Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

Master's Thesis

Cryptocurrencies: Examining Exchange Rate Trends using Data-Analysis and Clustering Algorithms

Sven Alexander Giesselbach

Supervisor: Prof Dr. Christian Bauckhage

Second Grader: Prof. Dr. Rainer Manthey

December 5th 2014

Acknowledgments

I have been very lucky to be surrounded by great people during my whole studies. I want to thank those people and bring out some people which have been especially helpful during my studies and the work on this thesis.

Firstly I want to thank Prof. Dr. Christian Bauckhage for giving me the privilege of working with him, especially on such an interesting research topic. He helped me not only by supervising my work, but also by giving me constant feedback and ideas. Thank you for the endless interesting discussions on the research and related topics.

Next I want to thank Prof. Dr. Rainer Manthey who helped me throughout my whole studies of Computer Science. He helped me in his function as professor, advisor and mentor but also as a person.

Among all my friends I want to particularly thank Christian Langer, Joël Janai and my best friend Patrick Kurtis for taking the time to read through this thesis and give me useful feedback.

I also want to thank my girlfriend Elisabeth Bauch for being understanding in a time full of stress, work and no sleep.

Most importantly I want to thank my parents Vesna and Leo Giesselbach who gave me all the help and love possible to start and finish my studies.

Lastly I want to thank Angelina Leveska for being a wonderful grandmother and teaching me that education is one of the most important parts of life.

Contents

Chapter 1

Introduction

1.1 Motivation

Payment methods have undergone a big evolution over the course of the last few thousand years. While at first trades of goods or services were used as payment, soon monetary societies took over. Around 2200 BC the first money in the form of commodity money has been used. Commodity money is money whose value is determined by the material it is built of. It was succeeded by commodity-backed money, which represented certificates for certain amounts of goods as for example rare metals. The money which is used today is called fiat money and it has no intrinsic value. Its value is reliant on the trust of people in the authorities which issue the money. Now, today's money which is not only used for payment and saving, but also as measurement of value for goods, is undergoing a new development.

Virtual currencies start to gain the interest of many people. In the last few years especially cryptocurrencies have been on a constant uprise of attention. The most popular of them is undoubtedly Bitcoin. In its peak 1 Bitcoin (BTC) was worth over $1100. With such high values Bitcoins do not only represent an alternative for fiat money but a very serious investment alternative.

Predicting the value of stocks is a non-trivial but very lucrative problem. Not only broker but also private persons use stocks as assets with which they try to earn and save money. Especially in times like this when interests are very low, stocks are a serious alternative. Nevertheless the development of stocks has been studied excessively not only by financial experts but in recent years also by data analysts.

Cryptocurrencies are a very young currency form that has not yet been studied in such an extend. This thesis will focus on the analysis of cryptocurrencies given the example of Bitcoins. The exchange rate will be analyzed with the help of analytical and statistical tools. Then Bitcoins are compared to well-known stock markets and other interesting time series. As a main part of this work a method is presented which can be used to generate prototypical trend developments using clustering. These prototypes are then

used to segment the Bitcoin time series and also build models which try to predict further development of the Bitcoin exchange rates. A survey of different clustering and prediction algorithms is given.

The results of this work can be used to build more sophisticated prediction models. They could for example serve as features for future models.

1.2 Cryptocurrencies

Cryptocurrencies are virtual currency schemes that use cryptography in order to produce new units of payment and to verify transactions. Cryptocurrencies typically do not rely on a centralized institution but highlight decentralized control.

The first and most popular scheme is the Bitcoin scheme but more than 500 cryptocurrencies have been available in October 2014. Noticeable other cryptocurrencies are for example Dogecoins, Namecoins, Litecoins and many more. Many of the cryptocurrencies are just slight variations of the Bitcoin scheme.

All of them share a public, but more or less anonymous, record of all completed transactions. The safety of this transaction record is guaranteed by cryptocurrency miners. Miners use their computers to validate and timestamp transactions by performing intense computations. In exchange for their computing power miners are also rewarded with units of the cryptocurrencies.

The number of units of a cryptocurrency is usually bound to an upper limit. This is to avoid hyperinflation and to introduce a scarcity of resources. Typically not all units are available instantaneously but become available over the course of time.

Hopes are that cryptocurrencies will be widely accepted and recognized as currencies soon. Some services and goods can already be purchased with cryptocurrencies.

1.2.1 Bitcoins

The Bitcoin scheme is to this date the most adopted cryptocurrency scheme. The exact protocol is described in Section 2.1. This small Subsection is to give an overview of basic information, advantages and disadvantages of the Bitcoin scheme and to explain its relevance. The methods in this thesis are all applied to Bitcoin market data.

Figure 1.1: The Bitcoin logo (source: bitcoin.org)

Like most of the cryptocurrencies, Bitcoins are decentralized. This means that transactions do not have to be performed via institutes like banks. This implies freedom to pay and get paid whenever wanted. Transactions can be made to anyone in the world that also uses Bitcoins. Furthermore this means that transaction fees, as they are taken by banks, are no longer necessary.

With its blockchain system, which is a sequence of all verified transactions made with Bitcoins, all transactions can be reviewed. With its cryptographic scheme and the irreversibility of transactions, fraud can be prevented. The information on the blockchain do not contain personal information and are thus secure and anonymous. All of this are clear benefits for merchants.

Yet Bitcoins are still in the early stage of development. Recent events such as high level wallet theft and bugs have led to drastic changes in the Bitcoin value, the worst resulting in a $700 drop of value. With its small units in circulation and the still not large acceptance, the Bitcoin value is prone to volatility. The currency cannot yet be regarded as stable and secure.

1.3 Goal of this thesis

This thesis has 3 main goals. The first goal of this thesis is to give an insight into the exchange rate of Bitcoins using common tools in data analysis, time series analysis and finance. The relation of Bitcoin markets to stock markets and other financial quantities is measured. Some of the features of the Bitcoin markets are analyzed.

The second goal is the extraction of prototypical trends of time series and the segmentation of time series into a sequence of trends. The trends lead to the third goal which is the prediction of further developments of the time series based on the given trends and the trend sequence.

1.4 Related Work

The related work on Bitcoins is rather sparse. There are only few papers concerning the analysis of the transactions. Most former work on Bitcoins focused on different aspects.

The paper **Bitcoin: a Money-like Informational Commodity** [1] tries to give a top-level classification of Bitcoin. In the paper Bitcoins are classified as *"money-like informational commodity"*.

In **Quantitative Analysis of the Full Bitcoin Transaction Graph** [2] Ron and Shamir present statistical results on the whole Bitcoin transaction history. They analyze how users spend Bitcoins, how large the ratio of saved to spent Bitcoins is and how

users try to protect their Bitcoins. They also make an in-depth analysis of large scale transactions. Similarly **Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace** [3] investigates the Silk Road marketplace and its sales. Silk Road is an international marketplace that is hidden by the Tor service and utilizes Bitcoins as currency. The paper gives information about the items sold, which are primarily controlled substances and narcotics, and about the behavior of the sellers. It also analyses the money-flow and the impact on the economy.

Other papers like **The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries** [4] regard the Bitcoin mining process as a game and investigate it under the aspect of game theory. They propose that Bitcoins have to be regulated by governmental like structures or will otherwise be open to attacks regarding its game-like nature.

Yet other papers suggest changes to the Bitcoin scheme or even new cryptocurrencies. In **Zerocash: Decentralized Anonymous Payments from Bitcoin** [5] the authors aim at installing a cryptocurrency scheme that guarantees completely anonymous transactions and is still as efficient as the Bitcoin scheme.

Excessive research has been performed in the area of Subsequence Time Series Clustering. The article **A Review of Subsequence Time Series Clustering** [6] summarizes the results of the research and arranges the papers of the field in time. They split the research into 3 historical parts. As historical event, that splits the research, they consider the publication of the paper **Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research** [7] which claimed that the results of clustering subsequences are meaningless because they are essentially random clusters which cannot be differentiated from randomly generated data. The proof of this was widely accepted and led to a change of the field of subsequence clustering. In most of the studies addressed by the paper large subsequences of time series are clustered. The clusters are then used to find similar patterns in other time series or in the same time series. Aside of changing the similarity measure from euclidean distances to other measures, altering the data has been proposed as a good way of bringing sense into the clusters as for example in **Useful Clustering Outcomes from Meaningful Time Series Clustering** [8]. **A Symbolic Representation of Time Series, with Implications for Streaming Algorithms** [9] proposes to convert the subsequences of the time series into symbolic representations which then can be used together with algorithms of other areas such as text processing or bioinformatics.

In **Application of Markov Chains to Analyze and Predict the Time Series** [10] Liu uses Markov Chains on price data time series to predict future price ranges. He simply categorizes every time point by the level of the price it represents and then uses the Markov Chains on the categorized data.

1.5 Contribution of this thesis

To the best of the knowledge of the author, this thesis is the first to analyze the Bitcoin exchange rate time series with the presented time series analysis tools. The periodicity of the time series is inspected as well as the relation to other markets and related time series.

Approaches of Subsequence Time Series Clustering are presented in which the values of the time series are normalized to certain boundaries to provide comparability of the subsequences. Clustering is used to extract trends of the time series. A survey of known and rather unknown Clustering algorithms, such as DESICOM, is given and for the first time subsequence time series clustering is applied to the exchange rate time series of Bitcoin markets.

The extracted trends are used to segment the time series and represent it as a sequence of trends. Markov Chains and Language Models are evaluated on their suitability for predicting the future developments of the time series, considering the former sequence of trends the time series consist of.

Chapter 2

Applicational Foundations

2.1 Bitcoins

Satoshi Nakamoto introduced the peer to peer Bitcoin scheme [11]. It aims at decentralizing transactions and obviating the need of central institutions which control the transaction flow. Instead of being based on trust, Bitcoins should provide proof for transactions and put an end to fraud. The following subsections give a short introduction into the way the Bitcoin system works.

2.1.1 Transactions

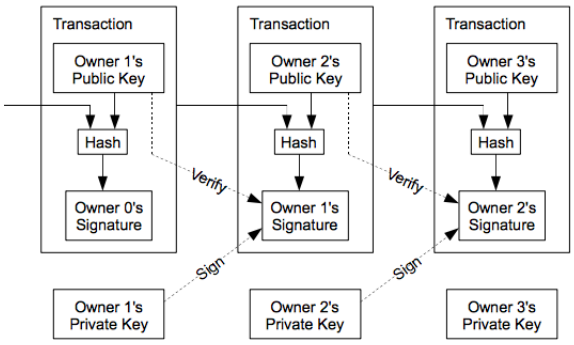


Figure 2.1: Chain of Bitcoin Transactions (source: [11] page 2)

Bitcoins are defined as a chain of digital signatures. Transactions extend these chains. In a transaction the owner of a Bitcoin digitally signs a hash of the previous transaction chain and the new owners public key, with their own private key. This new signed hash is then appended to the end of the coin. In order to avoid double spending of Bitcoins all transactions have to be publicly announced. Only the first transaction of a Bitcoin to

be publicly recorded is accepted as valid. This is established by a system in which the majority of the users have to agree that the current history of transactions is valid.

