Master Thesis

Customer Segmentation Algorithms in Financial Services

Florian Lorber

Graz, January 2013

Institute for Software Technology
Graz University of Technology

Supervisor/First reviewer: Univ.-Prof. DI Dr. Prof. Alexander Felfernig

Abstract

This thesis worked on improving the mechanisms to retrieve a suitable database for *case based reasoning*. This is a method to decide actual questions relying on the data of previous similar situations stored in a database. Refining that data base leads to more accurate results. In this thesis, the data of actual customers was taken to predict the behavior of new customers. This was done in cooperation with a financial service provider, using real data.

To prevent working with the data set of all customers at once, which is more time intensive and less effective, clustering was used to retrieve the most relevant users as an expressive base for the case based reasoning. In this thesis the Algorithm "Affinity Based Clustering with Intelligent Weights" (ABCIW) was developed, by adopting the "Affinity Based Clustering Algorithm" from the field of computer vision to the field of data mining and enhancing it with a trained weighted distance function. Using this algorithm, a subset of sample users could be created for case based reasoning that achieved a significant raise in both recall and precision, which are the commonly used measures to measure the efficiency of data mining algorithms.

To evaluate ABCIW, it has been tested in a detailed study on different data sets and against different other algorithms.

German Abstract

Im Rahmen dieser Arbeit wurden das Gebiet Data Mining und die Technik *Case Based Reasoning* behandelt. Es wurden Wege gesucht, geeignete Fälle für fallbasierte Schlussfolgerungen zu bestimmen, mit denen Firmen das voraussichtliche Kaufverhalten zukünftiger Kunden anhand der Daten schon bestehender Kunden vorhersagen können. Um nicht alle Kunden für diese Vergleiche heran ziehen zu müssen, wurden mittels Clustering signifikante Kunden als Fallbasis ausgewählt. Weniger Datensätze in der Fallbasis führen zu kürzeren Berechnungszeiten und bei einer richtigen Auswahl der Fälle auch zu signifikant besseren Ergebnissen. Nach der Untersuchung mehrerer Methoden wurde der Algorithmus "Affinity Based Clustering with Intelligent Weights" (ABCIW) entworfen und implementiert. Dieser ist eine Erweiterung des Algorithmus "Affinity Based Clustering", welcher ursprünglich aus dem Gebiet der Computer Vision stammt. Um den Algorithmus zu verbessern, wurde er mit einer gewichteten Distanzfunktion kombiniert, welche nach entsprechendem Training eine deutliche Verbesserung von Recall und Precision, den beiden gängisten Maßen für die Effizienz von Data Mining Algorithmen, erreichen konnte. Der Algorithmus wurde an einem realen Datensatz aus der Wirtschaft getestet und erreichte dabei bessere Ergebnisse als vergleichbare Algorithmen aus dieser Branche.

Dedication

First of all, I would like to thank my supervisor, Dipl.-Ing. Dr.techn. Prof. Felfernig, for his prompt and informative answers to all of my questions, his inspiring suggestions and in general for all the help he gave me.

I would also like to thank Dipl.-Ing Christoph Zehentner, for introducing me into the topic and helping me get through the toughest parts.

And of course I want to thank my parents, for both the mental and the financial support they gave me through the years of study.

Florian Lorber
Graz, 2013

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Place, Date Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Ort, Datum Unterschrift

Contents

Chapter 1

Introduction

1.1. Motivation

The major motivation for participating in the project presented in this thesis consisted of two parts:

- **The interest in working on a practical project, direct from economy.**
  Due to the steadily increasing amount of data that companies gain from their customers, the need for efficient methods to process the data grows increasingly stronger and stronger. It is a problem that concerns companies in any business.

- **The interest in exploring a state of the art research topic.**
  There is still no universal solution to the occurring problems, even though research on this topic has dramatically increased in the last decade. There are plenty algorithms to any subject of data mining, but there are still a lot of open questions and contributions that can be achieved.

1.2. Goals

The general goal of this thesis was to improve the effectiveness of the customer analysis approaches already exploited by the projects partner financial service providers. When this thesis started, the project was already quite advanced and it was already determined, that case based reasoning (Aamodt and Plaza (1994)) was so far the best approach to solve the task of customer behavior prediction. Since the quality of case based reasoning is restricted by the quality of the database it uses, the main goal was to find a suitable algorithm to determine expressive customers for that database. This was achieved by clustering (Chapter 2.8) the customers into groups with similar attributes. The most significant users of each group, the so called cluster centers, built the data base for case based reasoning.

The work done for this thesis consisted of four steps:

- **Investigation of clustering algorithms**
  Exploring different clustering algorithms in order to find most suitable ones.
- **Implementation of the algorithm of choice**
  In this case, this has been the Affinity Based Clustering Algorithm (Frey and Dueck (2007)).
- **Improvement of the Affinity Based Clustering**
  By adding a genetically optimized weighted distance function, it was enhanced to the algorithm "Affinity Based Clustering with Intelligent Weights".
- **Evaluation of the algorithm**
  Testing the algorithm against the original algorithm, against others and on different data sets.

1.3. Contribution

The main work done during this thesis was the evaluation of different clustering algorithms for customer segmentation. That led to the discovery of the affinity based clustering algorithm for this research area. It is one of very few algorithms that really takes all instances at the same time into account as cluster centers. It starts with the distance of each data point to every other data point, to determine whether one of them could be a suitable cluster center of the other one. Then it compares the values and the more data points consider point A to be their cluster center, the higher is point A's value for data point B. Then in iterations, those values are refined and finally lead to a small and precise amount of cluster centers. The number of cluster centers does not have to be defined manually as in most other clustering algorithms, but is computed by the algorithm.
The contribution achieved during this thesis was the porting of affinity based clustering to the field of customer segmentation and enhancing it by a genetic training of a weighted distance function. The genetic computation of the weights was achieved by a combination of **a)** a simple genetic algorithm (Mathew (2005)), to come near to local maximums and b) a hill climbing algorithm (Mitchell et al. (1993)) to reach the exact maximum weights once the genetic algorithm was close enough. The evaluation on different real world data sets was done by case based reasoning using the cluster centers as a basis and predicting the buying behavior of customers. The data sets were split into two parts. The first part was processed by the algorithm to train it. The customers of the second part, whose actual behaviors were already known, but not accessible for the algorithm, where used to evaluate it. The predicted behavior was then compared with the real behavior and the results were measured in terms of recall and precision (Makhoul et al. (1999)). It showed that clustering the data using basic affinity based clustering already increased the case based reasonings effectiveness compared to data clustered by k-means clustering (Berkhin (2002)) and the genetic optimization could raise the case based reasoning results by up to 10 %.

1.4. Overview

The second chapter of this thesis deals with related work. It gives a basic overview of the topic ”Data Mining”, by explaining different classifiers, different clustering algorithms and evaluation measures for data mining. It also explains the machine learning technique case based reasoning. It also gives an overview of the results that have already been achieved during this project, before the clustering algorithm had been implemented.

The third chapter will deal with ”Affinity Based Clustering”, which describes the origins of the algorithm and its advantages. It also shows a few examples of where the algorithm has already been used. A main part of the chapter will explain, what kind of changes were made to the algorithm and how that improved the results, compared to the original version. Some different distance functions are discussed and the genetic training of the Weighted Euclidean Distance is described.

Chapter 4 analyses the results that were accomplished and evaluated on different data sets. Chapter 5 will discuss issues for future work and, finally, the conclusions of the thesis will be summarized in Chapter 6.

Chapter 2

Related Work

2.1. Data Mining

A good percentage of every paper that focuses on Data Mining, (e.g. Witten and Frank (1999); Fung (2001)), starts the introduction with the statement, that the amount of data that is accessible in the modern world has increased incredibly fast and will grow even faster in the future. But the more data is accessible, the higher the need for methods to extract the relevant parts of the information. It has become a huge part of the research in artificial intelligence to train our computers on filtering and interpreting the accessible data and turning it into a representation understandable and usable to humans.

Definition

> Data mining is part of knowledge discovery which deals with the process of identifying valid, novel, potentially useful and ultimately understandable patterns in data. (Pal (2004))

These patterns might be similarities in face recognition, important paragraphs in text mining, or, like in this case, patterns to predict the future behavior of customers. Data Mining usually deals with enormous amounts of data, too big to process by hand. Consequently the used algorithms have to be able to manage these amounts of data and to process them in reasonable time.

This thesis will concentrate on the most business relevant side of Data Mining, that, according to (Olson and Delen (2008)), is separated into three parts:

- **Customer Profiling:**
  identifying those subsets of customers most profitable to the business
- **Targeting:**
  determining the characteristics of profitable customers who have been captured by competitors

- **Market-basket analysis:**
  determining product purchases by consumers, which can be used for product positioning and for cross-selling

For financial service providers, ”Customer Profiling” is the most useful part, because every new customer can be assigned to a suitable subset of customers with a similar buying behavior and the company can determine, which kind of product it should offer to the new customer. Reducing the amount of useless advertising and informing a sales representative with the topics their client might be interested in, can save time and help to earn a lot of money.

Since there are many different approaches to customer profiling, there is a need for a general evaluation measure, that can evaluate the quality of a data mining algorithm. Therefore, (among others) two main measures have been defined: Precision and Recall. (Makhoul et al. (1999))

> ” *Recall* is the proportion of True Positive cases that are correctly Predicted Positive. *Precision* denotes the proportion of Predicted Positive cases that are correctly True Positives. ” (Powers (2011))

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2.1}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.2}$$

	positiv	negative
predicted positive	True Positive	False Positive
predicted negative	False Negative	True Negative

Table 2.1.: Definition of ”True Positive”, ”False Positive”, ”False Negative” and ”True Negative”.

Before the next example, that is gonna show the calculation of precision and recall, one has to understand that the classification algorithms predict only one behavior at a time, so for this behavior (e.g. ”The customer will buy the product”) it returns ”true” or ”false”. If it predicts ”true” and the customer really buys the product, it predicted ”True Positive”, if the customer does not buy, it predicted ”False Positive”. Table 2.1 visualizes the definitions of the different possible combinations of predictions and outcomes. Now, if the algorithm predicts the behavior of 10 new customers and 3 of them were predicted positive, but only 2 of those really buy the product and two that have been predicted negative buy it as well, precision and recall can be calculated.

The precision in this example equals the ”True Positive” (=2) hits divided by the ”True Positive” (=2) plus the ”False Positive” (=1) so it is 2/3.

The recall is ”True Positive” (=2) hits divided by ”True Positive” (=2) plus the ”False Negative” (=2), so it is 2/4.

It is easy to increase precision at the cost of recall and vice-versa: just predicting that everybody buys the product would raise *recall* to 100%, but the cost of *precision* would be enormous. So, depending on the original purpose a company has in mind, they have to find a suitable algorithm, that offers the desired relation between the two values.

A company has to decide for themselves, whether they want to exaggerate advertisement (use an algorithm with high recall) and send drop mail to some people who will not buy the product, or if they accept the risk of loosing some potential customers, by sending as few advertisements as possible (use an algorithm with high precision).

The two measures also have been combined, to create a single measure to scale the quality of the algorithms in one value. The most common combination is the f-measure, that is produced by the harmonic mean of precision and recall (Sasaki (2007)):

$$F = \frac{2 * P * R}{P + R} \tag{2.3}$$

Since those measures are used in various research fields, there are a lot of variations to satisfy the different goals (Mizzaro (2001)). But for the evaluations in this project, precision, recall and the original f-measure will be sufficient.

2.2. Machine Learning

Although machine learning uses a lot of common algorithms with data mining, there is quite a difference between the two terms:

- Data Mining
  The process of discovering useful patterns, automatically or semiautomatically, in large quantities of data.
- Machine Learning
  Things learn when they change their behavior in a way that makes them perform better in the future.
  (Witten and Frank (1999))

According to this definition of machine learning, the learning algorithms of interest in this thesis are data mining algorithms that are able to improve their results over time. In this sense most of the methods described in this thesis are part of both fields, Data Mining and Machine Learning.

Machine Learning algorithms can be divided into two classes: supervised and unsupervised. (Kotsiantis (2007))

Supervised algorithms do not only get the data points, but also the labels that they shall be matched to. So, if a new data point is retrieved, the algorithm has to determine which label to put on it. For

example, customer segmentation is the separation according to the buying behavior, hence the labels could be ”extravagant”, ”average” or ”economical”. Supervised learning algorithms applied to that example can connect each customer to one of those labels.

Unsupervised algorithms only get the data points and, therfore, have to decide for themselves which connections might exist between them. They might form groups of customers as well, but not according to pre-defined labels but according to their internal rules, that might not be comprehensible to observers. The most common unsupervised algorithm is clustering, which will be explained in detail in the last section of this chapter.

2.3. Case Based Reasoning

Case Based Reasoning is a method being used by humans every day. It’s the processing of actual or future events and problems by comparing them with other previously experienced and solved problems, using methods that have worked and avoiding those that have failed. A doctor, for example, uses Case Based Reasoning by comparing the symptoms of a patient with the symptoms of patients he has already cured, in order to find a suitable treatment.

Usually, Case Based Reasoning (CBR) consists of the following steps:

- Occurrence of a new problem
- Retrieval of the most similar case
- Reuse of the solution that worked in the found case
- Evaluation of the result
- Memorizing the case (if the evaluation was good)

As described by (Aamodt and Plaza (1994)), Case Based Reasoning is a problem solving technique, that has a fundamental advantage compared to most of the other techniques: It does not need to know about the domain it is used in, nor does it have to calculate how and why a solution helped on a previous problem, it simply searches for the case that was most familiar to the actual one and tries to apply the same solution that was used there.

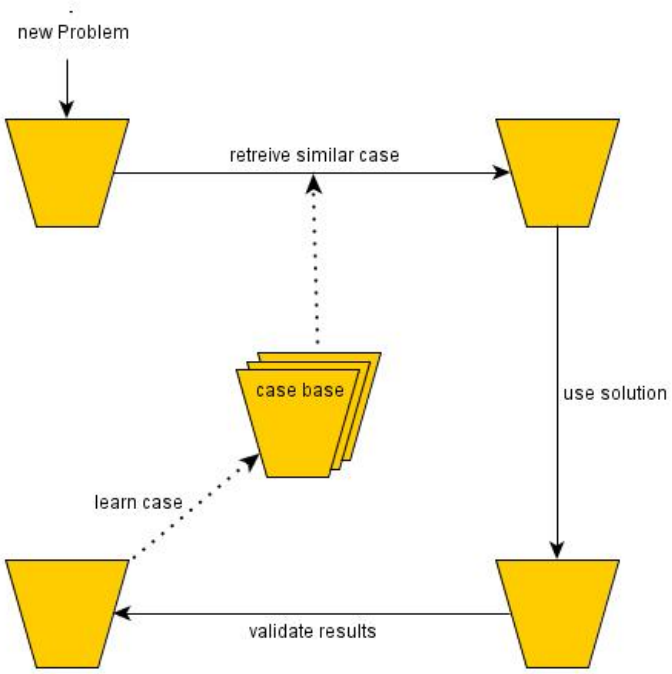

Figure 2.1.: Illustration of Case Based Reasoning, based on (Watson and Marir (Watson and Marir)).

Applied to Customer Profiling, this means that CBR can search for the customer in the data base which is most similar to a new client and predict the buying behavior of the new client, by comparing to the behavior of the customer in the data base.

Consider Table, 2.2, that gives a very simplified, small example of a customer data base.

ID	Age	Income	Married	Debts	Classifier
0	20	1200	yes	0	true
1	43	2300	no	5000	false
2	34	1700	no	0	false
3	22	0	yes	3000	false
4	51	3100	no	0	true

Table 2.2.: Example customer database, there are no real attributes and no real values used. The customers where classified by whether or not they have a live insurance policy.

Now imagine a new customer, with the attributes shown in 2.3.

ID	Age	Income	Married	Debts	classifier
5	33	2000	no	2000	?

Table 2.3.: Example Customer for classification.

