TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Data Mining in Actuarial Performance Indicators using Self-Organizing Maps

Ausgeführt am Institut für

Stochastik und Wirtschaftsmathematik
der Technischen Universität Wien

unter Anleitung von

Associate Prof. Dipl.-Ing. Dr.techn. Stefan Gerhold

durch

Stephanie Schmid, BSc.

Clemens-Holzmeisterstraße 31
6166 Fulpmes

May 8, 2017

Unterschrift

Abstract

Data Mining becomes a vital aspect in data analysis and clustering is a potential tool of Data Mining. In this work we apply the Data Mining method of an artificial neural net, namely Self-Organizing Maps, on a dataset containing information about new business contracts.

We explain neural nets in general and how Data Mining can be applied in insurance companies. By means of the classical vector quantisation process we explain the algorithm of Self-Organizing Maps and its parameters. We then apply the algorithm to a data sheet provided by the actuarial life department of Allianz Elementar Lebensversicherungs-AG to get a better insight into parameters affecting the new business margin, going beyond the already widely performed analyses.

The outcomes show clear evidence that Self-Organizing Maps can cluster this data into individual groups. Though the results support the assumption of a highly non-linear correlation between the new business margin and the parameters leading to it, only a small amount of data can be used for analyses. A suggestion on how to set parameters leading to a more stable, higher new business margin is made nevertheless.

Because of limitations concerning data, software and time, Self-Organizing Maps are not the ideal solution for analysing this kind of data. Especially due to the required time-consuming, manual pre- and postprocessing it is not recommended to use the methods presented in this work on a regular basis on this particular data sheet.

Danksagung

An dieser Stelle möchte ich die Möglichkeit nutzen, meinen außerordentlichen Dank auszudrücken. Großer Dank gilt meiner Familie und besonders meinen Eltern, die mich wo immer sie konnten unterstützt haben. Ebenso meinen Freunden daheim, die trotz langer Zeiten der Abwesenheit immer für mich da waren. Meinen Studienkollegen, von denen viele zu guten Freunden wurden und die gemeinsam mit mir Teile des Weges gingen, möchte ich danken. Sie haben mich in den anstrengenden Zeiten des Studiums bestärkt weiterzumachen und mit mir die Tiefen der Mathematik erforscht. Ich verdanke ihnen anregende Gespräche, die geteilte Leidenschaft für Diskussionen und immer wieder neue Ideen.

Weiters danke ich der Allianz Elementar Lebensversicherungs-AG, die mir durch ihre Datenfreigabe und finanzielle Unterstützung ermöglicht hat, diese Arbeit zu schreiben. Hier durfte ich im Aktuariat Leben unter Andreas Mehl und Peter Hüttner frei und unter dem ständigen Bewusstsein der Hilfsbereitschaft arbeiten. Allen aus der Abteilung, die mich in den letzten Monaten meines Studiums begleitet haben, gilt mein Dank.

Ich danke auch Stefan Gerhold, der mir als wissenschaftlicher Betreuer auf Seiten der TU Wien zur Seite stand. Mit Motivation und neuen Ideen konnten durch ihn viele Fragen beantwortet werden. Weiters gebührt mein Dank allen Professoren der TU Wien, die mich im Rahmen meines Studiums gefordert und gefördert haben.

Contents

List of Figures

Chapter 1

Introduction

When the collection and analysis of large amounts of data is getting very cheap, this accelerates firstly the known process of hypothesis, falsification and new hypothesis. Secondly, interesting connections, interesting questions sometimes evolve only from careful data analysis. [...] *This is a fundamentally different approach, because now the computer analysis does no longer tell whether the assumption was right or wrong, but requests to look at a relationship.*

Viktor Mayer-Schönberger [53]

A vivid and often used example to use data is customer targeting. Small pieces of data from websites are stored on computers or mobile devices by browsers. These so-called cookies allow the website to save actions or preferences over time. This process is said to aid a better user experience but also helps homepage operators to get individuals' data. With this information visitors can get customized advertisement, leading to a better (lower cost, higher or equal returns) cost-benefit ratio for a company, not necessarily dealing with this particular website.

While more and more companies are already aware of the increasing importance of data collection and analysis, insurances have often fallen back. Busily employed with their long known analyses and models the search for new data only starts reluctantly. Although the data used in this work might not be referred to as Big Data, the dataset is sufficiently vast to start showing how Data Mining methods can be implemented for insurance based data. The problem to overcome is the rigid view of big insurance companies to only maintain own model-based data with standard and classical statistics. The mixture of statistical approaches with new Data Mining techniques may give better insights than the method of following or denying a given hypothesis.

Introductory example:

Imagine you call all people you know to a big soccer field. When all have arrived you tell them to ask each other about some defined attributes like age, place of residence, yearly income, height and so on. Their task is to find the person with the most similar properties. While they pass over the field (rearrange) you manage to get a high view point above the crowd.

When they are done you ask them to raise flags in the colour according to their age: those younger than ten shall hold up a dark blue flag, between the age of ten and 20 a light blue one and so on to the oldest ones raising a dark red flag. From your position above the crowd you take a picture of all the flags colouring the whole soccer field. Then you tell them to stay in place and raise a flag with the colour according to their place of residence: the more west they live, the more blue the flag should be, and red for east. After taking a picture you call the next characteristic and so on.

For each attribute you take a photo of the colour distribution on the field. What you get are several multicoloured pictures with the properties of your acquaintances. This colour pattern corresponds to the colour-coded maps visualised within this work. Finally you can put up the photos next to each other and inspect dependencies. You could see groups of younger people in blue and clusters of older people in red. Or you could see a correlation i.e. between age and gross income. Continuing in this manner you will discover further relationships among the defined attributes.

The algorithm we use in this work is programmed to do this: projecting high dimensional data (in this example attributes of your friends; in the following work insurance contracts) to a lower-dimensional representation.

In this work we talk about Data Mining in general and neural networks in particular. We use an artificial neural network called *Self-Organizing Map* (*SOM*) to find groups of similar newly entered contracts of Allianz Elementar Lebensversicherungs-AG. First we would like to test the ability of the algorithm to cluster the data into comprehensible groups. Secondly we would like to know, if we can find clusters and data points leading to an above average new business margin without using the actuarial internal model data. Within the clusters, the data should be as similar as possible, leading to an expected similar new business margin. In SOMs high-dimensional data can be visualised on surfaces easy to understand. Some guidance of what and what not to interpret to the plots is given.

A question is whether it is possible to reproduce results of statistical analyses with neural networks. In addition we would like to speed up the analyses about newly entered contracts and visualise the outcomes better than it is possible right now. The data sheet contains all customer related variables to calculate the *Market Consistent Embedded Value* (*EV* or *MCEV*), which is a construct allowing insurance companies and their contract portfolio to be (re-)valued. Expectedly this measure is depending on contract and finance market related factors. Due to the complexity actuaries are interested in determining dependencies within the customer oriented calculation. The data worked with in this thesis includes customer related values as well as individual contribution components of the EV. It consists of more than 9000 variable vectors (i.e. contracts).

Although neural networks are in discussion for more than three decades, no best practice method is known for each application. Therefore a lot of discussion goes into pre-analyses and selection of model parameters. As in most operating knowledge-based systems, definition of the relations is based on human evaluation [31].

This thesis is meant for readers interested in Data Mining and financial structures. It will give an overview of the mathematical, data analysis and financial side of the problem. The intention for this work was to be readable and understandable for non-experts of both finance and data analysis. Therefore, some basics are explained for either side.

Text highlighted like this shows the outcomes of previous discussions. It sums up what will be used in subsequent chapters.

The remaining work is structured as follows: First we discuss some mathematical and statistical issues that will influence the application. The second chapter deals with the overview of Data Mining in general and in insurance groups. Additionally, we explain how different neural networks work. Chapter 4 gives a detailed description of Self-Organizing Maps and their precursors of Learning Vector Quantisation. All theoretical background is then applied in Chapter 5 where we explain the decision process that led to the shown visualisations. Exemplary conclusions are shown by variables in use. In the last chapter we mention possible extensions of the problem and how they may be implemented.

Chapter 2

Mathematical Preliminaries

Among the many Data Mining methods advertised in literature, there are only a few fundamental techniques. The actual underlying model representation used by a particular method typically comes from a composition of a small number of well-known options. The mathematical background - shortly recapitulated in this chapter - is of significant importance within the topic of Self-Organizing Maps which will be discussed later on.

2.1 Distance Measurement

2.1.1 Definition. A function dist is a *distance function* if it satisfies the following:

(D1) $\mathrm{dist}(p_1, p_2) = d \in \mathbb{R}_{\geq 0}$,

(D2) $\mathrm{dist}(p_1, p_2) = 0 \Leftrightarrow p_1 = p_2$,

(D3) $\mathrm{dist}(p_1, p_2) = \mathrm{dist}(p_2, p_1)$

for two points p_1, p_2 in the parameter space Θ. A distance function is called a *metric*, if it satisfies (D1) - (D3) and additionally the *triangle inequality*:

(Δ) $\mathrm{dist}(p_1, p_2) + \mathrm{dist}(p_2, p_3) \geq \mathrm{dist}(p_1, p_3)$.

2.1.2 Remark. This definition fits the intuitive view of the direct relation between smaller distances and similar objects. Similar objects get lower values in the distance function. Note the possibility of defining a similarity function sim analogous to the distance function with the difference of having higher values for similar objects. (D2) has to be replaced appropriately with e.g. $\mathrm{sim}(p_1, p_2) = \infty \Leftrightarrow p_1 = p_2$.

2.1.3 Example. As stated in Ester and Sander [15] the most common distance functions for data points $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ with numerical or categorical entries are:

– the *Euclidean distance*

$$\mathrm{dist}(x, y) = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2},$$

– the *Manhattan distance* with the absolute value function $|.|$

$$\mathrm{dist}(x, y) = \sum_{i=1}^{d} |x_i - y_i|,$$

– the *L_p-metrics*

$$\operatorname{dist}(x,y) = \sqrt[p]{\sum_{i=1}^{d}(x_i - y_i)^p}, \quad p \in \mathbb{R}_{>0},$$

– or the *Hamming distance* counting the different entries

$$\operatorname{dist}(x,y) = \sum_{i=1}^{d} 1 - \delta(x_i, y_i),$$

using *Kronecker's delta*

$$\delta(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{if } x \neq y \end{cases}.$$

2.1.4 Example. (cf. [15]) Also text and finite sets can be analysed. Possible distance functions for these kinds of data are the following:

– Let x and y be finite sets consisting of $x = \{x_1, \dots, x_d\}$ and $y = \{y_1, \dots, y_d\}$ respectively. Here the function $|.|$ counts the elements in a set. A distance can be defined as proportion of different data in x and y:

$$\operatorname{dist}(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}.$$

– In a document D, let be given a vocabulary T with terms t_i. The document is then represented as $r(D) = \{f(t_i, D)\}_{t_i \in T}$, with $f(t_i, D)$ symbolising the number of occurrences of t_i in the document. If we use an injective damping function $g : \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0}$ such as e.g. the logarithm, applied per element, we can consider the vectors $g(r(D)) = \{g(f(t_i, D))\}_{t_i \in T}$. The distance between two documents D_1 and D_2 can then be defined via the cosine of the angle between those vectors:

$$\operatorname{dist}(D_1, D_2) = 1 - \left| \frac{\langle g(r(D_1)), g(r(D_2)) \rangle}{\|g(r(D_1))\| \cdot \|g(r(D_2))\|} \right|$$

with the scalar product $\langle ., . \rangle$ and the therewith defined length of vectors $\|.\| = \sqrt{\langle ., . \rangle}$.

2.2 Principal Components

2.2.1 Definition. Let $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_n)$ be real-valued random vectors with existing expected value of $\mathbb{E}(X)$, $\mathbb{E}(Y)$ and variance of $\mathbb{V}(X)$, $\mathbb{V}(Y)$, respectively. The *covariance matrix* (*cov*) of X and Y is then defined by

$$cov(X,Y) := \mathbb{E}\left([X - \mathbb{E}(X)]^T [Y - \mathbb{E}(Y)]\right).$$

With a positive variance the *standard deviation* σ is defined as the square root of the variance and we can define the *correlation matrix* (*cor*) by

$$cor_{XY} := diag(\sigma_X^{-1}) \cdot cov(X,Y) \cdot diag(\sigma_Y^{-1})$$

with $diag(\sigma^{-1})$ being the matrix with the reciprocal of the per-element standard deviation σ in the principal diagonal.

2.2.2 Remark.

If X and Y are standardized with an expected value of 0 and a variance of 1 for every entry this leads to:

$$cor_{XY} = \mathbb{E}(X^T Y) = cov(X, Y).$$

Let X be standardized with an expected value of 0 for each entry. Then the correlation matrix cor_{XX} is symmetric and diagonalisable. In addition there exists an orthonormal basis $O = (o_1, \dots, o_n) \in \mathbb{R}^{n \times n}$ for $\mathbb{R}^n$ which consists of eigenvectors $o_1, \dots o_n$ of cor_{XX} such that

$$cor_{XX} O = OD$$

with the diagonal matrix of eigenvalues $D = diag(\lambda_1, \dots, \lambda_n)$ with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$.

The higher the eigenvalue, the more variance is explained by the corresponding eigenvector. Therefore this so-called *principal component analysis* (*PCA*) reveals a part of the internal structure of the data. When a multivariate dataset is visualised in a high dimensional space, the purpose of PCA is to reduce the dimensionality of observation vectors by trying to span the data in those dimensions where the dataset has most variance. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

Using PCA for data analysis is, according to [32], an idea from the 1930s and widely used today. However in its pure form PCA does not incorporate information on how objects should be compared (cf. [66]).

We will use PCA for preprocessing as well as in pre- and post-analysing our data.

2.3 Classification and Clustering

The objective of clustering is to gather the most similar data points in the same group, called cluster. The best clustering method gets most equal data together and has significant dissimilarities between clusters. Significance in this case relies upon the analysis' purpose and user. Possible definitions of similarity and distance can be seen in Section 2.1.

2.3.1 Definition.

(i) A *cluster* or *class c* is a subset of given data with some kind of similarity within itself and some distance to other subsets.

(ii) *Classification*, also referred to as *pattern recognition* is applying a function that maps or classifies a data item into one of several predefined classes or clusters $C = \{c_1, \dots c_m\}$.

A number of clustering techniques exist, broadly classifiable into *hierarchical* and *partitioning* methods. Hierarchical ones iteratively merge or distribute data points into a number of clusters. The former borders stay preserved. This method directly produces binary trees.

The objective of using partitioning methods is to separate a dataset into some disjoint subset of points, such that the points lying in each subset are as similar as possible.

Both methods can work divisive or agglomerative, meaning it is possible to start with one single cluster and separate it into more or join data points into a class and possibly end up with only one class for all data, respectively.

Another way to subdivide the methods is hard and fuzzy clustering. In hard clustering the developed cluster will have their well defined boundaries. Thus a particular data point will belong to exactly one cluster. On the other hand in fuzzy clustering a particular data point may belong to the different clusters with a different membership value, e.g. with 60% it belongs to cluster A and 40% to cluster B. (cf. [59], [18], [31])

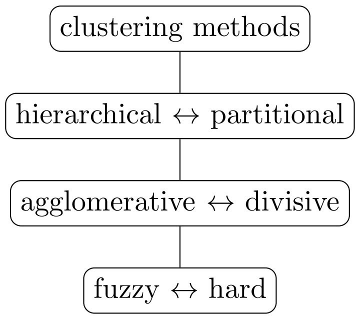


Figure 2.1: Possible general classification of clustering techniques.

2.3.2 Example. (Nearest Neighbour, cf. [32]) The most trivial classification or pattern recognition method is to compare the unknown pattern (f.i. a combination of data points or pixels) with all known reference patterns on the basis of some criterion for the degree of similarity in order to decide in which class the pattern belongs. Since natural patterns usually have a high dimensionality, plenty of computations are needed, unless a small representative set of reference-vectors from each class can be used for comparison. In this *Nearest Neighbour* (*NN*) method, the mutual distance is computed for all reference patterns and the pattern is classified according to the smallest value.

As the distributions of the reference patterns of different classes usually overlap, the simple NN method does not reliably define the class borders. A method widely used is to consider K nearest reference-vectors to the sample to be classified. The class is then identified according to the majority of classes encountered among those K nearest neighbours. It can be shown that the class borders defined by this *K-Nearest Neighbour* (*KNN*) method are very close to the statistically optimal borders derived according to the Bayesian theory of probability. (cf. [7])

2.3.3 Example. (K-means, cf. [13]) Another traditional clustering method with similarities to the algorithm used later is *K-means*. To approximate the distribution of high-dimensional input vectors using a smaller number of suitable selected reference- or model vectors one minimizes the average quantisation error (see Definition 4.3.3). The difference between input sample and reference-vectors is then defined as an error. By comparing all input vectors with all reference-vectors, one can identify the model vectors with the smallest error. In practice, each cluster can be represented by one or several reference-vectors.

The major drawback of the K-means algorithm is the high dependency on K. Upon changing K, different kinds of clusters may emerge. Good initialisation of the model is crucial as some of the clusters may even be left empty if the initial values are far from the distribution of the data.

In the neural net we use hard clustering.

2.4 Conditional Inference Trees

Objects can be automatically classified and decision making simplified by the method of decision trees. They consist of a root, any number of internal and at least two end nodes. Every inner node represents a decision and every end node an answer to the question asked.

2.4.1 Definition. A *dendrogram* is a tree-structured graph for visualising hierarchical clustering of data. The root represents one single cluster, containing the full quantity of data. Leafs and nodes show clusters holding at least on data point. An inner node contains the consolidation of all its child nodes.

Roughly speeking the algorithm used is as the following:

(*i*) Test the null hypothesis.

Stop if this hypothesis cannot be rejected.

Otherwise select the input variable with the strongest association to the response.

(*ii*) Implement a binary split in the selected input variable.

(*iii*) Recursively repeat the former steps.

2.4.2 Remark. Dendrograms are decision trees applied for separating data. They can visualise recursive binary partitioning of dataset.

2.4.3 Example. (Decision trees are regression models) A decision tree is a simple representation of classifying examples. A tree is grown by splitting the source into subsets based on an attribute value test. We want the tree to find the variable the input responds most, which is calculated by regression. The process is repeated on each derived subset in a recursive manner. Splitting ends when the subset of a node has all the same value of the targeted variable or when splitting no longer adds value to the predictions.

A root can f.i. be the weather outlook, having three branches (e.g., sunny, overcast and rainy), each representing values for the attribute tested. With the variables of temperature, humidity and windiness we want to find out how many hours of that day can be used to play football. (see [51] for a step by step calculation)

In the case of this work we feed the tree model with our data sheet wanting two branches every time we split. For us it is important, which variable the algorithm chooses for separating the data. This shows us, how the split groups get more homogeneous.

The problem with this tool for regression analysis is the overfitting and a selection bias towards covariates. Pruning the tree can solve the overfitting problem. The variable selection bias though still affects the interpretability. Unbiased procedures have been suggested for some special cases often lacking a common theoretical foundation. The solution worked with was to use a unified framework for recursive partitioning which embeds regression models into the theory of conditional inference procedures. (cf. [26])

These *conditional inference trees* are statistics-based and use non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. The general null hypothesis is independence between all input variables and the response. If the general null hypothesis has to be rejected, a partial null hypothesis of the strongest dependency between one of the input variables and the response is tested. In our application the input variables are the given contract variables and the response is the new business margin (see Definition 3.2.1).

The association to the response is measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response. The evaluation of this p-value is adjusted by either the methods of Remark 2.4.4, univariate-statistic or using values of the test-statistic.

