision tree consists of nodes and borders. The topmost node (without any parent node) is called “root”. A node without a child node is called “leaf”. Nodes that have parent and child nodes are called “interior nodes”. The interior nodes represent the splitting of the included object sets. An interior node has at least two child nodes (sons). If every interior node has exactly two child nodes

# Decision Trees (II)

Graphically, decision tree models divide the dataspace in a large number of subspaces and search for the variables which are able to split the dataspace with the greatest homogeneity. We can think of the decision tree as a map of different path. For a distinct combination of predictor variables and their observed values, we would enter a specific path, which gives the classification in the leaf of the decision tree.

The decision tree approach
does not require any

# Decision Trees (III)

Source: http://iopscience.iop.org/article/10.1088/1749-4699/5/1/015004

# Overview of important Decision Tree Methods

| Name | CART | ID3 | C5.0 | CHAID | Random Forests |
|---|---|---|---|---|---|
| **Idea** | Choose the attribute with the highest information content | One of the first methods from Quinlan; uses the concept of information gain | Like ID3 based on the concept of information gain | Choose the attribute that is most dependent on the target variable | Construct many trees with different sets of features and samples (randomly). Result by voting. |
| **Measure used** | Gini-Index | Information gain (entropy) | Ratio of information gain | Chi-square split | Optional, mostly Gini-Index |
| **Type of** | Binary | Complete, | Complete, | Complete, | Complete |

# Introductory Example

# Splitting with Entropy in ID3

# Calculating the Information Gain

The information gain is a measure, that shows (by combination of the entropies) the appropriateness of an attribute for splitting:

where m = number of values (here two: light, strong), ti = number of data sets with strong or light wind (8 resp. 6), t = total number of data sets (14) and entropy(t) = entropy before splitting.

# Decision using ID3

Information gain (outlook) = 0.246

Information gain (humidity) = 0.151

Information gain (wind) = 0.048

Information gain (temperature) = 0.029

We choose the attribute with the largest information gain (here: outlook) for the first splitting.

As solution we obtain the following tree:

# Decision using C5.0

ID3 tends to favor attributes that have a large number of values, resulting in larger trees. For example, if we have an attribute that has a distinct value for each record, then the entropy is 0, thus the information gain is maximal.

To compensate for this, C5.0 is a further development that uses the information gain ratio as a splitting criterion:

In the case of our example the GainRatio of Windy is

and the GainRatio of Outlook is

# Handling Numerical Attributes

Numerical attributes are usually splitted binary. In contrast to categorical attributes many possible splitting points exist .

The splitting point with the highest information gain is looked for. For this, the potential attribute is sorted according to its values first and then all possible splitting point and the corresponding information gains are calculated. In extreme cases there exists n-1 possibilities.

# The CART Algorithm

The CART algorithm (Classification And Regression Trees) constructs trees that have only binary splits. Like C5.0, it is able to handle categorical and numerical attributes.

As a measure for the impurity of a node t, CART uses the Gini Index. In the case of two classes the Gini Index is defined as:

# Splitting in CART

# Coherence between Entropy and Gini Index

**Remark: Entropy has been scaled from (0, 1) to (0, 0.5)!**

# Overfitting (I)

Most decision tree algorithms partition training data until every node contains objects of a single class, or until further partitioning is impossible because two objects have the same value for each attribute but belong to different classes. If there are no such conflicting objects, the decision tree will correctly classify all training objects.

If tree performance is measured from the number of correctly classified cases it is com-mon to find that the training data gives an over-optimistic

# Overfitting (II)

The learner overfits to correctly classify, the noisy data objects

Noisy or dirty data objects

# Random Forest (I)

Random forest is an ensemble classifier that consists of many decision trees.

For every tree a subset of the data objects and a subset of features is randomly chosen. Then the tree is constructed usually using the Gini Index.

In the end, a simple majority vote is taken for prediction.

Algorithm :

1. Create n samples from the original data. Frequent sample size is 2/3.

2. For each of the samples, grow a tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample m* of the m predictors and choose the best split from among those variables.