There are many different ways to calculate the similarity between customers, and this will be discussed in its own subsection in Chapter 3.4.1. For the purpose of explaining CBR, it should be enough to compare the customers by manually comparing their attributes. The most similar customer in the data base is ID 1. He/she is the second most equal in age, has the most equal income, they are both not married and both have debts. So to classify the new customer, the behavior of customer ID 1 is used. Since he/she has no insurance policy, the result of the prediction is *false*.

2.3.1. Advantages

The main advantage of CBR is the fact that it works even if there is no prior knowledge on the topic and if there is no algorithmic solution to a problem. Other advantages, as listed in (Kolodner (1992)), are:

- CBR provides a way to evaluate the current results, even if there are too many unknowns yet, to evaluate the solution in common ways, e.g. if no algorithmic method can be used. The evaluation can use the values of the reference case, replacing its missing parameters with the parameters that were evaluated in the old case.
- The experience of previously saved cases also includes the experience of the mistakes that have been made, or the problems that occurred. A well built system might alert the user to complications that might arise and it might even suggest ways to avoid them.
- Also, each new problem that has been solved serves as a new case base in the future, so the algorithm is learning and improving over time, becoming more accurate and precise.

Another important advantage is the explanation part: CBR does not only assist with a solution to upcoming problems, it can also show how this solution has already solved a similar problem, making the decision much more comprehensible. If there is any doubt on the advice that CBR is giving, it can be massively reduced by showing cases where the exact same advise had already helped.

2.3.2. Disadvantages

The primary disadvantage of CBR is that two customers with the exact same attributes may still decide totally differently on whether to buy a product or not. So, taking a single case as a basis of the CBR might, regardless how high the similarity is, still have the possibility to fail. Fortunately there is a solution to that, which will be discussed in the next chapter: Data Preprocessing by Clustering. By

centralizing certain groups of customers to single representative exemplars, statistical outliers with unusual behavior are not taken into account.

Another problem caused by the large amount of data that has to be processed is the needed computing power. Imagine a data base of fifty thousand customers, each having a hundred different attributes: This can slow down the system, making it nearly unusable. Again, this problem is reduced if the data is clustered, since due to the clustering, that can be done off-line before the CBR is executed, only the cluster centers have to be taken into account, when searching for a similar case to be used as a basis for the CBR.

Errors in the database might also tend to worsen over time, if new cases are not validated after classification. If one faulty case is used to classify another case, the error will duplicate, raising its probability to be used as case base. If a clustering is executed on the data base, that also increases the chance that a cluster might build up around the faulty bases.

2.4. Other classifiers

Classifiers are formal methods that try to deduce the classes that new data instances belong to, by observing and comparing the attributes of the new and old instances. They are initialized by getting the labels of the possible classes and previous instances for which the classes are already known (Michie et al. (1994)). In astronomy, for example, this might mean the connecting of new found space objects with labels like ”star”, ”supernova” or ”planet”, while in customer segmentation it is the connecting of customers to labels like ”extravagant” or ”economical”.

CBR is the classifier that was used in the practical part this thesis. But to give a good overview of the topic, the next subsections will describe other common classifiers.

2.4.1. Linear classifiers

A linear classifier classifies by applying a linear combination to the attributes of the data instances, meaning that the attributes are multiplied with certain constants. These constants are determined by the previous data points and can be calculated in different ways, depending on the used algorithm. The most common linear classifiers are probably the Fisher Linear Discriminant and the Euclidean Distance Classifier (Skurichina and Duin (2002)).

Generally, linear classifiers are divided into two subclasses, generative and discriminative classifiers (Ng and Jordan (2002)):

- **Generative Classifiers**

  Generative classifiers learn a model of joint probability, p(x,y), of the inputs x and the labels y. Then they use the Bayes Rule, to calculate p(y|x) and determine the label that most likely fits to the input. Algorithms that belong to this class are the Fishers Linear Discriminant and Naive Bayes Classifier. (Rubinstein and Hastie (1997))

- **Discriminative Classifiers**

  Discriminative Classifiers learn a mapping from the inputs to the possible labels, trying to maximize the quality of the output on a given training set. Some examples are the support vector machines or logistic regression. (Rubinstein and Hastie (1997))

2.4.2. Neural Networks

Artificial Neural Networks imitate their natural counterparts: They consist of artificial neurons and weighted, directed connections between them (2.2). Messages are sent over the connections and amplified or inhibited according to the weights of the connections. During the training phase, data is sent over those connections, updating the weights and enhancing the input / output relationship (Kriesel

(2007)). With the attributes of the customers of the training data as input and their classification as output, neural networks can sense relations in the training data.

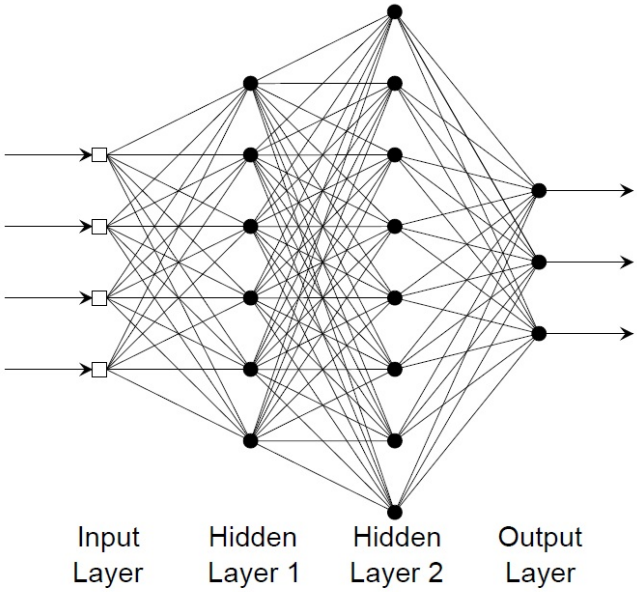

Figure 2.2.: Neuronal network with multiple layers, taken from (Dony and Haykin (1995)).

Neural Networks are used in many different fields of study and also have some huge advantages when used for classification (Zhang (2000)): They are data driven, self-adaptive methods, that can update and enhance without needing any knowledge of the underlying model. Being nonlinear models, neuronal networks are flexible enough to model complex real world relationships between data points. More importantly, they can estimate posterior probabilities that provide the basis for establishing classification rules.

The biggest disadvantage, however, is the fact that the results are very hard to explain by common reasoning. As a consequence, tools using neural networks often have problems being accepted in the real world.

2.4.3. Hidden Markov Models

Hidden Markov Models are a powerful tool to model processes that vary over time. They consist out of two parts: the hidden process, that is a Markov Chain consisting of states and transition probabilities and the observable process, that produces observable emissions at each moment, according to a state-dependent probability distribution. (Chai and Vercoe (2001); Dymarski (2011))

The most common example for Hidden Markov Models is the weather prediction (Rabiner (1990)). Consider a Markov Model of the weather, measured by three definitions: sunny, rainy and cloudy

and let there be a transition probability matrix A (2.4) that states the probabilities for each weather to occur, depending on the actual weather.

$$A = \begin{pmatrix} R>R & R>C & R>S \\ C>R & C>C & C>S \\ S>R & S>C & S>S \end{pmatrix} = \begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 0.3 & 0.3 & 0.4 \end{pmatrix} \tag{2.4}$$

Then for a specific weather on one day, the probability for a chain of weather conditions over the next days (e.g.: rainy, sunny, cloudy, sunny) can be calculated. Assuming we know that it rains on the first day (hence the probability for the rain is 1), the probability for sunshine after rain, according to A, is 0.5. Then the probability for clouds after a sunny day is 0.3 and sunny days after clouded days occur at a chance of 0.4. So the over all chance for that chain is 1 * 0.5 * 0.3 * 0.4 = 0,06 = 6%.

Yet, though they belong to the most powerful classifiers, they can not be used for customer analysis, for they need timed or at least sequenced observations to work. The most common usages are speech recognition, or document evaluation (where one letter / word follows the other and the probability for the next one changes depending on the actual one).
The only field, where Hidden Markov Models can be used for customer classification is the field of Customer Relationship Management (CRM) (Sepideh and Aaghaie (2011)), where the data of sequenced purchases are given. CRM tries to establish a long time relationship between the customers and the companies, and therefore it tries to keep track of the ”state” the customers are in. If they are satisfied / loyal they are probably going to stay at the company, but if not, the company has to try to persuade customers that turn towards disloyal.

2.5. Former Results

As mentioned in the introduction, the data mining project of our industrial partner was already quite advanced, when this thesis started. This section is about the results, that were already achieved, taking the information from (Zehentner and Bulic (2011)). The evaluation was performed on the same financial service data set, that was used for evaluation purposes in this thesis. The only difference in the data was, that the data used in the former experiments was preprocessed by attribute selection, using the method of correlation-based feature selection (Hall (1999)). The idea behind that approach is that meaningful features were highly correlated with the outcome of the classification, but as uncorrelated with each other, as possible. This reduced the data to the 91 main attributes which contain the most useful information, without worsening the results.

The main processes used in the original study were CBR with a randomly picked case base, Chaid Tree, Random Committee and the J48 Tree, which will briefly be described in the following subsections.

2.5.1. Chaid Tree

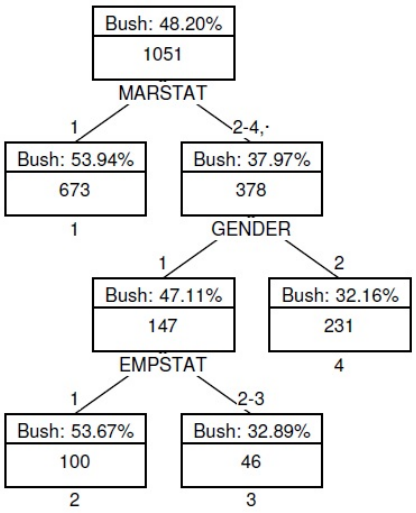

Figure 2.3.: Illustration of the Chaid Tree, taken from (Magidson and Vermunt (2004)).

This procedure, the Chi-square Automatic Interaction Detector, had already been used by the financial service provider and it was the only data mining support they had had before the project started. It creates a decision tree, that shows which variables have the greatest influence on the outcome of the prediction. According to (Wilkinson (1992)), the Chi-square Splitting Criterion that was used in the algorithm, was used to increase the time performance, on cost of accuracy. But since the time performance turned out to be in a suitable range anyway, this made the Chaid Tree an inferior version compared to other automatic interaction detector trees. Its main benefit is the advantage all decision trees have compared to other methods: The easy explanation of the result, as it can be seen in the graphical representation of the tree (see Figure 2.3).

(Magidson and Vermunt (2004)) provide a good example of the graphical representation with a Chaid tree applied on voting. It is shown in Figure 2.3 and deals with the voting behavior of 1051 citizens, with 48% of Bush supporters. The three splits that are done depend on *marriage status*, *gender* and *employment status*. Marriage status is separated into four sections (1=”married”, 2=”widowed”, 3= ”divorced”, 4= ”never married”), the sections 2 to 4 being merged in the tree. Gender obviously has two possible outcomes (”male” and ”female”) and the employment status is separated into ”employed”, ”unemployed” and ”retired” (”unemployed” and ”retired” are again merged). So the first split separates the married (673) from the unmarried (or divorced / widowed) (378) voters and the second split separates male (147) from female (231). The last split isolates the employed male (100) from the unemployed (or retired) male (46). The leaves (end-nodes) of the tree are numbered according to their values, the leaf with the highest percentage (53,94 %) getting the lowest number. Knowing the basic structure it was built from, the graph easily shows that the best sectors for Bush

are the married voters and the unmarried but employed males.

2.5.2. J48 Tree

J48 is the WEKA implementation of the C4.5 Algorithm, a decision tree learning algorithm. According to (Kohavi and Quinlan (1999)) it uses case bases like CBR and in each node of the tree it divides the cases by the attribute giving the highest information gain. Information gain increases, if the probabilities for the different decisions are mostly alike. Maximum information gain means, that the resulting subsets are of nearly equal size. In contrary to the Chaid tree, J48 trees do not try to divide into the subsets with the most different prediction outcome, but into the subsets that split up the overall information the best. Table 2.4 shows example data that could be used to train a J48 Tree, the corresponding tree can be seen in Figure 2.4. The first separation is done using the marriage status as criterion, since the two resulting subset are nearly of same size (2/3), while using gender would have split into two subsets of size four and one and using income would have split into three sets of size one, two and two. The married subset, containing three customers, is then split using the gender as criterion. The returning subsets, two males and a female, share the same classification outcome, hence the algorithm does not process them further. The unmarried subset it divided by income, leaving two subsets of size one.

ID	Gender	Income	Married	Classifier
0	male	none	no	true
1	male	low	yes	false
2	female	high	yes	true
3	male	low	no	false
4	male	high	yes	false

Table 2.4.: Example data for building a J48 tree, the classifier defines whether a customer has a life insurance policy.

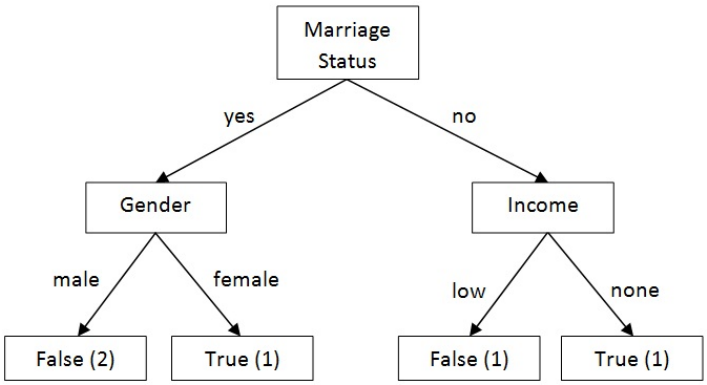

Figure 2.4.: The J48 tree for the data in Table 2.4.

2.5.3. Random Committee

Random Committee (Guo et al. (2006)) is defined as a collection of algorithms that are randomly chosen from an over-all pool of available algorithms. The random committee in this case consisted of an amount of randomly chosen classifiers, taken from all the classifiers that were supported by the used framework, WEKA. The framework will be described in detail later on (Section 2.6).
Since the committee was chosen randomly in each test run, the used classifiers can not be described here.

2.5.4. Results

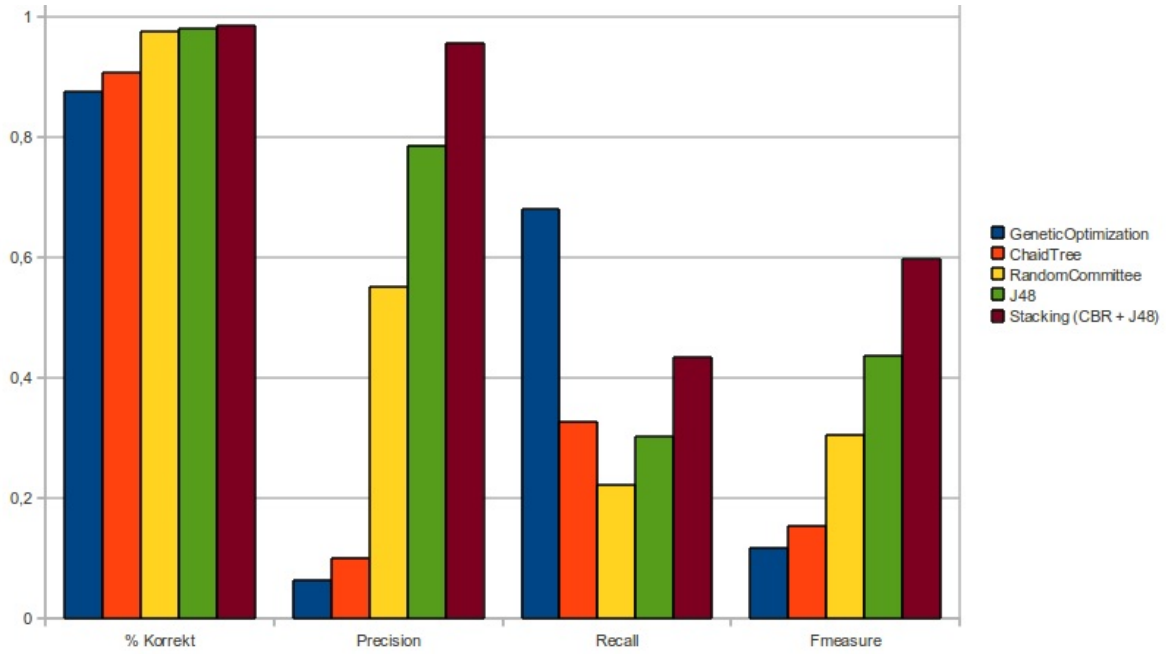

Figure 2.5.: Statistic taken from ((Zehentner and Bulic, 2011, p. 22)).