2.4.4 Remark. (cf. [67], [26])

1. Bonferroni Method: This correction compensates for the increasing likelihood of incorrectly rejecting a null hypothesis. The increase happens when multiple comparisons are done and the chance of a rare event increases. This is done by testing each individual hypothesis at a significance level of α/m. Here α is the desired overall significance and m the number of hypothesis given. A downside of this correction is the increasing probability of a reduced statistical power by producing more falsely negatives.

2. Monte Carlo or Permutation Method: Here pseudo datasets are created by sampling observations without replacement from the original data. The pseudo dataset is used to calculate the p-value. Whether the minimum p-value from the pseudo dataset is lower or equal to the actual p-value is recorded by a counter. This process is repeated a large number of times. The proportion of the counter to the total number of repetitions is the adjusted p-value used. The p-values adjusted by this method incorporate all correlations and distributional characteristics of the original dataset.

2.4.5 Remark.

- For more information about the basic statistical understanding of null hypothesis and p-values see e.g. [20].
- The splitting criterion when creating the trees was based on the adjusted p-values. The maximal p-value was 0.05 in order to split the nodes in the pre-analyses of this work.
- The statistical approach to split the trees only if the p-value reaches a certain significance ensures the right sized tree to be grown. No form of pruning or cross-validation is needed.

The conditional inference trees (grown with the four p-value evaluation methods mentioned before) were compared with a stepwise regression tree. As expected, the regression tree focused on variables with a higher variance. The conditional inference trees led to a rather similar result within the four runs concerning depending variables.

We use the function `ctree` from the `R`-package `party` with a minimal splitting criterion of 0.95 four times, using the four different p-value adjustments. The outcomes of dependencies will be used as a criterion for selecting training variables.

2.5 Graphs

2.5.1 Definition.

(*i*) A finite, simple, directed *graph* $G = (V, E)$ is composed of a non-empty finite set of vertices or nodes V and a set of edges $E \subseteq V \times V$.

(*ii*) $P(i) := \{j \in V : (j, i) \in E\}$ is the *set of all direct predecessors* of the node $i \in V$.

(*iii*) $S(i) := \{j \in V : (i, j) \in E\}$ is the *set of all direct successors* of the node $i \in V$.

(*iv*) A node i with $P(i) = \emptyset$ is a *source*, if $S(i) = \emptyset$ it is a *sink*.

(*v*) With an integer l let

$$e_j := (i_{j-1}, i_j) \in E \quad \text{with} \quad i_0, \dots, i_l \in V \quad \text{and} \quad j = 1, \dots, l.$$

A sequence of edges $(e_1, \dots, e_l)$ is called *path* from i_0 to i_l with the number of edges l as its *length*.

(*vi*) A path with coinciding first and last node is called *cycle*. A graph without any cycles is *acyclic*.

2.5.2 Remark. Let $G = (V, E)$ be an acyclic graph. Let V be partitioned into non-empty sets as

$$V = V_0 \,\dot\cup\, V_1 \,\dot\cup\, V_2 \,\dot\cup\, \dots \,\dot\cup\, V_k \,\dot\cup\, V_{k+1} \tag{2.5.1}$$

for some $k \geq -1$ in the following way: V_0 contains all sources. Reduce G by all vertices in V_0 and all edges starting in one of them. The resulting graph is again acyclic. Let V_1 be the set of all sources of the sub-graph, etc. With this partitioning, $i \in V_j$ holds if the longest path from a source to i has length j. Moreover the elements of V_{k+1} are sinks and

$$i \in V_j \Rightarrow P(i) \subseteq V_0 \cup \dots \cup V_{j-1} \quad \text{and} \quad S(i) \subseteq V_{j+1} \cup \dots \cup V_{k+1}.$$

2.5.3 Remark. Any tree in 2.4 is actually a graph in a different context. Sources correspond to roots and sinks to end-nodes. We use trees for pre- and post-analyses and graphs for defining neural networks. The nodes in V_0 then represent the entries of an input vector.

Chapter 3

Data Mining and Big Data

3.1 Definitions

When speaking about Data Mining and Big Data, the most common problem is the missing definition of terms. Before continuing the discussion about neural modelling we will first state how the terms in use are defined.

3.1.1 Remark. It has to be said that the definitions in Definition 3.1.2 and Definition 3.1.4 are by no means the official ones or the only right ones. In fact, the Berkeley School of Information published an article asking 40 thought leaders and got 40 different definitions for Big Data and Data Mining (cf. [14]). Forbes Magazine found and discussed an additional twelve definitions (cf. [45]). Many coincide with the *theme of the three Vs*: volume, velocity and variety, meaning the data is too big and too complicated to be processed manually; but some are rather different, taking their purpose into account as well. The definitions mentioned in this section are only the author's favourite and are not to be understood uniquely.

3.1.2 Definition.

(*i*) A set of facts, for example entries in a database, is known as *data*.

(*ii*) *Big Data* is a general term for the widely, mostly unfiltered collected data that cannot be processed manually because of its vastness.

(*iii*) *Data Mining*, also referred to as *Knowledge Discovery in Databases* (*KDD*), is the non-trivial process of data-driven analysis in databases to extract from an immense wealth of details valid, currently unknown, potentially useful and understandable structures, connection or patterns. (cf. [49])

3.1.3 Remark. Fayyad et al. [17] proposed to refer to Data Mining as a particular step of the KDD process, applying specific algorithms for extracting patterns from data. When using this definition, data preparation, selection and transformation are outsourced as well as incorporation of appropriate prior knowledge and proper interpretation. See Figure 3.1.

3.1.4 Definition. (cf. [17])

(*i*) The notion *interestingness* is an overall measure of the pattern value to a user. Interestingness combines validity, novelty, usefulness and simplicity of the connection found via Data Mining.

(*ii*) We consider a *pattern* to be knowledge about data exceeding some interestingness threshold. It is an expression describing a subset of the data or a model applicable to the subset.

3.1.5 Remark. Interestingness can be defined explicitly via functions or through expert's choice. In the definition of patterns, knowledge is only user orientated, domain specific and determined by whatever functions and thresholds the user chooses.

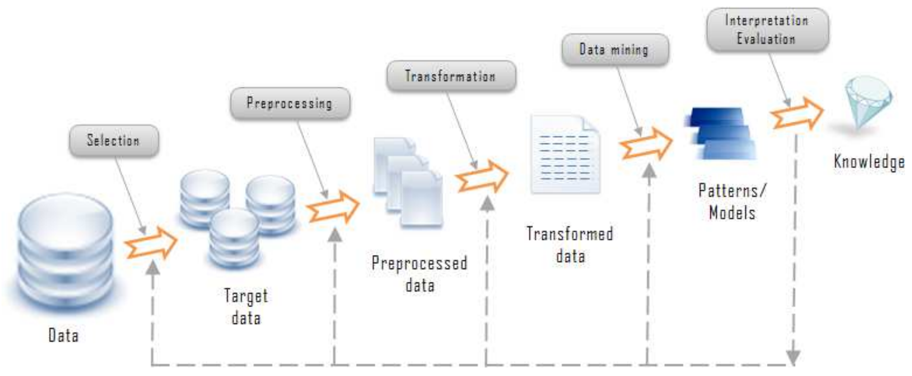


Figure 3.1: Data Mining as a part of the KDD-Process. Picture taken from [60].

3.2 Application in Insurance Companies

General use in corporations

The goals of Data Mining in companies can be summarized in verification and discovery, whereby the discovery can be separated in prediction and description. Reuß [49] proposes a generic process model to embed Data Mining in business processes. This model focuses on actions, not on data analysis and is modelled as a cycle with continuous cost-benefit consideration. As Data Mining is only one step in the analysis cycle, we will not go deeper into this approach, only showing the idea in Figure 3.2.

In addition to a process, a company should perform a systematic potential assessment. Business processes with a capability of improvement by using Data Mining should be identified. After that, the potential has to be rated monetarily. The actions to be processed also have to be valued with regards to the impact on critical success factors. Possible applications are sketched in Figure 3.3.

It has to be mentioned, that no best practice method does exist and analysis depends always on the users (e.g. companies, researchers, scientists) and their goals.

To find potential processes, appropriate data suitable for a Data Mining technique have to be detected. With regard to insurance companies there are multiple internal and external data sources. Internal sources include client information and model assumptions. Examples for external sources are official statistics or financial market data.

Data from financial markets have to be used carefully in neural networks for the reason that despite the importance of information retrieval from financial markets, there is a lack of metrics considered to manage specifically financial data (cf. [48]).

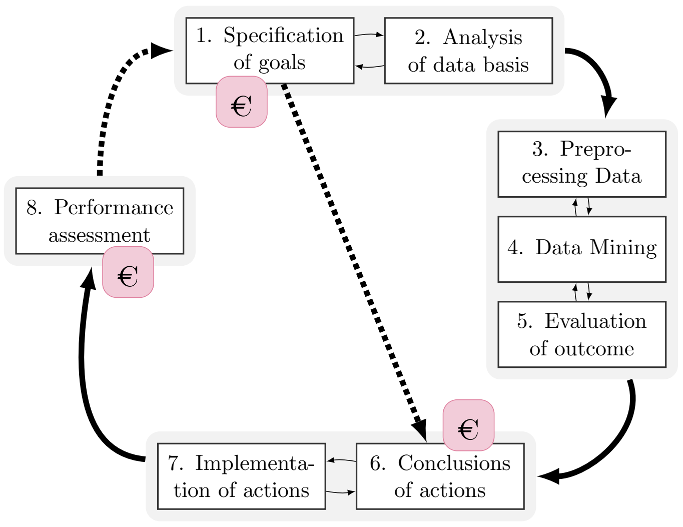

1. Specification of goals
2. Analysis of data basis
3. Prepro-cessing Data
4. Data Mining
5. Evaluation of outcome
6. Conclusions of actions
7. Implementa-tion of actions
8. Performance assessment
€
€
€

Figure 3.2: A generic circulation model to embed Data Mining in business processes. The steps in the process marked with a €-Symbol are the ones where a cost-benefit analysis can and should take place. See [49] for further informations.

key process	possible application field for Data Mining
product development	analysis of competitiveness and profitability of a tariff
	determination of significant risk factors
product distribution	determination of customers keen to sign up
	determination of cross-selling potential
application handling	automated risk audit
portfolio management	lapse prevention and retention promotion
	determination of customer value
	supervision of portfolio development
claims handling	fraud detection
	estimation of claims amount and claims reserve

Figure 3.3: Problem-sided potential analysis showing possible application fields of Data Mining in a general meaning, sorted into key processes of insurance companies. (cf. [49])

Life insurance

In the insurance market, data evolve naturally by clients' contracts. Especially in life insurance, policyholders provide personal information on their own accord to an extensive amount.

In addition to information received directly from the insured person, models are calculated on the insurance side to forecast the benefits of each contract signed. Natural major income sources are new contracts. Definition 3.2.1 are the official Allianz Group definitions (cf. [1]). We would like to know if there is additional information to be found to separate (concerning the new business margin) profitable contracts beforehand without using models to actually calculate the indicators.

3.2.1 Definition. (cf. [1])

(i) The *present value of future profits* (*PVFP*) describes the future, statutory shareholder profits. It is calculated after tax projected to emerge from operations and assets backing liabilities, including the value of unrealised gains on assets backing policy reserves.

(ii) The term *O&G* covers the time value of the financial *options and guarantees*. It is determined based on stochastic techniques with no closed form solution.

(iii) The *Risk Margin* (*RM*) is the cost of providing an amount of available financial resources equal to the solvency capital requirement, necessary to support the insurance and reinsurance obligations over the lifetime of those obligations. It is calculated with a 6% cost of capital rate, after tax and a 100% capitalisation of risk capital in line with Solvency II requirements.

(iv) The *value of new business* (*vnb*) is the additional value to the shareholder created through the activity of writing new business. It is defined as PVFP after acquisition expense over- or underrun, minus O&G, minus RM, all determined at issue date.

$$vnb = PVFP - O\&G - RM$$

(v) The *present value of (net) new business premiums* (*PVNBP*) is the value of future premiums on new business written during the year discounted at reference date. It is the present value of projected new regular premiums, plus the total amount of single premiums.

(vi) The *new business margin* (*nbm*) is the ratio of the value of new business to the present value of new business premiums.

$$nbm = \frac{vnb}{PVNBP}$$

3.2.2 Remark.

- The PVFP is the value of the company's profit (= income - cost) stream expected from policies already in force.
- The RM has replaced the calculation of the cost of non-hedgeable risks and is required for the financial, non-interest rate risk, insurance and operational risks that cannot be removed through capital markets.
- The nbm is the equivalent to value of sales in other industries. It is expressed as a percentage. The higher this figure, the more of each premium shareholders receive as profit.

- Most often the considered period for the vnb is a year. One has to keep in mind that in life insurance the first year normally is a loss-making period. This is among others due to acquisition costs and the long policy duration. Therefore projections of future cash in- and outflow are a bigger issue in life insurance than in other types of insurances. These and other cash flows have to be discounted to their value as of the date of valuation, called the *present value*.
- All these calculations take place longing to calculate the *Market Consistent Embedded Value* (*MCEV*). It represents shareholders' economic value of the in-force life and pension business of an insurance company. Future new business is not included. Allianz Group has decided to base and publish its MCEV results following a balance sheet approach, which is explicitly allowed in the MCEV principles (cf. [16]), using the Solvency II Market Value Balance Sheet. (cf. [1])

3.2.3 Definition. (cf. [10]) The *lines of business* (*lob*) in traditional life insurance used in this work are term life insurances, endowments and annuities. All can be participating or non-participating contracts. The former are usually the major component of the business in force and are expected to pay dividends (allocate bonuses) to the policy holder. These dividends (bonuses) are the so called *policyholder participation.*

- Term life contracts provide insurance over a specified time period. In general, term products do not have a nonforfeiture value and are written for relatively short periods, like 1, 4, or 15 years. They often have an automatic or optional renewal feature to extend the coverage for another period or offer the option to convert it into a whole life contract.
- Endowments are similar to whole life contracts. Additionally, they provide a maturity survival lump sum benefit, if the policy holder survives until the end of the coverage period.
- An annuity contract is an insurance policy promising to make a series of payments for a fixed period before death (e.g. three years) or over a person's lifetime (until time of death). The premium of an annuity contract can be paid in a one-time payment or periodically over a period of deferral before the annuity payments start.

Contracts can also be classified as follows:

- by the number of lifes covered
    - single life
    - joint life
    - joint and last survivor
- by the method of premium payment
    - single
    - fixed or flexible periodic
    - capital transfer from another policy or payment source
- if existing, by its funds investment strategy
    - in a general investment account of the company
    - in a separate pool account (variable annuities)

- by when payouts start or happen
    - · immediately upon payment of a single premium
    - · deferred to an age or date specified in the contract
    - · only in case of death
- by the nature of payouts
    - · options how annuity payouts are distributed
    - · whether a refund feature or a death benefit exists
    - · the duration and frequency of the benefit payout
- by tax treatment

3.2.4 Remark.

- Many parameters in actuarial models depend on the lines of business. As we will see, the data will naturally split up by the different lines.
- A common type of non-traditional life insurance is the *unit-linked insurance* (*ULI*). Another type would be the index-linked insurance with an according investment logic.
- ULIs are insurances with a speculative share. The saving component of the premium is invested in assets. Those can be real estate, bonds and stocks. Regardless of the actual interest income, in traditional insurances only the guaranteed actuarial interest is credited to the reserve. Policyholder participation is not affected by that.
  In contrast to that, ULI gives the opportunity to invest in assets with a higher risk-benefit margin. Gains and losses are forwarded to the policy holder every year. In an extreme case, the policy holder loses all contributions paid. Therefore often a balanced mix of conventional and unit-linked is used. (cf. [10])

A possible goal for Data Mining in actuarial data would be to gain more insight in the indicators leading to a high or low nbm. This is, what we are aiming at in Chapter 5.

3.3 Remarks on Selection of Data Mining Methods

Each Data Mining technique typically suits some problems better than others. Thus, there is no universal Data Mining method, and choosing a specific algorithm for a particular application is according to Fayyad et al. "something of art"[17]. In practice, a large portion of the application effort can go into properly formulating the problem (asking the right question) rather than into optimizing the algorithmic details of a particular Data Mining method. (cf. [17])

3.3.1 Example. Decision tree classifiers can be useful for finding structure in high-dimensional spaces and in problems with mixed continuous and categorical data (because tree methods do not require distance metrics). However, classification trees might not be suitable for problems where the true decision boundaries between classes are for example described by a second-order polynomial.

Data Mining methods can be categorized into four groups according to their properties. The classification can happen *supervised* or *unsupervised*. If the system is told how the clustering has to be performed it is supervised, otherwise not.

If no classes are known a priori, supervised classification is not possible. If the samples nevertheless fall in a finite set of categories, e.g. because of their similarity relations, we can classify them unsupervised.

On the other hand we can have a look at what the technique does with the input. If the output is fed back and used again it is a *feedback* method, else a *feedforward*. (cf. [43])

Feedforward networks are the most common ones as they are rather easy to learn. Backpropagation can be considered to be in this group. Feedback, also-called *recurrent*, networks are more complicated to implement and learn but better match the human brain.

3.3.2 Remark.

- In 2009 Grochowski and Włodzisław [19] discussed how neural networks frequently miss simple solutions that can be discovered by a more constrained learning method. Therefore, it is proposed to take a close look at the data before even preprocessing and afterwards again before training any model.

- The problem in comparing different Data Mining techniques was also discussed from the side of comparing neural networks to classic statistical techniques.
  Paliwal and Kumar [42] collected 73 studies talking about this issue. In 58% of the cases covered, feedforward neural networks outperformed the traditional method. One of the advantages of neural networks is their capability of automatically approximating any non-linear mathematical function. This aspect is particularly useful when the relationship between variables is complex or not known and hence difficult to handle in a statistical way. Finding the optimal configuration of neural networks is very time consuming though. The problem with the articles evaluated was the different handling of the two approaches (classical statistics versus neural networks.). The final neural network architectures used to train were obtained iteratively by trying various networks. Similar attempts have not been made with the traditional techniques. (cf. [42]) Thus, comparing and ranking them may not be possible.

There are still no established criteria for deciding which methods to use under which circumstances. Many of the approaches are based on crude heuristic approximations to avoid expensive search required to find optimal solutions.

We will use a feedforward neural network Data Mining method with unsupervised learning after applying regression and decision tree methods for pre-analysis and parameter selection. The data is well known by experts. This domain knowledge will help extensively to reduce calculation time and classification errors.

3.4 Review of Neural Networks

We will explain the basics of neural networks before discussing the one used in detail.

The neuron in an *artificial neural network* (*ANN*) emulates the behaviour of a human brain as far as it is known to work. Those neurons are connected within a complex network by their *synapses*. The relative strength or weakness of these connections can be modified. The neural net learns by a modifying process based on external stimuli. (cf. [70])

Relation between biological and artificial neural networks

Kohonen [32] stated some main characteristics relating artificial networks to biological ones:

- analogue representation and processing of information, allowing any degree of asynchronism of the processes in massively parallel computation without interlocking,
- ability to average conditionally over sets of data,
- fault tolerance and recovery from errors, which is a great survival value,
- adaptation to changing environment and emergence of information processing function by self-organisation in response to data.

Single Neurons

The basic building block of an ANN is the *artificial neuron*. The ANN is viewed as a structure that imitates the interconnected system of neurons in a human brain. Thereby the brain's components are replaced by hardware elements or simulated by using software. Depending on what the neural network has learned before, electric signals passed between neurons are either inhibited or enhanced. This is similar to the way in which the brain's neurons pass on electrochemical signals.