To establish such a system a timestamp server is needed. Timestamp servers work by hashing the blocks of items, that need to be timestamped, and then publicly announcing the timestamp. A new timestamp will then again hash the blocks of items, that need to be timestamped, but also include the former timestamp. This way a chain of timestamps is created that validates all transactions.

2.1.2 Proof-of-Work

The peer to peer distributed timestamp server, which is used for Bitcoins, uses a proof-of-work system. This system has two constraints for every timestamp. When hashed with SHA-256 the message has to be smaller than a given target number and contain a certain number of zero bits at its beginning. The timestamp is created by hashing the previous hash, the new transactions and a nonce. Finding the right nonce to meet the constraints requires a lot of computational effort. This makes altering the block sequence almost impossible.

When such a proof-of-work is found it will be broadcast to all nodes that work on finding proof-of-work for other blocks. If they accept the new proof-of-work they use it as hash for the next proof-of-work and work on the next block. Always the longest block-chain is regarded as the correct block chain.

Until the preset upper bound for Bitcoins has been reached, the first transaction in a block will always contain a coin for a miner, as incentive for doing the work of confirming transactions. Other incentives are transaction fees that can be offered by the payer of a transaction.

2.1.3 Privacy

Privacy is a clear concern when considering that all transactions are announced publicly and the public keys of every Bitcoin wallet are also available. The key to privacy is to not link the public keys to personal information. An additional security measure can be the usage of new private/public key pairs for every transaction.

2.1.4 Bitcoin Value

This thesis works mainly on the exchange rates of Bitcoins to US-Dollar (USD). The price of Bitcoins changes with the demand and supply. A high interest and trust in the currency will lead to higher prices, while events such as a theft will lower the price, because they

result in less trust. Ultimately, it is the price that people are willing to pay for Bitcoins, that determines its value.

2.2 The Data

The main data that is used in this thesis has been obtained by `www.bitconcharts.com`. It is data from the BTCE market and the Mt Gox market. The raw data is a sequence of transactions and states the time of the transaction, the amount that has been transferred, as well as the exchange rate in US-Dollar. For this thesis only the time of the transaction and the exchange rate are relevant. The transaction amount is omitted. The following table shows the number of transactions and the start- and end-date of the data.

	BTCE	Mt Gox
Transactions	10.740.061	8.295.809
Start-Date	14/08/2011	18/07/2010
End-Date	25/02/2014	25/02/2014

Table 2.1: Number of Transactions and Start- and End-Date of the Raw Data

The BTCE market averaged about 11.580 transactions per day in the given period. It reached its peak of 333.107 transactions on the 10^{th} of September 2011 and its low with 4 transactions on the starting day, the 14^{th} of August 2011. Mt Gox averaged 6.318 transactions per day. The most transactions took place on September 17^{th} 2010 with as many as 66.293 transactions and the least transactions were on the day of its creation on August 17^{th} 2010.

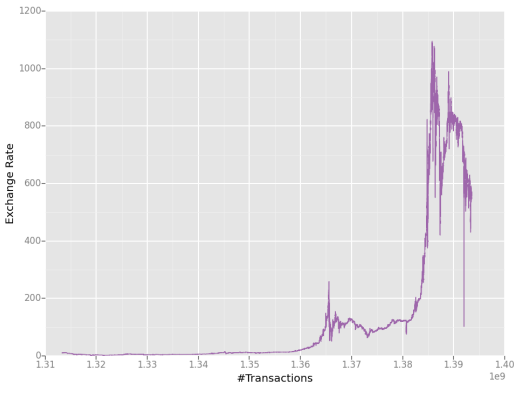


Figure 2.2: Exchange Rate of BTCE - All Transactions

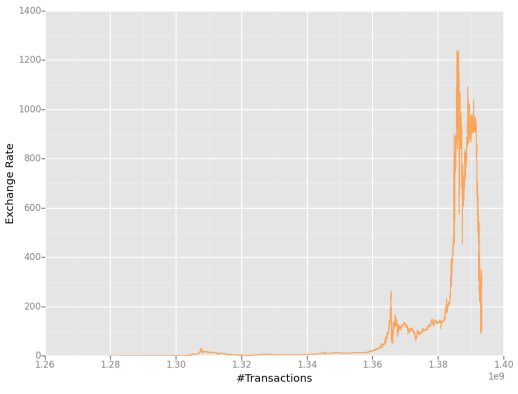


Figure 2.3: Exchange Rate of Mt Gox - All Transactions

Using every transactions for trend prediction and segmentation, would utilize an unnecessarily high amount of mostly redundant data. Transactions are not equally distributed among days and many times the exchange rate only changes marginally from one transaction to another. Often transactions with equal exchange rates follow each other. This introduces a lot of redundancy and it does not allow the data to be split into equally large time intervals because of the irregular timing of the transactions. Considering that the mean standard deviation of the courses per day is around 3 for both markets, it makes sense to sample the exchange rates on a daily basis without distorting the time series too much. Therefor the daily mean is taken as exchange rate value for each day.

The data is given in unixtime. Unixtime states time points by counting the seconds passed from January 1^{st} 1970 at 00:00 am. This second count can easily be converted to the actual date. Common programming languages, like python, provide functions that convert unixtime into datetime and account for leap years, seconds and so on. Once the time is converted for each transaction, transactions of each day are summarized and their mean exchange rate value per day can be calculated.

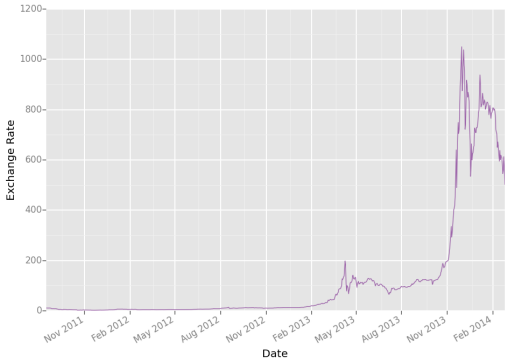


Figure 2.4: Daily Exchange Rate of BTCE - Daily Mean

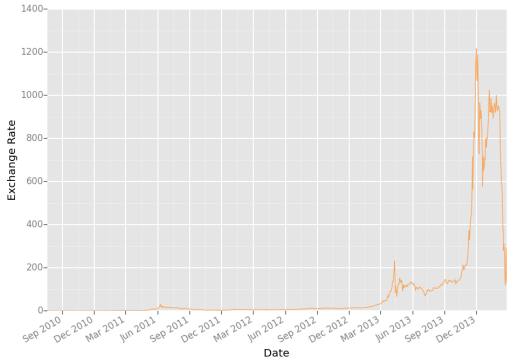


Figure 2.5: Daily Exchange Rate of Mt Gox - Daily Mean

This reduces the amount of data tremendously. The BTCE series now contains 930 entries and the Mt Gox data is reduced to 1313 entries. Table 2.2 shows some basic statistics about the Exchange Rates of both markets for the given data.

Although the time curves of the exchange rates in both markets are very similar their mean, low and peak differ substantially. The long period, in which Mt Gox existed but BTCE did not yet exist, influences the mean course of MT Gox to be a lot smaller than the mean course of BTCE. Also the crash of the Mt Gox market in February 2014 affected the course substantially.

	BTCE	Mt Gox
Entries	930	1313
Mean Exchange Rate	121.72	90.66
Lowest Exchange Rate	2.13	0.05
Highest Exchange Rate	1050.28	1213.34

Table 2.2: Exchange Rate Statistics of the Per-Day Time Series

Chapter 3

Methodical Foundation

This chapter focuses on the advanced algorithms which are used to segment and predict time series trends as described in chapter 5. Due to the length constraints of this thesis it gives only a short overview and a brief introduction into every algorithm rather than an in-depth discussion.

The first part of the methodical foundations discusses clustering and several clustering algorithms. In addition to the commonly known K-Means algorithm, Affinity Propagation, HMMs and DESICOM are explained and shortly discussed. The latter 2 also offer similarities or transition probabilities between the clusters/states and can thus be used for cluster- or respectively state-transition prediction. In the second part of the Methodical Foundations, High-Order Markov Chains and Language Models are discussed, which also can be used to predict state sequences.

3.1 Clustering

Clustering is a form of unsupervised learning. This means that the training data, which is learned from by the model, does not contain any labels. Thus the training data consists solemnly of input values. Usually clustering algorithms try to partition the input data into homogeneous disjoint groups. Those homogeneous groups are typically represented by an instance of the input value (exemplar) or e.g. a mean of instances which is regarded as cluster center. The instances can be of higher dimensions. All of the instances which belong to the same group are represented by the according cluster center or exemplar. This can be regarded as dimension-reduction. New instances can be assigned to the clusters by checking which cluster center is most similar to them in the feature space. Closeness can be measured by the Euclidean Distance or other similarity measures. One goal of clustering is to create clusters which have a low within-class scatter and a high between-class scatter. This means that instances which belong to the same group should be similar to each other, while instances of different groups should be dissimilar and thus different clusters should be clearly distinguishable.

In this thesis clustering algorithms will play a crucial role for extracting prototypical intervals of time series which are then used to segment time series. The following subsections describe the clustering algorithms which will be used in this thesis and highlight differences between the approaches. Finally chapter 5 explains how clustering can be used to segment the time series of the Exchange Rate Trends.

3.1.1 K-means

K-means clustering is one of the most popular clustering approaches. As the name implies, this approach partitions n observations into k clusters. The clusters are represented by their cluster centers, which are the means of all instances belonging to a cluster. Those means are called centroids. This implies that the cluster centers do not have to be in the set of observations. They rather are new instances which best represent the observations. These centroids serve as representation of the observations which they are assigned to and can be considered as their prototypes.

Setup and Goal of K-Means

Given a set of n d-dimensional observations $O = \{o_1, o_2, o_3, ..., o_n\}$. K-means aims at finding k clusters which best partition the set of observations into k disjoint sets $K = \{K_1, K_2, K_3, ..., K_k\}$. The way K-Means is defined, a best partitioning is a partitioning which minimizes the within-cluster sum of squares. In other words this means that K-Means aims at finding a set K^*, which minimizes the sum of the distances of the observations to the cluster center they belong to. With k_i as the centroid of partition K_i this can be expressed formally as:

$$arg\min_{K} \sum_{i=1}^{k} \sum_{o \in K_i} \|o - k_i\|^2$$

This is equal to finding k clusters with the smallest Euclidean distances to their observations.

Initialization

The K-Means algorithm is divided into two repeating steps in which clusters are assigned to the observations and new cluster means are calculated. As a starting point, k clusters have to be generated in order to begin with the assignment of the observations and the latter refinement of the centroids. There is a vast amount of heuristics for choosing the initial cluster centers. A commonly used strategy is the so called 'Forgy' method [12].