Figure 2.5 shows the results of the presented methods, measured in terms of precision, recall and the f-measure. Since the J48 Tree resulted in a high precision and the CBR showed a good recall, it was a logical next step to combine them. The two methods were stacked, meaning that both CBR and J48 were trained on the same part of the instances and a third classifier decided which of the methods to use for classifying a specific new data point. This led to the best results in terms of precision and f-measure and to the second best results in terms of recall.

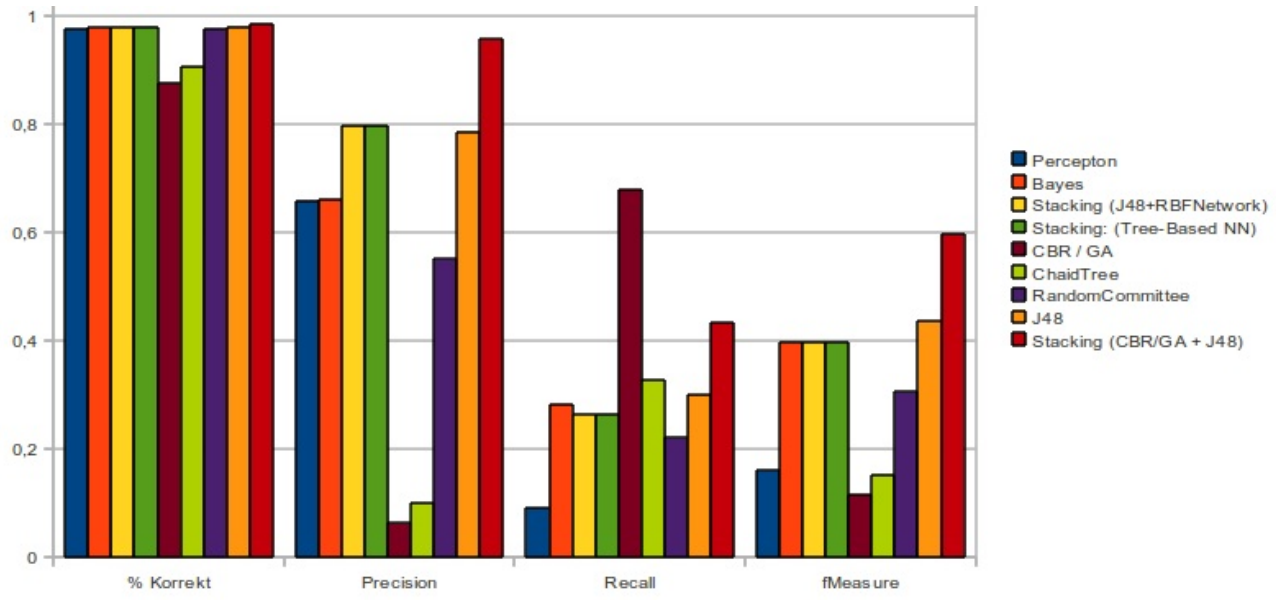

Figure 2.6.: Statistic taken from ((Zehentner and Bulic, 2011, p. 23)).

Stacking was also compared to a series of other stackings and to other learning methods, like neuronal networks, or Bayes classification. Still it could exceed all other methods, as shown in Figure 2.6. So the best way to improve the results was to improve case based reasoning or the J48 tree. And since both of the algorithms strongly rely on the cases they used for their training, which were so far randomly picked, the focus of this thesis was to improve those cases.

2.6. WEKA

WEKA (Waikato Environment for Knowledge Analysis) is an open source Java framework for data mining, that originated in a project of the University of Waikato, New Zealand, in 1992. According to (Hall et al. (2009)), it was developed due to the need of a universal framework, that would allow easy and universal access to different state of the art algorithms for data mining and machine learning.

Besides all the included algorithms, the framework provides an infrastructure for data loading, saving and manipulation, for the evaluation of results and for extending it with new algorithms. This made it to one of the most accepted tools, both in academia and in business. In addition, it comes with a book, (Witten and Frank (1999)), that does not only give a great introduction into WEKA, but also into the whole topic of data mining and the most common algorithms.

Big frameworks like WEKA also help to introduce standards for any input and output types: E.g. it can read nearly any form of data, since you can implement your own open and save functions. Yet if you use the format recommended by WEKA (arff) you can use the ArffViewer, a graphical interface to view the files, no matter how big they are. And there is an already implemented way to open and save them with only one command.

The most used part of WEKA is the explorer (Figure 2.7), a graphical interface that gives a quick and easy access to most of the provided functions of WEKA. The explorer is separated into 6 sectors as can be seen in Figure 2.7:

1. **Preprocess:** This section allows to load *arff* - files, databases or simple spreadsheets. It gives a numerical and graphical representation of the distribution of the instances for each attribute. Also, the data can be preprocessed here by applying filters.
2. **Classify:** The classification section is where most of the magic is done. One can choose between about 100 provided algorithms and any algorithm that was manually implemented and integrated into Weka. It allows different options regarding the validation, gives detailed output in form of confusion matrices (Kohavi and Provost (1998)) and calculated precision, recall and f-measure. A confusion matrix has the size L x L where L is the amount of possible labels. It compares the predicted labels with the actual labels, giving a good overview on how many predictions failed. In Section 4 some confusion matrices can be seen.
3. **Cluster:** There are already about 10 different clustering algorithms implemented in WEKA, which can be executed in this tab. WEKA can also evaluate the returned clusters, by splitting the data into a training and a test set.
4. **Associate:** Associators try to find rules in the data that use subsets of the attributes to conclude about other attributes. An example of such a rule would be that customers who buy bread and honey most probably also buy butter. (Agrawal and Srikant (1994)) states that those rules are very useful for cross-marketing, catalogue design and many other applications. Again, there are several different algorithms provided.

5. **Select attributes:** In this section different algorithms are provided which help to reduce the number of attributes. They filter out unimportant attributes and leave only attributes that have enough influence on the results of the classifiers.

6. **Visualize:** The last section proides a statistical visualization of the data set using scatter-plots.

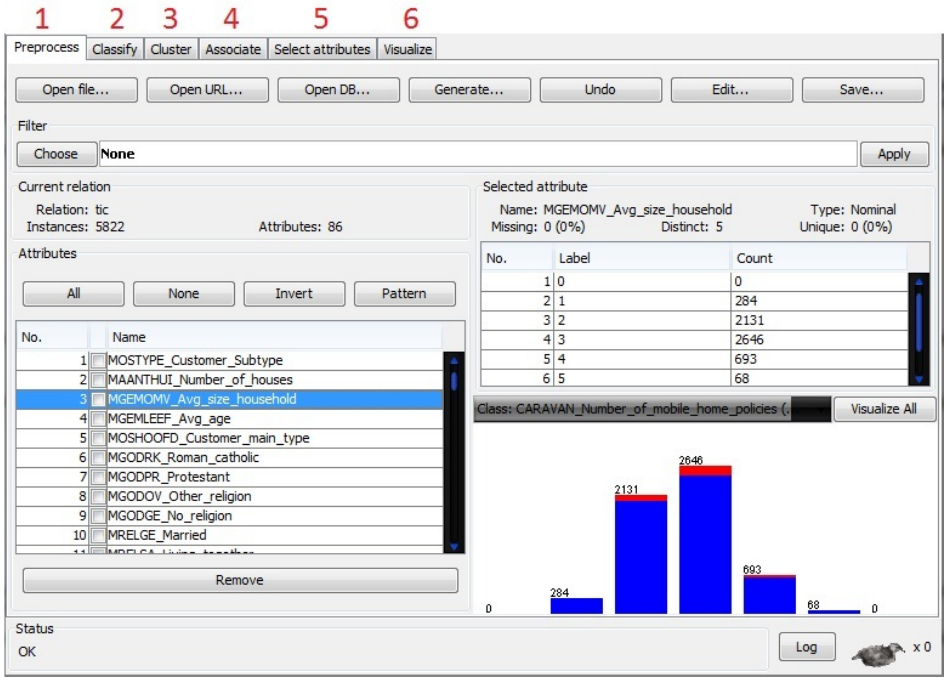

Figure 2.7.: Screenshot of the front screen of the Weka Explorer, with a dataset from the CoIL Challenge 2000 (http://www.liacs.nl/ putten/library/cc2000/report2.html). It visualizes the distribution of the data set according to the criteria of the size of their households.

Additionally WEKA also offers a second sector, the Experimenter. The WEKA experimenter is provided with an own tutorial of 40 pages, (Scuse and Reutemann (2008)), and allows users to define their own experiments, which e.g. could be the comparison of different algorithms. Then it can automatically evaluate the results of each of them, comparing them one to the other. It can also be parameterized with a chosen amount of iterations and it can be executed on different data sets.

2.7. Clustering

Clustering is an unsupervised machine learning technique that takes a large amount of collected data and tries to divide it into separate regions, clusters, according to it's attributes, sorting similar data items into the same group.

To represent the idea of clustering graphically, usually 2D data sets are used (see Figure 2.8). Similarity is then defined as the distance between the data points.

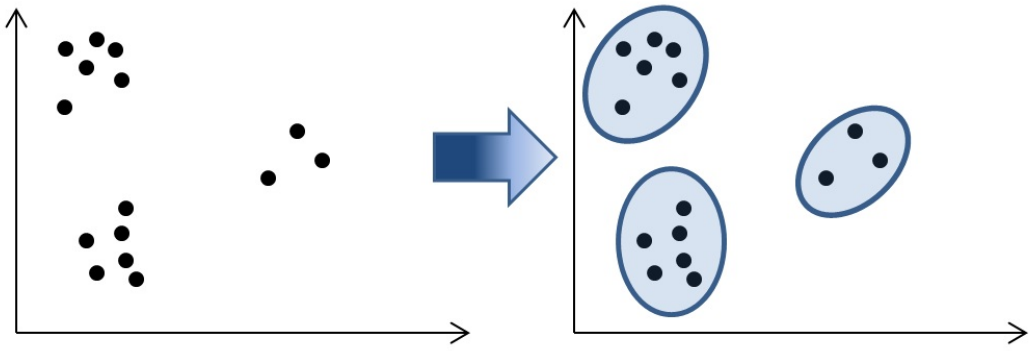

Figure 2.8.: Clustering of 2D points.

The mathematical definition of clustering is the following (see (Fung (2001))) :

> Let $X \in R^{m \times n}$ be a set of data items representing a set of m points x_i in R^n. The goal is to partition X into K groups C_k such that every data belonging to the same group are more "alike" than data in different groups. Each of the K groups is called a cluster.

Each of the clusters can then be represented by a cluster center, also called exemplar: The cluster center is the data point with the most characteristic traits for the cluster. So when a new data point is available, its distance to all the cluster centers is calculated. The center with the least difference determines the cluster the new data point will fit into.

In face recognition (see (Beumer et al. (2006))) the computer is able to distinguish features such as size, position and color of the eyes, color of the skin, etc. and then categorize them into common clusters. For example, a cluster could then consist out of the data of all Caucasian faces with blue eyes. The cluster center would then be the data of the face with the most common Caucasian skin color and blue eyes in the most average position.

In text mining documents are clustered according to their most common key words and a text using the phrases "data mining" and "clustering" in every second sentence will most probably be sorted into the cluster about data mining, if there are enough texts about that topic so it will generate its own cluster.

Clustering is a very important topic in data mining, because the part that is time intensive (the calculation of the clusters on training data), has to be done only once. Also the resulting data set can be used to classify new exemplars be used as a base for other algorithms as well. Since most of the data mining algorithms are very time consuming, reducing the data to only the cluster centers saves a lot of time. With a good clustering algorithm it can also improve the results of algorithms that run on the cluster centers instead of the full dataset.

According to (Fung (2001)), finding the ideal approach to a clustering problem means finding the global solution to a non-linear optimization problem and therefore it is an np - hard problem. So the goal is not to find the perfect solution, but to accomplish the best compromise possible.

2.7.1. K-Means Clustering

K-Means Clustering one of the most widely used clustering algorithms. (Berkhin (2002)) stated in the year 2002, that it was ”by far the most popular clustering tool used in scientific and industrial applications.” It can be applied to most of the data sets and it usually gives a reasonable good result. It therefore provides a good basis to compare against other algorithms.

(Wagstaff et al. (2001)) describes the functionality of the algorithm as follows:

1. Define the amount of clusters, k.
2. Assign each instance to its closest cluster center.
3. Update each cluster center to be the mean of all the instances assigned to it.
4. Repeat step 2 and 3 until there is no further change.

Depending on the starting position of the cluster centers, the results can diverge, since the algorithm only returns local minima. There are several different ways on how to set up the starting configurations. The most common ways are to start them at random positions, or to start them equally distributed.

For easier understanding, a graphical representation of the algorithm is shown in Figure 2.9, on the basis of an easy example of two dimensional data points. The similarity between the data points is simply calculated by their distance and the cluster centers are initiated randomly. In this example, the cluster centers do not need to be actual data points, but are freely selectable at any position. The initial two cluster centers (the unlabeled dots) where placed randomly. In the first step, the instances A and B are assigned to the upper cluster center, and the remaining three instances are assigned to the lower center. Then the centers are placed center between their assigned data points. So the center that A and B where assigned to lies exactly on the middle between them. Then the algorithm repeats these two steps one more time to reach the optimal positions for the cluster centers.

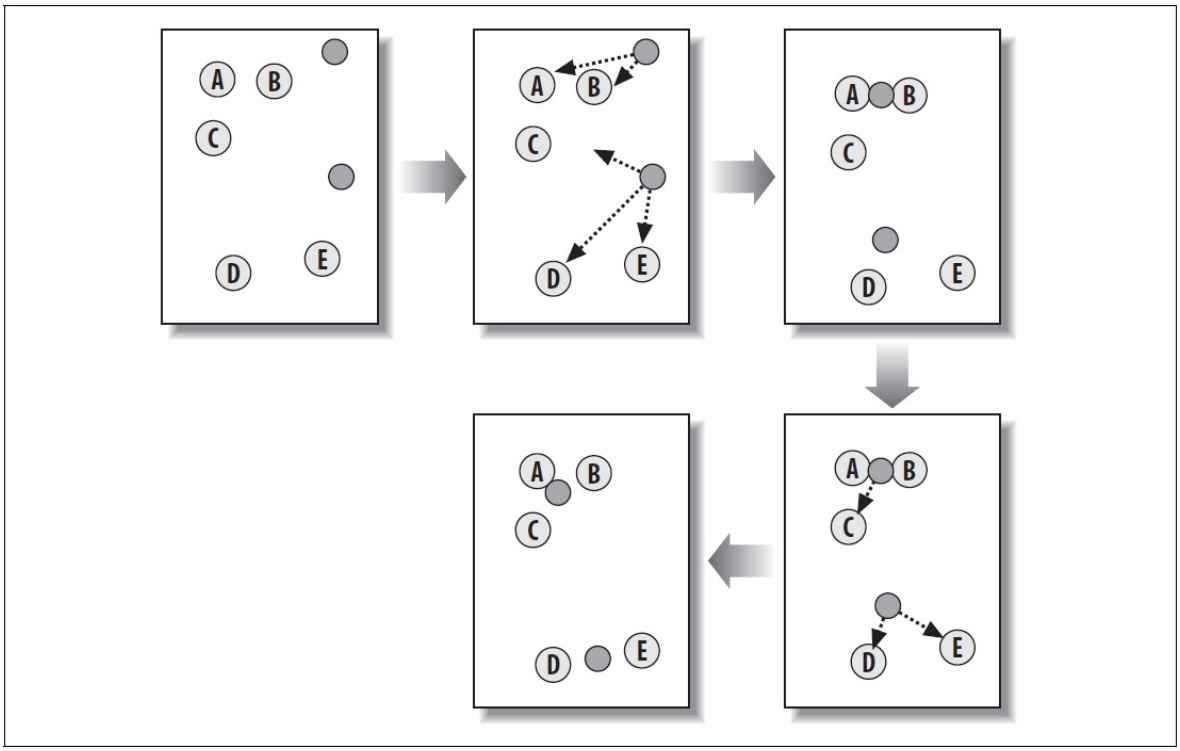

Figure 2.9.: Illustration of k-means, taken from (Segaran (2007)).