A biological neuron, as shown in Figure 3.4, has the following key elements:

- the *soma*, which is the central body of the neuron;
- the *axon*, which is attached to the soma, electrically active and produces the pulse emitted by the neuron; and
- the *dendrites*, which are electrically passive and receive inputs from other neurons by means of a specialised contact called a *synapse*.

In the artificial neuron, those three elements are simulated, see Figure 3.5.

3.4.1 Definition. (cf. [69]) Let $\mathcal{X} \subseteq \mathbb{R}^n$ for a positive integer n, $\mathcal{Y} \subseteq \mathbb{R}$ and mappings

$$s : \mathcal{X} \longrightarrow \mathbb{R} \quad \text{and} \quad \sigma : \mathbb{R} \longrightarrow \mathcal{Y}.$$

A data quadruple $(\mathcal{X}, \mathcal{Y}, \sigma, s)$ is then called *artificial neuron*. $\mathcal{X}$ and $\mathcal{Y}$ are called *input* and *output value sets* with n denoting the number of inputs. The mapping s is called *potential function* and σ the *output map*. The *transfer function* f is the composed input-to-output mapping

$$f = \sigma \circ s : \mathcal{X} \longrightarrow \mathcal{Y}.$$

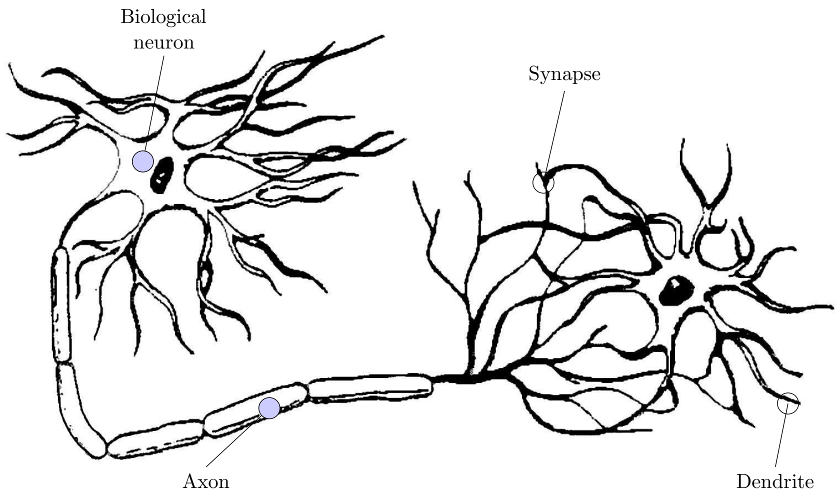


Figure 3.4: Biological neuron. The human brain has about $8.6{\cdot}10^{12}$ neurons and 10^{14} synapses. (cf. [70]) For comparison only: a honey bee has approximately $9.6 \cdot 10^5$ neurons (cf. [38]), an elephant about $2.6 \cdot 10^{13}$ (cf. [23]).

3.4.2 Definition. (cf. [69]) An artificial neuron $(\mathcal{X}, \mathcal{Y}, \sigma, s)$ is called *perceptron* or *McCullon-Pitts neuron* (cf. [37]) if

$$\begin{aligned} s(x) &= s(x_1, \dots, x_n) = \sum_{i=1}^{n} w_i x_i - \theta \\ \sigma(z) &= \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \end{aligned}$$

with *weights* $w_i \in \mathbb{R}$ and a *threshold* $\theta \in \mathbb{R}$.

3.4.3 Remark. Perceptrons are the simplest way to model a neuron. With their output mapping, only binary signals can be used and $\mathcal{Y} = \{0, 1\}$. The neuron only transmits any signal (fires), if the threshold is reached, i.e. $\sum_{i=1}^{n} w_i x_i > \theta$.

3.4.4 Example. Other examples for output mappings are:

- the Fermi function $\sigma(z) = \frac{1}{1+e^{-z}}$.
- the modified hyperbolic tangent $\sigma(z) = \frac{1}{2}(\tanh(z) + 1)$.

Input signals can be viewed to stem from receptors of connected neurons. Signals are modified at the synapses and condensed into a single signal at the soma. The neuron is activated only if the quantity delivered to the potential function surpasses a certain threshold. This threshold can vary from neuron to neuron.

The dendrites are simulated by weighting the inputs. The soma is going to be simulated by the potential function (s) and the axon by the output mapping (σ), bringing together weights, information and the activation threshold. The weighted inputs are processed by the neuron and the result is passed through an activation function.

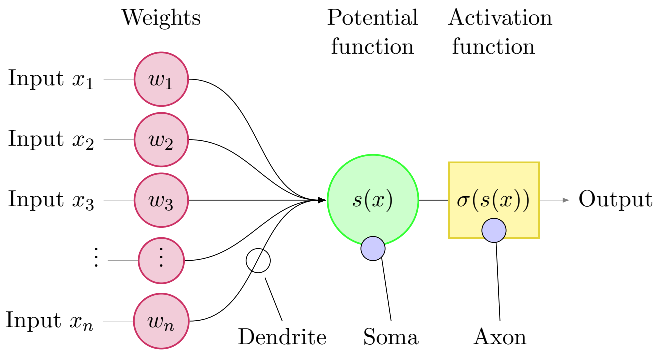


Figure 3.5: A single artificial neuron. Several artificial neurons with potentially different mappings and thresholds linked to each other compose a neural network architecture.

With a given distance function dist, the threshold $\theta = 0$, a matrix $W \in \mathbb{R}^{n\times m}$ for an integer m and $W_{.j} = (w_{1j}, \dots, w_{nj})$ we will use the following mapping for each neuron $j = 1, \dots, m$:

$$\begin{aligned} s_j(x_1,\dots,x_n) &= \underset{k=1\dots m}{argmin}(\text{dist}((x_1,\dots,x_n), W_{.k})) \\ \sigma_j(s_j(x_1,\dots,x_n)) &= \begin{cases} 1, & \text{if } j = s_j(x_1,\dots,x_n) \\ 0, & \text{if } j \neq s_j(x_1,\dots,x_n). \end{cases} \end{aligned}$$

Neural nets

A single artificial neuron has limited generality. It is often advantageous to link several neurons in order to increase the number of functions representable.

Mathematically a neural net resembles a graph. Feedforward nets are acyclic whereas feedback networks have at least one cyclic section.

3.4.5 Definition. (cf. [69]) Let $G = (V, E)$ be a (finite, simple, directed) acyclic graph with a partitioned V as in Remark 2.5.2. Let v_i be the cardinality of the direct predecessors $P(i)$ ($v_i := |P(i)|$), $\mathcal{X}_i \subseteq \mathbb{R}^{v_i}$, $\mathcal{Y}_i \subseteq \mathbb{R}$, $s_i : \mathcal{X}_i \longrightarrow \mathbb{R}$ and $\sigma_i : \mathbb{R} \longrightarrow \mathcal{Y}_i$. A *feedforward network* is composed of the graph G and a family of neurons $(\mathcal{X}_i, \mathcal{Y}_i, \sigma_i, s_i)$, each associated to one of the non-source vertices $i \in V \setminus V_0$. The source vertices $i \in V_0$ consist of the input parameters.

3.4.6 Definition. (cf. [69]) A feedforward network with

$$V = V_0 \,\dot\cup\, V_1 \,\dot\cup\, V_2 \,\dot\cup\, \dots \,\dot\cup\, V_k \,\dot\cup\, V_{k+1}$$

like in (2.5.1) is a *k-layered feedforward network* for some integer $k \geq 0$ if

$$i \in V_j \Rightarrow P(i) \subseteq V_{j-1} \quad \text{and} \quad S(i) \subseteq V_{j+1}$$

for $j = 0, \dots, k+1$, setting $V_{-1} = V_{k+2} = \emptyset$.

3.4.7 Definition. (cf. [69]) A *recurrent* or *feedback network* consists of a (finite, simple, directed) graph $G = (V, E)$ with source-vertices V_0 and a family of artificial neurons $(\mathcal{X}_i, \mathcal{Y}_i, \sigma_i, s_i)$ each associated to one of the non-source vertices $i \in V \setminus V_0$. For some set $Q \subseteq \mathbb{R}$, let $\mathcal{X}_i = Q^{\upsilon_i}$ and $\mathcal{Y}_i = Q$ for all i with $\upsilon_i = |P(i)|$. Each neuron then has a transfer function $f_i : Q^{\upsilon_i} \longrightarrow Q$. For $\upsilon = |V \setminus V_0|$, an integer t and $i = 1, \dots, \upsilon$ we set

$$q_i^{(t+1)} = f_i((q_j^{(t)})_{j \in P(i)}).$$

We then call $q_i^{(t)} \in Q$ the *state of neuron* i at time t. The vector $q^{(t)} = (q_1^{(t)}, q_2^{(t)}, \dots, q_{|V|}^{(t)})$ is called the *state of the network* at time $t \geq 0$ with initial starting value $q^{(0)}$.

3.4.8 Definition. (cf. [69]) Consider a feedforward network, V partitioned as in Remark 2.5.2 and the case of full interconnection, i.e.

$$i \in V_j \Rightarrow P(i) = V_{j-1} \text{ and } S(i) = V_{j+1},$$

for $j = 0, \dots, k+1$. We call V_0 the *input layer*, V_{k+1} the *output layer* and all remaining $V \setminus (V_0 \cup V_{k+1})$ *hidden layer*. The associated neurons are called hidden neurons and output neurons, respectively.

3.4.9 Remark. With linking several artificial neurons the calculation capacity of the net rises. Although every neuron has an output of $\mathcal{Y} \subseteq \mathbb{R}$ the whole net can have an output layer in $\mathbb{R}^m$ with an integer $m > 0$. We will use a neural net with no hidden layer. Our input consists of $n > 0$ nodes where n is the dimension of the input vector as well as the number of nodes in V_0 ($n = |V_0|$). Let $(x_1, \dots x_n) = X \in \chi$ be the input vector with n variables for the neuron to take into account. $\chi \subseteq \mathbb{R}^n$ shall denote the dataset in use, consisting of a finite amount of data vectors $X \in \mathbb{R}^n$, see Definition 4.1.2.

Teaching a network

There are two fundamental approaches to train a neural network: *supervised* and *unsupervised*. They coincide with the classic statistical approach of supervision. The core difference is that the supervised learning algorithm needs a desired solution to be known a priori, whereas an unsupervised training algorithm does not. With a random initial connection, so to say knowledge, a supervised algorithm will iteratively modify weights until the desired solution is achieved. An example for this is explained in Section 4.1, where we get deeper into a particular kind of neural network. In contrast, unsupervised training algorithms allow the neurons to compete with each other until winners emerge. The resulting values of the neurons determine the class to which a particular dataset belongs.

3.4.10 Example.

- Marcic and Ribari [35] compared backpropagation to the use of SOM, both being neural models, for the classification of multi spectral remote-sensing images. They concluded that for the experiments performed, backpropagation is "a little superior"[35] to SOM. The major drawback of backpropagation is the high learning time needed when using large samples. By using SOM, their experiments have shown the possibility of overcoming this drawback. They concluded that the SOM method could provide a successful discriminate between different spectral classes, which was their experiment's aim.

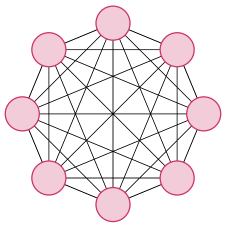
- The *Hopfield network* is a recurrent network where all nodes are attached to each other and work as input nodes and output neurons (see Figure 3.6a). The network is trained by setting the value of the nodes to a desired pattern after which the weights are computed. The weights do not change after this. Once trained, the network will always converge to one of the learned patterns because the network is stable only in those states. (cf. [25])

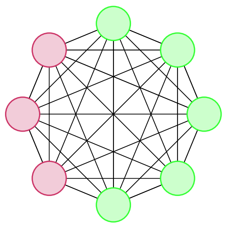
- *Boltzmann machines* are similar to Hopfield networks with all neurons attached to each other. Here, some nodes are marked as input (first use) and others are only used for calculations and hidden (see Figure 3.6b). The input nodes become the output at the end of a full network update. It starts with random weights and learns through backpropagation or contrastive divergence. Cells can get any value and we repetitively go back and forth between the input and the hidden nodes. The activation is controlled by a global temperature value, which - if lowered - diminishes the energy of the cells. This decreased energy causes their activation patterns to stabilise. See [24] for more information.

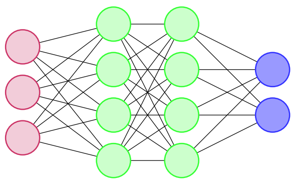
- The so-called deep learning is often performed by *deep believe networks* with one input layer, one output layer and several hidden layers (see Figure 3.6c). These networks have been shown to be effectively trainable stack by stack. So every hidden layer has to learn how to encode the previous layer. This means several small networks are arranged one after the other. Once trained or converged to a (more stable) state through unsupervised learning, the model can be used to generate new data. See the original paper [4] and [57] for a presentation about generating text with a network like this.

- A SOM does not have hidden layers. Every node in the input layer is directly connected to all neurons in the output layer. It learns through a learning rule derived from the-winner-takes-it-all principle (see Section 4.1). In addition to this, the neurons in the output layer are arranged on an one or more dimensional grid (see Figure 3.6d). This is the network's peculiarity. Through this structure, the output can be visually analysed. Furthermore, SOMs are a particularly robust form of unsupervised neural networks (cf. [65]). We will talk about it more in Chapter 4.

We will use Self-Organizing Maps as a clustering and visualisation Data Mining technique.


(a) Hopfield Network. Every neuron is input and output.


(b) Boltzman machine. Some neurons are hidden.


(c) Deep Believe Network with more hidden layers.

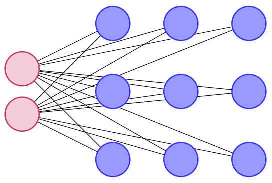

(d) Self-Organizing Map. Output neurons are arranged in a lattice.

Figure 3.6: Sketches of examples of ANNs. In (a) and (b) the input nodes (red) act as well as output. Hidden layers are coloured green, output neurons blue.

Chapter 4

Self-Organizing Maps

Self-Organizing Maps (*SOM*s, also referred to as *Kohonen Maps*) are used to create an ordered representation of multi-dimensional data which simplifies complexity and reveals meaningful relationships. Since their introduction by Prof. Teuvo Kohonen in the early 1980s in [31], those have been the technological basis of many applications as well as the subject of many thousands of publications.

4.0.11 Example. A with no means exhaustive list of already used applications is the following:

- Image coding and compression, e.g. in [34]
- Medical imaging and analysis, e.g. in [6]
- Speech analysis and recognition, e.g. in [40]
- Clone detection in telecommunications software systems, e.g. in [2]
- Prevention of cyber-attacks, e.g. in [22]
- Robotics including collision-free navigation, e.g. in [21]
- Full-text analysis, e.g. in [11]
- Fuzzy logic analysis, e.g. in [5]
- The Travelling-Salesman Problem, e.g. in [28]
- Data processing and analysis

4.0.12 Remark. The very last point of this list is the topic we will talk about. Using SOMs for data processing is useful in profiling customers, computer systems performance tuning, user identification, forecasting in information systems and, most important for us, exploratory analysis of financial and economic data.

To explain how SOMs are working, we take a step back and look at Learning Vector Quantisation.

4.1 Learning Vector Quantisation

Learning Vector Quantisation (*LVQ*) is a neural network which can be seen as a precursor to the SOM. It sorts a bundle of vectors into several classes, partitioning the data. After learning from a training dataset with given classes, similar vectors of unknown class can be assigned within the learned classification scheme. (cf. [52])

4.1.1 Remark. For a simple classification of e.g. vectors according to their length or direction, no neural network is needed. It gets interesting, when the data is more complicated and although for some data a classification may be given, no explicit projection algorithm (e.g. vectors according to length or customer according to age and residence) is known. The only purpose for stating the LVQ in this work is to describe the principle of the-winner-takes-it-all, which will be used by the SOM.

Notation

4.1.2 Definition. (Notation) Consider a feedforward network with n inputs (V_0), m output neurons and no hidden neurons. Suppose the threshold θ is zero. The output neurons are grouped into $k \leq m$ classes $c \in C = \{c_1, \dots, c_k\}$. Let $X = (x_1, x_2, \dots, x_n) \in \chi$ be an *input vector* or *train vector* from an available input or train dataset $\chi = \{X_1, \dots, X_s\} \subseteq \mathbb{R}^n$.

Furthermore for training, we need

$$W = (w_{ij})_{i=1\dots n, j=1\dots m} \dots \text{the initialised } \textit{weight matrix}.$$

An *iteration* t is the period of using every available input vector $X \in \chi$ from the training set once. A training *step* is a part of the training process, when one input vector is passed to the network and weight vectors are modified. When having s vectors X, an iteration consists of s steps. We call the amount of iterations T.

4.1.3 Remark.

- As $k \leq m$ we have less or equal classes than output neurons. Every class may be represented by several neurons in the output layer. Vertices in the output layer therefore can but do not have to be bundled in clusters.
- With the notation of Definition 2.5.1 each entry of $X \in \chi$ is given to a vertex in the input layer V_0. The output layer is V_1. Every neuron from the input layer is fully connected to every node in the output layer via the algorithm used.

4.1.4 Remark. The columns of the weight matrix are the so-called *reference- or codebook-vectors*. They should cover the space spanned by the input vectors and the individual classes as evenly as possible. These m codebook-vectors have n components. Each codebook-vector is assigned to exactly one neuron from the output layer. Our goal is to find representative vectors so each X is quantised into a particular neuron. This classifying or clustering is done by trying to find an encoding scheme which is a mapping from X to a so-called codeword in the columns of W such that the average distortion is minimized. The distance or distortion measure has to be "chosen appropriately according to specific applications" [27].

4.1.5 Remark. A concept for illustration of the vector quantisation method in pattern recognition and for neural networks in general is the *Voronoi tessellation* [32]. Figure 4.1 exemplifies a two-dimensional space where a finite number of codebook-vectors is shown as points, corresponding to their coordinates. This space is partitioned into regions, bordered by lines that would be in general hyperplanes. The borders are chosen in a way that each partition contains a reference-vector that is, in the sense of a given distance measure, a nearest neighbour to any vector within the same partition. These lines of the neighbouring reference-vectors together constitute the Voronoi tessellation. All vectors in the corresponding partition of the Voronoi tessellation are said to constitute the *Voronoi set*.

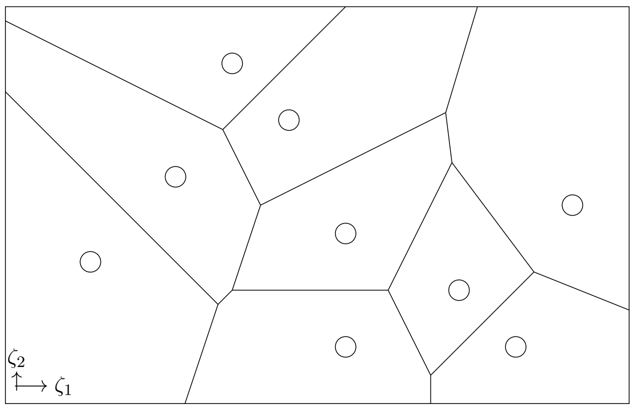


Figure 4.1: Voronoi tessellation partitioning a two-dimensional (ζ_1, ζ_2) space into regions around reference-vectors, shown as points in this coordinate system. All vectors (ζ_1, ζ_2) in the same partition have the same reference-vectors as their nearest neighbour. (cf. [32], [56])

Algorithm

With the notation from above we search for the most similar column vector in the weight matrix W to the input or train vector X.