3. Predict by aggregating the predictions of the n trees (majority votes).

# Random Forest (II)

Voting-Principle of Random Forest:

To avoid overfitting effects , the size and the depth of the trees can be restricted .

# Introductory Example

**websites (features)**

1=visited

0=not visited

**ad (target)**

1=clicked

0=not clicked

Giant sparse matrix!

One matrix for every ad!

Why not classical linear regression?

It is possible to implement a linear regression on such a dataset where Y={0,1}.

Problems:

The predicted values of the linear model can be greater than 1 or less than 0

e is not normally distributed because Y takes on only two values

The error terms are heteroscedastic (the error variance is not constant for all values of X)

Source: Bichler (2015): Course Business Analytics, TU München

# Logistic regression (I)

Logistic regression is a regression model where the dependent variable is categorical. The classical logistic regression is a binary classifier, where the dependent variable has two states. The output of a logistic regression model ranges between 0 and 1.

Logistic regression uses the logistic function (or Sigmoid function) because it can take an input with any value from negative to positive infinity, whereas the output always takes values between zero and one and hence is interpretable as a probability.

It is defined as:

# Logistic regression (II)

If we set

the logistic function can now be written as:

We interpret F(x) as the conditional probability that the class attribute has the value 1 with the given input vector x.

The coefficients ß0 and ß can be estimated via Maximum Likelihood Estimation.

The parameter ß0 represents the unconditional probability of “Y=1” knowing nothing about the feature vector x.

The parameter vector β defines the slope of the logit function. It determines the extent to which certain features contribute for increased or decreased likelihood to “Y=1”.

The output of a logistic model is a probability. To use this for classification purposes:

If the predicted probability is > 0.5 the label is 1

and otherwise 0.

# Functionality of Human Neurons

## A Look into the Nervous System

## Design of a Neuron

# An Easy Example (I)

**f(x) = Activation function**

**e.g.**

**where t = Stimulus threshold**

# An Easy Example (II)

**f(x) = Activation function**

**e.g.**

**where t = Stimulus threshold**

# Functionality of a Neuron

**For the case of n inputs, we can rewrite the neuron's function to**

**with b = -t. b is known as the perceptron's bias. The result of this function would then be fed into an activation function to produce a labeling**

**This results in a linear classifier. Finally, we have to pick a line that best separates the labeled data. The training of the perceptron consists of feeding it multiple training samples and calculating the output for each of them. After each sample, the weights w are adjusted in such a way so as to minimize the output error, defined for example as accuracy or MSE.**

Source: http://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks

# The Multilayer Perceptron

**The single perceptron approach has a major drawback: it can only learn linear functions. To address this problem, we'll need to use a multilayer perceptron, also known as feedforward neural network. Here, we add layers between the input and the output layer, so-called hidden layers** . The hidden layer is where the network stores it's internal abstract representation of the training data.

Input Neurons : receive signals from the outer world .

Hidden Neurons : have an internal representation of the outer world .

Output Neurons : pass signals to the outer world .

## Types of Activation Functions

A linear composition of linear functions is still just a linear function, so most neural networks use non-linear activation functions:

**tangens __ __ hyperbolicus**

logistic function (sigmoid)

# Design of a Multilayer Perceptron

- The Backpropagation algorithm is used for calculating the weights. In a training phase, the weights are iteratively calculated using training data sets in such a way that the difference between the calculated and the expected (true) results is minimized. Because the simultaneous calculation of all weights is not possible, they must be found via a learning process. The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent.
- The procedure in principle:
  - (1) Define the initial weights
  - (2) Put the training set into the input layer
  - (3) Calculate the result (value of the output layer) via successive processing one layer after the other
  - (4) Compare the output values and target values and calculate the difference
  - (5) Iterate steps (2) to (4) for every training set
  - (6) Calculate the total error. Adjust the weights beginning with the output layer towards the input layer (backpropagation)
  - (7) Iterate steps (2) to (6) until the total error reaches the defined error level or the number

# Adjusting the Weights (I)

The error of a training set i is calculated using the quadratic deviation between the values $o_{ij}$ of the neurons of the output layer and their corresponding true values $t_{ij}$.