The biggest disadvantage of K-Means clustering is a problem for most of the clustering algorithms. One has to manually determine the amount of clusters, that shall be returned. Figure 2.10 gives a good example of how the different amounts of cluster centers can give a totally different output. (a) shows the original data points, (b) (c) and (d) show different results for two, four and six cluster centers. While the clusters of (b) and (d) seem very logical, the clusters of (c) are not of the same size and obviously their connection is not as strong as in the other examples. So if one has no prior knowledge on how many clusters might result in the most significant cluster centers, different amounts of cluster centers have to be tested. This requires a lot of test runs to compare the results and there is no guarantee that the optimal amount is ever found.

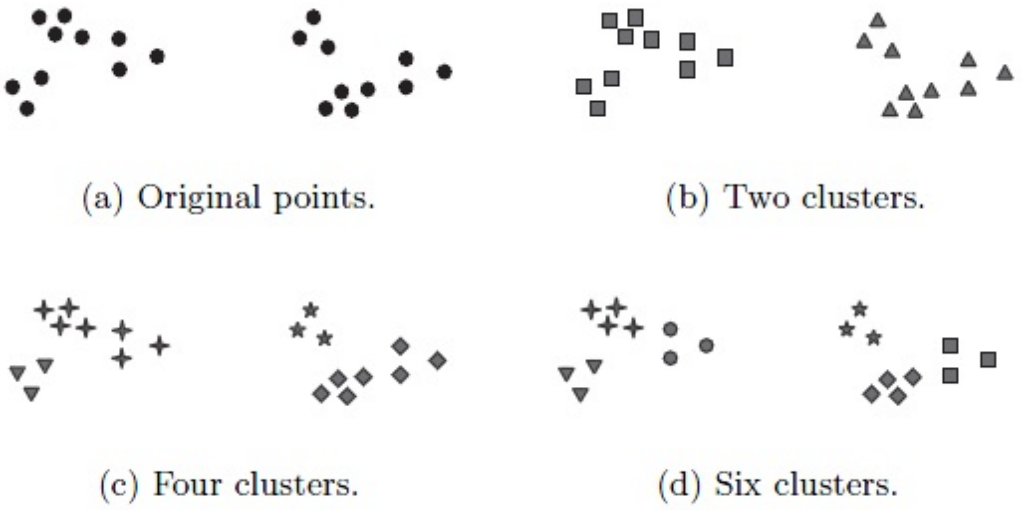

Figure 2.10.: Different amounts of cluster centers, taken from (Tan et al. (2005)).

Another disadvantage of K-Means clustering is that it does not have any way to deal with non numeric values in the data. These values can only be handled externally, by preprocessing the data and replacing non numeric values by numerics that somehow represent the original value. For example, if the non numeric values are restricted to ”low”, ”medium” and ”high”, they can be replaced with values like ”0”, ”1” and ”2”. Or, if one wants them to take more importance in the clustering, they can be replaced by ”-1000”, ”0”, ”1000”, or values with even higher difference. The problem there is that one needs background knowledge about the domain to find proper values.

2.7.2. Other clustering algorithms

(Böhm et al. (2010)) gives a good overview of research directions in the field of clustering: The article divides the algorithms into the main groups **PDF-Based Clustering** and **Density-Based Clustering**, though there are also other, more specific, algorithms that do not fit into those labels. (Segaran (2007)) also includes **Hierarchical Clustering** into the list.

PDF-Based Clustering algorithms use a model of a probability density function to describe the clusters. The most well known algorithms of this kind are the K-Means and the EM (Expectation Maximization) algorithms. Those algorithms need a pre-defined amount of clusters and are mostly suitable to detect spheric Gaussian clusters, meaning the data points have to be normally-distributed around the centers.

The specialty of EM clustering is that it does not assign a data point to a single cluster, but defines a probability for every data point to every cluster. Each cluster is represented by a standard variance

and an expectation which are iteratively updated. As in K-Means clustering, EM clustering alternately updates which points belong to a cluster and then assigns the cluster center to the middle of these points. An illustrative example is given in (Zhaohui Tang (2002)). It shows the improvement of 5 cluster centers over 3 iterations (see Figure 2.11).

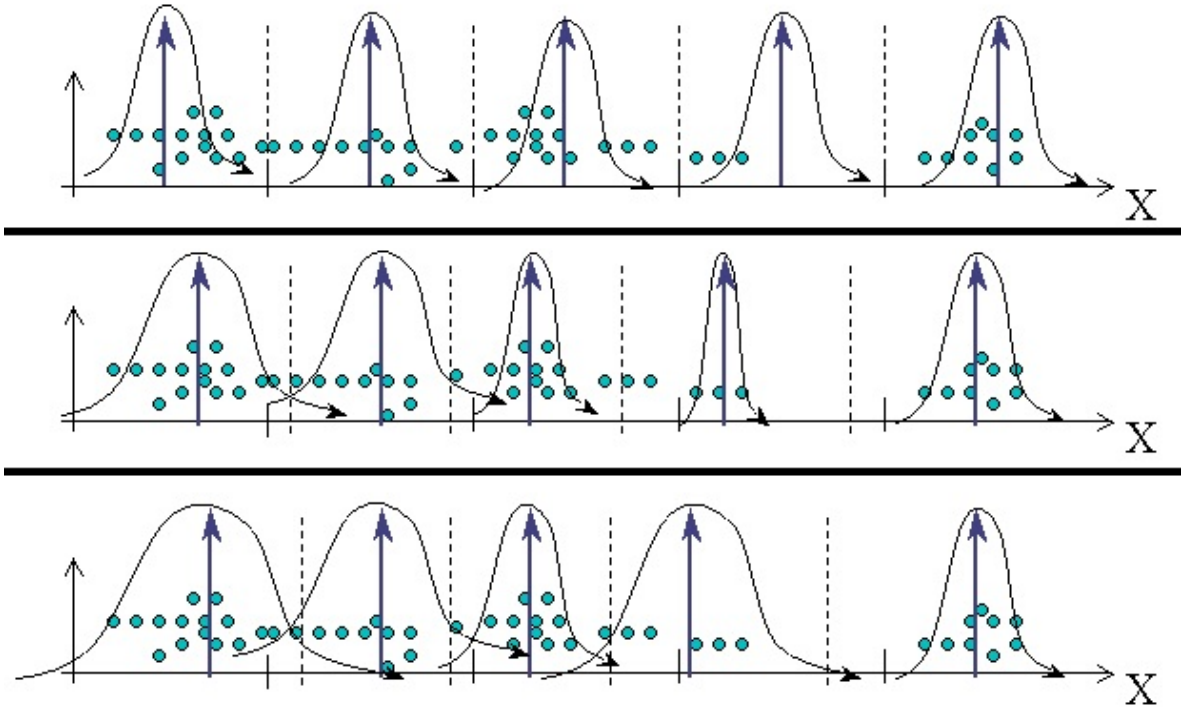

Figure 2.11.: Illustration of EM Clustering, taken from (Zhaohui Tang (2002)).

Density-Based algorithms try to detect regions of high density that are separated by regions of low density. The most common algorithm is the DBSCAN, that usually provides good results, but depends too much on the input parameters that have to be set and tested by the user. Those two parameters are the minimal amount of data points that have to be in each cluster and the neighborhood distance ε. There are three definitions needed to explain the algorithm:

1. The data point p is *directly density reachable* from point q if the distance between the points is shorter than ε and in addition q has at least the minimal amount of data points as neighbors that are closer than ε.
2. Two points p, q are *density reachable* if there exists a chain of points p, p_1, ..., p_n, q so that each point is directly density reachable from its predecessor.
3. Two points p, q are *density connected*, if there exists a point x such that p and q are density reachable from x.

Examples of density reachability and density connectivity can be found in Figure 2.7.2. Using this

(a) Density Reachability (b) Density Connectivity

Figure 2.12.: Density Reachability and Connectivity explained on a Figure from (Sander et al. (1998)).

definitions, the DBSCAN algorithm collects all density connected points to the same cluster.

Hierarchical Clustering (see (Segaran (2007)) works by merging similar groups. It starts with each data point as its own group and in each iteration it merges the two most similar groups, until all data points are in one group (see Figure 2.13).

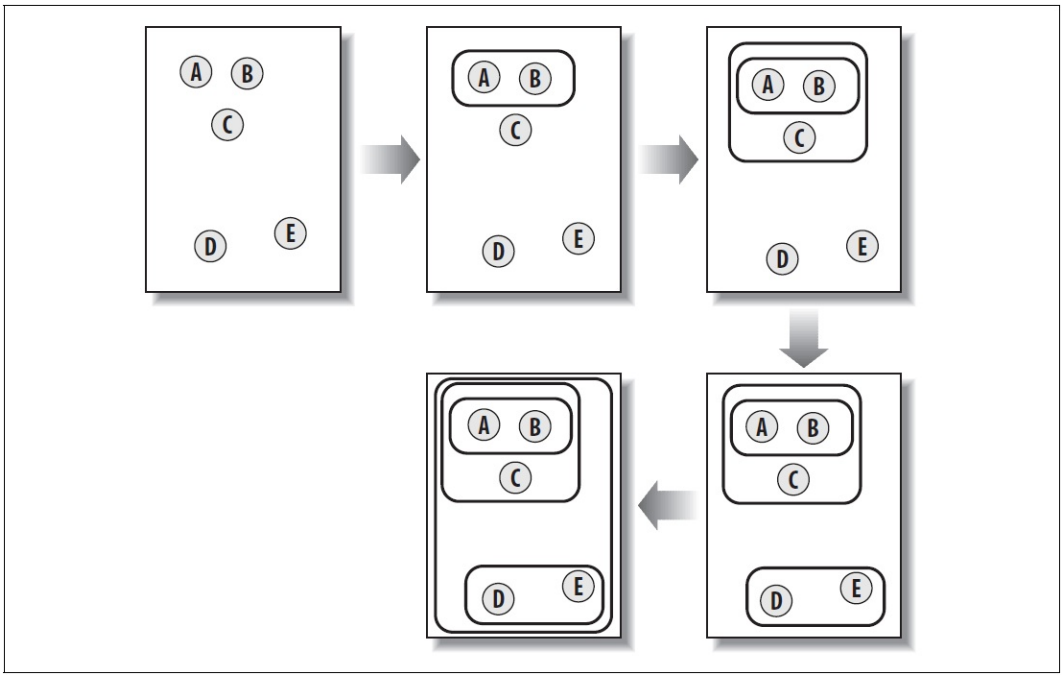

Figure 2.13.: Illustration of hierarchical clustering, taken from (Segaran (2007)).

It also stores the previous groups, hence the decisions that were made can be traced back and a **dendogram** (see Figure 2.14) can be built. Using the dendogram, one can individually decide the amount of clusters, by varying how deep one explores the graph. The big disadvantage of the algorithm is the fact, that it does not define any cluster centers and therefore it can't be used for data

preprocessing for CBR.

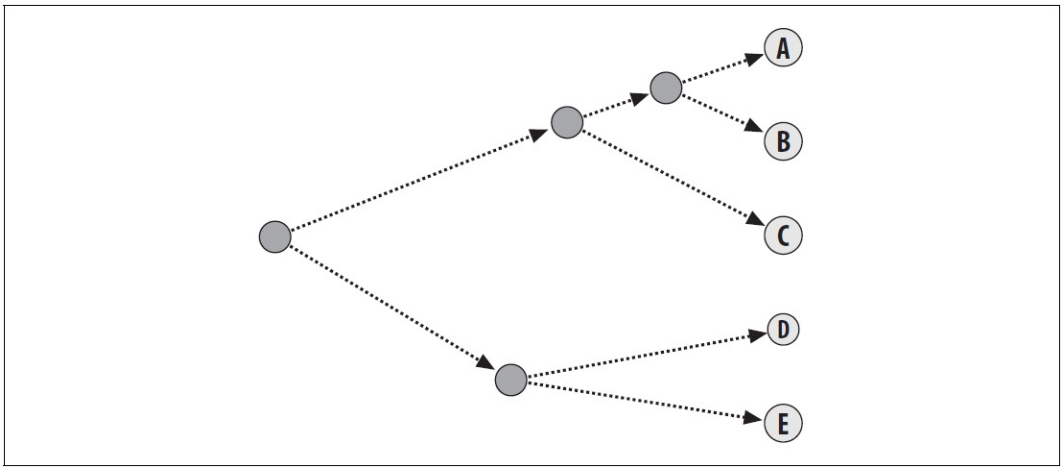

Figure 2.14.: Dendogram according to the example in Figure 2.13, taken from (Segaran (2007)).

2.8. Other applications of data mining

Beside the application in financial services, data mining has several other areas of application, such as those enumerated in (Kantardzic (2002)): Health Care, biological data analysis, telecommunications, recommendations, tourism, and so on. The next few subsections will give a short overview of some different uses and will explain how and why data mining is used there.

2.8.1. Data mining in health care

In the field of health care, data mining is used in at least 4 different ways, as shown in (Koh and Tan (2005)): It can increase **treatment effectiveness** by comparing symptoms, medicine prices and the effectiveness of those medicines. If groups of people with the same disease are observed while getting different treatments, the best price / effect ratio can be found.
It is also used in **healthcare management**, by profiling patients and their medical state. It is used in reducing the average time people have to stay in hospital, it can calculate the optimal distribution of medicine to the different hospitals and it can calculate the probabilities of certain diseases for specific patients.
Data Mining is also used in the **detection of frauds** in the health care system. It can detect abnormal behavior in the patients claims to clinics and laboratories and therefore find inappropriate prescriptions and it can also find illegal insurance claims.

2.8.2. Communications

(Weiss (2004)) gives an overview of the value of data mining in telecommunications: It contains the most effective techniques to detect **frauds**. By comparing the usual behavior of their customers with a real time view on their actions, the companies can detect for example stolen phones and can automatically deactivate them if the thief suddenly starts to use it for expensive international calls or calls to premium rate numbers.
Customer profiling is a worthy usage of data mining in telecommunications too. The companies can identify customers that are about to leave them and can send offers of reward if they stay. Additionally they can forecast the overall behavior of their customers, to gain knowledge of future trends, adjusting prices and special offers to gain the maximal income or to gain a maximal growth in customers.

2.8.3. Recommendations

Data Mining is also a very valuable source to recommender systems. These systems, as described in (Burke et al. (2011)) or (Schafer (2005)), can help users find and rate useful items. That might be a tool collecting the users preferences to recommend suitable holiday destinations, or a site like amazon*, analyzing former purchases to recommend other things to buy as well.
Some common combinations of data mining and recommender systems are described in (Schafer (2005)). First of all they mention **customer clustering**. It can cluster customers with similar preferences and then the average buying behavior of users in the same cluster can be used to find suggestions for any user in that cluster. Though that might lead to a small loss in accuracy, it can lead to a great speedup in online calculation.
Another great example for the combination of those two areas is the **item-to item correlation**. It can be used to match the items a user is currently buying to items that he might yet need. So if, for instance, the user buys different types of common spices, but does not have oregano in his list yet, he might be willing to buy that too, if it gets recommended to him. Item-to-item correlation can work on single instances as well as on the whole shopping basket or the whole previous buying history of the customer. Additionally (Schafer (2005)) states that one of its advantages is its adaptiveness to current, brand new purchases, as it can instantly interact with the first two or three items a user bought, and recommend him the next one.