4.1.6 Definition. Let $\operatorname{dist}(x, y)$ be a given distance function working on $\mathbb{R}^n$. Calling the neuron by its number ($\in 1, \ldots, m$), we say ℓ is the *winning unit*, if it satisfies

$$\ell = \underset{j=1\ldots m}{argmin}(dist(X, W_{.j})). \tag{4.1.1}$$

4.1.7 Remark.

- Using Definition 3.4.1 the net's winning neuron ℓ for the input X is calculated using for every neuron $j \in 1, \ldots, m$ the mappings

$$\begin{aligned} s_j(X) &= \underset{k=1\ldots m}{argmin}(\operatorname{dist}(X, W_{.k})) \\ \sigma_j(s_j(X)) &= \begin{cases} 1, & \text{if } j = s_j(X) \\ 0, & \text{if } j \neq s_j(X). \end{cases} \\ f_j(X) &= \sigma_j(s_j(X)) \in \{0, 1\} = \mathcal{Y}_j \\ \ell(X) &= \sum_{j=1}^{m} j \cdot f_j(X) \end{aligned}$$

In case of two or more outputs of the argmin function one of the results is chosen arbitrarily.

- This means the winning unit is the neuron connected to the weight matrix' column with the minimal distance to X.
- See Figure 4.2 for a notational overview. We set $k = m$ in this Figure.
- Figure 4.3 shows an example of a two-dimensional grid of nodes usable as a rectangular ordered map.

$$\begin{array}{cccccccc}
 & & & & \ell & & & \\
 & & & & \| & & & \\
 & \Bigg\{ & c_1 & \cdots & c_\ell & \cdots & c_{k=m} & \Bigg\} = C \\
X & & & & & & & \\
\| & & & & & & & \\
\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} & & \multicolumn{5}{c}{\begin{pmatrix} w_{11} & \cdots & w_{1\ell} & \cdots & w_{1m} \\ w_{21} & \cdots & w_{2\ell} & \cdots & w_{2m} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{n\ell} & \cdots & w_{nm} \end{pmatrix}} & = W \\
 & & & & \| & & & \\
 & & & & \multicolumn{1}{c}{codebook - vector} & & &
\end{array}$$

Figure 4.2: Overview of the notation in Definition 4.1.2 and Remark 4.1.7

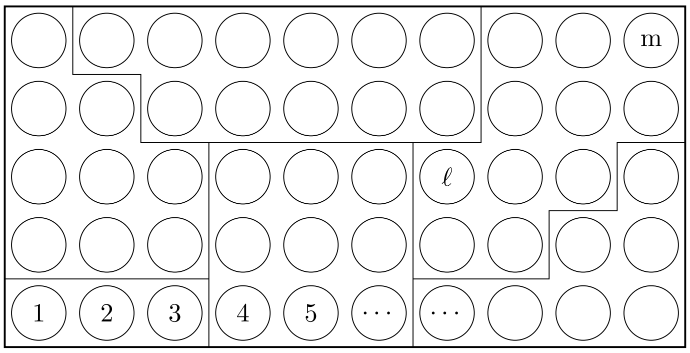


Figure 4.3: Basic understanding of the output layer with m nodes, $k = 6$ clusters and the current winning unit ℓ.

For the winning unit, the respective column in the weight matrix W will be modified. Depending upon whether or not the class is correct the conferring column is getting closer to or further away from the train vector. Therefore the class affiliation of the train vector has to be known in the LVQ learning process.

4.1.8 Definition. Let ℓ be the winning unit at step t with the least distance between the (currently used) vector X and every column in the weight matrix W. *Modifying the weight matrix*'s column in the LVQ is defined as

$$W_{.\ell}^{(t+1)} = \begin{cases} W_{.\ell}^{(t)} + \alpha \cdot (X - W_{.\ell}^{(t)}), & \text{if } \ell \text{ and X are in the same class.} \\ W_{.\ell}^{(t)} - \alpha \cdot (X - W_{.\ell}^{(t)}), & \text{if } \ell \text{ and X are not in the same class.} \end{cases}$$

The parameter $\alpha = \alpha(t, T) > 0$ is called *learning rate* and decreases with rising iteration.

4.1.9 Remark. Depending on the problem it may be possible to have a decreasing learning rate by step and iteration. In this case t in Definition 4.1.8 would be a counter of both.

This algorithm of choosing a train vector $X \in \chi$, finding the winning unit and adjusting its corresponding weight vector is repeated until convergence or an individually defined abort criterion is reached.

After the learning process the net can be fed with an unclassified vector. The net then will sort this input into the neuron with the most similar associated weight vector and find the class by its own.

4.1.10 Remark.

- LVQ is, just as SOMs, a feedforward neural net. We need the iterations to use different input to train the net. The output neurons are not fed back to the network.
- The set of codebook-vectors obtained in the way above is not yet ordered. This means the codebook-vectors are not able to reflect any structures of the data. Concerning data organising and retrieval this is the main disadvantage of *vector quantisation* (*VQ*).
- Another principal characteristic weakness: the final state may depend, even significantly, on the initialisation of the codebook-vectors in the above updating process.
- As we will see, a SOM is a LVQ with a constraint on spatial ordering.

4.2 Kohonen Maps

Kohonen Maps or Self-Organizing Maps (SOMs) are further developments of the LVQ process.

4.2.1 Remark. It is worth remembering that the principle of SOMs resembles the biological function of sensory perception and processing in the brain. Typically multidimensional stimuli are projected into planar areas, i.e. two-dimensional spaces, in the cortex. Similar stimuli activate neighbouring neurons in the cortex-area. The high-dimensional space of input signals is projected onto a two-dimensional map in a way, that neighbourhood relations in the signalling area broadly survive. Typical applications of Kohonen Maps are therefore processes in need for mapping a high-dimensional signal to a lower-dimensional space.

4.2.2 Definition. When talking about SOMs, the output layer is called *competitive layer*. Its neurons are arranged in a lattice with a lower dimensionality than the input value space χ.

4.2.3 Remark.

- Typically the grid used within SOMs is two-dimensional as seen in Figure 3.6d. For some utilisations a one- or three-dimensional grids may also be useful. In this work we will focus on two-dimensional grids.
- In the grid, a metric has to be defined. Although every metric can be chosen, the Euclidean is used predominantly. As well as the metric, grid size depends upon application. Once a grid is chosen, it will not be changed.
- In $\mathbb{R}^2$ quadratic, hexagonal or rectangular arrangements are primarily used (cf. [52]).
- Every neuron is uniquely defined through its (one-, two- or three-dimensional) coordinates in the grid. In our case, indices of all nodes would be a two-dimensional set. We will refer to an output neuron by its number, like in Figure 4.3, not its place in the grid to stay independent from the topology and dimensions of the grid.

SOMs learn unsupervised with the principle of the-winner-takes-it-all, already explained in Section 4.1. The neurons in the output layer are competing for the highest stimuli. Training the net means, as in LVQ, using every input vector iteratively and mostly randomly. After finding the winning unit we adjust the weights of this neuron and those in an appropriate radius or neighbourhood as well.

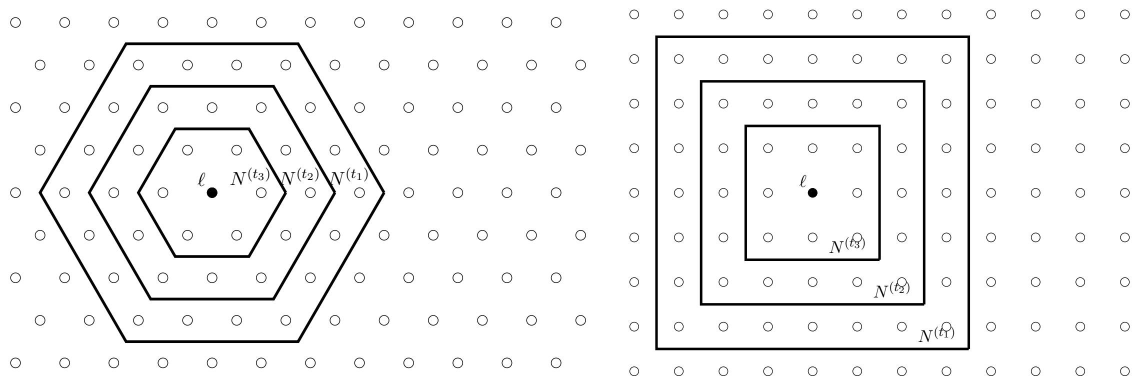
4.2.4 Definition. With a winning unit ℓ we call N_ℓ the set of neighbouring nodes around the node.

4.2.5 Definition. With a distance $\widehat{\text{dist}}(\ell, j)$ between a winning unit ℓ and a neuron j we say the neuron j is currently, at time t, in the *neighbourhood* N_ℓ of ℓ, if the distance is smaller than a given threshold $\rho^{(t)}$. In other words:

$$j \in N_\ell \Leftrightarrow \widehat{\text{dist}}(\ell, j) \leq \rho^{(t)}.$$

4.2.6 Remark.

- In Definition 4.2.4 no definition of neighbourhood is stated. It is dependent of the nets topology and the distance measure induced by its topology. Definition 4.2.5 already uses the topology we will use in our application.
- It is essential that initially N_j is very wide, covering for example half of the network, whereas at the end of the process N_j only contains the closest neighbourhood cells. Due to this condition it is possible to produce a globally ordered map. An example of a decreasing neighbourhood can be seen in Figure 4.4. If N_j would contain only the unit j, the map would be degenerated into a disordered zero-order topology model. (cf. [29])
- There are two different distances to be considered: On the one hand the metric in the grid, which defines the topological neighbourhood of the winning unit ($\widehat{\text{dist}}$); on the other hand the distance between the input vector X and the weight vector (dist). Every function satisfying Definition 2.1.1 can be used for both distance relations, always matching the need in applications.



(a) A grid with hexagonal arrangement. (b) A grid with quadratic arrangement.

Figure 4.4: A decreasing (with Euclidean distance measure) circular neighbourhood on a two-dimensional, (a) hexagonal and (b) quadratic grid arrangement.

Until a predefined condition is fulfilled, the training will be pursued. The aim of the algorithm is to correlate certain output neurons with a particular area in the input space. This is accomplished by activating the winning units neighbourhood in every training-step as well.

Under the condition of a convergent net, clusters of similarity occur between the vectors. In contrast to the LVQ, where the classes are given by the train vectors, the SOM autonomously generates a clustering for multidimensional vectors or data. Therefore, the topology stays preserved under the algorithm of Kohonen Maps.

Algorithm

(1) Initialise the weight matrix $W = (w_{ij})_{i=1\ldots n, j=1\ldots m}$.

(2) Choose a train vector $X = (x_1, \ldots, x_n)$ randomly from the input space χ.

(3) Determine the winning unit ℓ with the least distance between the weight vector $W_{\cdot\ell} = w_{\cdot\ell}$ and the input vector X. This is done by activating the neuron via the mapping functions in Remark 4.1.7.

(4) Adjust the weight vector W for all neurons j in the neighbourhood of winning unit ℓ:

$$W_{\cdot j}^{(t+1)} = W_{\cdot j}^{(t)} + \alpha \cdot \mathrm{h}_{\ell j}\,(t) \cdot \mathrm{dist}(X, W_{\cdot j}^{(t)})$$

with

$\alpha = \alpha(t) > 0 \ldots$ a monotonously decreasing learning rate,

$\mathrm{h}_{\ell j}(t) \ldots$ a monotonously, in dependence of the distance to the winning unit decreasing function and

$\mathrm{dist}(X, W_{\cdot j}^{(t)}) \ldots$ the distance already used for finding the winning unit.

(5) Iterating step (2) to (4) until the net stabilises. $(t = t + 1)$

4.2.7 Remark.

ad (1) The initial entries for W can be selected as random values. According to Kohonen [32] a much faster ordering and convergence follows if the initial values are selected as a regular, two-dimensional sequence of vectors taken along a subspace spanned by the two largest principal components of the input data vectors. (cf. [33], [32])

ad (2) Via this randomness the choice is subject to a given probability distribution. X can therefore as well be seen as a random vector.

ad (3) The winning unit is determined in the same way as in LVQ, see the algorithm on Page 30.

ad (4) The so-called *neighbouring function* $\mathrm{h}_{\ell j}(t)$ influences the training result. The definition whether a neuron is in the neighbourhood of another depends on the topology of the net and users' preferences. Hereby, the weight vectors initially do not have to be similar. In general the function $\mathrm{h}_{\ell j}(t)$ is monotonously decreasing depending on the iteration t and the distance between the winning unit ℓ and the neuron j for which the weight vector should be changed. The distance function used for defining the distance between the neurons does not have to be the same as for finding the winning unit.

ad (5) For grids with a few hundred nodes, as we will have in our application, Kohonen [33] supposes the radius of the final neighbourhood function to be one grid spacing. So the number of final convergence steps may be roughly 100 to 500 times the number of nodes, totalling to a hundred thousand recursive steps for a final convergence. The training should end up with no weights of winning node's neighbours being updated.

Examples

4.2.8 Example. (Learning rates, cf. [55], [36]) Possible learning rates, as used for instance in the MATLAB SOM Toolbox [36], would be linear (4.2.1), inverse-of-time (4.2.2) or power series (4.2.3) functions. A heuristic function (4.2.4) is also discussed. Here T is the number of iterations or steps and t is the order number of the current iteration or step, respectively.

$$\alpha(t) = 1 - \frac{t}{T} \tag{4.2.1}$$

$$\alpha(t,T) = \frac{1}{t} \tag{4.2.2}$$

$$\alpha(t,T) = (0.005)^{\frac{t}{T}} \tag{4.2.3}$$

$$\alpha(t,T) = \max\left(\frac{T+1-t}{T}, 0.01\right) \tag{4.2.4}$$

4.2.9 Example. (Neighbourhood functions, cf. [55]) Bubble (4.2.5) or Gaussian functions (4.2.6) are usually used as neighbourhood function.

$$\mathrm{h}_{\ell j}(t) = \begin{cases} \alpha(t), & \text{if } \widehat{\mathrm{dist}}(\ell, j) \le \rho^{(t)}, \text{ i.e. } j \in N_\ell \\ 0, & \text{else} \end{cases} \tag{4.2.5}$$

$$\mathrm{h}_{\ell j}(t) = \alpha(t) \cdot \exp\left(\frac{-\widehat{\mathrm{dist}}(\ell, j)}{2\sigma^{(t)}}\right) \tag{4.2.6}$$

where

$\widehat{\text{dist}}(\ell, j)$ defines the distance between the winning neuron ℓ and the neuron j,

ρ denotes a in t monotonically decreasing parameter coding the borders of a neighbourhood, like a radius and

$\alpha(t)$ and $\sigma^{(t)} \geq 0$ are monotonically decreasing functions with α being one of the above learning rates.

4.2.10 Remark. Stefanovič and Kurasova [55] discuss how the choice of neighbouring functions and learning rates influences the result in the SOM. They evaluated the quantisation error, see Definition 4.3.3, to show how well the neurons of the trained network adapt to the input vectors. As the quantisation error is the average distance between the data vectors in χ and their winner-neurons ℓ, this valuation is vivid too. They concluded that the smallest quantisation error is achieved with inverse-of-time, power series or heuristic learning rates.

4.2.11 Example. (cf. [70]) An example on how a SOM classifies fruits based on their characteristics is shown in Figure 4.5. Here, three features are encoded and a maximum of five different classifications will be learned.

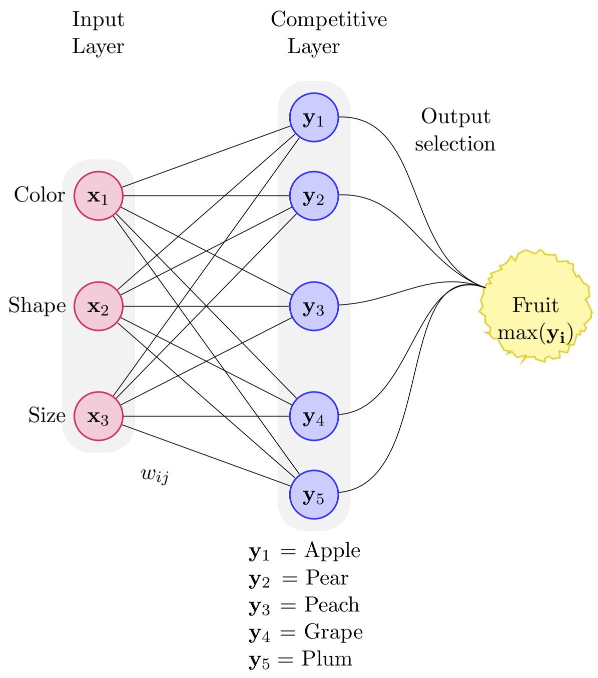


Figure 4.5: Example on how a Self-Organizing Map with a one dimensional grid can assign fruits.

With the above notation: $n = 3$, $m = 5$ and $k = 5$. The activation level of each neuron is calculated and the neuron associated to the weight matrix vector having the most similar properties is found. The output is the neuron number that corresponds to a specific category. For example with a purple colour, medium size and round shape the entries in the weight matrix for plum is supposedly close all three times, whereas grape is only suitable for two attributes. On the other hand it can make sense to consider white and red grapes, leading to $m = 6$. Therefore, they would have two nodes, both grouping the fruit in question to grapes.

4.3 Validation

The disadvantage of SOMs is that, as with other neural network techniques, it is not simple to explain why the trained network produces a particular result. According to [58], it is even less straightforward in the case of SOM to quantify a network. This is because the data clustered by the map have no a priori classification to measure a classification performance. The resulting map is totally dependent on the data and learning parameters. In this section we will investigate several ways to identify working maps.

U-matrix

The method of *Unified Distance Matrix* or *U-matrix* was developed to detect non-linearities in resulting mappings. The basic idea is to use the same metric that was used during a learning process to compute distances between adjacent reference-vectors. This method can be used to visualise topological structure of the SOM output layer and therefore also the topology in the input space.

4.3.1 Definition. (cf. [63]) Let c_{ij} denote the neurons in a rectangular lattice $C \in \mathbb{R}^{x \times y}$. For each c_{ij} four different types of distances to its immediate neighbours and one additional (dz) for the diagonal neighbours can be defined:

$$\begin{aligned}
dx(i,j) &= \operatorname{dist}(c_{ij}, c_{ij+1}) \\
dy(i,j) &= \operatorname{dist}(c_{ij}, c_{i+1j}) \\
dxy(i,j) &= \operatorname{dist}(c_{ij}, c_{i+1j+1}) \\
dyx(i,j) &= \operatorname{dist}(c_{ij+1}, c_{i+1j}) \\
dz(i,j) &= 0.5 \cdot (dxy(i,j) + dyx(i,j))
\end{aligned}$$

With an arbitrarily chosen $dc(i,j)$ the following $(2x-1) \times (2y-1)$-dimensional diagram is called *unified distance matrix* or *U-matrix* U:

index	...	$2j-1$	$2j$	$2j+1$	...
$2i-1$	...	$dz(i-1,j-1)$	$dy(i-1,j)$	$dz(i-1,j)$	...
$2i$	...	$dx(i,j-1)$	$dc(i,j)$	$dx(i,j)$	...
$2i+1$	...	$dz(i,j-1)$	$dy(i,j)$	$dz(i,j)$	...