The sum of the errors of all h training objects is the total error value E:

# Adjusting the Weights (II)

The function E has to be minimized. Because it depends on the output neurons o j , it automatically depends on their weights to the precedent layer(s) :

Thus, the weights have to be found where E is minimal.

Examples of Error functions with two weights:

# Adjusting the Weights (III)

To minimize the error (cost) function E the backpropagation algorithm uses the method of gradient descent . This method searches those weights, where the vector containing the partial first derivatives of the error function (gradient) is equal to the zero vector (minimum):

To adjust the weight w ij , which connects neurons i to j, the formula is:

where a represents a predefined learning rate , which defines the step length of each iteration in the negative gradient direction and x i denote the output value of neuron i .

The adjusted weight is then computed via

# Principle of Gradient Descent (I)

Gradient descent is used to find the minimum of the error function . It works iterative. In an 1-dimensional world, we define the error by

The error function is at minimum if the error is equal to zero.

The prediction is the result of a combination of input and weight

The weight as the dynamic component is now adjusted until the error is at minimum. Starting with an initial weight, gradient descent jumps step by step into the minimum by adjusting the weight. The adjustment is done by calculating the direction and the amount for a step via

Now, the weight is adjusted via

After repeating this several times, the minimum is reached.

# Principle of Gradient Descent (II)

The formula

represents the derivative of the error to the weight.

A derivative is a term that is calculated as the slope (or gradient) of a graph at a particular point. The slope is described by drawing a tangent line to the graph at the point. So, if we are able to compute this tangent line, we might be able to compute the desired direction to reach the minima.

Since the weight only indirectly affects the error, the chain rule must be applied

# Principle of Gradient Descent (III)

Gradient Descent isn’t perfect. When the gradients are too big we might overshoot so much that we’re even farther away than we started

This problem is destructive because overshooting this far means we land at an even steeper slope in the opposite direction. This causes us to overshoot again even farther.

If the gradients are too big, we can make them smaller. We do this by multiplying them by a single number between 0 and 1 (such as 0.01). This fraction is typically named alpha.

Thus, the adjustment of the weights is done by

Source: https://iamtrask.github.io/2015/07/27/python-network-part2/

# Backpropagation Step by Step (I)

In the following, the backpropagation process will be demonstrated using a simple Neural Network consisting of three layers: Input layer with two inputs neurons, one hidden layer with two neurons, and output layer with a single neuron:

Our initial weights will be: w 1 = 0.11, w 2 = 0.21, w 3 = 0.12, w 4 = 0.08, w 5 = 0.14 and w 6 = 0.15.

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (II)

Our dataset has one sample with two inputs and one output with the values inputs=[2, 3] and output=[ 1 ]. We will use given weights and inputs to predict the output. Inputs are multiplied by weights; the results are then passed forward to next layer:

For reasons of simplification, no activation function is used in the neurons.

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (III)

The network output, or prediction, is not even close to actual output. We can calculate the difference or the error as following:

Our main goal of the training is to reduce the error or the difference between prediction and actual output. Since actual output is constant, “not changing”, the only way to reduce the error is to change prediction value. The question now is, how to change prediction value?

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (IV)

By decomposing prediction into its basic elements we can find that weights are the variable elements affecting prediction value. To change prediction value, we need to adjust the weights:

We do this using Backpropagation. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point:

For example, we update w 6 :

*We can picture gradient descent optimization as a hiker (the weight coefficient) who wants to climb down a mountain (cost function) into a valley (cost minimum), and each step is determined by the steepness of the slope (gradient) and the leg length of the hiker (learning rate).*

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (V)

The derivation of the error function is evaluated by applying the chain rule:

To update w 6 we can apply the following formula:

Similarly, we can derive the update formula for w5 and any other weights existing between the output and the hidden layer:

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (VI)

When moving backward to update $w_1$, $w_2$, $w_3$ and $w_4$ existing between input and hidden layer, the partial derivative for the error function with respect to $w_1$, for example, will be as following:

We can find the update formula for the remaining weights $w_2$, $w_3$ and $w_4$ in the same way.