2.8.4. Scientific Data Mining

One of the probably most unique approaches in using data mining is the use for purely scientific purposes (Embrechts et al. (2005)). It is not market driven, therefore it does not only aim on increasing

*www.amazon.com

profit or effectiveness. It is set up with totally different data, usually consisting of continuous and very precise values.
One great application is the scientific mining of **astronomy based data**: (Borne (2009)) mainly focuses on the data produced by the Large Synoptic Survey Telescope, which will produce a 3-gigapixel image every 20 seconds every night for ten years and plans are not only, to store those in databases for later data mining, but also to identify and classify all astronomical events happening in those images. The possibilities and challenges offered by that amount of data are outstanding.
The most common tasks with astronomical data are clustering and classification. The usage of clustering is self-explanatory, since clusters of ”nearby” stars form galaxies and clusters of galaxies form superclusters like for example the Milky Way. The further our telescopes reach, the more clusters can be found. The act of classification is used to automatically sort new instances to labeled classes like ”stars”, ”supernovas” or ”galaxies”.

Chapter 3

Affinity Based Clustering

This chapter deals with the clustering algorithm that was finally used to improve the base for the Case Based Reasoning algorithm (Section 2.3). While other clustering algorithms only look at a few, mostly randomly chosen cluster centers and try to iteratively improve them to find a good set of cluster centers, affinity based clustering, as it was introduced in (Frey and Dueck, 2007), takes all the data points into account and considers each of them as a potential cluster center. By sending messages between the data points, it iteratively enhances the values that are used to calculate their fitness as cluster centers, until it results in a clear set of cluster centers.

3.1. The basic Algorithm

Affinity based clustering takes as its input the similarities *s(i,k)* between all pairs of data points and for each data point its preference *s(k,k)*, a real number indicating its initial likeliness to be chosen as a cluster center. The calculation of the similarities is the only part of the algorithm that needs the attributes of the data points (in fact, it is the only part that needs the input data at all) and it happens only once, at the beginning of the algorithm. Therefore, preprocessing of the data by attribute selection is not necessary to improve the runtime of the algorithm.

Two sorts of messages are sent: responsibilities and availabilities. Responsibilities *r(i,k)* reflect how well suited the point k is, to represent i as a cluster center, comparing k with other possible cluster centers for i.

The availability tells how well suited it would be for i to take k as a cluster center, summing up the values of other data points, that would take k as a cluster center.

At the beginning, *a(i,k)* is initialized with 0. Then, in each iteration, first the responsibilities are

updated according to the following formula:

$$r(i,k) = s(i,k) - max\{a(i,k') + s(i,k')\}(k' \neq k) \tag{3.1}$$

So, in the first iteration, *r(i,k)* is set to the similarity between *i* and *k* minus the maximal similarity that *i* has to any other data point. Later, after *a(i,k')* has been updated for a few times and the according data points have already been assigned to other cluster centers, they might decrease *s(i,k')* enough, so the data points are not used in the calculation. The self-responsibility *r(k,k)* is, according to the formula, set to the initial likeliness of the data point minus the best similarity that it has to another data point.

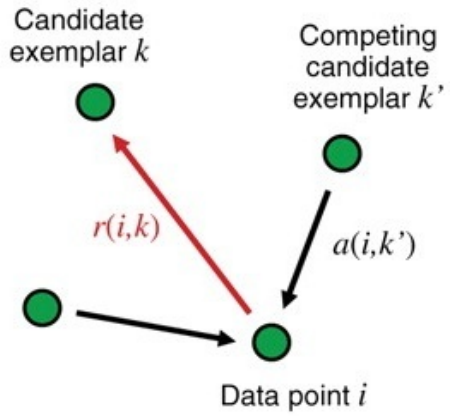

Figure 3.1.: Calculation of responsibility Frey and Dueck (2007).

Once the responsibilities have been calculated, the availabilities are updated:

$$a(i,k) = min\{0, r(k,k) + \sum_{i' \notin \{i,k\}} max\{0, r(i',k)\}\} \tag{3.2}$$

That means that the availability of two data points *i* and *k* is set to the self-responsibility of *k* plus the positive responsibilities, *k* gets from other data points. The negative responsibilities are not taken into account, because it does not matter if the data point is a bad cluster center for those data points that are not going to be in its cluster anyway. The only thing of importance is that the exemplars are similar to the points in their own cluster.

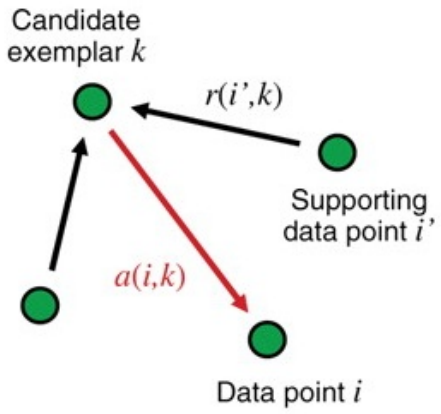

Figure 3.2.: Calculation of availability Frey and Dueck (2007).

The self-availability *a(k,k)* uses only part of above formula, summing only the positive feedback from other data points, not taking into account the self responsibility:

$$a(k,k) = \sum_{i' \neq k} max\{0, r(i',k)\} \tag{3.3}$$

The algorithm can be interrupted after any iteration, though it is recommended to let it run until no further change in the predicted cluster centers happens. The cluster centers are calculated by combining a and r. For any data point i, the data point k that maximizes *a(i,k) + r(i,k)* is the predicted cluster center. That also holds true if $i == k$. In each iteration, the connections between the data points and their centers grow stronger. A graphical example of these connections can be seen in Figure 3.5. The example comes from Frey and Dueck (2007) and shows the clustering of 25 two-dimensional data points. In the beginning, the connection between all the points is very weak, being only a little stronger between close points. Then in each iteration, the connection between distant points decreases, and the connection between the probable cluster centers and the points of their clusters grows stronger.

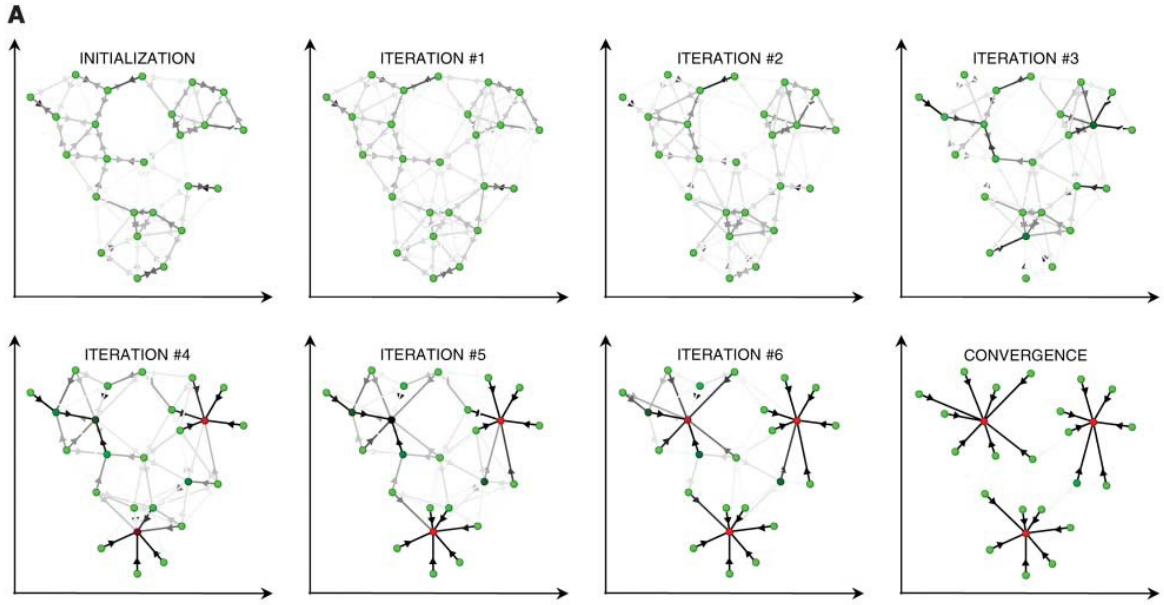

Figure 3.3.: Affinity based clustering on 2-dimensional data points Frey and Dueck (2007).

To give a better insight into the algorithm, the following will show a concrete example of the whole work flow of the algorithm. Again, the data points as seen in Table 3.1 below, are two-dimensional data points. The similarity between two points is measured by the negated distance between the points, calculated via the Euclidean Product.

X-Value	3	2	3	1	1	2	0	1	0	4
Y-Value	4	3	2	5	6	5	0	0	2	4

Table 3.1.: Two-dimensional data points used for an example calculation of the Affinity Based Clustering algorithm.

First of all, the similarities between all the points are calculated and can be seen in Table 3.2. Note the diagonal is storing the preference of each data point. Since none of the points are preferred as cluster centers, they are all set to an equal random value. The value ”-2” was picked for this example because the algorithm determined faster than with other tested random values.

	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9	Point 10
Point 1	-2.0	-1.41	-2.0	-2.23	-2.82	-1.41	-5.0	-4.47	-3.6	-1.0
Point 2	-1.41	-2.0	-1.41	-2.23	-3.16	-2.0	-3.6	-3.16	-2.23	-2.23
Point 3	-2.0	-1.41	-2.0	-3.6	-4.47	-3.16	-3.6	-2.82	-3.0	-2.23
Point 4	-2.23	-2.23	-3.6	-2.0	-1.0	-1.0	-5.09	-5.0	-3.16	-3.16
Point 5	-2.82	-3.16	-4.47	-1.0	-2.0	-1.41	-6.08	-6.0	-4.12	-3.6
Point 6	-1.41	-2.0	-3.16	-1.0	-1.41	-2.0	-5.38	-5.09	-3.6	-2.23
Point 7	-5.0	-3.6	-3.6	-5.09	-6.08	-5.38	-2.0	-1.0	-2.0	-5.65
Point 8	-4.47	-3.16	-2.82	-5.0	-6.0	-5.09	-1.0	-2.0	-2.23	-5.0
Point 9	-3.6	-2.23	-3.0	-3.16	-4.12	-3.6	-2.0	-2.23	-2.0	-4.47
Point 10	-1.0	-2.23	-2.23	-3.16	-3.6	-2.23	-5.65	-5.0	-4.47	-2.0

Table 3.2.: Similarities of the data points introduced in Table 3.1. The preferences (the values in the diagonal) are equally set to a random value.

In the next step, the initial responsibilities $r_{i,k}$ are calculated by setting them to the similarity $s_{i,k}$ minus the maximal similarity point i has to any other point. After that the availabilities (Table 3.4) can be calculated by setting them to the self responsibilities $r_{k,k}$ plus the positive responsibilities k received from other points. Values higher then zero are replaced by zero. Now for each data point i one can find the data point k that maximizes $a(i,k) + r(i,k)$, which is the predicted exemplar. Table 3.5 shows the cluster centers of the current example. Due to the simplicity of the example, the cluster centers can be calculated after one iteration and do not change by increasing the amount of iterations. Finally Figure 3.4 shows the graphical representation of the points and their cluster centers.

	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9	Point 10
Point 1	-1.0	-0.41	-1.0	-1.23	-1.82	-0.41	-4.0	-3.47	-2.6	0.41
Point 2	0.0	-0.58	0.0	-0.82	-1.74	-0.58	-2.19	-1.74	-0.82	-0.82
Point 3	-0.58	0.58	-0.58	-2.19	-3.05	-1.74	-2.19	-1.41	-1.58	-0.82
Point 4	-1.23	-1.23	-2.6	-1.0	0.0	0.0	-4.09	-4.0	-2.16	-2.16
Point 5	-1.82	-2.16	-3.47	0.41	-1.0	-0.41	-5.08	-5.0	-3.12	-2.6
Point 6	-0.41	-1.0	-2.16	0.41	-0.41	-1.0	-4.38	-4.09	-2.6	-1.23
Point 7	-4.0	-2.6	-2.6	-4.09	-5.08	-4.38	-1.0	1.0	-1.0	-4.65
Point 8	-3.47	-2.16	-1.82	-4.0	-5.0	-4.09	1.0	-1.0	-1.23	-4.0
Point 9	-1.6	-0.23	-1.0	-1.16	-2.12	-1.6	0.0	-0.23	0.0	-2.47
Point 10	1.0	-1.23	-1.23	-2.16	-2.6	-1.23	-4.65	-4.0	-3.47	-1.0

Table 3.3.: Initial responsibilities of the data points introduced in 3.1.

	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9	Point 10
Point 0	1.0	0.0	-0.58	-0.17	-1.0	-1.0	0.0	0.0	0.0	-1.0
Point 1	0.0	0.58	-0.58	-0.17	-1.0	-1.0	0.0	0.0	0.0	-0.58
Point 2	0.0	-0.58	0.0	-0.17	-1.0	-1.0	0.0	0.0	0.0	-0.58
Point 3	0.0	0.0	-0.58	0.82	-1.0	-1.0	0.0	0.0	0.0	-0.58
Point 4	0.0	0.0	-0.58	-0.58	0.0	-1.0	0.0	0.0	0.0	-0.58
Point 5	0.0	0.0	-0.58	-0.58	-1.0	0.0	0.0	0.0	0.0	-0.58
Point 6	0.0	0.0	-0.58	-0.17	-1.0	-1.0	1.0	-1.0	0.0	-0.58
Point 7	0.0	0.0	-0.58	-0.17	-1.0	-1.0	-1.0	1.0	0.0	-0.58
Point 8	0.0	0.0	-0.58	-0.17	-1.0	-1.0	0.0	0.0	0.0	-0.58
Point 9	-1.0	0.0	-0.58	-0.17	-1.0	-1.0	0.0	0.0	0.0	0.41

Table 3.4.: Initial availabilities of the data points introduced in 3.1.

Points	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9	Point 10
Centers	Point 1	Point 1	Point 1	Point 4	Point 4	Point 4	Point 7	Point 7	Point 7	Point 1

Table 3.5.: Each data point and its assigned cluster center.

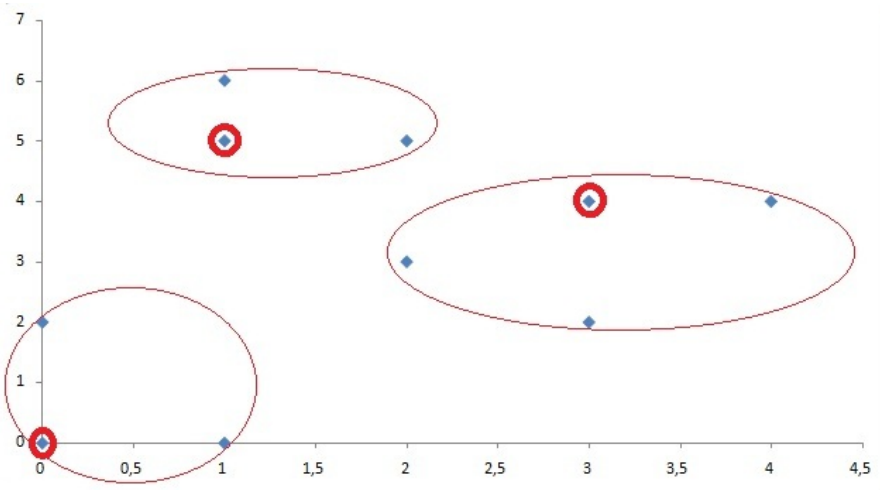

Figure 3.4.: Graphical representation of the data points from 3.1 and their cluster centers.

3.2. Affinity based clustering in customer segmentation

Assigned to customer segmentation, the principal usage of the algorithms is quite clear: the data points are the customers with their attributes and the goal of the clustering is to fit them together into groups that most likely show equal behavior. The similarities between the customers that are needed as input for the Affinity Based Clustering are described by negative distance, so that a difference of

zero means, that the two data points are equal, which of course is the maximal similarity possible. Distance between data points is described by the summed distance of their traits.