4.3.2 Remark.

- Figure 4.6 shows a 2×2 grid with a quadratic arrangement and the corresponding 3×3 U-matrix with the defined distances of Definition 4.3.1 sketched.
- The definition of the U-matrix in a hexagonal, two-dimensional lattice is analogue. It arises even easier as the definition of a cross-distance like dz is needless. See Figure 4.7 for a visualisation.
- As a result of its definition the U-matrix contains a geometrical correct approximation of the vector distribution in the net. In order to apply a vector metric for all components, the range of the components has to be the same, otherwise the entries with larger absolute range dominate the vector metric.
- For visualisation we use the U-matrix to capture the relative distances between the map units: the lighter the colour, the farther away (higher distance) unit weight vectors are

from each other. Separate clusters in the data therefore have a lighter band between them on the map. Visually, this resembles a mountain range separating two plateaus on a geographic map.

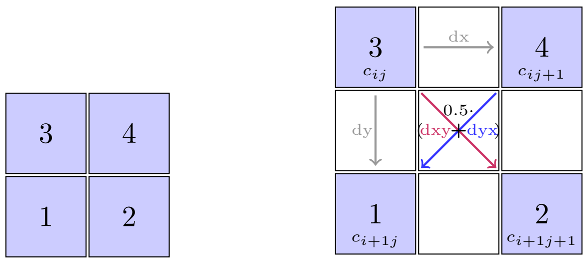


Figure 4.6: Structure of a U-matrix with the distances from Definition 4.3.1 added. Every neuron is labelled by numbers and its name according to the definition ($c \in C$).

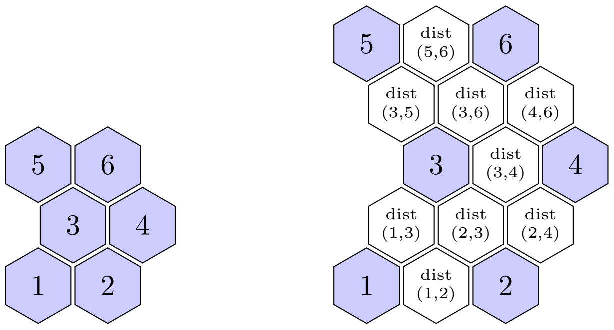


Figure 4.7: Structure of a U-matrix and its original 2×3 lattice with hexagonal arrangement. Distances in the U-matrix are added. The nodes are numbered for a better understanding.

Quantisation and Topological Error

To measure map quality, the accuracy and order of the map can be a criteria. The accuracy can be measured by the quantisation error. An example of order measure is the topographic error.

The quantisation error is a measure of how well the inputs are represented in the output space. One possible optimal selection of the winning node is to minimize the average expected square of the quantisation error.

4.3.3 Definition. (cf. [32], [58]) Remembering the winning unit ℓ being a function of the input vector $X \in \chi$ and all other units, the *quantisation error* for our application can be written as

$$E_q = \frac{1}{s}\sum_{i=1}^{s} dist(X_i, W_{\cdot\ell}).$$

4.3.4 Definition. (cf. [58], [30]) The *topographic error* E_t of a map is defined as

$$E_t = \frac{1}{n} \sum_{i=1}^{n} \delta(X_i)$$

with

$$\delta(X_i) = \begin{cases} 0, & \text{if best and second best unit } (\ell \text{ and } \zeta \text{ respectively}) \text{ are adjacent.} \\ 1, & \text{otherwise.} \end{cases}$$

4.3.5 Remark. Definition 4.3.4 arises from the following idea: The units having the best and the second best matching weight vectors, $W_{\cdot\ell}$ and $W_{\cdot\zeta}$ respectively, for input data X on the map have to be adjacent to indicate continuity of the mapping. Consequently some points between X and $W_{\cdot\zeta}$ in the input are mapped to $W_{\cdot\ell}$ and the remaining to $W_{\cdot\zeta}$.

Chapter 5

Application

In this section the application of the previous theoretical chapters is stated. We will go through the process of preprocessing the data with experts' knowledge, classic statistical measures such as decision trees and deleting not-usable data. The neural network of a SOM will then be applied in `R` using the pre-implemented `som`-function. The calculations are performed with `R` version 3.2.1 [47], on a 64 Bit Windows 7 Professional operating system with 16GB RAM. The set-up will be explained as well as the chosen visualisation and groupings.

5.1 Objectives

The new business margin, already explained in Section 3.2, is an important indicator for the performance of life insurance products. As the margin is the fraction between the vnb and the PVNBP we would like to focus on how the values of nbm and vnb are dependent on the other parameters. New information shall be gathered using the benefits from extensive analysis already made before using neural networks. A focus is on finding patterns and clusters in the data leading to a more focused acquisition strategy for new customers in the future. The other main issue is the selection of the best visualisation method for the SOM used.

5.2 Preprocessing

Data in use

The input data in use is an information table about newly entered contracts in a traditional, non-ULI life insurance product (i.d. endowments, annuities or term life insurances). It consists of 9083 information vectors taken from an internal data warehouse with the effective date of June 30$^{\text{th}}$, 2016. This is relevant and system-representative data as this month is assumed to be non-extreme, having average indicator values.

Each of the vectors representing a new contract consists of 128 variables. This raw data is presented as a mix of factorial, symbolic and numeric values on an individual contract basis. No personal information can be extracted from the outcomes in this work. Personal data is either deleted or encoded during the preprocessing treatment.

In addition to the dataset, domain knowledge was the main input. Specialists working in the life actuarial department for several years gave advice concerning the insurance based data. During the process it got clear that maintaining a historically grown data warehouse is difficult and may have been upstaged for a longer period. Therefore, not all data could be used or interpreted as objected.

Selection

Selecting the data scope in relation to the objective of a study is important for any kind of data analysis. Neural networks are not different to other analysis tools in this perspective. Deboeck [13] warns of being lazy using all available data on a particular subject rather than a selective set relevant to the objectives.

Starting with the above mentioned raw data of 128 variables we begin to delete columns with all different entries, e.g. position keys and byte markers for the internal data warehousing. A main key for every information vector will stay, so that the connection between the SOM and the data table can be reviewed. Some variables also have the same entry for each of the vectors, meaning they are either only used as placeholders for future usage or happen to be the same because we only use the slice of non-ULI life insurance in our analysis. With clearing these data, we end up with 71 variables to start our preprocessing.

Roughly speaking we have two kinds of data in the raw state:

(*i*) data to delete, consisting of e.g. keys, variables with all identical entries, personal data and groupings made up by other variables and

(*ii*) data to keep, having symbolic, semi-symbolic and numeric entries.

5.2.1 Remark. The maximum available set χ consists of 9083 vectors with 71 entries.

Transformation

In this case semi-symbolic means, that up to a certain point the values are actual numeric but have extreme values to encode some information. E.g. annuity payments may be declared for a life-long period, which is coded with a symbolic 9999 while those declared only for several years are represented by the actual number of months. To avoid deformation from these values we overwrite them to match current life expectancy values. This will not interfere with other variables since the original symbolic value would never be reached.

After having taken care of symbolic and factor-valued variables we end up with an all numeric field of non-scaled entries.

Experience of e.g. Kohonen [33] has shown that a rather good first strategy is to scale every feature in such a way that the variance of every variable becomes the same. Alternatively, one can adjust the range from minimum to maximum of every variable to the same values. This is necessary because the data range of each feature varies from column to column. When an ordered SOM has been constructed, it can be used directly for statistical classification of the input items. (cf. [33])

If no preprocessing is applied it may influence the clustering and the ultimate shape of a SOM map. For this reason we apply a widely used approach (cf. [13]) and standardize all data based on the standard deviation after centring.

Pre-Analyses

With the use of conditional inference trees we avoid having a bias towards high variance values. A standard regression tree was grown as well as using the `rpart`-Package (cf. [61]) which showed this phenomenon. Leading to an over-interpretation, the regression tree grants more importance to variables with a higher variance.

The conditional inference trees grown had a p-value lower than 0.001 in 96.5% of splitting points. The remaining still had a p-value below 0.04 and appeared only in end nodes, which had to contain at least 100 input vectors to remain interpretable. Boxplots were used to analyse these nodes with respect to homogeneity. This led to a first expectation towards certain groupings.

With the information drawn from the trees, parameters were chosen to be fed into the neural net.

5.3 Map Tuning

Kangas et al. [29] state there is no guarantee for the original Self-Organizing Maps defining the reference-vectors that the rate of misclassification is minimized. Even with identical setting, repeated training of a SOM can lead to different mappings because of the random initialisation (cf. [29]). However, taken from the experience of Wehrens and Buydens Lutbarde [66], the conclusions drawn from the map remain consistent. A repeated training of several maps before drawing conclusions is nevertheless recommended (cf. [66]).

Each time we decide the map to learn something we have to select the setting:

(i) map size and arrangement

(ii) learning rate and neighbourhood function

(iii) training data variables

(iv) initial values for the codebook-vectors

Comparison of different set-ups improved the map to the extent shown in the following.

Map Size and Arrangement

It is necessary to design a net that is sufficiently large to absorb the information we want it to learn but small enough to minimize the time used for training. In a SOM following characteristics have to be defined:

– the number of input nodes n,

– the number of output nodes m.

This means even though we do not explicitly specify the number of classes we actually limit it with the map size (m).

If the map is used for classification each class of samples is approximated by several codebook-vectors. The exact location of the codebook-vectors in the inside the categories is insignificant. As a matter of fact, only a few codebook-vectors in the inside might be sufficient. On the contrary, most of the resolution (i.e. more codebook-vectors) is needed at the class border where the class distribution intersects and causes misclassification. In general, the variance of samples in a class is a more important factor. This is due to the following fact: a very dense class can be describable even by a single codebook-vector. In contrast, in case of a large variance, more vectors must be spread into the region in order to define the forms of class borders with sufficient accuracy (cf. [29]).

A fraction of the order of 10% of the total number of samples (number of available input vectors $X_1, \ldots, X_s$) is recommended as number of output neurons (m) for studying clusters as we do. There were two sizes of output maps used: a 10×10 and a 30×30 grid, leading to 100 and 900 nodes, respectively. The first one was used to gather an average 90 input vectors into every node which would then represent a cluster. As the data did not optimally fit into this kind of map the latter was introduced and is shown here. With the goal of having 10 vectors gathered together in a node we can look for classes of nodes. Clusters of the smaller grid would then be able to stay, but can be distinguished more precisely with a smoother transition.

It was decided to concentrate on a 30×30 grid with hexagonal arrangement, $m = 900$ and n depending on the training run.

Learning Rate and Neighbourhood Function

Via the `kohonen`-Package in `R` (cf. [66]), a linearly decreasing learning rate from 0.05 to 0.01 was used.

After one third of the iterations only winning units are adapted. At that stage the procedure in use is equal to K-means with the advantage of SOM being a spatially constrained form of the K-means clustering.

Concerning the neighbourhood we choose a circular one with Euclidean distance like shown in Figure 4.4.

Training Data

As a first intention all 71 variables have been used to test the program. Although a smooth map was achieved, a better selection of variables led to a more distinct result. After several versions of training it became clear that the focus of evaluation will lie on three training models or run specifications. The stipulations of each model are:

(train.run 1) Totally relying on the decision tree outcome. This led to 12 training variables. Those include

- the age of entry of the first insured person,
- the number of months until maturity,
- the annuity deferral period,
- the duration of payout phase,
- the tariff based sum insured and
- the expected overall profitability for the main contract (in terms of the nbm, which is not the nbm itself).

(train.run 2) Adding numeric variables to those from the first model, ending up with 33 variables. The additional variables include

- information about the possible second person insured, like age,
- technical parameters for reserve calculations,
- the internal splitting of the profitability on main and additional contracts (riders) and
- actuarial parameters like the minimum guarantee interest rate.

(train.run 3) Using the output of the first model as initial values for the weight matrix for a run with all training variables of the second model.

5.3.1 Remark. Actually the second and third run have the same input. It was investigated into replacing the randomly chosen initial values with ones already trained with other data. The last two runs are an example of how initial values can change the outcome.

Initial Codebook-Vectors

The first learnings were conducted using a randomly chosen field of initial vectors from the training dataset. After finding groups to focus on, it was intended to get this group in the centre of the map by using the outcome of the first model as initial values for the third to get a more focused view. The idea was that the group with the most relevant nbm should be in the middle to find variables which interact the most. This did not work out as the groups stayed preserved comparing the second and third run. Nevertheless, the outcomes of the third training run is shown. Keeping in mind, that all training runs shown are only representations of several trainings with the same stipulations.

Random initial values have been chosen without replacement from the training set.

For train.run 3 the codebook-vectors of the first run, being 900×12-dimensional were used by binding this data field to a randomly generated field for the 20 new variables. This led to a 900×33-dimensional matrix usable as initial values for the weight matrix.

5.4 Visualisation with Heatmaps

5.4.1 Definition. *Heatmaps* or *colour maps* are false colour visualisations for data matrices. They are dependent from the values received, the surface topology and the colour palette chosen.

5.4.2 Remark. The term heatmap implies a relation to temperature, which can be misleading. Depending on the application, heatmaps can encode almost every data input, if assigned to the right surface. From the input data to the output figure an abstraction from values to temperature-colouring takes place.

5.4.3 Example. Examples of heatmaps used for visualisation are

– encoding temperature on a geographical map or a house, showing its insulation. An example for this can be seen in Figure 5.2b. Here the abstraction step from colour to temperature is omitted.

– colouring a country by its districts or a continent with its states, as seen in Figure 5.2a.

– where players on a soccer-field stay more often.

– the clicking rate on an internet homepage.

5.4.4 Remark. The appearance of heatmaps is strongly influenced by the chosen colour palette. There is a heavy discussion about which palettes can be recommended for which application. (cf. [8], [9], [39]) No best practice method could yet be found for all applications in use. Palettes can be grouped by continuousness and their usage of colour:

1. Sequential: suited to ordered data, progressing in lightness steps from low to high.

2. Diverging: puts equal emphasis on mid-range critical values and extremes at both ends of the data range.

3. Qualitative: for representing nominal or categorical data, as they do not imply magnitude difference between legend classes. Hues are used to create the primary visual differences between classes.

See Figure 5.1 for examples from the `RColorBrewer`-Package in `R` (cf. [41]). In [12] the equal application of dendrograms (see Definition 2.4.1) and heatmaps is shown.

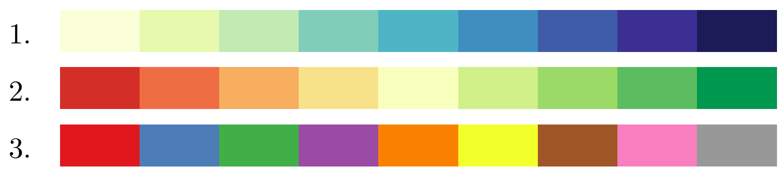
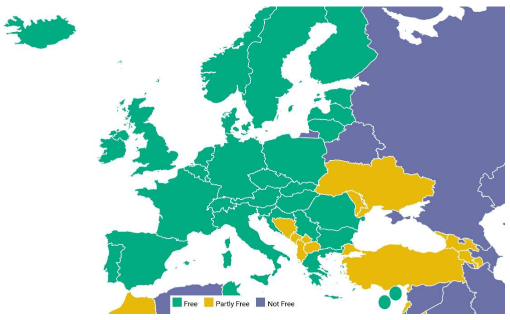


Figure 5.1: Examples of colour palettes with numbering accordingly to Remark 5.4.4. (cf. [62], [41])

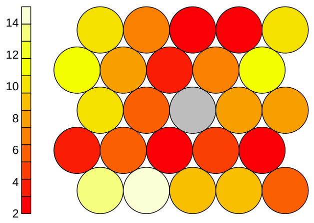
The standard procedure of `R` only allows circular nodes. As we would like to exploit the plots at best, a sub-programme to create hexagons within the arrangement was implemented. There already exist programs having pre-implemented this visualisation like [65] or the MATLAB SOM Toolbox [64]. See Figure 5.3 for a comparison of node shapes and colour palettes.



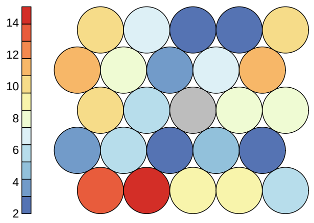
(a) European countries coloured by their status of freedom according to a report of the Freedom House organisation. See [46].

(b) A thermal image of a house with continuous colouring in the rainbow colour-palette. See [54] for more information.

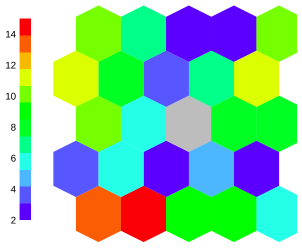
Figure 5.2: Examples of heatmaps with a (a) three-colour and a (b) continuous colour palette.



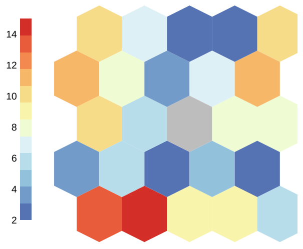
(a) Original plot
with `heat.colors`-palette.



(b) Original plot
changed to `"RdYlBu"`-palette.



(c) Hexagonal nodes
with `rainbow`-palette.



(d) Hexagonal nodes
with `"RdYlBu"`-palette.

Figure 5.3: Heatmaps of SOMs with the `wine` example-data within the `kohonen`-package in different colour-palettes and the pre-implemented circular versus hexagonal node-shapes.

We will use a heatmap corresponding to the grid size and arrangement chosen before. The `"RdYlBu"`-palette from the `RColorBrewer`-package (cf. [41]) was chosen as colour scheme for the property plots. The U-matrix will be plotted in grey-scales, and the node-counts using the `rainbow`-palette. The nodes will be represented as hexagonal shapes.

5.5 Discussion

A key ingredient in this step was domain expertise, as expected in [13]. As in insurance based data certain variables always depend on each other, some of the groups can easily be assigned to a type of tariff. The following groups can be identified, differing from their surrounding by their specifications:

(1) Lifelong annuity in payout phase.

  Every node in this group has a higher average present value of premiums then overall average. The highest age for the second person insured and no expected income from further premiums is also visible.

(2) Lifelong individual annuity in saving phase.

  This group in average has a lower minimum guarantee interest rate than the other annuities.

(3) Lifelong group annuity with rider for occupational cover.

  Contracts in this group were rather signed by brokers and have an age shift.

(4) Endowment with a one-time death benefit.

(5) The only group with discount for large sums.

  This group can largely be identified by the tariff area, mainly entered by corporate customers and providing group conditions.

(6) Group with rather low nbm.

  The contracts in this group belong to a single industrial client. It can be identified a single premium payment and a certain sales party.

(7) Supplementary insurance (rider) to the former group.

  It has the same parameters concerning payment and sales party as group 6. As these are additional insurances they are not treated as standalone contracts and therefore do not have costs (so called loadings). This group contains as well the highest and the lowest nbm, separated by the age of the insured person. Younger policy holders lead to the highest, older ones to the lowest nbm in the whole contract-pool.

(8) High average nbm.

  This group is very heterogeneous concerning the line of business, payment frequency and policy holder participation. It is the most interesting group as most nodes in the group declare contracts having a nbm above average.

(9) Remaining term insurance related nodes, having a nbm above average.

(10) Remaining endowments related nodes with nodes having a nbm below average.

Most of these groups can be identified in every run of all three training models. In the first run groups 2 and 3 can not be clearly differentiated, so they stay together within two separated areas. Groups 9 and 10 are spread out as remaining term life or endowment insurance contracts in all three runs. Some of the outcomes can be verified externally. We will go through exemplary variables and show their connection and interpretation.