In this method k randomly chosen observations are picked as cluster centers. It is very important to note that K-Means will not always find a globally optimal solution. The solution is highly dependent on how the initial cluster centers are chosen. This implies that random assignments of the initial clusters, as for example in the 'Forgy' method, can lead to different results in different runs. The results always represent locally optimal solutions.

Assigning the Observations to the Clusters

Once an initial choice of cluster centers has been generated, the observations have to be assigned to the cluster centers in order to build the clusters. This step of K-Means is called the **Assignment step** or **expectation step** with respect to the expectation-maximization algorithm. Since the goal of K-Means is to minimize the within-cluster sum of squares or to minimize the sum of the euclidean distances, this yields an easy way to find the appropriate cluster center for an observation o_i. The most suitable cluster center is always the cluster center k_j which is closest to the observation o_i in euclidean distance. So for every observation o_i K-Means is aiming to find the cluster center k_j for which

$$\|o_i - k_j\|^2$$

is minimized.

Updating the Cluster Centers

Once the observations have been assigned to their cluster centers, they can be used to refine the clusters. The next step is called the **update step** or **maximization step**. In order to minimize the scatter within the clusters, the new cluster centers will be placed as centroids, in the middle of all observations which belong to them. The calculation of the new centroid k_i^* of a cluster i corresponds to calculating the mean of all observations $o_j \in K_i$

$$k_i^* = \frac{1}{|K_i|} \sum_{o_j \in K_i} o_j$$

Updating the centroids not only affects the position of the centroids, but also which observations are closest to them. This leads to new assignments and a crucial part of the K-Means algorithm.

Iterations and Stopping Criteria

The new assignment of observations signals that the algorithm has not yet converged. K-means is not done after one iteration of assignment and updating, but runs as many iterations as needed, to reach a state in which no new assignments are made. In this case a local optimum has been found.

In order to find possibly better solutions, K-Means does not have to be run once but can be run multiple times. The best clustering of all runs can be determined by picking the partitioning with the lowest within-cluster sum of squares.

Discussion

The K-Means clustering algorithm does not guarantee a globally optimal solution. The quality of the solution is highly dependent on the initial set of clusters which is chosen.

Another factor which crucially impacts the quality of the solution is the number of clusters k. If k is chosen poorly, the partitioning of the observations will be of low quality. It is hence very important to choose an appropriate number of clusters.

Yet K-Means is a very easy to understand and powerful clustering algorithm.

3.1.2 Affinity Propagation

Affinity Propagation [13] clusters data based on measurements of similarity between instances. In contrast to K-Means Affinity Propagation aims at finding instances of the observations which best represent the other observations. This means that the clusters are represented by actual observations which are called exemplars. Affinity Propagation does not specify a fixed number of clusters beforehand.

While K-Means starts with an initial set of clusters, Affinity Propagation regards every point as possible exemplar. It utilizes a network between the instances. The instances recursively communicate with each other via messages. The messages contain information, that represent the affinity of one instance to a possible exemplar. Hence the name Affinity Propagation.

Initialization

The input of Affinity Propagation is a similarity matrix, containing pairwise similarities for data points d_a and d_b. The euclidean distance is used as similarity measure. The similarity s is defined as:

$$s(a, b) = -\|d_a - d_b\|^2$$

This similarity indicates how well suited data point d_b is to serve as exemplar for data point d_a.

Another input for Affinity Propagation is the preference value $s(b, b)$ for each data point. A data point which is assigned a higher preference value than others is more likely to be an exemplar. If all instances are equally likely to be chosen as exemplars, they should all be assigned the same preference value. The preference values highly influence how many clusters are found.

Messages between Instances

The messages which are sent from point to point serve two purposes. They are used to determine which data points are exemplars and which exemplar a data point belongs to.

The first message is called **responsibility**. The responsibility message $r(a, b)$ which is send from data point d_a to data point d_b tells data point d_b how well suited it is to be the exemplar for data point d_a with respect to all other potential exemplars d'_b.

The other message which is passed between data points is the **availability** $av(a, b)$. The availability is sent from data point d_b to data point d_a and signifies how suitable d_b is as exemplar for d_a, considering how suitable d_b is for all other data points d'_a.

Initially all availabilities $av(a, b)$ are set to zero.

Responsibility and Availability

The update rule for the responsibilities is as follows:

$$r(a, b) \leftarrow s(a, b) - \max_{b' \neq b} \{av(a, b') + s(a, b')\} \tag{3.1}$$

Since the availabilities are set to zero initially, the meaning of the responsibility update is very simple. It is the similarity of a to the potential exemplar b, subtracted by the largest similarity of a to the other potential exemplars b'. The self-responsibility $r(b, b)$ is set to the inputted preference value $s(b, b)$, subtracted by the largest $s(a, b)$, with $a \neq b$ for each data point.

The availability $av(a, b)$ update formula is defined by the following equation:

$$av(a, b) \leftarrow \min \left\{ 0, r(b, b) + \sum_{a' \notin \{a,b\}} \max\{0, r(a', b)\} \right\} \tag{3.2}$$

The availability represents the accumulated evidence from all data points d_a as to how well d_b is suitable to be their exemplar. It does not take negative responsibilities into account since this would bias the choice of exemplars towards those which are similar to all instances. This accounts for the cases in which the exemplar only represents some data points well and is very ill-suited for data points which belong to other exemplars.

The self-availability $av(b, b)$ is defined as:

$$av(b, b) \leftarrow \sum_{a' \neq b} \max\{0, r(a', b)\} \tag{3.3}$$

It describes the accumulated suitability of b to serve as exemplar for other points. Availability values are limited to a maximum value of 0 to minimize the influence of very high responsibility values by some data points.

Interplay of Responsibility and Availability

It is easy to observe that responsibility and availability mutually influence each other. While the availability of the data points is initially 0 it will quickly drop for non-exemplars to a negative value. Negative availability values affect the responsibility as can be seen in equation 3.1.2. The responsibility of exemplars will increase because the availability of non-exemplars decreases. Vice versa the responsibility of non-exemplars will decrease.

This again influences the availability of exemplars and non-exemplars. Exemplars which now have higher responsibilities also have higher availabilities, while non-exemplars will have lower responsibilities.

Update and Decision Rule

In every iteration of Affinity Propagation availabilites and responsibilities are updated and messages are sent. The algorithm usually converges fast. Amongst other, some suitable ending criteria for the algorithm are:

- A certain preset amount of iterations has been completed
- The changes by the messages are below a given threshold
- The assignment of the exemplars did not change for a certain amount of iterations

The exemplar of a data point can easily be calculated by combining availability and responsibility values. The exemplar for a data point d_a is the data point d_b which maximizes

$$av(a, b) + r(a, b)$$

In order to avoid fluctuations, which can occur while updating availabilities and responsibilities, the influence of messages is cooled down with a damping factor λ. A message m_i at iteration i is set to

$$m_i = \lambda \cdot m_{i-1} + (1 - \lambda) \cdot m_{update}$$

where m_{update} is the update value which was computed in iteration i.

Discussion

Affinity Propagation is clearly different than K-Means. It depends on the application domain whether the differences are beneficial or not. Affinity Propagation is more flexible in a sense that it doesn't restrict the cluster-search to a preset number of clusters. But on the other side it is hard to find suitable preference-values which limit the number of clusters to a useful range. In the literature the Median and the Minimum value of all similarities are recommended as preference values, when all instances are equally suited to serve as exemplars. Depending on the context these values often do not limit the amount of clusters to a useful range. This can yield a trial and error process for finding suitable preference values.

Depending on the context, it can be also an up- or downside that the clusters are represented by exemplars. This can be very useful if new instances cannot be easily generated but also bad if none of the given instances is very suitable to represent the other instances.

3.1.3 Hidden Markov Models

Given a sequence of observations, the HMMs [14] task is to find representative states and assign a label or state to every observation in the sequence. The assumption is that the sequence of states is generated by Markov processes. Markov processes, the HMM and how HMMs can be used for clustering will be explained in the following subsections. Since HMMs are a broad topic this Subsection focuses only on those parts which are most important to understand the general concept and the algorithms needed for the purpose of this thesis.

Markov Processes

One of the main assumptions when using HMMs is that the state and observation sequences are described by Markov processes. Markov processes are 'memoryless' stochastic processes. In the discrete time-case which is relevant for this thesis, memoryless means that the probability of observing a state s_t at time t is only dependent on the state s_{t-1} observed at time $t-1$. This is known as the **Markov property**. Given a set of random variables $X = \{X_t\}_{t\in\mathbb{N}}$ and a set $S = \{s_1, s_2, s_3, ...\}$ of states the Markov property is defined as

$$\mathcal{P}(X_n = s_n | X_{n-1} = s_{n-1}, X_{n-2} = s_{n-2}, ...X_0 = s_0) = \mathcal{P}(X_n = s_n | X_{n-1} = s_{n-1})$$

Description of an HMM

These are the notations which will be used to explain the HMM in the following subsections:

- observation sequence $OS = (OS_1, OS_2, OS_3, ..., OS_l)$
- l = length of observation sequence
- n = number of model states
- m = number of distinct observations
- distinct states $S = \{s_1, s_2, s_3, ..., s_n\}$
- distinct observations $O = \{o_1, o_2, o_3, ..., o_m\}$
- state transition probability matrix T_S
- state observation probability matrix O_S
- I_S the initial probability distribution of the states

T_S is a matrix of dimension $n \times n$. Every entry ts_{ij} determines the probability of a transition to state j given state i. O_S is of dimension $n \times m$ and $os_i(o)$ states the probability of observation o given state i. I_s determines the probabilities of starting in a particular state. T_S, O_S and i_s are sufficient to define an HMM.

Finding an HMM model for a given observation sequence

There are many known applications and many suitable problem tasks for HMMs. The one which is important for this work is the situation of knowing only an observation sequence, the desired number of states and the number of observations. The goal is to make conclusions about the states given the observations. It is clear to see that this is related to clustering where there were observations and it was tried to find representative prototypes.

The desired output is an HMM model $hm = (T_S, O_S, I_S)$ which best suites the observation sequence OS given the constraints on n and m. Two algorithms are combined to solve this problem. The forward algorithm and the backward algorithm which can be used to solve other HMM related problems.