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (VII)

In summary, the update formulas for all weights will be:

We can rewrite the update formulas in matrices:

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (VIII)

With the derived formulas we can now adjust the weights:

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

# Backpropagation Step by Step (IX)

... and use the new weights to recalculate the example:

The new prediction 0.26 is bit closer to the output than the previously predicted one 0.191. We repeat now the same process until error is close or equal to zero.

Source: http://hmkcode.github.io/ai/backpropagation-step-by-step

Problems with fixed Training and Test Samples

Method 1 optimize

Test data is used for two things:

Optimize the model training

Select the best model via testing the model quality

Method 2 optimize

Method 3 optimize

This contradicts the idea of independent testing and results in:

Endogenization of the test data

Selection Bias

… optimize

Rule : NEVER use any information from the test data for model training !

# Addressing the Endogeneity Problem

*Predictive* _ Model_

*Validation Sample*

- Training and test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set ( Selection Bias ).
  - Example of differentOLS models as a result of different samples:
  - To avoid such problems, one can use so-called resampling methods.

# Cross Validation

- In Data Science cross validation can be used for model selection and adjustment. In these cases, cross validation is applied to the training data set. For every iteration, k-1 folds are used for model fitting and the remaining fold for testing the model (Validation). Every time, the quality measure (e.g. accuracy) for the validation fold is captured. At the endof this step, the average and the standard deviation of the measures are calculated. The best model is the one with the best ratio in high average and low standard deviation.
- Once the model type and its optimal parameters have been selected, a final model is trained using these hyper-parameters on the *full* training set, and the generali-zation quality is measured on the test set.

# Cross Validation and Grid Search

- Partition the Dataset into a training and test set
- For every hyperparameter value combination apply cross validation
- For the combination with the highest (mean) quality calculate the final model with the complete training set
- Test the final model with the test set
- Compare the accurracies of training and test with regard to overfitting

Calculate the mean quality of the validation folds, e.g. mean accurracy or mean F1

## Cross Validation and Grid Search in Python

- Performs cross validation with the given hyperparameter combinations and manages the evaluation process

Using the original libraries and functions

KNeighborsClassifier()

cross_val_score()

- Performs cross validation

DecisionTreeClassifier()

RandomForestCl … ()

# Variants of Hyperparameter Optimization

- 1. Grid Search
- Grid search sequentially goes through a preselected list of permutations for each hyperparameter and evaluates the entire search space.
- 2. Random Search
- Random search selects values for hyperparameters at random within a predefined distribution.
- While a grid search is able to find the best model given the provided options, limited compute resources means that in practice, the search space selected will have to be limited. A random search on the other hand does not iterate over the entire search space.

# Other Variants of Cross Validation

- 1. Repeated Cross Validation
- 2. Nested Cross Validation
- Different composition of the folds by random selection.

# Cross Validation in Time Series

- In the case of time series, classical cross validation may cause problems. If we choose random samples and assign them to either the test set or the training set we are quickly in the situation of using values from the future to forecast values in the past. But we want to avoid future-looking when we train our model. If there is a temporal dependency between observations, we must preserve that relation during training and testing.
- A procedure that can be used for cross validating a time series model is cross validation on a rolling basis. Start with a small subset of data for training purpose, forecast for the later data points and then check the accuracy for the forecasted data points. The time frame for the forecast is then included as part of the next training dataset and subsequent data points are forecasted and so on.
- Scikit-learn provides a class *TimeSeriesSplit* to do this.

## Ensemble Methods

Ensemble methods use different models (created via different data sets, feature sets or methods) that are simultaneously applied to the same problem. The results are sent to an aggregating operation that produces the final result.