The preferences of the data points are not predefined, since there is no prior knowledge about which of them might serve as a good cluster center. If, for some reason, some of them would be preferred as cluster centers, one could raise their preferences. However, in the work done in this thesis, they were always set to equal values. The preferences in the tests done during the evaluation were set at zero and also at the mean of the similarities, the minimum of the similarities and the median of the similarities*. Each of these values tried, resulted in a different amount of cluster centers. The best result was achieved in the test carried out using half of the minimal similarity as preference.

During this thesis, ABC and ABCIW where implemented for WEKA, which is a Java framework, and Java does not normally support matrices of the size that is needed for customer segmentation, an external package, the Universal Java Matrix Package (UJMP) has been included. It allows the usage of matrices of far larger size than normal Java matrices and even matrices that are too huge to be kept in the main memory, by saving them in an own file on the hard drive. Therefore, it achieves the ability to work with the data amount that was given, yet at the price of calculation speed, since the algorithm works with the matrices all the time and for every value that has to be read or written, the hard drive has to be used.

Availabilities and responsibilities were calculated just like the algorithm described, since the algorithm really fits for customer segmentation. The next section will show other usages of the Affinity Based Clustering and how it often has to be changed to fit the according problem, but for customer segmentation no severe changes to the core functionality had to be made.

Very basic implementations of ABC show the tendency to end up in loops, switching back and forth between two sets of values. To prevent the possibility of these oscillations, a damping factor was used combining availabilities and responsibilities with their copies from the last iteration. The matrices were multiplied with this damping factor and their copies from the last iteration were multiplied with one minus the damping factor, then the two results were added. The damping factor which was used in most of the tests done, was set to 0.7, as suggested by the inventor of ABC. Several tests with other values did not once show better results. This factor did not only prevent oscillations, but also ensured that the changes being done during an iteration were not too severe and therefore statistical outliers during one iteration would be of no consequence.

*The mean value of an amount of numbers is their sum devided by their amount, while the median is the middle element of the sorted values. For the example 1 8 2 12 7, the mean would be 6, whereas the median would be 7.

3.3. Other Usages

Affinity Based Clustering has already proven itself in many fields of study [†] and to understand the algorithm and its potential better, a few examples of its use will be described. Also these examples give a good idea of what had to be improved to make Affinity Based Clustering work for customer segmentation.

3.3.1. Medical Treatment Portfolios

Affinity Based Clustering has led to a great improvement in the construction of treatment portfolios Frey et al. (2008): A treatment portfolio tries to determine a subset of vaccine sequences that provides the maximum epitope coverage for a HIV genome population. The problem had a quite severe difference to customer segmentation, since there are two different sets of data points: The treatments (the vaccine sequences) and the targets (the genome population). The targets have to form clusters around the treatments, whereas treatments and targets should not influence themselves. The initial similarities *s(i,k)* (where *i* is a treatment and *k* is a target) are set to the utility of the treatment to the target. So its not clustering by similarity, but by how good the treatment works.

The preferences are set to the negative cost of the treatment, so treatments with higher costs will be assigned to a ”garbage collector” data point, that filters out the unused treatments.

The algorithm itself works as described in the previous section, except that messages are only sent between treatments and targets. The goal was to find the smallest and cheapest set of treatments that cures the maximum of the genome population.

Evaluation showed that affinity propagation not only worked faster, but also more effectively than K-Median and greedy algorithms.

3.3.2. Document clustering

In Frey and Dueck (2007), a short demonstration of Affinity Based Clustering for document clustering was given. The data was preprocessed by deleting all words with less then 5 characters and deleting all punctuations. Then the similarities between two sentences were calculated by the negative cost it took to encode one sentence by the words of the other and a dictionary of all the words in the document. If a word was equal in both sentences the costs were set to the negative logarithm of the number of words in the sentence (the coding costs of the index of the word in the sentence) otherwise it was set to the negative logarithm of the number of words in the dictionary. (the coding costs of the index of the word in the dictionary). The initial preferences for the sentences was set to the costs of coding all words in the sentence using the dictionary.

[†] http://www.psi.toronto.edu/index.php?q=affinity propagation

There has not been a real study on the results, but they used the method on the paper in which they presented it, which was mainly about Affinity Based Clustering. The 3 top rated exemplar sentences, that really cover the most important parts of the description of Affinity Based Clustering and therefore show the correctness of the clustering, were the following:

- Affinity propagation identifies exemplars by recursively sending real-valued messages between pairs of data points.
- The number of detected exemplars (number of clusters) is influenced by the values of the input preferences, but also emerges from the message-passing procedure.
- The availability *a(i; k)* is set to the self responsibility *r(k; k)* plus the sum of the positive responsibilities candidate exemplar *k* receives from other points.

3.3.3. City reachability

Another published usage of Affinity Based Clustering (Frey and Dueck (2007)) was filtering easily reachable American cities from other American cities, by clustering them using the data of estimated commercial airline travel time. The similarities were set to the negated time costs of traveling time between the cities, taking into account factors like stopover delays and head or tailwind. The challenge in calculating city reachability was that the similarity was not symmetric, due to the headwinds and different stopover points on outward and inward flights. But since Affinity Based Clustering, unlike many other clustering algorithms, checks both the values from *i* to *k* and the values from *k* to *i*, it could process the data and gain a result of 7 cities that served as cluster centers and are the most reasonable decisions to fly to if you want to travel further to any other city in their clusters. The resulting cities are ”Seattle”, ”Calgary”, ”Denver”, ”Minneapolis”, ”Atlanta”, ”Philadelphia” and ”Toronto”.

3.4. Distance Function

The three previous examples give a good overview of Affinity Based Clustering and its usage. They all use basically the same algorithm, but there is one part that is different in all of them and turns out to be the most important part in using the algorithm: the calculation of the similarities.

The similarity between sentences in a document and the similarity between different gene treatments is fundamentally different and so is the similarity between customers. So the main work in adapting Case Based Reasoning for customer segmentation was finding the best measure for similarity.

Generally, similarity is the negation of distance, where distance is usually calculated by the summed distances of the attributes. The higher the distance, the lower the similarity. So the calculation of the similarities can be done by using a negative distance function, allowing the use of all the already

defined distance functions (Lewicki and Hill (2006)), that have already been approved by the scientific communities. Some of them will be shown in the following subsection.

3.4.1. Different Distance Functions

To demonstrate some typical distance functions, the next few subsections will calculate the distance between the customers shown in Table 3.6.

ID	Age	Income	Debts	Years of work
0	20	1200	0	2
1	43	2300	5000	23
2	34	1700	0	7

Table 3.6.: Example customers.

Euclidean Distance

Euclidean Distance (Lewicki and Hill (2006)) is probably the most used distance measure:

$$distance(x,y) = \{\sum_i (x_i - y_i)^2\}^{1/2} \tag{3.4}$$

Using the first two customers from 3.6 the calculated euclidean distance is $[(20-49)^2 + (1200-2300)^2 + (0-5000)^2 + (2-23)^2]^{1/2}$, resulting in a distance of 5119. The customer with the ID 0 and the customer with the ID 2 have a distance of 500 and Customer 1 and 2 have a distance of 5035.

The measure is simply the geometric distance between the attributes of the two individuals. It provides certain advantages, mainly based on the fact that it is usually used on raw data: It does not depend on the other data points, so if a new data point is added, it does not affect the already calculated distances. It usually only works with numeric data, but with some supervision it can also process non-numeric values.

Yet it also has a big disadvantage due to processing of raw data. The difference in the dimensions between the attributes can cause a severe over-valuation on the higher dimensional attributes. For example, if one of the attributes of a customer is his monthly rent and another attribute is his yearly income, then the rent would be about 3 to 4 digits long, while the income would be around 5 digits long. So it would have far more influence on the distance.

The **Squared Euclidean Distance** works the same, but without the square root. Therefore high scaled attributes will grow even bigger.

Power Distance

Using this measure, one can individually decide the weight he/she wants to place on the distance of attributes versus the weight he wants to place on the total distance (Lewicki and Hill (2006)):

$$distance(x,y) = \{\sum_i (x_i - y_i)^p\}^{1/r} \tag{3.5}$$

p is the weight placed on the difference of individual attributes and r is the weight placed on the larger differences between data points. Setting p and r to 2, gives the euclidean distance.

Again the customers from Table 3.6 are used to demonstrate the distance function, with p set to 1.5 and r set to 2. The low value for p means that each attribute with a high difference between the two customers will carry far less weight.

The distance between the first two customers is calculated by $[(20-43)^{1,5} + (1200-2300)^{1,5} + (0-5000)^{1,5} + (2-23)^{1,5}]^{1/2}$ and results in 25, the differences between the other customers are 10 and 24. So the difference between Customer 0 and 2 is once again the lowest, but the ratio between the differences has changed greatly.

Cosine Similarity

This measure is already a similarity, hence for the usage in Affinity Based Clustering, it can be used directly, without inverting its sign. It calculates the angle between the two data points, treating them as vectors:

$$similarity(x,y) = \frac{x*y}{||x||*||y||} = \frac{\sum_i x_i * y_i}{\sqrt{\sum_i (x_i)^2 * \sum_i (y_i)^2}} \tag{3.6}$$

The similarities between the customers from Table 3.6 is 0,41 between customers 0 and 1 and also between customers 1 and 2. The similarity between customer 0 and 2 is 0,99, so again those customers are the most equal ones.

In Qian et al. (2004), cosine distance was compared to the euclidean distance and there was mainly one conclusion: If the data is already clustered, or it is high dimensional, or if the vectors are normalized by size, the results of cosine distance and euclidean distance tend to become more and more similar. Since the data in customer segmentation is quite high dimensional, this will also apply to customer segmentation.

3.4.2. Weighted Euclidian Distance

Evaluation of the above distance functions has shown that none of them fits significantly better for customer segmentation than the simple euclidean distance function. Yet, since the input data is not

normalized and so many of the high dimensional attributes gain far too high attention, improvements still need to be made.

The solution lies in the **Weighted Euclidean Distance**, described by Agostino and Dardanoni (2009), that adds a quite simple, but effective improvement to the euclidean distance.

$$distance(x,y) = \{\sum_i w_i * (x_i - y_i)^2\}^{1/2} \tag{3.7}$$

where w_i are the attribute specific weights, that can not only be used to decrease the difference between high and low dimensional attributes, but can also be used to set a higher attention on values that were already proven influential by former experiments or prior knowledge of the topic. As an example, the monthly income might be an important attribute when trying to decide whether a customer might want a loan, and should therefor have a higher weight than most other attributes. The weights are usually kept around 1, so that values that are not known and stay on the standard value, are not affected.

A simple way to calculate weights is to use the difference between the maximal and the minimal values of each attribute and calculate the weights relative to those values, so that they have a common dimension. In the special case of weights that are set to 1 divided by the standard deviation of x_i and y_i over the sample set, the distance function is called **Mahalanobis Distance**, further described in Durak (2011).

3.4.3. Genetically Optimization

These weights already give a bias in the right direction, but the effects of the weights can be improved much further, by genetically improving them.

A genetic algorithm, as defined by Mathew (2005), is based on the principle of genetic evolution, by encoding possible solutions of a problem in a chromosome-like representation and evolving them over time. It starts with an initial population consisting of such chromosomes that were produced in a simple way, like using random values. Then in iterations, new generations are built by mutation and recombination of the older generations. The algorithms always use a fitness function that can calculate the fitness of a chromosome, by deciding how good it solves the initial problem. The higher the fitness of a chromosome, the higher its chance to survive, meaning that it is more likely to be picked as a member of the next generation.

Applied on the weighted euclidean distance, the chromosomes are lists of double values, that serve as the weights during distance calculations. Initially they were set to random values reaching from 0.5 to 1.5.

The final algorithm for calculation and the updating of those weights consisted of the following steps, shown more formally in Algorithm 1:

- **Step 1: Create the initial population**
  The initial weights were randomly set and saved in separate files (one file per population member), containing one weight per attribute. The size of the population was kept relatively low, due to time reasons, so most of the test runs were executed with a population size between 10 and 30.

- **Step 2: Mutate**
  A random amount of attributes of each chromosome was mutated. Different values for that amount were tested, the final setting was a five percent chance for each attribute to be mutated. The data sets consisted of about 150 different attributes which made a mean of about 7 mutated attributes. Most mutations decrease the fitness, so too many mutations at once seldom result in a positive outcome.
  A mutation was done by multiplying the weight with a random value between 0.5 and 1.5.

- **Step 3: Crossover**
  With a probability of 10 percent for each chromosome, a crossover (Kim and Street (2004)) with another chromosome was conducted. A random weight was selected and all weights after that were replaced by the weights of the other chromosome. So the descendant of the next generation owns DNA of both its parents.

- **Step 4: Determine the fitness**
  The fitness function was the time consuming part of the algorithm. The weights created by the mutations and crossovers were used to build and execute an Affinity Based Clusterer on a data set that usually contained about 400 customers. The resulting exemplars were saved and then used by the Case Based Reasoning algorithm, to be evaluated with a 10 fold cross validation. 10 fold cross validation means, that the data set is split into 10 parts using 9 parts as training data and 1 part to validate the results. This is repeated 10 times, each time with a different part as the validation part, so in the end each part was used for validation at least once. The effective fitness was then calculated by the mean f-Measure of those 10 validations.

- **Step 5: Selection**
  Since neither the implementation of the mutation, nor the implementation of the crossover increased the size of the population, the selection was done by comparing the fitness of the resulting children with the fitness of their parents. If the children evolved, they were taken into the next generation, discarding their parents. If they degenerated, they would usually be discarded, still there was a slight chance they would be taken into the next generation anyway, depending on the amount of iterations already done: That chance was set at the beginning of the algorithm, usually to 30% and was halved in each iteration. Such mechanisms are used to escape local maxima at the beginning of the algorithm, while there is still a chance to reach the global maximum. The fewer iterations that are left, the lower is the chance to still find a better configuration, if a locally good solution is skipped.

- If the maximum number of iterations was not yet reached, the next generation was processed, by returning to step 2.

Algorithm 1 Genetic optimization algorithm for optimizing the weights. Lines 8-9 show the mutation, occurring with a probability of 5% per weight. Lines 13-18 show the crossover with a randomly chosen second parent, happening for 10 % of the children. Lines 19-24 show the selection between children and parents. At the beginning, there is a high (30%) chance for a degenerated child to still be selected (Lines 23-24). In each iteration this possibility decreases (Line 29). The hill climbing algorithm (Line 32) can be seen in Algorithm 2.

```
1:  heat ← 30%
2:  Population ← createPopulation(size)
3:  for i ← 1 to maxIterations do
4:    for all parent ∈ Population do
5:      childs ← ∅
6:      child ← parent
7:      for all w ∈ weights(child) do
8:        if random(0,100) < 5 then
9:          w ← w * random(0.5,1.5)
10:       end if
11:     end for
12:     childs ← childs ∪ child
13:     if random(0,100) < 10 then
14:       child ← parent
15:       parent2 ← randomElement(Population)
16:       child ← crossOver(parent1, parent2)
17:     end if
18:     for all child ∈ childs do
19:       if fitness(child) > fitness(parent) then
20:         parent ← child
21:       else
22:         if random(0,100) > heat then
23:           parent ← child
24:         end if
25:       end if
26:     end for
27:   end for
28:   heat ← heat/2
29: end for
30: best ← getBestChromosome(Population)
31: result ← hillClimbing(best)
```

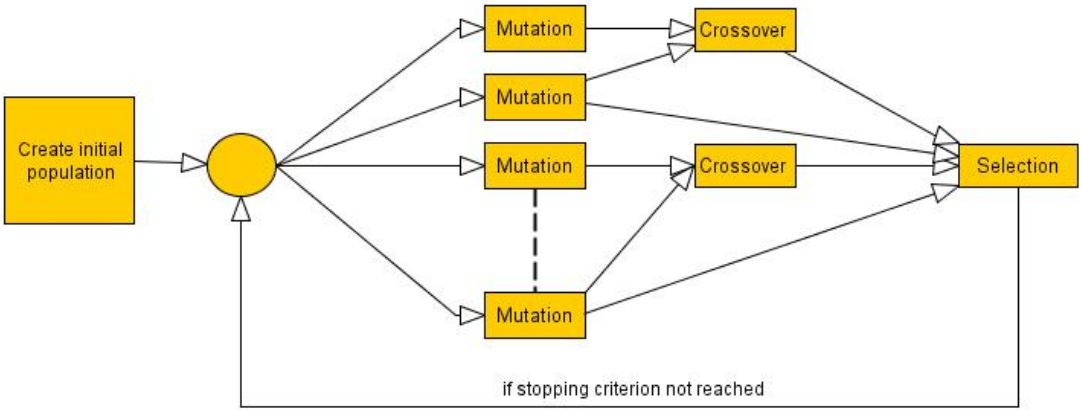

Figure 3.5.: Genetic algorithm: Mutations done at a chance of 5% per attribute, crossover at a chance of 10%.