5.5.1 Example. Examples for linear correlations understandable by the nature of tariffs are:

– Smoking behaviour is only recorded for term life insurances.

– The age of entry of the second person insured is the highest in annuities in payout phase.

U-matrix and Node Counts

A way to estimate the quality of the trained SOM is to examine its U-matrix representation, see Definition 4.3.1 (cf. [55]). In Figures 5.4a, 5.4c and 5.4e the U-matrix for each training run is shown. Following the standard, the U-matrix is plotted in grey scales. As expected, the U-matrix for insurance based data is not as informative as for other data.

5.5.2 Remark. The scales in Figure 5.4 have to be interpreted as follows:

- For the U-matrix light grey means high average distance from the code vector to those of the surrounding nodes.
- The node count is an enumeration on how many times the node was chosen as winning unit in the last iteration from zero to the maximum. So-called empty nodes with no projected input vector, are grey.

Figures 5.4b, 5.4d and 5.4f show the node count for every run. The aim was to space out the data evenly by an average of ten contracts (input vectors) per node (output neurons). The expected accumulation of similar policies into one node happened and is visualised in red colour. The redder a node, the more often the node was chosen as winning unit and therefore represents the most contracts. The maximum of vectors gathered in a single node is 52, 59 and 49 for training run 1, 2 and 3, respectively. The minimum for all is zero. These empty nodes are shown by grey colour in the right-hand figures.

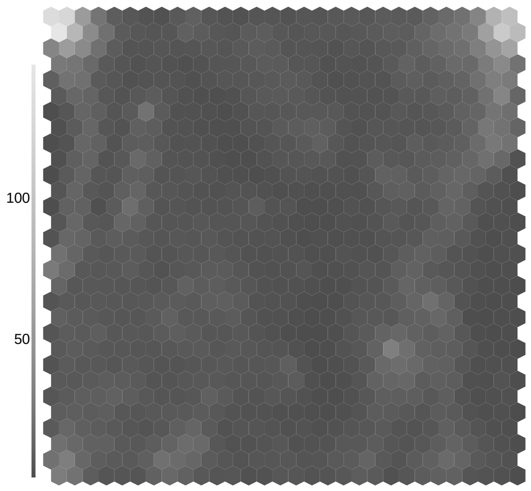
Working as borders, empty nodes appear slightly lighter in the U-matrix.

In all figures except Figure 5.4 empty nodes will be coloured by the overall average in the map. This leads either to a smooth-looking map or to border-like colouring. Both can be seen for example in Figure 5.5b. In addition to the colouring, found borders of the groups were added. These are meant to simplify recognition of the groups. Due to the algorithm not every group can be segregated in every heatmap. When considering the U-matrix in Figure 5.4, the highest neighbouring distances are in the same area as group 1, shown in the following figures.

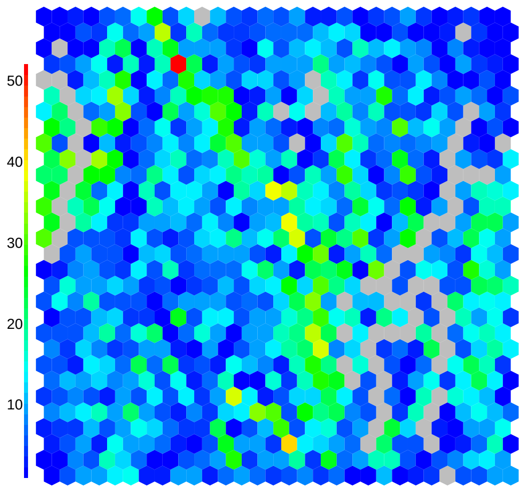
Groupings

Concerning the groups' borders, every group can be seen in every run. As we will find out, especially groups 1 to 7 are perfectly separable, where the others sometimes do not have clear borders. As mentioned before, the exact location of a group is not as important as its connection to others. Because of the chosen amount of nodes, all groups are represented by several codebook-vectors. Therefore the importance of whether or not the group is e.g. in the centre is decreasing.

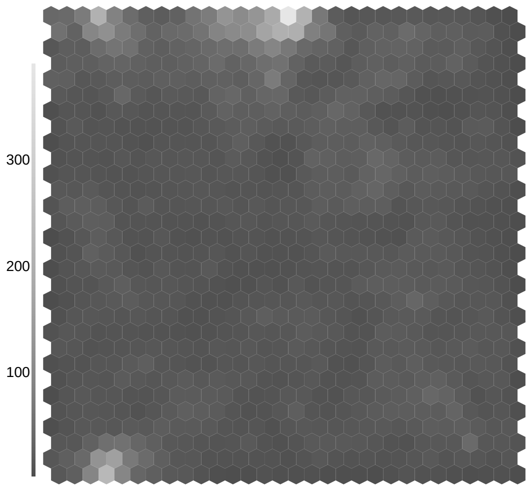
The groupings stay preserved whether or not the initial values are manipulated. There has been a rearranging of some groups (compare e.g. group 7 in training run 1 on the left side and in training run 3 on the upper right side) but little interchange between them (a contract from group 7 in training run 1 stays in group 7 through all runs performed). Groups having significant differences (e.g. group 7) do not interact or rearrange (contracts in group 8 may



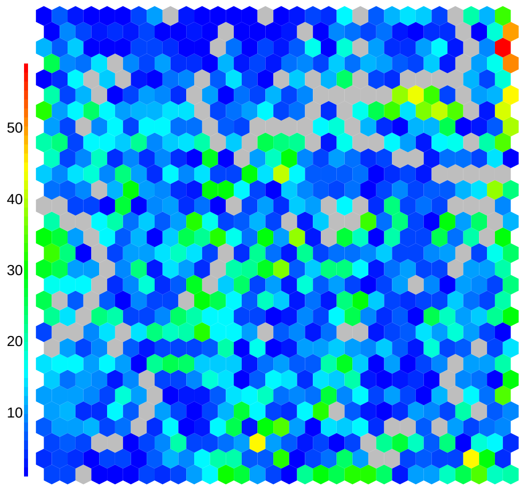
(a) U-matrix for training run 1.



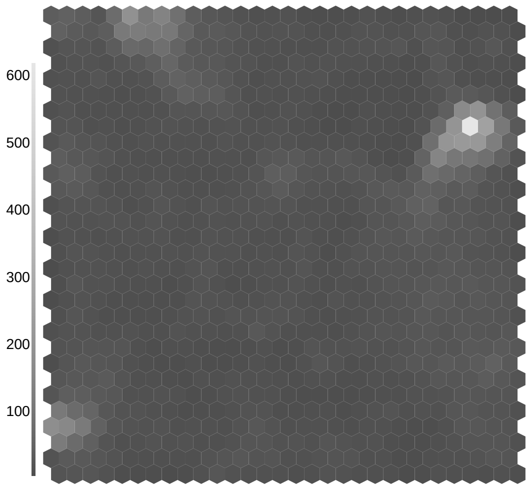
(b) Node counts for training run 1.



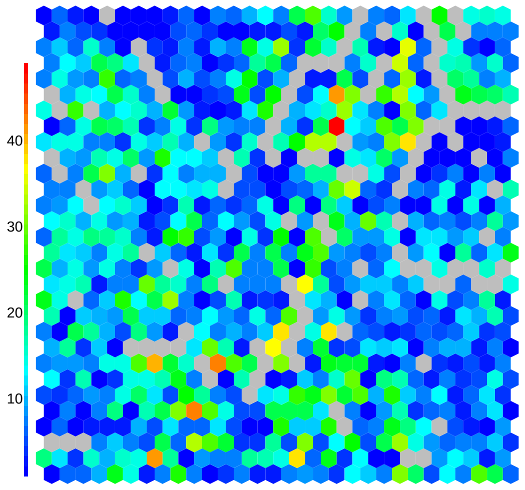
(c) U-matrix for training run 2.



(d) Node counts for training run 2.



(e) U-matrix for training run 3.



(f) Node counts for training run 3.

Figure 5.4: U-matrix showing the distance between neighbouring nodes and the plots showing how many contracts gather in each node.

end up in group 9 or 10 in some training scenarios). It became clear which of the groups have more in common than the others with more runs performed. For example groups 2 and 3 could be separated only by the additional runs, which means contracts in these groups seem to be similar with the information of the first run. This stays preserved as they are next to each other in every run. The additional information only helps to separate them. In the first run group 1 is not next to the other annuity contracts. This is also a good reason to follow the recommendation (cf. [32]) to always train a dataset several times and rely on information only after using different settings.

As another outcome, group 4 is always close to the annuity groups 1, 2 and 3. Group 7 has the most differences to other groups, which can be seen by the ever remaining border of empty nodes. Another indication is the complete separation of this group in some runs. As an example the third run shows group 7 on the upper right corner in contrast to the majority of term life insurance contracts (group 9, see Figure 5.6e). Also a part of group 9 can always be separated from the remaining. It was decided not to open yet another group and keep them in one. The reason for this ongoing separation is the fact of this sub-group being a term insurance contract with a second person insured.

Annuity Period and Payout Deferral

Figure 5.5 shows the payout deferral (i.e. month until the first payout) on the right (Figures 5.5b, 5.5d and 5.5f) and the duration or term of contract concerning payouts (i.e. payout deferral plus maximum duration of annuity payout phase) on the left side (Figures 5.5a, 5.5c and 5.5e). All three runs had both variables measured in months as training inputs.

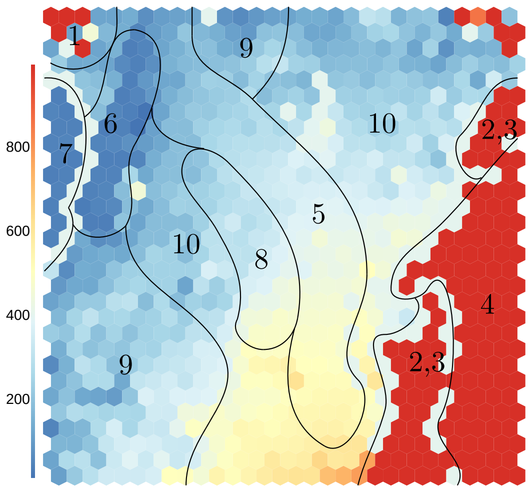
5.5.3 Remark. In Figure 5.5 the colours can be interpreted as following:

- Red colour shows a lifelong annuity payout phase on the left and a lifelong payout deferral (i.e. payout at time of death) on the right side.
- If no payout is made, the payout deferral and the duration of payout is set zero and coloured dark blue.
- The shades between are the months of payout and months of deferral of payouts, respectively.

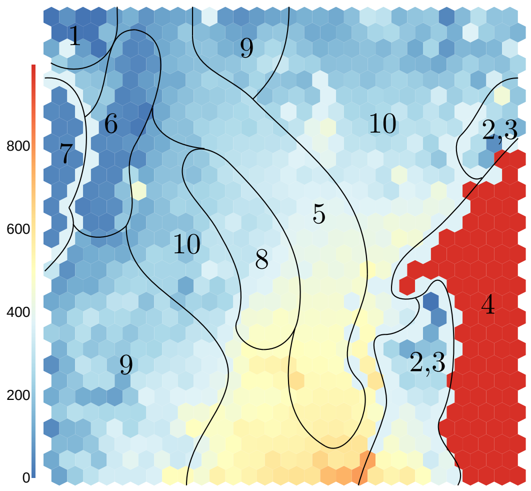
The red colour in Figure 5.5 codes lifelong annuity or the payout at time of death, respectively. Especially the first run (Figure 5.5a) shows a smoothly decreasing annuity payout-period from the lower right to the upper left corner. As anticipated in the U-matrix, group 1 is an exception for this increasing value. The other runs do not show the same smoothness as the first one. This is because other training variables have been added and give additional information, reducing the importance of a single variable. In this comparison, group 4 can be distinguished from groups 1 to 3, which are annuities, as we will see in Figure 5.6.

Line of Business and Smoking Behaviour

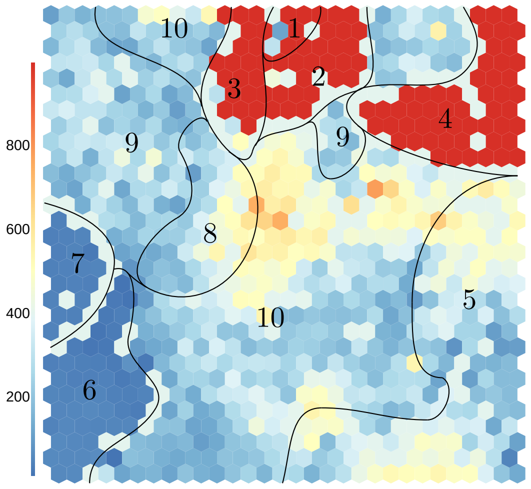
In Figure 5.6 the map is coloured in association with the lines of business (Figures 5.6a, 5.6c and 5.6e) and the smoking behaviour (Figures 5.6b, 5.6d and 5.6f). Contrary to Figure 5.5 none of the runs obtained any information about these variables. In all maps the empty nodes can easily be seen.



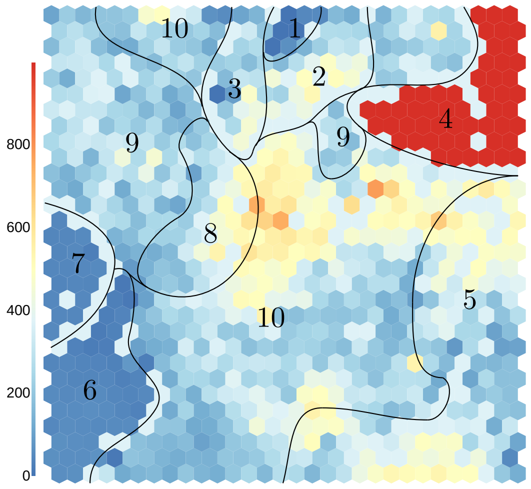
(a) Contract duration for training run 1.



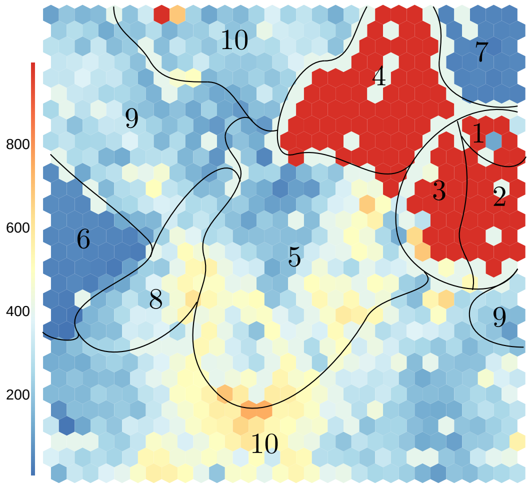
(b) Payout Deferral for training run 1.



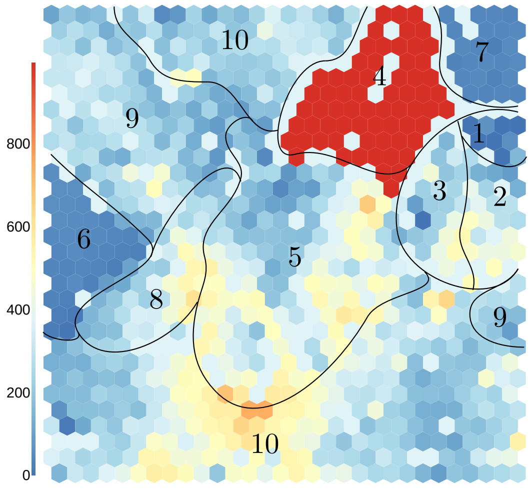
(c) Contract duration for training run 2.



(d) Payout Deferral for training run 2.



(e) Contract duration for training run 3.



(f) Payout Deferral for training run 3.

Figure 5.5: The duration of payout deferral and the contract with deferral and maximum annuity-payout phase in months.

5.5.4 Remark. As both of the variables shown only have three possible outcomes, the colours in Figure 5.6 can be interpreted as:

colour code	red	yellow	blue
line of business	term	annuity	endowment
smoking behaviour	unknown	smoker	non-smoker

All intermediate stages represent contracts of different values in those variables in one node. Empty nodes are coloured light blue for lob and light red for the smoking behaviour related figure.

The colouring in this figure is a good example of the mixture of different contracts in one node. Although having only three distinctive possibilities for one contract to be in a line of business, a node can collect insurances of different kinds. This leads to various colours in the heatmap. On one hand empty nodes result in a light blue or light red colouring. On the other hand mixtures result in orange and light blue shades, respectively for the lob and smoking behaviour visualisations. Especially in group 8 all lines of business are present with a small tendency to term. Groups 6, 7 and 9 are almost exclusively term contracts, whereas groups 1, 2 and 3 can be classified to annuities.

Comparing the line of business to the recorded smoking behaviour leads to conclusions that could have been drawn by logic as well. Smoking behaviour is only recorded in term life insurances. Therefore it should not lead to an over-interpretation comparing unknown and non-smoker. It can be said though, that a major number of contracts in groups 8 and 9, which are the only groups with recorded smoking behaviour, is non-smoking (dark blue). There are only few nodes with solely smokers (yellow), but from top to bottom the proportion of smokers rises. There is no known smoking behaviour in groups 6 and 7 although they are groups of term contracts.

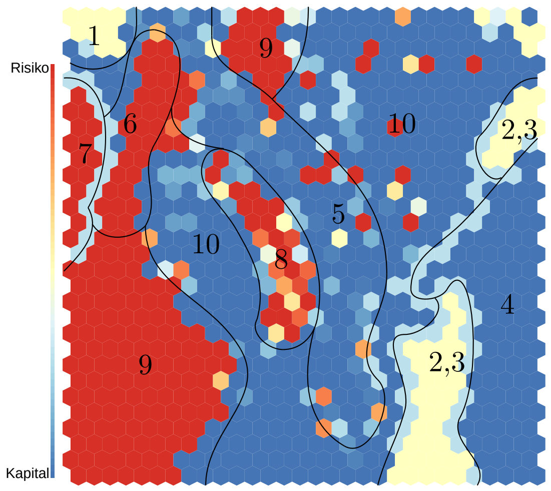
Comparing the three runs it seems obvious that the second and third one separate the contracts better than the first. Although main clusters are visible in the first run, outliers are more seldom in the runs with more training information.

Area of Tariff and Payment Frequency

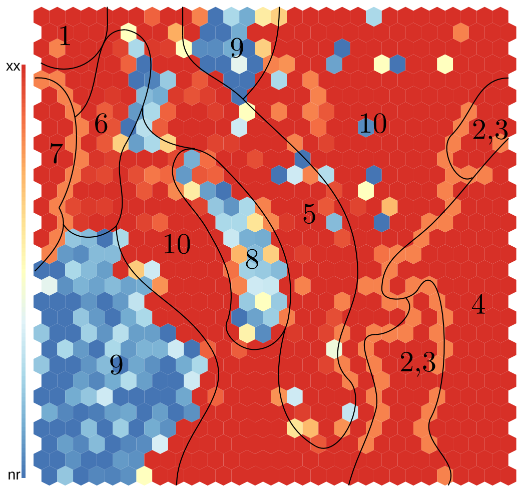
The so-called *area of tariff*, shown in Figures 5.7b, 5.7d and 5.7f, shows the speciality of group 5. The area of tariff is an identifier, chosen by experts to designate certain cost-loading and pricing levels to contracts. In the case of group 5 it is known these contracts are mainly group contracts signed by corporate clients. As customers signing several contracts with very similar specifications do not have the same per-capita costs, clients benefit from a discount for large sums. Altogether, experts chose to distinguish seven areas and encode them. For this reason, no labels are added to the key. Again, there are more than seven different colours to be seen in Figures 5.7b, 5.7d and 5.7f because of different areas ending up in one node. Of the 9083 contracts considered, 6835, or 75.25%, belong to one area of tariff. This area contains the standard term life insurances. The area in group 5 visible through the graph comprises 19.89% of all contracts. The remaining 441 indentures spread throughout the other five areas with the smallest one only containing three.