The most widely used classes of ensemble methods are:

Bagging

Boosting

Stacking

Bagging means to build multiple models from different subsamples of the training dataset and/or with different methods. The results are sent to an (weighted) voting operation that produces the final result.

Source: http://rasbt.github.io/mlxtend/ user_guide /classifier/ EnsembleVoteClassifier /

Boosting involves sequentially building an ensemble by training each new model instance to emphasize the training instances that previous models mispredict . Different variants exist, mostly based on tree methods. In general, any method can be used. This involves the usage of different methods at the different iterations when building the sequence of models.

Source: https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/

Stacking means to build multiple models (typically of differing types) and a supervisor model that learns how to best combine the predictions of the primary models. The inputs of the supervisor model (meta-classifier) are the outputs of the other models:

Source: http://rasbt.github.io/mlxtend/ user_guide /classifier/ StackingClassifier /

Types of Ensembles

Type 1:

consists of only a few models

each is a strong model

like few professional experts

risk of diverging opinions

risk of experts being biased to their experiences

Type 2:

consists of many models

each is a weak model as a principle

based on the idea of the wisdom of the masses

Random Forest and Gradient Boosted Trees are examples

# Predicting using Regression Methods

Example: Predicting House Prices

Function: Price = f(SquareFootage, Bedrooms, Age, SchoolRating)

Source: http://www.sclgsummit.org/uploads/presentation/8934b2d0be055a2261f5d0320f5b59bb.pdf

# Traditional OLS Regression Approach

Function:

Price = ß0 + ß1 * SquareFootage+ ß2 * Bedrooms + ß3 * Age + ß4 * SchoolRating

Source:
http://www.sclgsummit.org/uploads/presentation/8934b2d0be055a2261f5d0320f5b59bb.pdf

# Ordinary Least Squares Regression

# Measuring the Quality of Fit (I)

Measuring the quality of fit means to measure how well the predictions of a model match the observed data.

A commonly-used measure is the Mean Absolute Error (MAE) which can be calculated for the training and the test set

A variant is the Mean Absolute Percentage Error (MAPE) which expresses the error in percent

While MAE and MAPE are easily interpretable, using the absolute value of the error often is not as desirable as squaring this difference. Depending on how you want your model to treat outliers, or extreme values, in your data, you may want to bring more attention to these outliers or downplay them.

Consequently, the most used measure in regression is the Mean Squared Error (MSE) or its variant the Root Mean Squared Error (RMSE), which is the square root of the MSE.

# Measuring the Quality of Fit (II)

# Ridge Regression

Complexity can be measured as the size of the set of possible outputs for a given set of inputs.

In this example the interval 0 to x * represents the set of possible inputs. Function h 0 has the lowest complexity because there is just one output independent of the inputs. h 2 has the highest complexity because here the set of possible outputs is the biggest one.

# Complexity und Generalisation

## Mean Squared Error

# Different Complexities

$\lambda \to \infty$ : Lowest Complexity

the ridge regression coefficients are equal to zero. For every input, the result is β0.

$\lambda = 0$ : Relative High Complexity (linear Model)

the penalty term has no effect, and ridge regression will produce the least squares estimates.

Example:

Source:

James et al. (2013): An Introduction to Statistical Learning with R Applications, p. 215f.

# Handling High-Dimensionality (I)

OLS is not suitable for high-dimensional data. Especially when the number of features p is as large as, or larger than, the number of observations, OLS cannot be applied. _ _ Regardless of whether or not there truly is a relationship between the features and the response, OLS will yield a set of coefficient estimates that result in a perfect fit to the data, such that the residuals are zero.

The figure shows two cases. When there are 20 observations, n > p and the OLS line does not perfectly fit the data. When there are only two observations, then regardless of the values of those observations, the regression line will fit the data exactly. This is problematic because this perfect fit will almost certainly lead to overfitting of the data.

Source:

James et al. (2013): An Introduction to Statistical Learning with R Applications, p. 239f.

# Handling High-Dimensionality (II)

The figure illustrates the risk of applying OLS when the number of features p is large. The model R2 increases to 1 as the number of features increases, and the training set MSE decreases to 0. At the same time, the MSE on a test set becomes extremely large as the number of features increases.