Since genetic algorithms do not guarantee optimal results, especially if the fitness function is time intensive and it is not possible to run an appropriate number of iterations, the resulting weights are updated afterwards by an adaption of the Hill Climbing Algorithm (see Algorithm 2). A Hill Climbing Algorithm, (Mitchell et al. (1993)), is a local search method, that does not search for the global solution to a problem, but for the local maximum that lies closest to its initial configuration. If it is performed after an other algorithm, like in this case, after the genetic algorithm, it can improve the results, if the maximum has not been met exactly. The algorithm simply increases or decreases an attribute and evaluates whether the change achieved a higher fitness function. If the result was improved it continues. Figure 3.6 shows an example of how the fitness might change by increasing an attribute value. Starting at the marked spot, the Hill Climbing algorithm can not find the global maximum, because to reach it, it would have to decrease the fitness first. But it can reach the local maximum by decreasing the current attribute value.

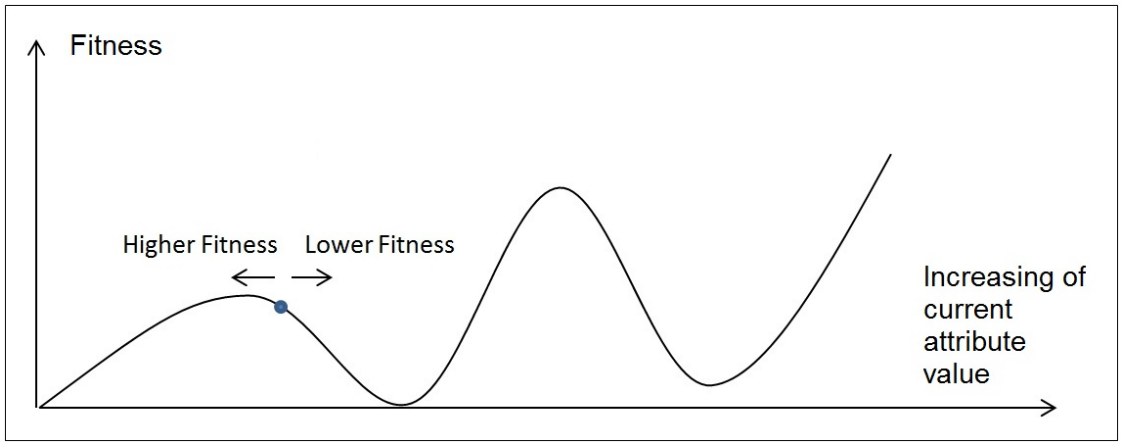

Figure 3.6.: A simplified illustration of the Hill Climbing algorithm, starting at a point where the global maximum is not reachable by the algorithm. The local maximum can be achieved by the algorithm through decreasing the attribute value.

Algorithm 2 This algorithm shows the Hill Climbing algorithm, applied to reach the local maximum nearest to the result found by the genetic algorithm by an offset of 0.3 per attribute.

1: **for all** $w \in weight(chromosome)$ **do**
2: $oldFitness \leftarrow fitness(chromosome)$
3: $w \leftarrow w + 0.3$
4: **while** $fitness(chromosome) > oldFitness$ **do**
5: $oldFitness \leftarrow fitness(chromosome)$
6: $w \leftarrow w + 0.3$
7: **end while**
8: $w \leftarrow w - 0.3$
9: **end for**
10: **for all** $w \in weight(chromosome)$ **do**
11: $oldFitness \leftarrow fitness(chromosome)$
12: $w \leftarrow w - 0.3$
13: **while** $fitness(chromosome) > oldFitness$ **do**
14: $oldFitness \leftarrow fitness(chromosome)$
15: $w \leftarrow w - 0.3$
16: **end while**
17: $w \leftarrow w + 0.3$
18: **end for**

The Hill Climbing algorithm (Algorithm 2) improved weight by weight, increasing each of them by 0.3 as long as the fitness was increased by those changes. After all weights were processed, it reduced each weight by 0.3, again as long as the fitness was increased. That process made sure that the local maximum was met by an offset of 0.3 at most, per weight.

3.5. Disadvantages of Affinity Based Clustering

There is only one huge disadvantage of Affinity Based Clustering: the size of the matrices that are built in the process grows too high and therefore the procedure gets really space and time consuming.

To create the similarities, each customer is compared with each other and the values are stored in a matrix. Those values are usually decimal digits, lets say they are stored as floats, so they take 4 byte of memory each. Now lets say, there are 50 000 customers in the data base. Processing all of them at once creates a matrix of 2.5 billion entries, so the resulting matrix grows up to 10 gigabytes of data. Then there are still the responsibilities and the availabilities to calculate. Additionally, to apply the damping factor, one also has to keep track of availabilities and responsibilities from the last iteration. So in total, that comes to 50 GB.

This is where the problem splits up into two parts: The occupied disk space and the processing time for this data. Nowadays disk space is not that much of a problem anymore, though if the amount of data raises to over hundred thousand customers, it might turn critical again, especially if you want to use fast disks, preferably SSD disks.

But the biggest problem is the processing time of data stored on the hard disk. The evaluation in this thesis was done on subsets of the original set that contained from 200 to 500 customers, so on comparatively small sets. Yet the calculation time between 200 and 500 customers differed significantly. A few thousand customers could be done on a normal laptop, but for the genetic algorithm which takes a few days to process requires a really strong work station, or preferably a cluster of work stations.

Chapter 4

Evaluation of Results

The biggest improvement for the Affinity Based Clustering was the training of the genetic fitness function. The implementation of the Affinity Based Clustering with Intelligent Weights, however, could not be tested against the original Affinity Based Clustering in the cases described in Section 3.3. The difference between those cases was mostly the usage of a different fitness function to adapt to the special problems they dealt with. Since the customer segmentation needed its own distance function, no direct comparison to the methods used for those problems could be carried out.

The results shown in the next few subsections were achieved using the different subsets of the dataset provided by the financial service provider that sponsored the project. The original set contained the data of about 60,000 customers, of whom about 1,500 were classified as positive. So the ratio was about 1:40. That created a handicap for the training of the Case Based Reasoning, for if it gets trained on subsets with randomly chosen customers, there might be a chance that not even one positively classified customer is in that set.
If only a few customers are classified true, they might not be significant enough to be used as a case, nor might they be significant enough to create their own clusters, when being clustered.
So the first few created subsets were not purely chosen at random, but it was made sure that the ratio was 1:1.

For this evaluation five data sets of 200-500 customers were created. The ABCIW was trained on all of them and then evaluated by Case Based Reasoning. The comparison was done against untrained Case Based Reasoning, trained Case Based Reasoning with a case base created by k-Means clustering (with different amounts of clusters) and trained Case Based Reasoning with Affinity Based Clustering without genetic optimization. The f-Measure that is used to express the results in this tests is a weighted f-Measure, meaning that the f-Measure of the true instances is multiplied by the number of true instances and added to the f-Measure of the false instances, also multiplied by their amount.

Despite the selection of the customers, the data was not pre-processed like the data in Zehentner and Bulic (2011), to give the clustering algorithm as much information as possible.

Classified as TRUE	Classified as FALSE	
62	138	actually TRUE
27	173	actually FALSE

Table 4.1.: Confusion matrix of the CBR, using the unoptimized Affinity Based Clustering results on the equally distributed 400 customers dataset. 62 customers were classified as true positive while 138 should have been classified as true, but were not. Only 235 of the 400 customers were classified correctly.

Classified as TRUE	Classified as FALSE	
95	105	actually TRUE
75	125	actually FALSE

Classified as TRUE	Classified as FALSE	
49	151	actually TRUE
20	180	actually FALSE

Classified as TRUE	Classified as FALSE	
62	138	actually TRUE
23	177	actually FALSE

Table 4.2.: Confusion matrix of the CBR, when using different k-Means clusterings approaches (12,23 and 35 cluster centers) for dataset 1. 220, 229 and 239 customers were classified correctly.

4.1. Dataset 1: 400 customers, equally distributed

The first part of the evaluation was the unoptimized Affinity Based Clustering, which resulted in 23 cluster centers. The trained Case Based Reasoning used with these cluster centers and 10 fold cross validation (see Section: 3.4.3) gave the confusion matrix as shown in Table 4.1.

The values lead to a precision of 62% and a recall of 58%. Added together, those values lead to the f-Measure of 55%.

To compare these values with k-Means clustering, 3 runs with k-Means were started. One with 12 cluster centers, one with 23 and another one with 35.

The results are shown in Table 4.2 and the corresponding f-Measure values, in the same order, are 0.54, 0.52 and 0.56. So, for the same amount of clusters, the Affinity Based Clustering was 3 % better, which is not really impressive, but shows that the decision to use Affinity Based Clustering is justified, for in its simplest version it already shows an improvement.

The best results however were achieved by the genetically optimized Affinity Based Clustering that resulted in 18 cluster centers. Of course, the same parameters used in the test with the untrained

Classified as TRUE	Classified as FALSE	
116	84	actually TRUE
64	136	actually FALSE

Table 4.3.: Confusion matrix of CBR executed on dataset 1, using the resulting clusters of ABCIW. 252 of the 400 customers were classified correctly.

version including damping-factor (see Section 3.2) and intial self preference were used. After applying the clusters to the Case Based Reasoning, the final confusion matrix (Table 4.3) was achieved.

This led to an f-Measure of 0.63, giving an 8 % improvement and it even reduced the amount of clusters. One has to keep in mind that a lower amount of cluster centers usually makes the Case Based Reasoning worse, for it has less data to compare with the new cases. Only if the fewer cases are really well suited and representive, fewer cases can lead to better results.

That set the focus on the examination of the cluster centers of the unimproved and the improved algorithm and it turned out that only 4 of the 18 improved cluster centers were not in the set of the 23 cluster centers found by the unimproved algorithm. So the improvement in the results was achieved by genetically updating the weights of four of the cluster centers and filtering out five irrelevant centers.

4.2. Dataset 2: 200 customers, equally distributed

To test the algorithm on smaller sets, another test run with only 200 customers was done. Again the evaluation was started with unoptimized Affinity Based Clustering, resulting in 23 exemplars.

The values of the confusion matrix of Table 4.4 showed that Affinity Based Clustering works very well on small data sets, even without the genetic optimization. The f-Measure was calculated with the value of 0.6. As in the previous evaluation, the algorithm had a tendency towards classifying instances as false. This means, that relying on this classifier would result in some customers who might be interested in a product not being asked to buy it.

k-Means clustering, this time with cluster sizes reaching from 12 to 24 and 36, constantly showed worse results (Table 4.5). The corresponding f-Measures are 0.54, 0.42 and 0.49, so at the same amount of clusters, the unoptimized Affinity Based Clustering increased the results by 18%. Surpris-

Classified as TRUE	Classified as FALSE	
44	57	actually TRUE
19	81	actually FALSE

Table 4.4.: Confusion matrix of CBR for dataset 2 using unoptimized Affinity Based Clustering.

Classified as TRUE	Classified as FALSE	
93	8	actually TRUE
73	27	actually FALSE

Classified as TRUE	Classified as FALSE	
80	21	actually TRUE
83	17	actually FALSE

Classified as TRUE	Classified as FALSE	
72	29	actually TRUE
69	31	actually FALSE

Table 4.5.: Confusion matrix of CBR using different k-Means clusterings (12, 24 and 36 cluster centers) for dataset 2.

Classified as TRUE	Classified as FALSE	
59	42	actually TRUE
34	66	actually FALSE

Table 4.6.: Confusion matrix of CBR for dataset 2, using the ABCIW clusters as basis.

ingly, both a lower and a higher cluster center amount showed better results with k-Means than the one chosen by Affinity Based Clustering.

However, the optimized Affinity Based Clustering resulted in only 12 clusters, which is the amount that brought the best results with k-Means too.

The f-Measure obtained with the optimized algorithm is only 2% better than the f-Measure of the unoptimized. But the confusion matrix (Table 4.6) shows, that the relation between true and false classifications has changed strongly towards true. While the first algorithm had only 63 positively classified instances, the optimized had 93 and instead of 44 true positive instances, the optimized algorithm had 59. In addition to that, the amount of cluster centers needed for that prediction was reduced once more to only half the amount of the original algorithm.

4.3. Dataset 3: 500 customers, ratio 5:1

The previous test runs were all executed on data sets of customers with an equal distribution between true and false classification. To test the algorithm on datasets with more realistic distributions, the data set had a ratio of 5:1. So 100 out of the 500 customers were classified as positive. The first test, with the unoptimized Affinity Based Clustering already showed quite good results (Table 4.7), with an amount of 22 clusters. Only 10 of the 100 positive customers were correctly identified, yet the

Classified as TRUE	Classified as FALSE	
10	90	actually TRUE
24	376	actually FALSE

Table 4.7.: Confusion matrix of CBR executed on dataset 3, using unoptimized Affinity Based Clustering.

Classified as TRUE	Classified as FALSE	
27	73	actually TRUE
48	352	actually FALSE

Classified as TRUE	Classified as FALSE	
14	86	actually TRUE
28	372	actually FALSE

Classified as TRUE	Classified as FALSE	
9	91	actually TRUE
21	379	actually FALSE

Table 4.8.: Confusion matrix of CBR for dataset 3, using k-Means clusterings (11, 22 and 33 cluster centers) to gain a basis for the reasoning.

weighted f-Measure reached a value of 0.72, which looks much better than it actually is. Here one has to take into account, that the f-Measure is weighted by the amount of true / false classified instances and the 400 false instances have far too much weight. The f-Measure of only the positive instances lies at only 0.14.

To keep the evaluation equal in all the test runs, 3 runs with k-Means clustering were executed: One with the half cluster amount (11), one with the same cluster amount (22), and one with one and a half times the amount (33). As shown in Table 4.8, the lowest amount of cluster centers reached the best values, clustering 27 of the 100 positive instances correctly. With a weighted f-Measure of 74%, this is the first time during this experiments, were k-Means clustering reached better values than the unoptimized Affinity Based Clustering, for it reached an f-Measure of 0.3 for the positive instances.

The optimized Affinity Based Clustering showed quite similar results, yet they were slightly better. Though the f-Measure was only 2% better, it managed to correctly identify 8 more positive customers, than k-Means (4.9). This may seem small as a number, but since it were 8 of 100 positive customers, this means an increase of 8% and on larger data sets the improvement would be far more significant.

Classified as TRUE	Classified as FALSE	
35	65	actually TRUE
54	346	actually FALSE

Table 4.9.: Confusion matrix of CBR for dataset 3, using the clusters gained by ABCIW.

4.4. Dataset 4: 500 customers, original distribution

The 5:1 ratio used in the above section was more realistic than the ratio in the earlier tests, however it was still far from the original distribution. The previous tests were good to show the advantages of Affinity Based Clustering against other clustering methods, yet they did not give a straight view of the quality of the algorithm for customer segmentation on real world data. This subsection shows how the algorithm behaves if the data is distributed like the original set, with a ratio of 1:38.

From the 500 customers in this set, only 14 were classified as positive. It is a huge challenge for a clustering algorithm to work with data that has only such few positive instances, because it is most likely that those data points have quite different attributes and will not form clusters on their own. So the probability of one of those data points becoming an exemplar is really low and if none of them is chosen as an exemplar, the Case Based Reasoning will not classify a single instance as positive. On the other hand, if one of them is chosen to be an exemplar and there are a total of fifteen exemplars, about one in fifteen classified instances will be classified true. But only one in thirty-eight instances is really positive.