5.5.5 Remark. For Figures 5.7a, 5.7c and 5.7e, following colour code fits:

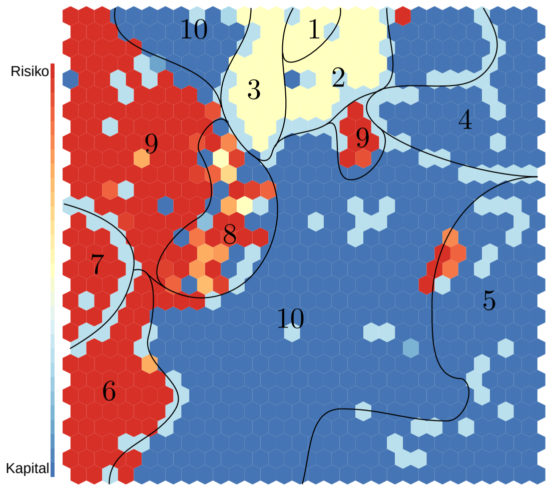
colour code	red	light blue	blue	dark blue
key	12	4	2	1
payment frequency	monthly	quarterly	semi-annual	annual or single-premium



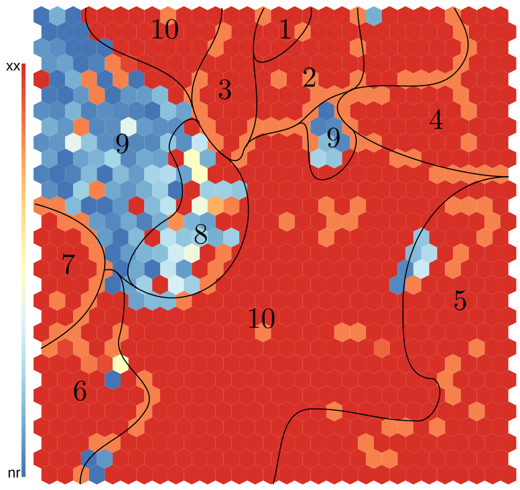
(a) Line of business for training run 1.



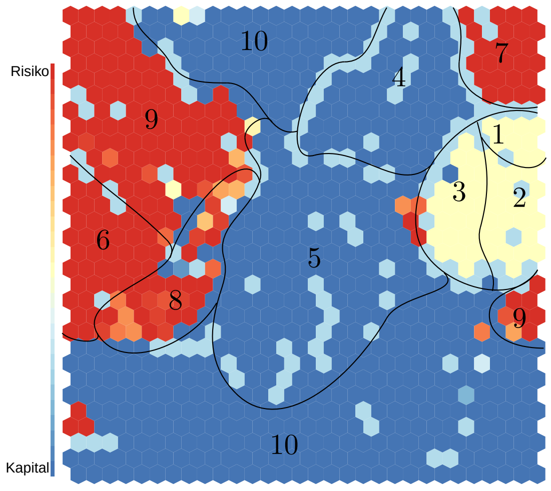
(b) Smoking behaviour for training run 1.



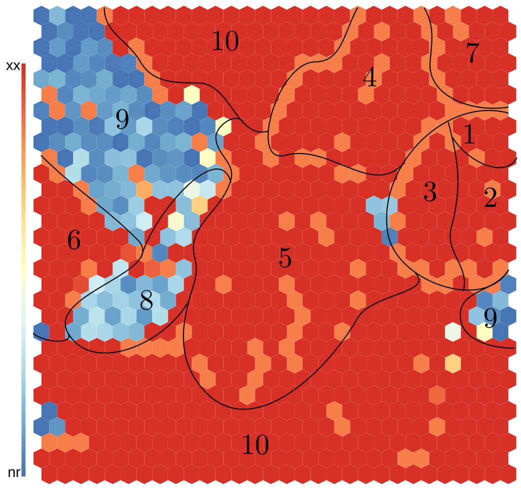
(c) Line of business for training run 2.



(d) Smoking behaviour for training run 2.



(e) Line of business for training run 3.



(f) Smoking behaviour for training run 3.

Figure 5.6: The lines of business: endowment (blue), annuity (yellow) and term life insurance (red) and the smoking behaviour: smoker (yellow), non smoker (blue) and unknown (red).

On the left hand side of Figure 5.7 the heatmaps for payment frequency are shown. The frequency of premium payment is strongly correlated with the so-called state of contract. There are three types of states: liable for premium payment, current annuity and single-premium. In groups 6 and 7 only single-premium payments occur. The only other group of single-premiums is a part of group 10 and always near group 6. The only group with current annuity is group 1. Coming back to Figures 5.7a, 5.7c and 5.7e it has to be considered, that a single-premium and the annuity in payout phase is encoded with the same colour as a yearly payment. This variable is always one of four possible year's proportion, 12 being a monthly, 4 a quarterly, 2 a semi-annual and 1 a yearly or single payment.

The blue colour in groups 1, 6 and 7 and nodes near or surrounding group 6 are single-premiums, where group 5 for instance has real yearly payments. This is a good example for the fact not to consider single heatmaps without using all information needed. In this case, looking at the colouring for the state of contract is sufficient to differentiate single-premiums and payouts from yearly payments.

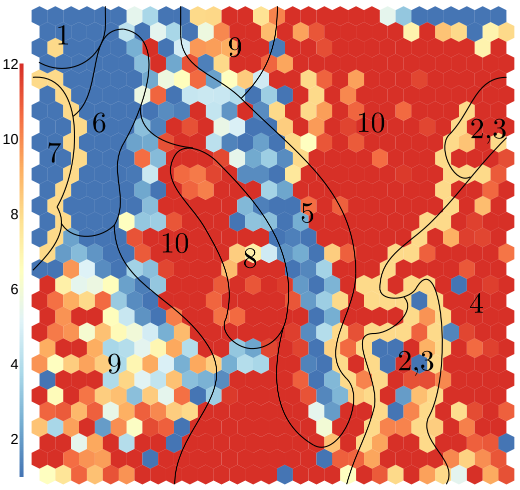
Although none of the runs got any information neither about the area of tariff nor about the payment frequency, all separated them in a comparable way. In all three runs group 5 can be detected. The conclusion of the majority of contracts belonging to one area can be drawn. It seems that the first run separates the different payment frequencies better than the runs with more training information but it draws back in the area of tariff. Not too much conclusion can be drawn from this fact because other runs with smaller training information show slightly better separation. The known linear correlation of a lower average payment frequency in endowment insurances and a higher one in term life insurances can be visualised when comparing to Figure 5.6.

New Business Margin and Value of New Business

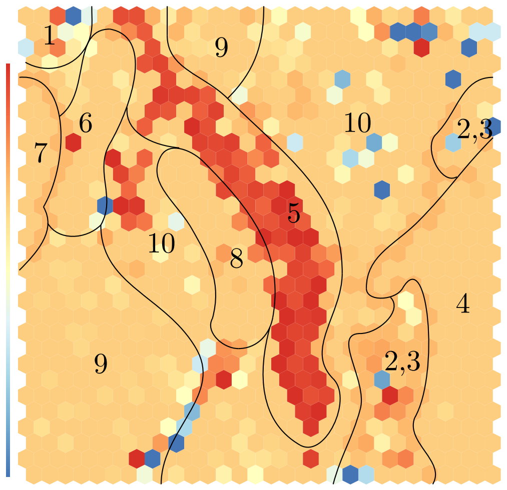
The targeted new business margin can be seen in Figure 5.9 next to the vnb. First the different scales have to be mentioned. Alongside the nbm the first run does not separate the contracts well enough, so the colours are downscaled. Between the right- and the left-hand side of Figure 5.9 the colour palette has been chosen differently.

Especially current annuities (in payout phase) have an enormous high vnb. This would impact the colouring of the map in a way, that only three to eight nodes could be discerned. After training the map with original data it was decided to cut the colouring at a 99.5% quantile. This affects 46 contracts, which are capped to the quantiles value of 2895.82€. A different `heatcolour`-Palette was chosen to keep this intervention in mind. The first run generally shows a lower vnb. This means the contracts with extreme values are not spread out but collected into individual nodes. As in the nbm to the right, the scale is slightly different from the first to the other runs.

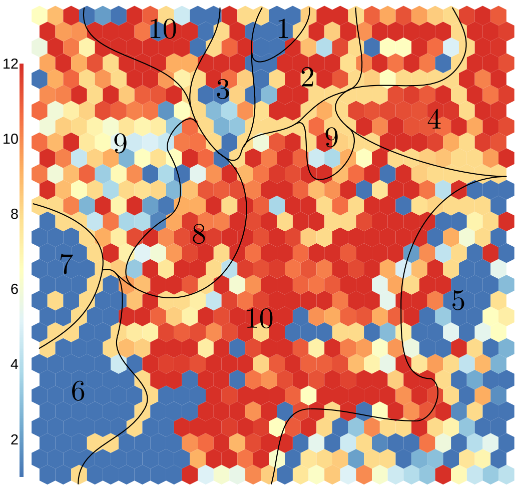
Although the nbm (in the case of profitability monitoring) is one of the most targeted indicators in the actuarial life department, the vnb is vivid too. It interacts with the nbm but a higher vnb does not directly lead to a high nbm. Contracts with a high vnb might also have a high PVNBP to cover the insurance. The high vnb in group 11 leads to an average nbm because of this fact. In every run, parts of group 10 have a higher than average vnb, but these also do not appear in Figures 5.9b, 5.9d or 5.9f for this reason. The interesting point is, that not all contracts behave like this. Group 7 is again the most distinguished group. It can be clearly separated from the others. Here one has to take into account additional information. Contracts ending up in group 7 have a unitary premium, not adapting on the policy holder's age. In addition to this they are not participating in profits. These two informations explain, why a slight change in age leads to a hight impact on the nbm. Younger policy holders pay more, older ones less than they would if the premiums would adapt by age. The profit participation



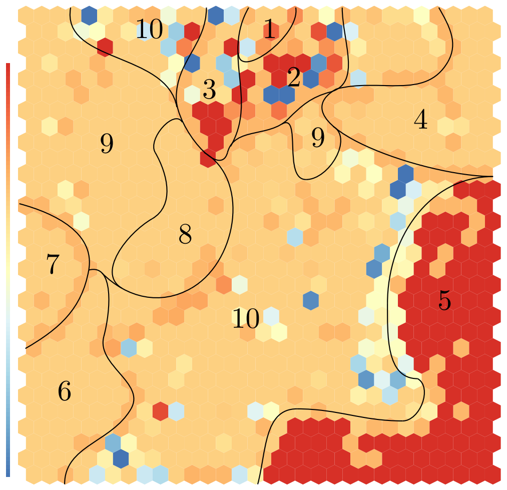
(a) Payment frequency for training run 1.



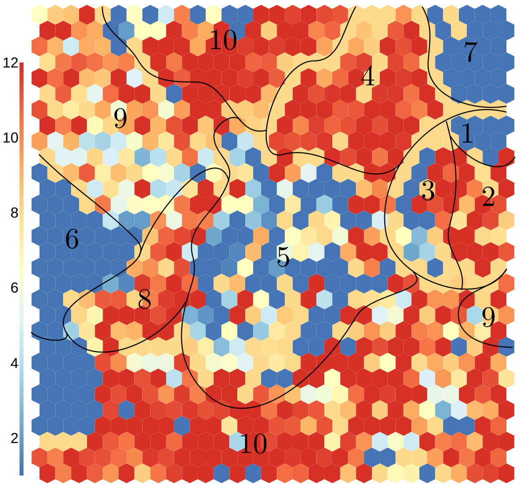
(b) Area of tariff for training run 1.



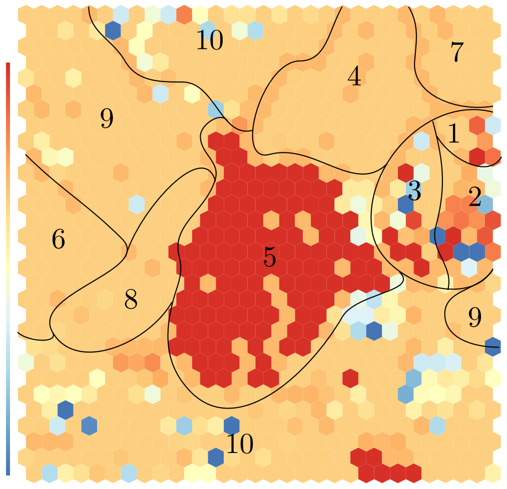
(c) Payment frequency for training run 2.



(d) Area of tariff for training run 2.



(e) Payment frequency for training run 3.



(f) Area of tariff for training run 3.

Figure 5.7: The payment frequency in parts of the year and the area of tariff. See text for more information.

smooths the new business margin of all other groups except group 7. Table 5.8 summarizes the minima, maxima, median, mean and the two main quantiles of all runs for three exemplary groups we focused on.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
input data	-0.6678%	0.0043%	0.0198%	0.0213%	0.0387%	0.2777%
Group 6 Run 1	-0.5295%	-0.1247%	-0.0643%	-0.0903%	-0.0335%	0.0718%
Group 6 Run 2	-0.5295%	-0.1274%	-0.0653%	-0.0923%	-0.0351%	0.0381%
Group 6 Run 3	-0.5295%	-0.1253%	-0.0650%	-0.0914%	-0.0344%	0.0381%
Group 7 Run 1	-0.6678%	-0.0333%	0.1223%	0.0461%	0.1697%	0.2777%
Group 7 Run 2	-0.6678%	-0.0333%	0.1223%	0.0461%	0.1697%	0.2777%
Group 7 Run 3	-0.6678%	-0.0333%	0.1223%	0.0461%	0.1697%	0.2777%
Group 8 Run 1	-0.0126%	0.0261%	0.0440%	0.0627%	0.1076%	0.1226%
Group 8 Run 2	-0.0880%	0.0927%	0.1062%	0.0851%	0.1113%	0.1209%
Group 8 Run 3	-0.0880%	0.0985%	0.1072%	0.0897%	0.1118%	0.1209%

Figure 5.8: A summary of the given values of new business margin in the raw data and specified groups, rounded to four digits.

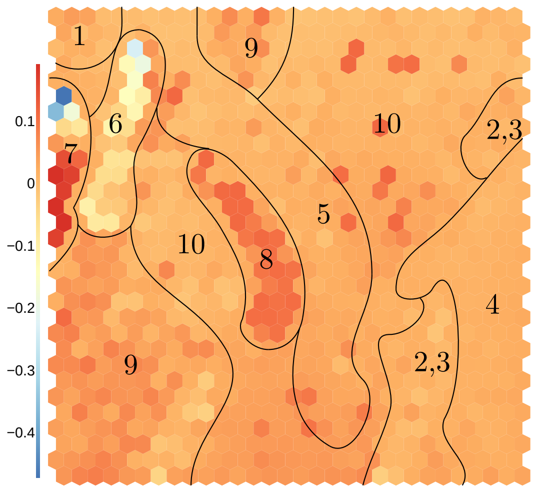
It can be seen, that in group 7 maximum and minimum of the input data is reached. In addition, all runs lead to the same summary. With this information and the border of empty nodes in all runs the high differences from this group to all others become obvious.

Group 8

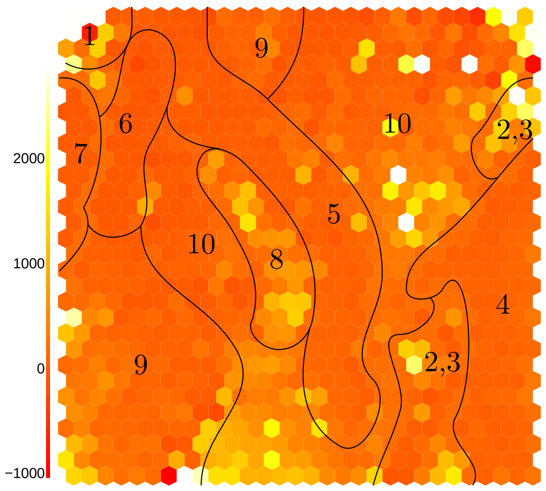
As group 8 has the most relevant new business margin, additional effort was made. In contrast to groups 6 or 7, it is not possible to attach the contracts in the group to a single parameter's characteristic. Additionally contracts within the group's borders interchange between groups. This means not all training runs contain the same contracts in group 8. Even within one training specification, when repeating the run (as was recommended), it is possible to have several contracts outside the border for some runs. On average, 230 vectors are projected to group 8, but looking at all runs, around 700 contracts appear at least once in the pool. Only 143 vectors are in the group 8 in more than 90% of runs. This encourages the a priori knowledge of the nbm being highly non-linear and depending on many variables.

The on average best and most new business margin comes from contracts with the following arrangements:

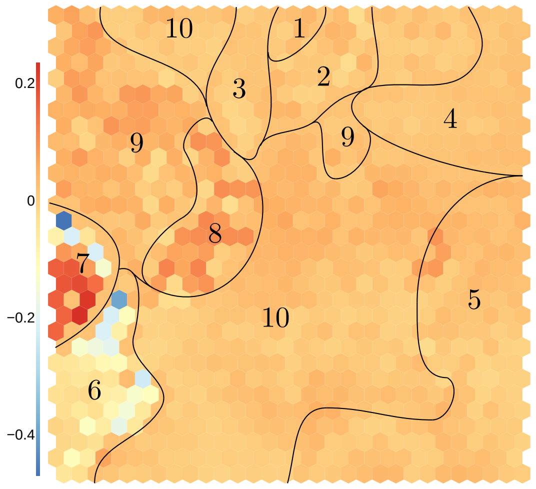
- More than 90% of contracts being in group 8 are term insurances of one particular tariff. Contracts being at least once in this group represent either the same tariff or endowments.
- With the age of entry between 21 and 42 and a payout deferral between 19 and 42 years the payout is shifted to the age of 60 in the contracts of group 8. On average those contracts sometimes in group 8 start payout at a slightly higher age, going up to 64. Leaving aside the contracts linked to a lifetime (i.e. payout at time of death or until death), the average age at first payout is insignificantly lower though the age of entry varies from zero to 81.
- Endowments in group 8 are not in their payout phase.



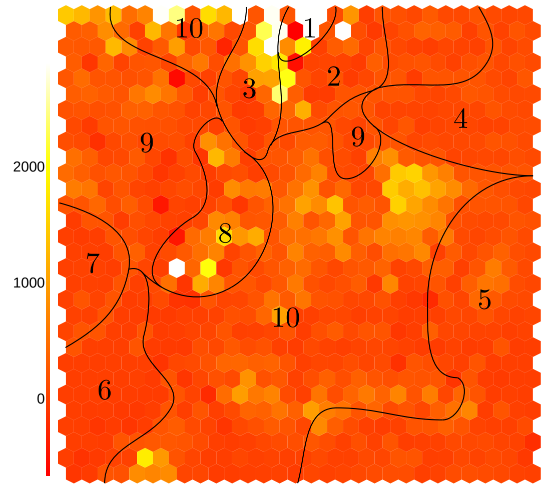
(a) New business margin for training run 1.