In contrast, methods like ridge regression are particularly useful for performing regression in the high-dimensional setting. Essentially, these approaches avoid overfitting by using a less flexible fitting approach than least squares.

Source: James et al. (2013): An Introduction to Statistical Learning with R Applications, p. 240f.

# Support Vector Regression

The Goal is to find a robust model with a high generalization ability.

SVR regards two sources of Robustness:

1. Eliminating Noise
2. Handling Complexity

# Insensitive Loss Function (I)

**-insensitive Loss**

does not penalize acceptable deviations (defined by )

# Insensitive Loss Function (II)

Using the e-insensitive loss function, only those data objects are considered in the estimation, which have a distance greater than e from the regression function:

e-insen-sitiveRegion

Every object inside the e-insensitive region is ignored. It is regarded as noise.

# Support Vector Regression (I)

Decision criterion:

*Choose the line with the smallest sum of error slopes with paying attention to the flatness of the regression line!*

# Estimating the SVR (Linear Case)

Nonlinearity (I)

**The linear case :**

**The nonlinear __ __ case :**

# Nonlinearity (II)

# Kernel Functions (I)

Kernel Functions are used to project n-dimensional input to m-dimensional input, where m is higher than n:

Any point x in the original space is mapped into the higher dimensional space. For reason of efficiency, the mapping is not performed in real but instead embedded in the model building process via the kernel function:

Instead of ß 0 + ß · x = y the following is used ß 0 + ß · F (x) = y

The main idea to use a kernel is: A linear regression curve in higher dimensions becomes a non-linear regression curve in lower dimensions.

# Estimating the SVR (Nonlinear Case)

# Kernel Functions (II)

A frequently used kernel function is the Polynomial Kernel Function:

where x and z are vector points in any fixed dimensional space and n is the order of the kernel.

In the case of order equal to 2, we get:

Source: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000173

# Kernel Functions (III)

A **nother frequently __ __ used __ __ kernel __ __ function __ __ is __ __ the __** Radial Basis __ **Function __ (__ RBF):**

It maps the data according a Gaussian function where Sigma ( s ) is a streching factor.

Different Sigmas

= Euclidean distance between x and z

Source: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000173

# Using Neural Network for Regression

Artificial neural networks are often used for classification because of the relationship to logistic regression. Neural networks typically use a logistic activation function and output values from 0 to 1 like logistic regression.

But the continuous output of a net must not be interpreted as a probability, so neural networks can be used too for regression, to model complex and non-linear relationships.

The Singlelayer Perceptron corresponds to a linear regression while a Multilayer Perceptron is able to approximate nearly any function regard-less of the complexity and nonlinearity.

Because of the high complexity of the MLP, the models are usually very sensitive and have a tendency to overfitting.

There exist regularization methods, which make the networks better at generalizing beyond the training data.(see http://neuralnetworksanddeeplearning.com/chap3.html)

# Neural Network (Multilayer Perceptron)

Source:
http://www.sclgsummit.org/uploads/presentation/8934b2d0be055a2261f5d0320f5b59bb.pdf

Source:
http://www.sclgsummit.org/uploads/presentation/8934b2d0be055a2261f5d0320f5b59bb.pdf

# Introductory Example

## Decision Tree for Predicting Fuel Consumption of Cars(in Miles-per-Gallon )

# Regression Trees

Some of the tree approaches can be used for regression too. They can be used for nonlinear multiple regression. The output must be numerical.

The figure shows a regression tree for predicting the salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.

The predicted salary is given by the mean value of the salaries in the corresponding leaf, e.g. for the players in the data set with Years<4.5, the mean (log-scaled) salary is 5.11, and so we make a prediction of e5.11 thousands of dollars, i.e. $165,670, for these players.

Players with Years>=4.5 are assigned to the right branch, and then that group is further subdivided by Hits. The predicted salaries for the resulting two groups are 1,000*e6.00 =$403,428 and 1,000*e6.74 =$845,346.