Forward Algorithm

The first algorithm is the forward algorithm. For a given model and an observation sequence it can find the likelihood that the observation sequence is observed, given the model. The resulting likelihood is:

$$\mathcal{P}(OS|hm)$$

This probability needs to be found, given the model $hm = (T_S, O_S, I_S)$ and the observation sequence $OS = (os_1, os_2, os_3, ..., os_T)$. Let $X = \{x_1, x_2, x_3, ..., x_T\}$ be the sequence of hidden states. This yields that the probability of the observation sequence, given the state sequence and the model, is

$$\mathcal{P}(OS|S, hm) = o_{x_1}(os_1) \cdot o_{x_2}(os_2) \cdot o_{x_3}(os_3) \cdot ... \cdot o_{x_T}(os_T)$$

The likelihood of the state sequence can easily be computed with

$$\mathcal{P}(X|hm) = i_{x_1}$$

It is also possible to show that

$$\mathcal{P}(OS, S|hm) = \mathcal{P}(OS|S, hm)\mathcal{P}(S|hm)$$

which leads to the following way to compute $\mathcal{P}(OS|hm)$:

$$\begin{aligned}\mathcal{P}(OS|hm) &= \sum_S \mathcal{P}(OS, S|hm) \\ &= \sum_S \mathcal{P}(OS|S, hm)\mathcal{P}(S|hm) \\ &= \sum_S i_{x_1} o_{x_1}(os_1) t_{x_1,x_2} o_{x_2}(os_2) ... t_{x_{l-1},x_l} o_{x_l}(os_l)\end{aligned}$$

This shows that the sum over all possible state sequences to yields the probability $\mathcal{P}(OS|hm)$. Summing over all possible state sequences which is in practice not feasible. This would take $2ln^l$ calculations.

To make this calculation feasible the forward algorithm which is also called α-pass is used. Partial observation sequences are defined as

$$\alpha_k(j) = \mathcal{P}(os_1, os_2, \ ... \ , os_k, x_k = s_j|hm).$$

The forward algorithm exploits that those partial sequences can be calculated recursively in n^2l operations with the following 3 rules:

1. $\alpha_0(j) = i_j os_j(os_0)$ for states $j = 1, 2, ..., n$
2. $\alpha_k(j) = \big[\sum_{v=1}^{n} \alpha_{k-1}(v) t_{vj}\big] os_j(os_k)$ for $k = 1, 2, ..., l$ and $j = 1, 2, ..., n$
3. $\mathcal{P}(OS|hm) = \sum_{j=1}^{n} \alpha_l(j)$

Backward Algorithm

With the backward algorithm the most likely state sequence given an observation sequence and an HMM can be calculated. In this case most likely means that as many as possible states are correct. The backward algorithm is also called β-pass. It is basically an α-pass in reversed order. This time define $\beta_k(j) = \mathcal{P}(os_{k+1}, os_{k+2}, ..., os_l, x_k = s_j|hm)$. Then again 3 rules are needed to calculate $\beta_k(j)$:

1. $\beta_l(j) = 1$ for states $j = 1, 2, ..., n$
2. $\beta_k(j) = \sum_{v=1}^{n} t_{j,v} os_v(OS_{k+1}(v)\beta_{k+1}(v)$ for $k = l-2, l-3, ..., 0$ and $j = 1, 2, ..., n$
3. Define $\gamma_k(j) = \mathcal{P}(x_k = s_j|OS, hm)$ for $k = 0, 1, ...l-2$ and $j = 1, 2, ..., n$, then $\gamma_k(j) = \frac{\alpha_k(j)\beta_k(j)}{\mathcal{P}(OS|hm)}$ and the state which maximizes $\gamma_k(j)$ is the most likely state at k

Forward-Backward algorithm

Finally the initially addressed problem can be solved. The forward-backward algorithm or Baum-Welch algorithm makes it possible to find an HMM with a fixed amount of observations and states, which best fits an observation sequence.

To find the model, a randomly initialized model is used and then refined. To refine it $\gamma_k(j,h) = \mathcal{P}(x_k = s_j, x_{k+1} = s_h|OS, hm) = \frac{\alpha_k(j)t_{j,h}os_h(OS_{k+1})\beta_{k+1}(h)}{\mathcal{P}(OS|hm)}$ is introduced with $\gamma_k(j) = \sum_{h=1}^{n} \gamma_k(j,h)$. With this definition everything needed to do the reestimation is given:

1. $i_j = \gamma_1(j)$ for $j = 1, 2, ..., n$
2. $t_{j,h} = \sum_{k=1}^{l-1} \gamma_k(j,h) \Big/ \sum_{k=1}^{l-1} \gamma_k(j)$ for $j = 1, 2, ..., n$ and $h = 1, 2, ..., n$
3. $os_h(g) = \sum_{k \in \{1,2,...,l-1\} \& OS_k = g} \gamma_k(h) \Big/ \sum_{k=1}^{l-1} \gamma_k(h)$

These steps can be repeated until a desired number of iterations has been completed, the probability of $\mathcal{P}(OS|hm)$ does not improve anymore or the improvements are below a certain threshold.

Discussion

HMMs clearly represent a different approach to clustering than seen in K-Means and Affinity Propagation. Yet it is easy to understand that the states represent some kind of cluster prototypes, whereas the observations can be seen as instances belonging to clusters. Like in K-Means it is possible to fix the number of clusters or respectively states beforehand.

In contrast to the K-Means and Affinity Propagation algorithms, HMMs also compute transition probabilities between clusters. This can be helpful for predicting state sequences and can also be used as input for the next clustering algorithm.

3.1.4 DESICOM

DESICOM [15] or DEcomposition into SImple COMponents is built upon the DEDICOM algorithm (amongst other: [16]). DEDICOM is an antonym for DEcomposition into DIrectional COMponents. Hence it is very useful to start this chapter with a quick explanation of DEDICOM and then use this as a basis to explain DESICOM. Both algorithms take

an (asymmetric) similarity matrix of elements as input and factor it into two matrices. One matrix that shows which component each element belongs to and a matrix that puts the components into relations.

DEDICOM

In contrast to K-Means the directional components, in which DEDICOM separates the elements, are not related in a spatial sense, but rather in a way similar to Affinity Propagation. DEDICOM seeks transitive relations among elements, which is similar to the propagated affinity values in Affinity Propagation. Given an asymmetric $n \times n$ matrix S of similarities between n elements, the main idea of DEDICOM is to factorize this matrix into

$$S = ARA' + E$$

Here A represents an $n \times r$ matrix which states for each of the n elements in which extent it is part of each of the r components. Matrix R is of shape $r \times r$ and contains the asymmetric relationship values between each of the r components. E is an $n \times n$ matrix of error terms. This yields the following relationship between two elements i and j:

$$\sum_k \sum_l a_{ik} a_{jl} r_{kl} + e_{ij}$$

In words this means that the relationship between two elements i and j is composed as the sum of the product of the relationship of i to every component l, the relationship of j to every component k and the relationship between the components j and k. The relationship is complemented by the error term for the elements i and j. The objective function of DEDICOM is to minimize the squared error or formally:

$$\|S - ARA'\|^2$$

This is to be treated equal as to minimize the error term.

There are various ways to optimize this algorithm but they are not to discussed here. There is a decent amount of literature available that discusses the DEDICOM algorithm.

A negative aspect of DEDICOM is that the affinities among the components expressed in R can also be negative. This makes it hard to interpret the results. DESICOM solves this problem.

DESICOM

DESICOM uses the same objective function as DEDICOM. Like DEDICOM, DESICOM tries to find a least squares fitting of the model as a Function of A and R:

$$o(A, R) = \|S - ARA'\|^2 \tag{3.4}$$

A constraint on this objective function is that each row in A has $r - 1$ zero values and only 1 non-zero value. The column in which this non-zero value is, corresponds to the component which the element in the row belongs to.

The objective function is minimized by an algorithm that updates one row of A in every iteration and then updates R.

The i^{th} row and column of S is denoted as r_i and c_i and the i^{th} element in the respective row or column is deleted. To minimize the objective function 3.1.4 over row i in A, which is referred to as a_i', the other rows of A are fixed and the objective function is split as follows:

$$\begin{aligned} g(a_i') &= c + \|c_i - A_i R a_i\|^2 + \|r_i' - a_i' R A_i'\|^2 + (x_i i - a_i' R a_i)^2 \\ &= \left\| \begin{pmatrix} c_i \\ r_i \end{pmatrix} - \begin{pmatrix} A_i R \\ A_i R' \end{pmatrix}_l a_i \right\|^2 + (x_{ii} - {a_{il}}^2 r_{il})^2 \end{aligned} \tag{3.5}$$

Note that in the second line only the l^{th} column of the matrix is considered, where l has to take the value that minimizes the function. The simplest way to find the appropriate value for l is, to try every possible value. But this can also be solved directly by minimizing a fourth degree polynomial $h(a_{il})$. The computation of $h(a_{il})$ is explained in more detail in [15] . Once the minimum $l = m$ is found, a_{im} is set to its corresponding minimum and all other a_{il} with $l \neq m$ are set to 0.

Now it is time to update R. This can be done by calculating the Moore-Penrose inverse [17] A^+ of A and solving $R = A^+ S (A')^+$. The described calculations lead to an iterative algorithm that stops once it reaches a local or hopefully global minimum. The algorithm can be rerun and A and R can be initialized differently in order to find better results.

Discussion

DESICOM offers a clustering approach which is based on transitive affinities of observations. In contrast to the other algorithms presented in this section, the algorithm is not fed with the observations themselves but rather with an asymmetric matrix of their affinities. DESICOM outputs not only the clusters for each of the observations, but also a relationship matrix R which shows affinities among the clusters.

Algorithm 3.1: DESICOM algorithm

Data: Asymmetric similarity matrix S
Result: at least locally optimal solutions for R and A

Choose r `// number of components`
Initialize A and R
Compute initial value $o^{ini} = \|S - ARA'\|^2$
for $i = 1$ ***to*** m **do**
    Assign $backup = a'_i$
    Compute $\begin{pmatrix} c_i \\ r_i \end{pmatrix}$ and $\begin{pmatrix} A_i R \\ A_i R' \end{pmatrix}$
    Find the $l = m$ that minimizes $h(a_{il})$
    if *after setting* $a_{il} = 0$ *for* $l \neq m$ *a column of* A *is all-zero* **then**
        roll back and set $a'_i = backup$
    else
        set $a_{il} = 0$ for $l \neq m$ and $a_{im} = h(a_{im})$
Update R
Compute $o = \|S - ARA'\|^2$ and **if** $o^{ini} - o < \epsilon$ **return** A and R **else** goto Step 4

3.2 Prediction

In this Section two methods are presented which can be used for statistical analysis of transitions between states. The first is high-order Markov Chains which uses Markov processes of higher order than the discussed HMMs. Instead of taking only one former state into account high-order Markov Chains can handle arbitrary histories. The second method is statistical language model creation. It is built upon high-order Markov Chains and thus directly related. It will be shown how language models can be used to predict state transitions.

3.2.1 Markov Chains

High-Order Markov Chain models [18] are built open the Markov property which was introduced in Section 3.1.3. In contrast to the HMM model introduced in that Section, High-Order Markov Chains keep a history of order k and make the probability of transiting to a state dependent on that history.