So, there can not be a perfect way to deal with the problem and, as expected, nearly all test runs with this dataset classified all data points to positive. The f-Measure, of course, looked really good in all of them, usually with values over 0.95, because 486 of the classified instances were classified correctly.

The solution to the problem is also the reason why the previous tests were run: The resulting cluster centers of those tests can also be used on other datasets, as long as all data points consist of the same attributes. Cluster centers that were achieved on data sets with equal distribution will classify about 50% of the instances to positive and those with a 1:5 ratio will classify 20% as true.

Using the exemplars of the first test as a base for Case Based Reasoning on the actual dataset, produced the confusion matrix in Table 4.10.

Classified as TRUE	Classified as FALSE	
6	8	actually TRUE
176	310	actually FALSE

Table 4.10.: Confusion matrix of the CBR executed on dataset 4, using the ABCIW results from dataset 4.

The results are as expected: about 200 of the instances were classified as positive, due to the high amount of positive cases in the case base. Correctly classifying 6 out of 14 true instances is quite good for such a data set, but the 176 false positive instances have to be improved drastically, for the test to be a success.

Therefor the cluster centers from the dataset with the 5:1 ratio were tested and the confusion matrix (Table 4.11) shows the improvement made: While classifying only 12 of the instances as true, the algorithm still identified 5 of the 14 positive instances. Now one could shift the results farther towards true classification and therefore reach more true positive instances, but only at the cost of classifying more instances as true, which are actually false. Without an empirical field study, one can't decide whether the ratio of the true positive to the false positive instances is more important than the ratio of the true positive to the false negative instances.

Out of curiosity, the Case Based Reasoning was executed with the whole data set (still the 500 customers dataset, not the original one) as case base and 10 fold cross validation, to have something to compare the results with. The results showed, that it does not classify a single instance as true. This shows that the clusterings do not only increase the speed of the Case Based Reasoning, but really improves the efficiency!

Classified as TRUE	Classified as FALSE	
5	9	actually TRUE
7	478	actually FALSE

Table 4.11.: Confusion matrix of the CBR executed on dataset 4, using the ABCIW results from dataset 3 (the dataset with the 5:1 ratio).

Chapter 5

Future Work

5.1. Field Study

In my opinion the most interesting and important are for future study would be the clustering of the whole dataset to use the result for a field study in cooperation with the financial service provider. Evaluation as it was done so far shows great results in numbers, but it is hard to say if those improvements have any effect on the real market. It is hard to predict, whether an employee of for example an insurance company would actually use a program like that.
Of course it would be logical to use the tool, since the results are better and statistically it would work for most of the new customers, but often long-term employees are skeptical to new innovations and without a real field study nothing is for sure.

Another reason that adds importance to the creation of such a field study is the lack of publications on this topic. There are many papers on closely related topics, e.g. field studies to the effect of prices and product placement to customer behavior (Granbois (1968); Lichtenstein et al. (1993)), but very few on verifying the predictions of data mining approaches for customer behavior prediction. The case study by Kim and Kim (1999) is probably the most closely related one to the topic of this thesis.

5.2. Explanations

To find real acceptance for any tool like this, the program would have to provide explanations to its predictions. No customer consultant with ten years of work experience would trust a black box device that takes a huge amount of numbers as input, takes some time for calculation and then prompts an unrelated amount of other numbers that somehow lead to an advise on how he should do his work.

In reality, the employee would only use the program if the results were explained logically and comprehensively. Using previous cases to show the similarity to the actual ones provides a good

starting point but the more sense the decisions make, the better. Some of that could be done by decision trees, that are pretty much self explaining and some of that could be done by the weights captured by genetic optimization. Determining the attributes that have the highest / lowest weights and finding reasonable explanations why the corresponding attributes have that much importance, would automatically give an explanation why customers with high values at that attribute should be favored.

5.3. Matlab implementation

Another topic for a future study, that is not of any scientific relevance, but might be fundamental for commercial use, is the outsourcing of the clustering into Matlab. There is already a Matlab version of the Affinity Based Clustering available, so the main attention should lie on an implementation of the genetic algorithm in Matlab and on finding a way to integrate the results back into WEKA. Since the algorithm mostly consists of matrix calculations, Matlab could create an enormous speed boost, compared to Java.

5.4. Manual Weights

In cooperation with a financial service provider an expert could help set initial weights per hand, according to his knowledge. For example, the yearly income should normally have far more weight for buying behavior than the age. A predicted weight table that was calculated by an employee with years of work experience would give a great basis for the genetic optimization since a part of the initial population could be set to those weights and by mutation and hill climbing, this would result in a quite high maximum. If the employee is told that his own experience was part of the process that created those results, he might be far more willing to accept them.

5.5. Divide and Conquer

To speed up the computation of the clusters a *divide and conquer* approach, as described by Trung Hoang Dinh (2004), could be implemented. Instead of processing all the data at once and calculating the similarities between all the data points, this algorithm would split the data into separate parts and work with various smaller matrices instead of one over sized matrix. The separately found clusters could then be merged, to accomplish a result for the whole data set. This might even be a reasonable add on for WEKA itself, that could be combined with any clustering algorithm and should bring very promising speedups.

Chapter 6

Conclusions

In this master thesis, the following goals have been achieved:

- Finding, implementing and evaluating the clustering algorithm, with the so far best suitability for customer segmentation with regard to Case Based Reasoning.
- Genetically improving the weights for the weighted distance function, with a combination of a genetic algorithm and the hill climbing algorithm.
- Incrementing the recall and precision values of the Case Based Reasoning compared to the original clustering algorithm, by an average of 10 percent.
- Proving the advantages of the algorithm, by doing an evaluation on real world data.

Bibliography

AAMODT, A. AND PLAZA, E. 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications 7*, 39–59. (Cited on pages 1 and 8.)

AGOSTINO, M. AND DARDANONI, V. 2009. Whats so special about euclidean distance? a characterization with applications to mobility and spatial voting. *Social Choice and Welfare 33,* 2 (August), 211–233. (Cited on page 42.)

AGRAWAL, R. AND SRIKANT, R. 1994. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 487–499. (Cited on page 19.)

BERKHIN, P. 2002. Survey Of Clustering Data Mining Techniques. Tech. rep., Accrue Software, San Jose, CA. (Cited on pages 2 and 22.)

BEUMER, G. M., TAO, Q., BAZEN, A. M., AND VELDHUIS, R. N. J. 2006. A landmark paper in face recognition. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*. FGR '06. IEEE Computer Society, Washington, DC, USA, 73–78. (Cited on page 21.)

BÖHM, C., PLANT, C., SHAO, J., AND YANG, Q. 2010. Clustering by synchronization. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '10. ACM, New York, NY, USA, 583–592. (Cited on page 24.)

BORNE, K. D. 2009. Scientific data mining in astronomy. *CoRR abs/0911.0505*. (Cited on page 29.)

BURKE, R. D., FELFERNIG, A., AND GÖKER, M. H. 2011. Recommender systems: An overview. *AI Magazine 32,* 3, 13–18. (Cited on page 28.)

CHAI, W. AND VERCOE, B. 2001. Folk music classification using hidden markov models. In *Proc. of International Conference on Artificial Intelligence*. (Cited on page 13.)

DONY, R. D. AND HAYKIN, I. S. 1995. Neural network approaches to image compression. In *Proc. IEEE*. 288–303. (Cited on page 13.)

DURAK, B. 2011. A classification algorithm using mahalanobis distance clustering of data with applications on biomedical data sets. M.S. thesis, Graduate School of Natural and Applied Sciences, Middle East Technical Universit. (Cited on page 42.)

DYMARSKI, P., Ed. 2011. *Hidden Markov Models, Theory and Applications*. InTech. (Cited on page 13.)

EMBRECHTS, M., SZYMANSKI, B., AND STERNICKEL, K. 2005. Introduction to scientific data mining. 317–365. (Cited on page 28.)

FREY, B. J. AND DUECK, D. 2007. Clustering by passing messages between data points. *Science 315*, 972–976. (Cited on pages 2, 31, 32, 33, 34, 38, and 39.)

FREY, B. J., DUECK, D., JOJIC, N., JOJIC, V., GIAEVER, G., EMILI, A., MUSSO, G., AND HEGELE, R. 2008. Constructing treatment portfolios using affinity propagation. In *In RECOMB*. 360–371. (Cited on page 38.)

FUNG, G. 2001. A comprehensive overview of basic clustering algorithms. Tech. rep. (Cited on pages 5, 21, and 22.)

GRANBOIS, D. H. 1968. Improving the study of customer in-store behaviour. *Journal of Marketing 32,* 4, 28–33. (Cited on page 57.)

GUO, L., MA, Y., WARD, R., CASTRANOVA, V., SHI, X., AND QIAN, Y. 2006. Constructing Molecular Classifiers for the Accurate Prognosis of Lung Adenocarcinoma. *Clinical Cancer Research 12,* 11 (June), 3344–3354. (Cited on page 17.)

HALL, M. 1999. Correlation-based Feature Selection for Machine Learning. Ph.D. thesis, University of Waikato. (Cited on page 14.)

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl. 11,* 1, 10–18. (Cited on page 19.)

KANTARDZIC, M. 2002. *Data Mining: Concepts, Models, Methods and Algorithms*. John Wiley & Sons, Inc., New York, NY, USA. (Cited on page 27.)

KIM, B.-D. AND KIM, S.-O. 1999. Measuring upselling potential of life insurance customers: Application of a stochastic frontier model. *Interactive Marketing 13*, 2–9. (Cited on page 57.)

KIM, Y. AND STREET, W. N. 2004. An intelligent system for customer targeting: a data mining approach. *Decis. Support Syst. 37,* 2 (May), 215–228. (Cited on page 43.)

KOH, H. C. AND TAN, G. 2005. Data mining applications in healthcare. *Journal of healthcare information management JHIM 19,* 2, 64–72. (Cited on page 27.)

KOHAVI, R. AND PROVOST, F. 1998. Glossary of terms. *Machine Learning 30,* 2, 271–274. (Cited on page 19.)

KOHAVI, R. AND QUINLAN, R. 1999. Decision tree discovery. In *Handbook of data mining and knowledge discovery*. University Press, 267–276. (Cited on page 16.)

KOLODNER, J. L. 1992. An introduction to case-based reasoning. *Artificial Intelligence Review 6,* 1 (Mar.), 3–34. (Cited on page 10.)

KOTSIANTIS, S. B. 2007. Supervised machine learning: A review of classification techniques. *Informatica 31,* 3, 249–268. (Cited on page 7.)

KRIESEL, D. 2007. *A Brief Introduction to Neural Networks, Zeta version*. Available at http://www.dkriesel.com. (Cited on page 12.)

LEWICKI, P. AND HILL, T. 2006. Statistics : Methods and applications. *Statistics 1st*. (Cited on pages 40 and 41.)

LICHTENSTEIN, D. R., RIDGWAY, N. M., AND NETEMEYER, R. G. 1993. Price perceptions and consumer shopping behavior: A field study. *Journal of Marketing Research 30,* 2, 234–245. (Cited on page 57.)

MAGIDSON, J. AND VERMUNT, J. K. 2004. An extension of the chaid tree-based segmentation algorithm to multiple dependent variables. In *GfKl*. 176–183. (Cited on page 15.)

MAKHOUL, J., KUBALA, F., SCHWARTZ, R., AND WEISCHEDEL, R. 1999. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*. 249–252. (Cited on pages 2 and 6.)

MATHEW, T. V. 2005. Genetic algorithm. Tech. rep., Department of Civil Engineering, Indian Institute of Technology Bombay. (Cited on pages 2 and 42.)

MICHIE, D., SPIEGELHALTER, D. J., TAYLOR, C. C., AND CAMPBELL, J., Eds. 1994. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA. (Cited on page 12.)

MITCHELL, M., HOLLAND, J. H., AND FORREST, S. 1993. When will a genetic algorithm outperform hill climbing? In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 51–58. (Cited on pages 2 and 45.)

MIZZARO, S. 2001. A new measure of retrieval effectiveness (or: Whats wrong with precision and recall). In *In Proceedings of the International Workshop on Information Retrieval (IR2001*. 43–52. (Cited on page 7.)

NG, A. Y. AND JORDAN, M. I. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems 2,* 14, 841–848. (Cited on page 12.)

OLSON, D. L. AND DELEN, D. 2008. *Advanced Data Mining Techniques*, 1st ed. Springer Publishing Company, Incorporated. (Cited on page 5.)

PAL, S. K. 2004. Soft data mining, computational theory of perceptions, and rough-fuzzy approach. *Inf. Sci. 163*, 5–12. (Cited on page 5.)

POWERS, D. 2011. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies 2*, 37–63. (Cited on page 6.)

QIAN, G., SURAL, S., GU, Y., AND PRAMANIK, S. 2004. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of 2004 ACM Symposium on Applied Computing*. ACM Press, 1232–1237. (Cited on page 41.)

RABINER, L. R. 1990. Readings in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter A tutorial on hidden Markov models and selected applications in speech recognition, 267–296. (Cited on page 13.)

RUBINSTEIN, Y. D. AND HASTIE, T. 1997. Discriminative vs informative learning. In *In Proceedings of the third International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 49–53. (Cited on page 12.)

SANDER, J., ESTER, M., KRIEGEL, H.-P., AND XU, X. 1998. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Min. Knowl. Discov. 2,* 2, 169–194. (Cited on pages 26 and 67.)

SASAKI, Y. 2007. The truth of the f-measure. Tech. rep., School of Computer Science, University of Manchester. (Cited on page 7.)

SCHAFER, J. 2005. The application of data-mining to recommender systems. *J. Wang (Ed.), Encyclopedia of data warehousing and mining (pp. 44-48). Hershey, PA: Idea Group Reference I*, 44–48. (Cited on page 28.)

SCUSE, D. AND REUTEMANN, P. 2008. *WEKA Experimenter Tutorial for Version 3-5-8*. University of Waikato, Hamilton, New Zealand. (Cited on page 20.)

SEGARAN, T. 2007. *Programming collective intelligence: building smart web 2.0 applications*. O'Reilly Series. O'Reilly. (Cited on pages 23, 24, 26, and 27.)

SEPIDEH, S. AND AAGHAIE, A. 2011. Introducing busy customer portfolio using hidden markov model. *Iranian Journal Management Studies* 4-2, null. (Cited on page 14.)

SKURICHINA, M. AND DUIN, R. P. W. 2002. Limited Bagging, Boosting and the Random Subspace Method for Linear Classifiers. *Patern analysis and applications 5*, 121–135. (Cited on page 12.)

TAN, P.-N., STEINBACH, M., AND KUMAR, V. 2005. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. (Cited on page 24.)

TRUNG HOANG DINH, A. A. M. 2004. The speedup of the cluster-based approach in the divide and conquer paradigm. Tech. rep., Department of Electrical and Computer Engineering, National University of Singapore, Singapore. (Cited on page 58.)

WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHRÖDL, S. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 577–584. (Cited on page 22.)

WATSON, I. AND MARIR, F. Case-based reasoning: A review. *The Knowledge Engineering Review 9*, 327–354. (Cited on page 9.)

WEISS, G. 2004. Data mining in telecommunications. In *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers, Kluwer Academic, 2005*. kluwer, 1189–1201. (Cited on page 28.)

WILKINSON, L. 1992. Tree structured data analysis: AID, CHAID and CART. In *Sawtooth/SYSTAT Joint Software Conference*. Sun Valley, ID. (Cited on page 15.)

WITTEN, I. H. AND FRANK, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (The Morgan Kaufmann Series in Data Management Systems)*, 1st ed. Morgan Kaufmann. (Cited on pages 5, 7, and 19.)

ZEHENTNER, C. AND BULIC, S. 2011. *Dokumentation: MyLife/ADAP-Predictor*. (Cited on pages 14, 17, 18, and 49.)

ZHANG, G. P. 2000. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C 30,* 4, 451–462. (Cited on page 13.)

ZHAOHUI TANG, J. Y. 2002. Performance study of microsoft data mining algorithms. Tech. rep., Microsoft. (Cited on page 25.)

List of Figures

List of Tables