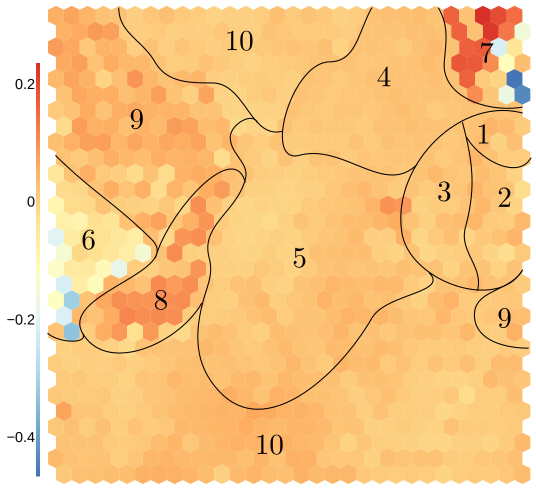
(b) Value of new business for training run 1.



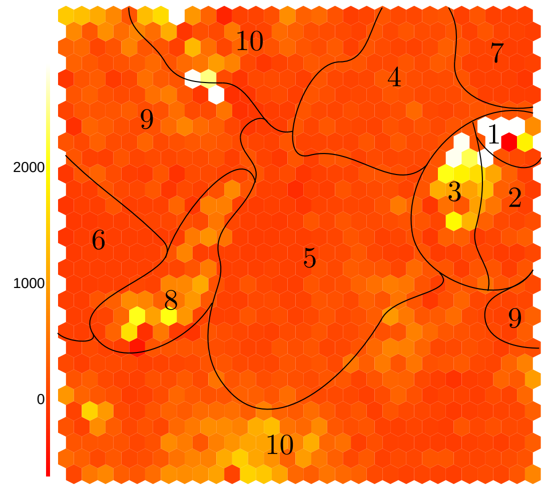
(c) New business margin for training run 2.



(d) Value of new business for training run 2.



(e) New business margin for training run 3.



(f) Value of new business for training run 3.

Figure 5.9: The targeted new business margin in comparison to the value of new business. See text for more information.

- The minimum guaranteed interest rate is 1% or 1.5%, which is neither the minimum of 0% nor the maximum of 3% still given under certain circumstances.
- The contracts being always in group 8 were sold by salaried sales forces or tied agents, not by brokers or other channels (which make up approximately one third).
- Concerning the subsidy for the actuarial reserve, group 8 contracts are always zero whereas others can go up to 185,346€ and contracts sometimes in group 8 have also a maximum of 2150€.
- The value of the intrayear Zillmer adjustment (cf. [44] for its definition) stays low within the contracts staying in group 8 (with an average of 25.17€) and those switching between group 8 and others (with 68.86€). For comparison, the all-over maximum is 998.17€ in the full dataset with an unweighted mean of 10.83€.

Drawing conclusions from the outcomes in this work is rather complicated, as most characteristics show only insignificant evidence. This is largely generated by unplanned use of parameters or their inconclusive entries in the original data.

Chapter 6

Conclusion and Further Research

In this work we discussed the Data Mining technique of Self-Organizing Maps. We applied it on a dataset consisting of parameters for calculation of the new business margin. After discussing Data Mining and its usage in insurance companies we reviewed neural networks in general. The algorithm of Self-Organizing Maps was introduced with the help of Learning Vector Quantisation. We applied the neural net on the preprocessed dataset and showed the outcome through several exemplary variables. As visualisation was one of the main focuses we also discussed the setting in number, arrangement and colouration of the nodes.

It was shown that the newly signed contracts split up in different ways. As expected, the line of business was one key driver, but also others were to be inspected. The main emphasis laid upon the group with a high new business margin. As this group offered a very heterogeneous pool of contracts, analyses revealed a dependence of the new business margin with many other parameters. The main problem with using this additional information, gathered with the neural network and the postprocessing process is the high non-linearity, making it difficult to implement the outcomes into future models.

Another difficulty appearing is widespread: unclean data. With a historically grown dataset and possible human interaction within the data, handling of data gets challenging. Some of the data entries are not used in the way they were intended to or are not used at all. Without additional knowledge about the data in use, significantly less results can be found or used. It is therefore recommended to clean data in a way that labelling and factorial entries are accurate and do not require additional manual interaction.

Automation and clear process structure can help to clean data and to analyse and evaluate data more easily. Nevertheless, in this work the outcomes of the SOMs can be relied on. Outcomes from former analyses can be re-enacted and an expected new side of the new business margin can be supported.

Experts should focus on non-annuity contracts with an insured person between 21 and 52 years of age. Contracts with a death benefit (first payout at time of death) are also more seldom to be found in the group with higher than average nbm. Only 143 contracts stay within this group for a significant number of runs. As this represents less than 2% the significance of this outcome is questionable. Yet hints in directions of which characteristics to choose can be drawn from the results.

This means Self-Organizing Maps are a possible tool to analyse data according to the new business margin. The problem of a high pre- and postprocessing expenditure remains as the data are not clean enough to have a ready-to-work status. To overcome this problem, apart from automation without human interference, more and different statistical methods than used in this work should interlock to get significant, reliable and interpretable results. Self-Organizing Maps can thereby be used as a pre-sorting algorithm for further neural networks and as a visualisation method.

Further research can include a time series analyses on these outcomes to back the assumption of June being an average and representative month concerning new business. The use of self-organisation processes for financial time series was discussed e.g. in [68]. Sarlin and Eklund [50] used fuzzy clustering of SOMs in application on time series. Dealing with financial data, Resta [48] introduced a SOM subclass maybe also useful for insurance based data. All of these ideas could be a starting point for further research.

Additionally, more input data may get to a result with higher resolution. A hybrid clustering method presented by Arumugam and Christy [3] using interacting SOMs and PCA is possibly useful for our purposes as well. There also exist several purchasable tools working with a wider range of standard statistical and SOM approaches (cf. e.g. [36] or [65]) than it was done in this work. Therefore further research should focus in the direction of combining more statistical tools and several datasets in a timeline manner.

Abbreviations

ANN	Artificial Neural Network
cf.	confer
cov	covariance
cor	correlation
$\mathbb{E}$	expected value
e.g.	exempli gratia
EV	Embedded Value
i.e.	id est
KDD	Knowledge Discovery in Databases
KNN	K Nearest Neighbours
lob	Line of Business
LVQ	Learning Vector Quantisation
MCEV	Market Consistent Embedded Value
nbm	New Business Margin
NN	Nearest Neighbour
O&G	Options and Guarantees
PCA	Principal Component Analysis
PVFP	Present Value of Future Profits
PVNBP	Present Value of (net) New Business Premiums
$\mathbb{R}$	real numbers
SOM	Self-Organizing Map
train.run	Training Run
ULI	Unit-Linked Insurance
$\mathbb{V}$	Variance
vnb	Value of New Business
VQ	Vector Quantisation

Bibliography

[1] Market Consistent Embedded Value Report 2015, 2015. URL https://www.allianz.com/v_1457705192000/media/investor_relations/en/results_reports/annual_report/ar2015/evr2015.pdf. Official Report - Allianz SE - Group Actuarial.

[2] J. Alspector, R. Goodman, and T. Brown, editors. *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, Inns Series of Texts, Monographs and Proceedings Series, 1995. Lawrence Erlbaum Associates.

[3] P. Arumugam and V. Christy. A Hybrid Clustering Method for Data Mining. *International Journal of Research and Scientific Innovation*, 3(7):87–94, 2016.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, volume 19, pages 153–160. MIT Press, 2007.

[5] H. R. Berenji and P. Khedkar. Adaptive Fuzzy Control with Reinforcement Learning. In *1993 American Control Conference*, pages 1840–1844. IEEE, June 1993.

[6] H. Bertsch and J. Dengler. Klassifizierung und Segmentierung medizinischer Bilder mit Hilfe der selbstlernenden topologischen Karte. In E. Paulus, editor, *Mustererkennung 1987: 9. DAGM-Symposium, Braunschweig, 29.9.–1.10.1987. Proceedings*, volume 149 of *Informatik-Fachberichte*, pages 166–170. Springer Berlin Heidelberg, 1987.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1996.

[8] D. Borland and R. Taylor. Rainbow Color Map (Still) Considered Harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, March 2007.

[9] C. Brewer. Color brewer 2.0, 2013. URL http://colorbrewer2.org.

[10] F. Buck, T. Kochis, D. Kunesh, M. McLaughlin, E. Robbins, D. Rogers, E. Schuering, B. Smith, and J. Zellner. *US GAAP for Life Insurers.* Society of Actuaries, 2000.

[11] J. M. Campanario. Using neural networks to study networks of scientific journals. *Scientometrics*, 33(1):23–40, 1995.

[12] P. Cock. Using R to draw a Heatmap from Microarray Data, 2009. URL http://www2.warwick.ac.uk/fac/sci/moac/people/students/peter_cock/r/heatmap.

[13] G. Deboeck. *Visual Exploration in Finance.* Springer Finance. Springer-Verlag London, 1st edition, 1998.

[14] J. Dutcher. What Is Big Data?, 2014. URL https://datascience.berkeley.edu/what-is-big-data/.

[15] M. Ester and J. Sander. *Knowledge Discovery in Databases*. Springer-Verlag Berlin Heidelberg, 1st edition, 2000.

[16] European Insurance CFO Forum. Market Consistent Embedded Value Principles, April 2016. URL `http://www.cfoforum.nl/downloads/CFO-Forum_MCEV_Principles_and_Guidance_April_2016.pdf`. Amended Market Consistent Embedded Value Principles.

[17] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.

[18] M. Gonçalves, M. de Andrade Netto, J. Ferreira Costa, and J. Zullo Jr. Data clustering using Self-Organizing Maps segmented by Mathematic Morphology and Simplified Cluster Validity Indexes: an application in remotely sensed images. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 4421–4428. IEEE, July 2006.

[19] M. Grochowski and D. Włodzisław. Constrained Learning Vector Quantization or Relaxed k-Separability. In C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, editors, *Artificial Neural Networks – ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14-17, 2009, Proceedings, Part I*, pages 151–160. Springer Berlin Heidelberg, 2009.

[20] W. Gurker. Angewandte Mathematische Statistik VO – 107.A06 – 7 . Lecture Notes, 2003.

[21] J. Heikkonen and E. Oja. Self-organizing maps for visually guided collision-free navigation. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 1, pages 669–672. IEEE, October 1993.

[22] T. Henning. Eine Self Organizing Map hilft bei der Prävention von Cyber-Angriffen, 2016. URL `http://www.datacenter-insider.de/eine-self-organizing-map-hilft-bei-der-praevention-von-cyber-angriffen-a-566966/`.

[23] S. Herculano-Houzel, K. Avelino-de Souza, K. Neves, J. Porfírio, D. Messeder, L. Mattos Feijó, J. Maldonado, and P. Manger. The elephant brain in numbers. *Frontiers in Neuroanatomy*, 8(46):1–8, June 2014.

[24] G. Hinton and T. Sejnowski. Learning and Relearning in Boltzmann Machines. In D. E. Rumelhart, J. L. McClelland, and PDP Research Group, CORPORATE, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, volume 1, pages 282–317. MIT Press, 1986.

[25] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 79, pages 2554–2558, April 1982.

[26] T. Hothorn, K. Hornik, and A. Zeilei. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

[27] Y. H. Hu. Lecture 38. Learning vector Quantization (LVQ). Lecture Notes, 2001. URL `http://homepages.cae.wisc.edu/~ece539/videocourse/notes/pdf/lec%2038%20lvq.pdf`.

[28] S. Joglekar. Using Circular Self-Organizing Maps to solve the Symmetric TSP, 2015. URL `https://codesachin.wordpress.com/2015/12/16/using-circular-self-organizing-maps-to-solve-the-symmetric-tsp/`.

[29] J. A. Kangas, T. K. Kohonen, and J. T. Laaksonen. Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1(1):93–99, March 1990.

[30] S. Kaski and K. Lagus. Comparing self-organizing maps. In C. von der Malsburg, W. von Seelen, J. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks*, volume 1112, pages 809–814. Springer Berlin Heidelberg, 1996.

[31] T. Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences*. Springer Berlin Heidelberg, 1989.

[32] T. Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences. Springer-Verlag Berlin Heidelberg, 3rd edition, 2001.

[33] T. Kohonen. Data Management by Self-Organizing Maps. In J. Zurada, G. Yen, and J. Wang, editors, *Computational Intelligence: Research Frontiers*, pages 309–332. Springer-Verlag Berlin Heidelberg, 2008.

[34] C. C. Lu and Y. H. Shin. A Neural Network Based Image Compression System. *IEEE Transactions on Consumer Electronics*, 38(1):25–29, February 1992.

[35] I. Marcic and S. Ribari. Comparison of a back propagation and a self organizing map neural networks in classification of TM images. *Internationl Archives of Photogrammetry and Remote Sensing*, 33:140–145, 2000. URL `http://www.isprs.org/proceedings/XXXIII/congress/part7/140_XXXIII-part7s.pdf`.

[36] MATLAB. *SOM Toolbox v2.1*. The MathWorks Inc., Natick, Massachusetts, 2015.

[37] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[38] R. Menzel and M. Giurfa. Cognitive architecture of a mini-brain: the honeybee. *TRENDS in Cognitive Sciences*, 5(2):62–71, February 2001.

[39] K. Moreland. Diverging Color Maps for Scientific Visualization. pages 92–103. Springer Berlin Heidelberg, 2009. URL `http://www.kennethmoreland.com/color-maps/ColorMapsExpanded.pdf`.

[40] S. Nakamura and T. Akabane. A neural speaker model for speaker clustering. In *1991 International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 853–856. ICASSP, April 1991.

[41] E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL `http://CRAN.R-project.org/package=RColorBrewer`. R package version 1.1-2.

[42] M. Paliwal and U. Kumar. Neural networks and statistical techniques: A review of applications. *Expert Systems with Applications*, 36:2–17, 2009.

[43] E. Pesonen, M. Eskelinen, and M. Juhola. Comparison of different neural network algorithms in the diagnosis of acute appendicitis. *International Journal of Bio-Medical Computing*, 40:227–233, 1996.

[44] M. Predota. *Prämienkalkulation in der Lebensversicherung : Einführung mit Beispielen aus der Praxis*. München : AVM, Impr. Meidenbauer, 2010. URL `http://katalog.ub.tuwien.ac.at/AC08049504`. Übungsbuch u.d.T. Prämienkalkulation in der Lebensversicherung.

[45] G. Press. 12 Big Data Definitions: What's Yours?, 2014. URL http://www.forbes.com/sites/gilpress/2014/09/03/12-big-data-definitions-whats-yours/#2a205abb21a9.

[46] A. Puddington and T. Roylance. Populists and Autocrats: The Dual Threat to Global Democracy. URL https://freedomhouse.org/report/freedom-world/freedom-world-2017.

[47] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2015. URL http://www.R-project.org/.

[48] M. Resta. Financial Self-Organizing Maps. In S. Wermter, C. Weber, W. Duch, T. Hoinkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, editors, *Artificial Neural Networks and Machine Learning*, volume 8681 of *Lecture Notes in Computer Science*, pages 781–788. Springer International Publishing, 2014.

[49] A. Reuß. Die Integration von Data-Mining in die Geschäftsprozesse von Versicherungsunternehmen. Presentation Notes, Universität Ulm, Sektion Aktuarwissenschaften, 2006.

[50] P. Sarlin and T. Eklund. Fuzzy clustering of the self-organizing map: Some applications on financial time series. In J. Laaksonen and T. Honkela, editors, *Advances in Self-Organizing Maps: 8th International Workshop, WSOM 2011, Espoo, Finland, June 13-15, 2011. Proceedings*, pages 40–50. Springer Berlin Heidelberg, 2011.

[51] S. Sayad. Decision tree - regression. URL http://www.saedsayad.com/decision_tree_reg.htm.

[52] J. Schmidt, C. Klüver, and J. Klüver. *Programmierung naturanaloger Verfahren*, volume 1. Vieweg+Teubner Verlag, 2010.

[53] S. Schmitt. "Ein Instrument der Verstehens". *DIE ZEIT*, 42:37, October 6th 2016. also to be found at: http://www.zeit.de/2016/42/big-data-wissenschaft-folgen-viktor-mayer-schoenberger/komplettansicht.

[54] SN/schwarz/shutterstock. Fehlersuche mit Thermografie, 2015. URL http://www.salzburg.com/nachrichten/rubriken/besteimmobilien/immobilien-nachrichten/sn/artikel/fehlersuche-mit-thermografie-137077/.

[55] P. Stefanovič and O. Kurasova. Influence of Learning Rates and Neighbouring Functions on Self-Organizing Maps. In J. Laaksonen and T. Honkela, editors, *Advances in Self-Organizing Maps: 8th International Workshop, WSOM 2011, Espoo, Finland, June 13-15, 2011. Proceedings*, Lecture Notes in Computer Science, pages 141–150. Springer Berlin Heidelberg, 2011.

[56] D. Stutz. tikz-voronoi-cells, 2016. URL https://github.com/davidstutz/latex-resources/tree/master/tikz-voronoi-cells.

[57] I. Sutskever, J. Martens, and G. Hinton. Generating Text with recurrent Neural Networks. URL http://techtalks.tv/talks/54425/.

[58] H. S. Tan and S. George. Investigating Learning Parameters in a Standard 2-D SOM Model to Select Good Maps and Avoid Poor Ones. In G. Webb and X. Yu, editors, *Advances in Artificial Intelligence*, volume 3339 of *Lecture Notes in Computer Science*, pages 425–437. Springer Berlin Heidelberg, 2004.

[59] D. Taniar. *Data Mining and Knowledge Discovery Technologies*. Advances in data warehousing and mining series. Idea Group Inc, 2008.

[60] O. Terzi. Monthly Rainfall Estimation Using Data-Mining Process. *Applied Computational Intelligence and Soft Computing*, 2012. URL `https://www.hindawi.com/journals/acisc/2012/698071/`.

[61] T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. URL `http://CRAN.R-project.org/package=rpart`. R package version 4.1-9.

[62] S. Turner. Use meaningful color schemes in your figures, 2009. URL `http://www.gettinggeneticsdone.com/2009/06/use-meaningful-color-schemes-in-your.html`.

[63] A. Ultsch and H. Siemon. Kohonen's Self Organizing feature Maps for Exploratory Data Analysis. In *Proceedings of International Neural Networks Conference*, pages 308–308, 1990.

[64] T. Vatanen, M. Osmala, T. Raiko, K. Lagus, M. Sysi-Aho, M. Orešič, T. Honkelae, and H. Lähdesmäkia. Self-organization and missing values in SOM and GTM. In P. Estévez and J. Principe, editors, *Neurocomputing*, volume 147, pages 60–70, 2015. Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012).

[65] Viscovery Software GmbH. Self-Organizing Maps, 2016. URL `https://www.viscovery.net/self-organizing-maps`.

[66] R. Wehrens and M. C. Buydens Lutbarde. Self- and Super-organising Maps in R: the kohonen package. *Journal of Statistical Software*, 21(5), 2007. URL `http://www.jstatsoft.org/v21/i05`.

[67] P. Westfall and S. Young. *Resampling-Based Multiple Testing*. John Wiley & Sons, Inc., 1993.

[68] H. Yin and H. Ni. Generalized Self-Organizing Mixture Autoregressive Model for Modeling Financial Time Series. In C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, editors, *Artificial Neural Networks – ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14-17, 2009, Proceedings, Part I*, pages 577–586. Springer Berlin Heidelberg, 2009.

[69] E. Zerz, U. Helmke, and D. Prätzel-Wolter. Mathematical Theory of Neural Networks. Lecture Notes University Kaiserslautern, University Würzburg, July 2001.

[70] J. Zirilli. *Financial Prediction using Neural Networks*. International Thomson Computer Press, 1st edition, 1997.