1st Order Markov Chains

Generally the probability of a sequence is calculated with

$$P(X_0 = x_0, ..., X_n = x_n) = P(X_0 = x_0) \prod_{t=1}^{n} P(X_t = x_t | X_{t-1} = x_{t-1}, ..., X_0 = x_0)$$

1st-Order Markov Chains utilize the Markov property and simplify this to

$$P(X_{t+1} = x_{t+1} | X_t = x_t, ..., X_0 = x_0) = P(X_{t+1} = x_{t+1} | X_t = x_t)$$

Recall that this means that the probability of a transition to a state is only dependent on the current state. For 1st-Order Markov Chains the probability of a sequence $X = x_0, x_1, ...x_n$ is obtained by calculating

$$P(X) = p_0(x_0)p(x_1|x_0)p(x_2|x_1)...p(x_n|x_{n-1})$$

The probability of observing x_k at time point k is

$$p_k(x_k) = \sum_{x_{k-1} \in S} p(x_k|x_{k-1})p_{k-1}(x_{k-1})$$

In words this is the sum of the probabilities of observing an element x_{k-1} at time point $k-1$, multiplied with the probability to transit to an element x_k. This represents the sum of the probabilities of all possible transitions to the element k.

This can be simplified to a matrix multiplication problem if assuming that there is only a finite amount of states $|S| = m$. Then a transition matrix P with $P_{xy} = p(y|x)$ can be defined and p_k can be computed as follows:

$$p_k = p_0 P^k$$

High-Order Markov Chains

In many cases the Markov assumption is not sufficient. It is not always enough to only rely on the current state to make assumptions about the next state. Often a history of observations is helpful in determining what is going to happen next. High-Order Markov Chains account for exactly this case. The following example displays the second order case:

$$P(X_{t+1} = x_{t+1} | X_t = x_t, ..., X_0 = x_0) = P(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1})$$

It is easy to see that this is still a Markov Chain, if making the following substitutions:

$$p(z|x,w) = P(X_{t+1} = z|X_t = x, X_{t-1} = w)$$
$$Z_t = (X_t, X_{t-1})$$
$$P(Z_{t+1} = (z,y)|Z_t = (x,w)) = \begin{cases} 0 & if x \neq y \\ p(z|x,w) & otherwise \end{cases}$$

The 2nd-Order case can be easily reduced to the 1st-Order case by creating a new state space $Z = S \times S$. This can be extended to any order k.

3.2.2 Language Models

Language models [19] are a common tool in language processing and speech recognition. N-Gram models are language models which predict words based on a sequence of n words. More clearly this means that an N-Gram model predicts a word from the $n-1$ previous words. This allows not only computing the probability of the next word but also of a sequence of words or a sentence. To achieve this, language models count word sequences in corpora. A corpus can be seen as a training set of words. Usually large data sets such as archives of newspapers are used to learn language models. N-Gram models are closely related to Markov Chains and use the same assumptions as them.

The use of Language Models

Language models are useful for a lot of different applications such as machine translation or speech recognition. In speech recognition the input can be very noisy. It is not always clear what a speaker has said, not even to humans. Assume there is a speaker that mumbles into a microphone. The speech recognition narrowed the most likely options of the sentence based on the audio input down to

1. "I go to the Jude Law."
2. "I go to the Jew law."
3. **"I go to the jeweler."**

A language model will indicate that the third option is the most likely option. Based on the training data it is built from it will recognize "go to the jeweler" as far more probable 4-Gram combination as for example "go to the Jude". Even a 3-Gram model should be sufficient in this case. The results all clearly depend on the training data. Older newspaper might for example never contain the 2-Gram "Jude Law".

While this is not a predictive usage in a sense that the next word is predicted, such predictive use-cases are very common today. Modern smartphones use language models to predict which words will be typed next. They offer suggestions based on known probabilities but also extend their knowledge by the texts which the user inputted.

Simple N-Grams

The key in building N-Gram models is counting occurrences of words and word combinations. The following example should advertise why it is not sufficient to simply check how often sequences of words have occurred in the training corpora but rather split them into N-Grams and their subsequences. Assume that the sentence-fragment "is the inventor of" is given and the probability that the next word is "Bitcoins" should be determined. One solution is to simply calculate the probability by counting how often the sentence fragment was followed by "Bitcoins" in the training data. This would result in:

$$P(Bitcoins|is\ the\ inventor\ of) = \frac{C(is\ the\ inventor\ of\ Bitcoins)}{C(is\ the\ inventor\ of)}$$

where $C(w)$ represents the count of the word sequence w. While this can result in a positive probability, this holds not anymore if the sentence fragment is changed in a way that its content is something not seen in the training corpus. If the sentence is modified to be "Barack Obama is the inventor of", then the probability of "Bitcoins" being the next word is equal to zero. In fact the probability of every word is equal to zero if the sentence fragment has not occurred in the training data. While it can be regarded as not too critical considering only this example, it is critical in many ways. Every sentence construct that has not been observed in the training data would result in a zero-probability according to the language model.

This is one of the reason why for N-Gram models the Markov assumption is used and extended to higher orders. Instead of regarding the whole sentence fragment as in the example above, a Bi-Gram model could be used which would decide upon the probability of the word "Bitcoins" only in the context of the word before. This means that the probability would be $P(Bitcoins|of)$. This can be generalized to N-Grams. Define w_1^n as a sequence of words $(w_1, w_2, w_3, ...w_n)$. The assumption for N-Grams is then

$$P(w_n|w_{n-N+1}^{n-1}) \approx P(w_n|w_1^{n-1})$$

In words this means that instead of looking at the whole history of words, the history can be approximated by only the last few words.

For the sake of simplicity Bi-Gram models are used for the rest of this subsection. The probability of a word sequence using Bi-Grams can be approximated with

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

To obtain the Bi-Gram probabilities $P(w_k|w_{k-1})$ the maximum likelihood can be used, which normalizes the counts over the training data set. For a Bi-Gram consisting of the words w_n and w_{n-1} and a count of $C(ww_{n-1})$ this leads to

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n}{\sum_w C(w_{n-1}w)}$$

The normalization is over all Bi-Grams which contain the word w_{n-1} as prefix. It is easy to understand that the sum of all of the Bi-Grams that contain the prefix w_{n-1} is the same as the Uni-Gram counts of w_{n-1}. This leads to

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n}{C(w_{n-1})}$$

Generalizing this result to N-Grams yields

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

Although this improves the results in many cases, there are still cases in which there are zero probabilities. This happens in cases where a N-Gram did not occur in the training data.

Smoothing and Backoff

To avoid such cases two techniques called smoothing and backoff can be used. The simplest way of smoothing is **Laplace smoothing** which simply adds one count to each possible observation and reestimates the probabilities by also adjusting the denominator appropriately. But this smoothing algorithm is too simple and adds too much probability mass to unseen N-Grams. This affects the probabilities of the known N-Grams drastically.

The **Good-Turing Smoothing** algorithm improves this behavior. It uses the count of singletons, things that have only been observed once, to estimate the probabilities of things that have never been observed. In general it keeps tracks of how many times objects have been observed and adjusts the counts in dependence of the next more frequent observations. This means that the counts of the objects which have been observed c times are adjusted by

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

and for the objects which have not been observed at all this is adjusted to

$$c^* = (c+1)\frac{N_1}{N}$$

where N is the total number of objects in the training set. This algorithm is still problematic in the case that any of the N_{c+1} counts is 0, which happens frequently. Interpolation and backoff algorithms are thus introduced. Interpolation algorithms interpolate the probabilities of N-Grams with (n-1)-Grams, (n-2)-Grams,...,Uni-Grams. Backoff algorithms fall back to the next lower category of N-Grams. This means that if there are no counts for N-Grams, they will backoff to the (n-1)-Gram counts.

Kneser-Ney Smoothing

The smoothing algorithm which is used in this thesis is the **Interpolated Kneser-Ney Smoothing** algorithm. Kneser-Ney smoothing incorporates the context in which words appear in the smoothing process. It is very easy to understand this with a well-known example. Assume that the language model should complete the sentence:

I can't see without my reading _ _ _ _ _ _ _ _ _

Now if the language model did not contain any Bi-Grams starting with reading, it will backoff to the Uni-Gram case. But the obvious choice *glasses* might have occurred less frequently in the training data than for example the word *Francisco*. The simple backoff algorithm would thus falsely conclude the sentence with *Francisco*. Kneser-Ney smoothing incorporates the fact that *Francisco* only appears in the context *San*. In contrast to that, *glasses* appears in many different contexts. Kneser-Ney thus estimates the probabilities on the number of contexts a word appears in. The interpolated form of Kneser-Ney has the following probability estimate:

$$P_{KN}(w_i|w_{i-1}) = \frac{C(w_{i-1}) - D}{C(w_{i-1})} + \beta(w_i)\frac{|\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}{\sum_{w_i} |w_{i-1} : C(w_{i-1}w_i) > 0|} \tag{3.6}$$

where D is an absolute discounting factor, subtracted from every count and $\beta(w_i)$ is a factor that is used to ensure that this stays a probability distribution. Note that discounting by an absolute factor D affects mostly the objects which have been counted few times. This is a desired effect because trust in their probabilities is lower than in the probability of more frequently observed objects.

Evaluating language models

The most common measure for language model evaluation is the **perplexity**. The perplexity of a language model expresses how likely a test set is given a language model. This means that given a sequence of words in a test set $W_T = (w_1, w_2, ..., w_N)$ the perplexity expresses the probability of this sequence normalized by the number of words N. The perplexity of N-Gram language models is calculated by

$$PP(W) = \sqrt[n]{\prod_{i=1}^{N}] \frac{1}{P(w_i|w_{i-1})}} \tag{3.7}$$

A lower perplexity means better results.

Discussion

N-Gram models are built upon the same principle as High Order Markov Chain Models. Both extend simple transition probabilities by a history of transitions. This can lead to better results if the current state really depends on more than one past state, but will distort the probabilities if the states do not depend on a history of states.

The Kneser-Ney Smoothing extends the N-Gram models. It enables a better handling of unseen elements. The backoff to observations that have been seen in more contexts gives the model more flexibility and also more accuracy in many cases.

Chapter 4

Analysis of the Exchange Rate Time Series

This chapter deals with the analysis of the exchange rate time series of the Mt. Gox and BTCE-USD markets. The effect of smoothing is studied and the derivatives of the time series are examined. Afterwards basic time series analysis methods are applied to the data and the results are discussed. The purpose of this analysis is trying to find periodicity and similarities to other economic quantities. Also the ARMA model is applied which is a very commonly used tool in time series analysis and also used for forecasting.

4.1 Basic Analysis

In this first Section the raw time series are analyzed and then basic analysis methods are applied and their effect is studied. This is to gain an understanding of the exchange rate development of the Bitcoin markets over time.

4.1.1 Smoothing

Smoothing time series reduces volatility at the cost of precision. To smooth the time series a Gaussian filter is applied to the data.