Source: James et al. (2013): An Introduction to Statistical Learning with R Applications, p. 304f.

# Constructing a Regression Tree (I)

Source: James et al. (2013): An Introduction to Statistical Learning with R Applications, p. 305f.

# Constructing a Regression Tree (II)

# Random Forests for Regression

Due to the usage of means as predictors a regression tree usually simplifies the true relationship between the inputs and the output. The advantage over traditional statistical methods is, that it can give valuable insights about which variables are important and where. But the prediction ability is poor compared to other regression approaches.

A much better prediction quality can be achieved with the creation of an ensemble of trees, use them for prediction and averaging their results. This is done, when applying the Random Forests approach to a regression task.

Regression Forests are an ensemble of different regression trees and are used for nonlinear multiple regression. The principle is the same as in classification, except that the output is not the result of a voting but instead of an averaging process.

The disadvantage of Random Forests is that the analysis, which aggregates over the results of many bootstrap trees, does not produce a single, easily interpretable tree diagram.

Frankfurt School Logo

# Comparing the Fitting Ability of one vs. many Regression Trees

## Single Regression Tree

## Average of 100 Regression Trees

# Limitations of Tree Methods in Regression

When applied to regression problems, tree methods have the limitation that they cannot exceed the range of values of the target variable used in training. The reason for this lies in their design principle, how the leaves of the trees are created.

Thus, Random Forests may perform poorly when the target data is out of the range of the original training data, e.g. in the case of data with persistent trends. A solution may be a frequent re-training in this case.

An important strength of Random Forests is that they are able to perform still well in the case of missing data. According to their construction principle, not every tree is using the same features.

If there is any missing value for a feature during the application there usually are enough trees remaining that do not use this feature to produce accurate predictions.

# k-Nearest Neighbors for Regression

k-Nearest Neighbors cannot only be used for classification but also for regression. The only difference in regression is that the prediction is not the result of a majority vote but of an averaging process.

A simple implementation of KNN regression is to calculate the average of the numerical target of the k-nearest neighbors. Another approach uses an inverse distance weighted average of the K-nearest neighbors. KNN regression uses the same distance functions as KNN classification.

Example:

# Introductory Example

Assume you are a wholesale distributor and each row of your dataset corresponds to a customer showing the following attributes:

1) FRESH: annual spending on fresh products (Continuous); 2) MILK: annual spending on milk products (Continuous); 3) GROCERY: annual spending on grocery products (Continuous); 4) FROZEN: annual spending on frozen products (Continuous) 5) DETERGENTS_PAPER: annual spending on detergents and paper products (Continuous) 6) DELICATESSEN: annual spending on delicatessen products (Continuous); 7) CHANNEL: customers buying channel (Nominal) 8) REGION: customers region (Nominal)

Your goal is to segment the users. That means finding similar types of users and bunching them together.

Why would you want to do this?

You might want to give different users different experiences. Marketing often does this; for example, to offer toner to people who are known to own printers.

You might have a model that works better for specific groups. Or you might have different models for different groups.

# Cluster Analysis

**Cluster analysis is a type of multivariate statistical analysis. It is used to group data into separate clusters. The main objective of clustering is to find similarities between data objects, and then group similar objects together to assist in understanding relationships that might exist among them. Cluster analysis is based on a mathematical formulation of a measure of similarity.**

**There are different types of cluster analysis methods:**

**Clustering Methods**

# Partitioning Cluster Methods

**The partitioning cluster methods divide the data into a predetermined number of** clusters.
The most popular technique is the K-Means algorithm.

Given a set of observations ( $x$ 1, $x$ 2,…, $x$ $n$ ), where each observation is a $m$ -dimensional real vector, $k$ -means clustering aims to partition the n observations into ( $k \leq n$ ) segments $S$ ={ $S$ 1, $S$ 2,..., $S$ $k$ } so as to minimize the within-cluster sum of squares (WCSS).

The objective is to find

where _ _ is the mean of points in $S$ $i$ .