Gaussian Filter

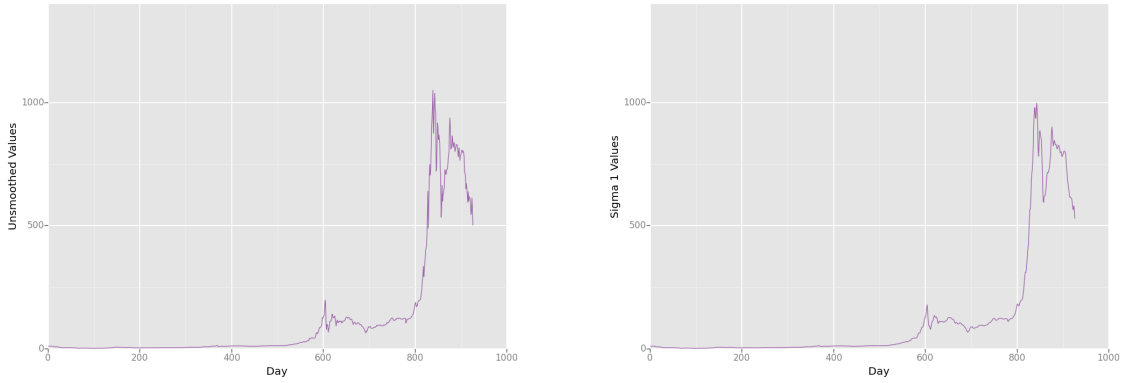
The Gaussian filter is the Gaussian function applied to the data by convolution. The one-dimensional Gaussian kernel is defined by

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

where σ is the standard deviation. It determines how wide the filter is. A wider filter incorporates more distant values into the new smoothed value. This means that the effect of smoothing gets stronger with a higher value for σ

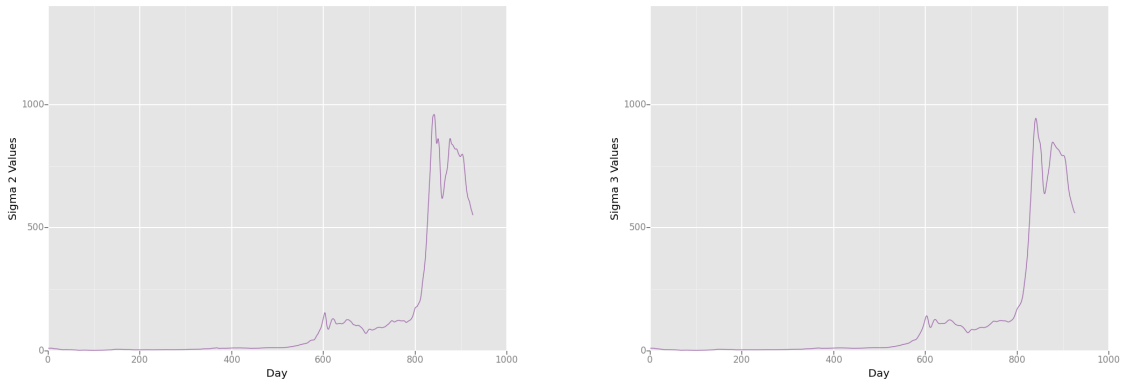
Applying the Gaussian Filter

The following 8 plots show the original time series for the BTCE and Mt Gox market and then the smoothed versions with $\sigma = \{1, 2, 3\}$.



(a) Unsmoothed

(b) Sigma 1

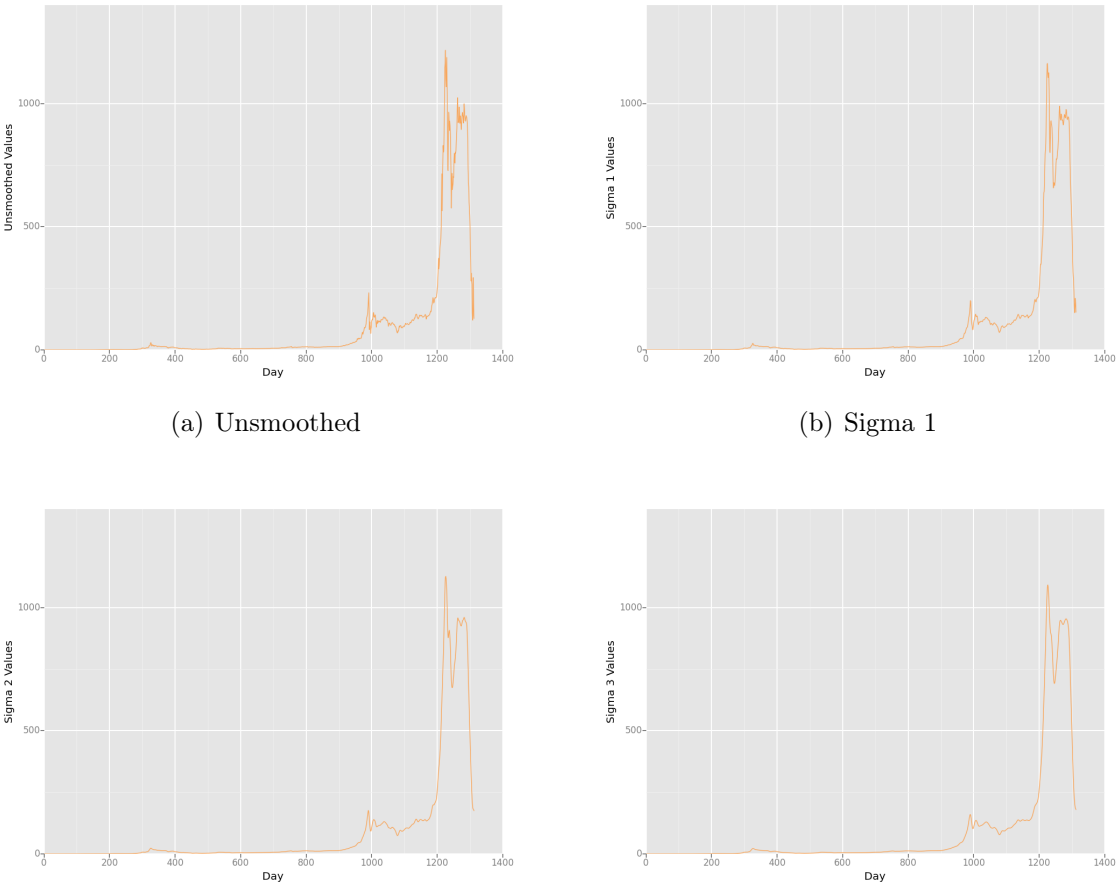


(c) Sigma 2

(d) Sigma 3

Figure 4.1: BTCE Time Series - Unsmoothed vs Smoothed

The mean value of the time series drops and both time series loose a lot of volatility due to the smoothing. To measure the difference between the original plot and the smoothed versions, the normalized root mean squared error between both time series can be calculated with



(a) Unsmoothed

(b) Sigma 1

(c) Sigma 2

(d) Sigma 3

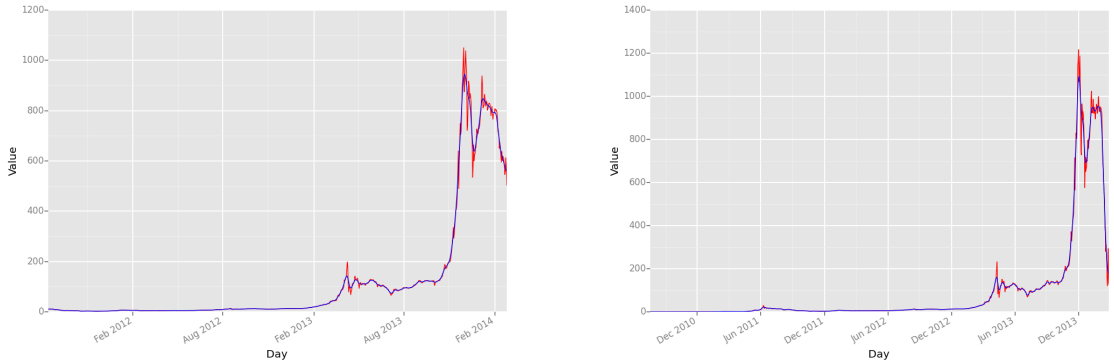
Figure 4.2: Mt Gox Time Series - Unsmoothed vs Smoothed

$$E_{NRMS} = \frac{\left(\frac{\sqrt{\sum_{t=1}^{n}(x_1(t)-x_2(t))^2}}{n}\right)}{x_{2_{max}} - x_{2_{min}}}$$

where $x_1(t)$ and $x_2(t)$ are the values of the time series at point t and n is the number of time points.

The following table shows how the NRMSE changes for increasing σ:

	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
BTCE	0.8%	1.3%	1.5%
Mt Gox	0.6%	1.1%	1.3%



(a) BTCE

(b) Mt Gox

Figure 4.3: Original time series (red) vs smoothed with $\sigma = 3$ (blue)

Although the errors are very low in figure 4.3 a drastically reduced volatility of the smoothed time series can be observed. Still the overall relative development of the time series is clear, even in the smoothed case.

4.2 Periodicity

Looking for periodicity in time series can be very helpful. Periodicity indicates the reoccurring of events and is thus useful for understanding the underlying processes of a time series and for predicting how the time series will evolve.

4.2.1 Fourier Analysis

The Fourier analysis is the spectral analysis of a function. It decomposes a time series into its frequencies which represent the cyclic components of the time series. Fourier analysis can be used to find seasonal fluctuations of a time series.

Concept

The concept of the Fourier analysis is to decompose functions into sums of periodic functions. The Fourier transform transforms a time-based function f into a frequency-based function $\hat{f}$. The Fourier transform $\hat{f}$ for non-periodic continuous time signals is defined as

$$\hat{f}(w) = \int_{\mathbb{R}} f(t) e^{-2\pi i \omega t} dt$$

$\hat{f}(w)$ specifies in which amount the the frequency w is involved in f. The decomposition in periodic functions becomes clear when applying Euler's theorem:

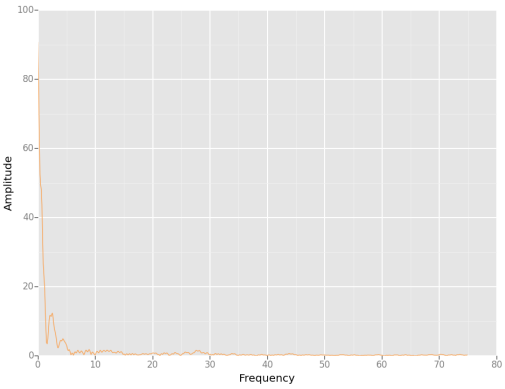
$$e^{-i2\pi t} = cos(2\pi ft) - isin(2\pi ft)$$

The computation of $\hat{f}$ can be transferred to the non-periodic discrete case. With the Fast Fourier Transform a very efficient algorithm for the discrete Fourier transform exists.

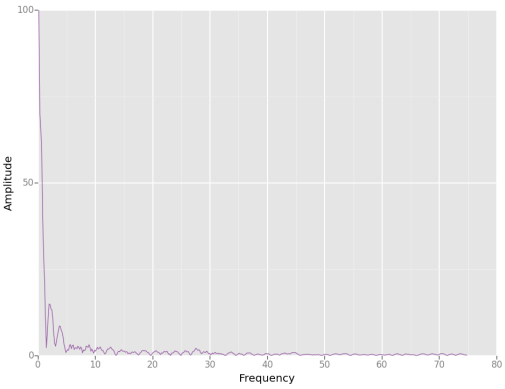
Application on the Raw Data

First the Fourier Transform is applied on the raw data of the Mt. Gox and BTCE Bitcoin Exchange Rates.

The Fourier Transforms of both series are very alike as can be seen in Figure 4.4. This is due to the similarity of both series. Other than that the Fourier Transform reveals that aside a peak at the zero-frequency there are not many other frequencies which are involved in both time series. All of the peaks are in rather low frequencies. This indicates that there are some long-term periodicities. But the low intensity of those peaks emphasizes that there are no strong periodic observations. The low frequencies point towards the uprising trend of the exchange courses.



(a) Mt Gox Fourier Transform



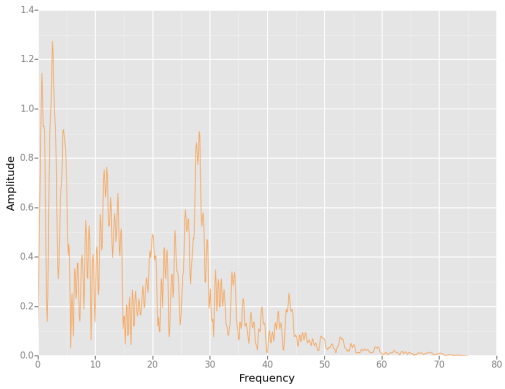
(b) BTCE Fourier Transform

Figure 4.4: Fourier Transform of the original time series

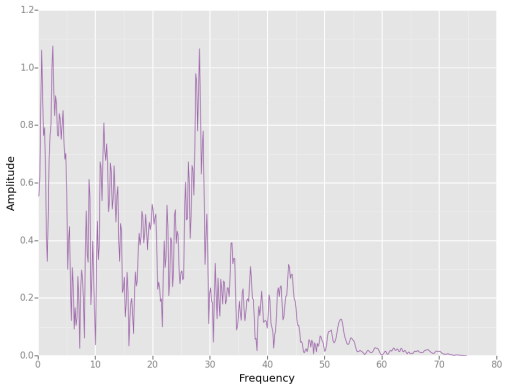
Application on the First Derivative of the Data

Next the Fourier Transform is applied to the first derivative of the time series. Recall that the first derivative represents the change of the original time series over time.

Figure 4.5 shows that the first derivative of the time series clearly has more periodic components than the original time series. The low- and mid- frequency responses are high enough to suggest that there are some periodicities in the long- and mid-term range. This indicates that the development of the exchange rate underlies periodic processes. Rises



(a) Mt Gox Fourier Transform 1st Derivative



(b) BTCE Fourier Transform 1st Derivative

Figure 4.5: Fourier Transform of the derivatives

and falls seem to repeat in certain time intervals. Note however that the amplitude is a lot lower than for the original time series.

4.2.2 Autocorrelation

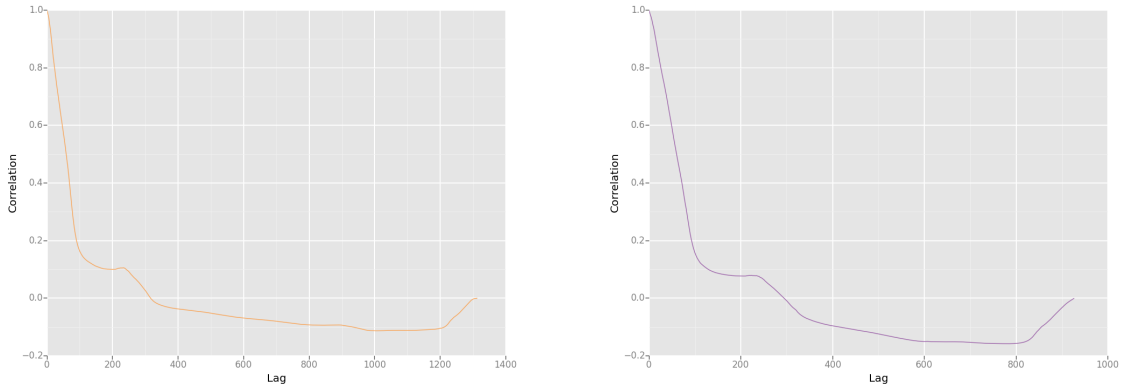
Autocorrelation is the correlation of a time series with a time-shifted version of itself. It can be used to find periodic patterns in a time series. The mathematical definition of the autocorrelation is expressed as the correlation of the time series at time point t and its shifted version at time s:

$$AC(s,t) = \frac{E[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s}$$

where σ is the standard deviation, μ the mean and $E[]$ the expected value which is technically the average. In words, this measures the deviation from the mean of each data point of the time series at time t and its shifted version s. This is a good indicator for a similar progression of both time series. Similar progressions at different time lags would mean that periodic behavior is given. The shift is implemented in a wheel-like way. The last value is not omitted but prepended.

Application on the Raw Data

The autocorrelation is first applied to the raw data of the time series.



(a) Mt Gox Autocorrelation

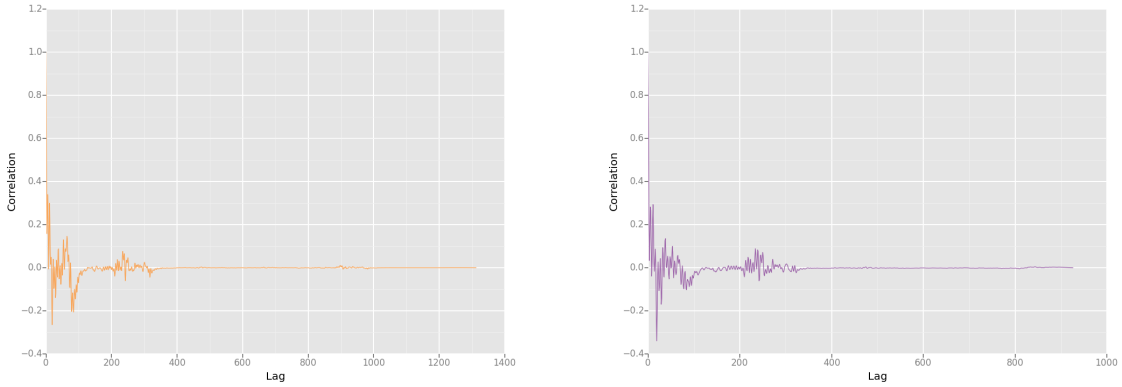
(b) BTCE Autocorrelation

Figure 4.6: Autocorrelations of the original time series

Again, the results show no indication of a periodicity in the data. Increasing the shift makes the autocorrelation quickly drop to about 0. There is no visible periodic pattern in either of the two autocorrelation diagrams.

Application on the 1st Derivative

Applying autocorrelation to the first derivative of the time series leads to the following results:



(a) Mt Gox Autocorrelation 1st Derivative (b) BTCE Autocorrelation 1st Derivative

Figure 4.7: Autocorrelations of the derivatives

Both autocorrelation plots show periodicity although not very intense. Sinusoidal patterns are recognizable which indicate repetitive patterns. Like the Fourier transform results this also indicates that the development of both series underlies periodic processes.

4.3 Correlations

In addition to the autocorrelation it is of interest to examine the correlation of both markets with each other and the correlation of the markets with other economic quantities. After the correlation between the BTCE and Mt Gox market, the correlation between the markets and the Dow Jones, the NASDAQ and the Google Trend for the word 'Bitcoin' is presented.

The correlation is a common tool in finance. It is used to understand whether different financial time series develop similarly. Correlation between stocks is used to develop risk-free assets. If two stocks are correlated, this implies that winnings will lead to higher

winnings and loss to worse losses. This is considered as risky. Two stocks with negative correlation minimize the risk because one stock will cover the loss of the other stock.

For calculation of the correlation the Pearson Correlation is used. The correlation of two time series X and Y is defined as

$$r_{x,y} = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}$$

where σ is the standard deviation. This is calculated for the time series with

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n})(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x_i and y_i correspond to the the i-th element of the time series and $\bar{x}$ and $\bar{y}$ are the means of both time series.

The correlation of the time series is not only measured starting from the same day. The shorter of the two time series, which in this case is the one that started later, is shifted along the longer time series. This reveals correlations between time series which do not occur instantaneously but after a certain time period. Such shifted correlations can, but need not show a causality between the development of the two time series. Eventually such correlations can be used for forecasting.

4.3.1 Correlation between BTCE and Mt Gox

The correlation between the BTCE market and the Mt Gox market should naturally be very high. This becomes very clear with a look at the similar time series. Since the BTCE market opened later than the Mt Gox market, which is hence longer, the BTCE time series is shifted along the Mt Gox time series and the correlation between the two markets is calculated.

Figure 4.8 reveals that naturally the correlation is highest when the two time series are aligned in time with a correlation value of 0.987. This means that the development of the two markets for this time span is alike. But it also shows one more very interesting result. When the BTCE data is aligned with the Mt Gox data of 237 days before, the correlation is as high as 0.983. This points to a periodicity of the development of the Bitcoin course over time.

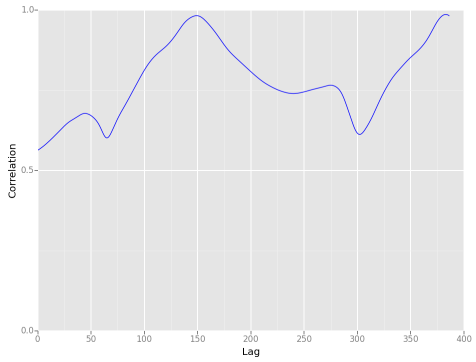


Figure 4.8: Correlation between the Bitcoin markets

4.3.2 Correlation between the Bitcoin Markets and the NASDAQ

The NASDAQ Composite index (data by `research.stlouisfed.org`) is a stock index which lists over 3.000 tech-companies in the United States. It belongs to the NASDAQ stock market which is the largest electronic stock market in the US. Amongst others it contains companies like Apple, Amazon, Microsoft, Google and so on. The NASDAQ Composite index is regarded as a good indicator for the development of other stock markets.