Procedure of K-Means:

**Step 1: Randomly partition the data objects into k clusters.**

**Step 2: Calculate the cluster centroids.**

**Step 3: Calculate the distance from every data point to all centroids**

**Step 4: If a data point is closest to its own centroid, leave it where it** __ __ is. If the data **point is not closest to its own centroid, assign** __ __ it to the cluster with **the closest centroid.**

**Step 5: Repeat the step 2 to 4 until a complete pass through of all** __ __ the data points **results in no data point changing from one** __ cluster to another.__

# Example of a K-Means Cluster Analysis

**Between cluster variance:**

**Within cluster variance:**

Finding the Optimal Number of Clusters (I)

**The aim of the cluster analysis is the segmentation of objects into clusters, which are preferably homogeneous in it selves and heterogeneous to each other. The less variance exists within the clusters and the more variance exists between the clusters, the better is the number of clusters.**

**Total variance:**

**Accumulated variance within the k clusters:**

**This results in the variance between the clusters:**

**with n = number of objects**

__ m = number of attributes__

__ __ **n k** __ = number of objects in cluster k__

__ __ **c k** __ = cluster k__

# Finding the Optimal Number of Clusters (II)

**If you put V in** __ on the ordinate and the number of cluster k on the abscissa, it often results in a curve with one or several kinks. At the point where exists the (first) significant kink, you can find the optimal number of clusters:__

**Total variance V tot**

**Between __ __ cluster variance V betw**

Within cluster variance **V in**

**Number of clusters**

# Finding the Optimal Number of Clusters (III)

Instead of visually identifying the optimal cluster number, we can calculate the distances from the points on the elbow curve to a straight line linking the first and the last point on the curve. The cluster number with the largest distance is then chosen as the one with the strongest kink.

**Number of clusters**

# Hierarchical Cluster Methods

- **There are two types of hierarchical cluster methods:**
  - **Agglomerative hierarchical clustering is a bottom-up clustering method. It starts with every single data object in a single cluster. Then, in each iteration, it agglomerates (merges) the closest pair of clusters by satisfying some similarity criteria, until all of the data is in one cluster.**
  - **Divisive hierarchical clustering is a top-down clustering method. It works in a similar way to agglomerative clustering but in the opposite direction. This method starts with a single cluster containing all data objects, and then successively splits resulting clusters until only clusters of individual data objects remain.**

# Process of the Hierarchical Cluster Analysis

# Measuring Similarity between Clusters (I)

Distance between two clusters is the distance between the closest points:

Complete Linkage:

Distance between two clusters is the distance between the farthest pair of points:

Distance between two clusters i and j is the distance between their cendroids :

# Measuring Similarity between Clusters (II)

Average Linkage:

Distance between clusters is the average distance between the cluster points:

Ward’s Method / Minimum Variance Method (only Agglomerative):

Ward’s minimum variance criterion minimizes the total within-cluster variance. At each step the pair of clusters is merged that leads to minimum increase in total within-cluster variance after merging. This can be calculated as the square of the distance between cluster means divided by the sum of the reciprocals of the number of observations in each cluster:

For a comparison of the methods see: Ferreira, L.; Hitchcock, D. B. (2009): A Comparison of Hierarchical Methods for Clustering Functional Data, http://people.stat.sc.edu/Hitchcock/compare_hier_fda.pdf

# Single Linkage Example (I)

Source: Fred, Ana: Unsupervised Learning, Universidade Técnica de Lisboa

# Single Linkage Example (II)

Source: Fred, Ana: Unsupervised Learning, Universidade Técnica de Lisboa

A dendrogram is a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering. The y-axis represents the value of this distance metric (e.g. euclidean distance) between the clusters.

In a dendrogram the widths of the horizontal lines give an impression about the dissimilarity of the merging object. Thus, a good cluster number might be at a point from where the width of the following horizontal lines is significantly smaller in length. The red line in the graph below shows such a point:

Counting the points that cut this line might be a good answer for the number of clusters the data can have. It is the number 6 in this case.