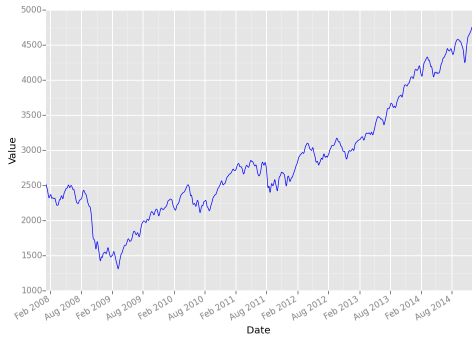
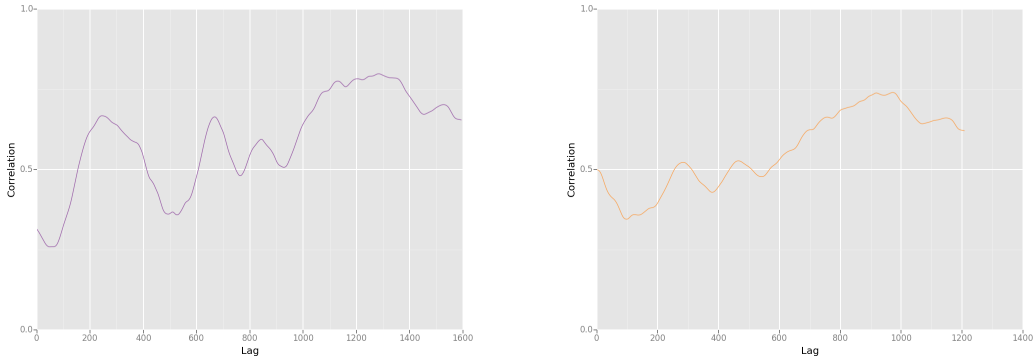


Figure 4.9: NASDAQ

It is of big interest to see whether the Bitcoin markets develop similar as the stock markets. This could imply that they are linked to each other or at least have similar underlying factors that influence their development.

Before calculating the correlation coefficient of the time series the NASDAQ Composite data has been supplemented with values. The NASDAQ Composite Data has no data

for weekends for example. Missing days in the time series have been supplemented by interpolating between the last day before and the first day after the gaps.



(a) BTCE vs NASDAQ

(b) Mt Gox vs NASDAQ

Figure 4.10: Correlation between the Bitcoin markets and the NASDAQ

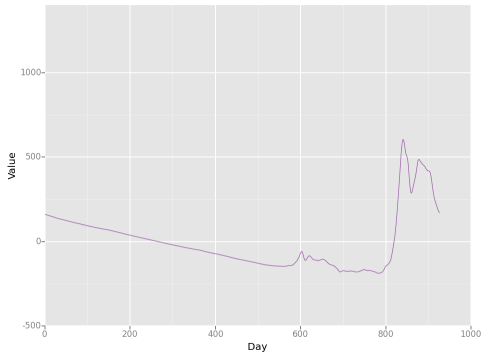
Both correlation diagrams show a clear correlation between the markets and the NASDAQ Composite index. For the BTCE market the correlation reaches a very strong value of 0.8. This value is achieved when measuring the correlation of the BTCE market with the values of the NASDAQ Composite one month before the start of the BTCE market. The highest correlation between Mt Gox and the index is about 0.74 and is reached around the start date of the Mt. Gox series. Interestingly the same correlation is reached again around 40 days later. This means that in this case the Mt Gox series is strongly related to future values of the index. The lower value of the Mt Gox series, in comparison to the BTCE series, can be explained with the long period in which only Mt Gox existed and Bitcoins were not yet very popular.

Detrend

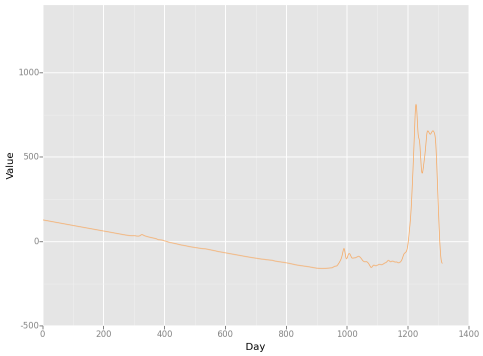
In its current form all the series contain trend information. The trend of the time series can be regarded as its long-term development. Detrending is the act of subtracting the trend from the data. In this case, this removes the constant growths that some economic quantities underlie. The long-term trend influences the correlation of the time series if it represents similar growths.

Assume that the trend of both series are linear. It can then be found by searching for the parameters a and b of a line which minimize

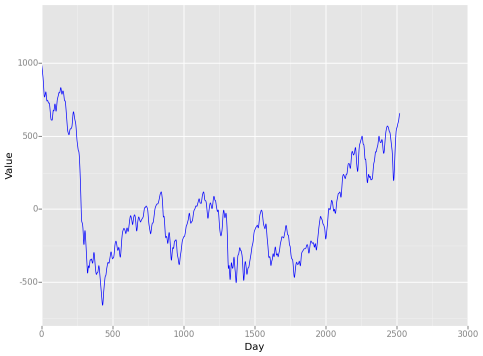
$$\sum_t [(a \cdot t + b) - y_t]^2$$



(a) BTCE - Detrend



(b) Mt Gox - Detrend

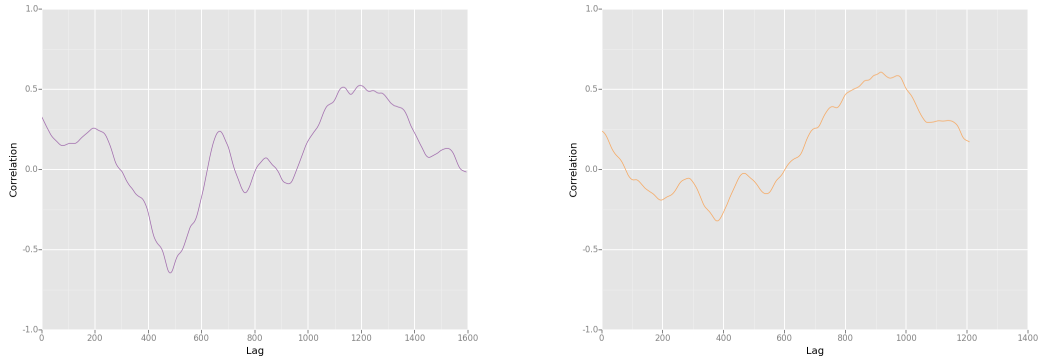


(c) NASDAQ - Detrend

Figure 4.11: Detrend - Bitcoins and NASDAQ

where y_t is the value of the time series at time point t.

After subtracting the trend it is possible to compute the correlations between the detrended Bitcoin markets and the detrended NASDAQ composite index.



(a) BTCE vs NASDAQ - Detrend

(b) Mt Gox vs NASDAQ - Detrend

Figure 4.12: Detrend - Correlation between the Bitcoin markets and the NASDAQ

Detrending leads to very different results for the correlations as can be seen in Figure 4.11 and Figure 4.12. Both markets show similar correlation graphs but the peak values differ significantly. For Mt Gox the strongest correlation of 0.6 is reached when both time series are about time-aligned with a -10 shift on the NASDAQ index. For BTCE there is also a relevant correlation of about 0.45 when both time series are aligned, but the strongest correlation is a negative correlation of -0.64 when the BTCE data is aligned with the NASDAQ index of 2 years ago. The strong negative correlation can also be seen in the Mt Gox graph and points at cyclic components.

4.3.3 Correlation between the Bitcoin Markets and the Dow Jones

In contrast to the NASDAQ Composite the Dow Jones Industrial Average index (data by `research.stlouisfed.org`) only covers 30 US companies which are not necessarily tech-companies. Similar to the NASDAQ, the Dow Jones is a good indicator for the development of stock markets. The companies in the index are chosen by the publishers of the Wall Street Journal. It contains companies such as IBM, Disney, Coca Cola etc. The plots in Figure 4.14 show the correlation of the Bitcoin markets and the Dow Jones.

The correlation for BTCE is higher than for Mt Gox. The correlation of BTCE and Dow Jones with a 4-month time difference is at a very high level of 0.81 while it is still about 0.7 for the the time-aligned correlation. For Mt Gox the high of 0.68 is reached at a 8-month shifted version but the time-aligned correlation is at a similarly high value of 0.66.

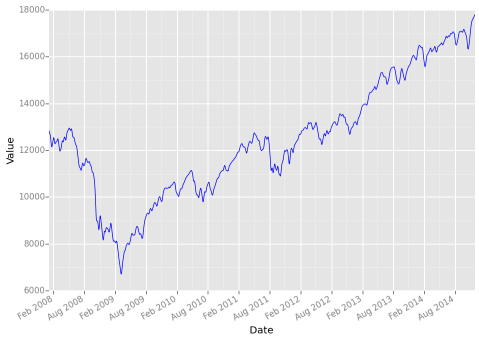
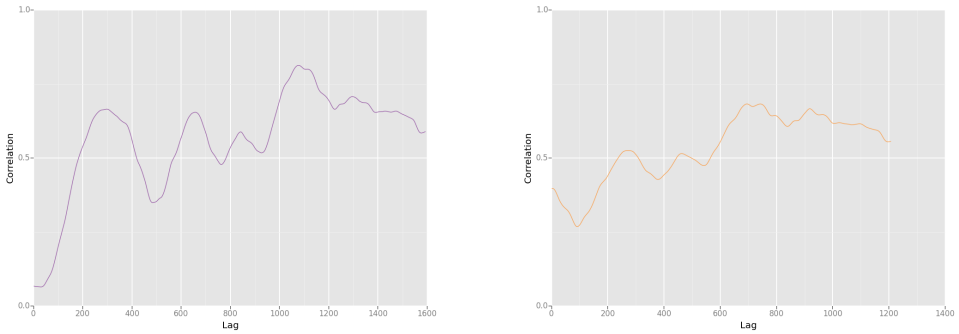


Figure 4.13: Dow Jones



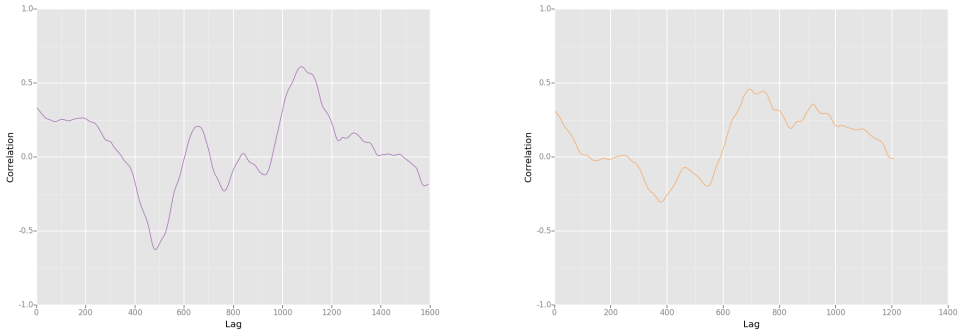
(a) BTCE vs DowJones

(b) Mt Gox vs DowJones

Figure 4.14: Correlation between the Bitcoin markets and the DowJones

Detrend

After detrending, the time-aligned correlations for both markets are very low with 0.34 for Mt Gox and even lower for BTCE with 0.12. This indicates that the markets are nearly uncorrelated to the Dow Jones index. Though it is of interest that again a cyclic component can be found. Both markets have strong negative correlations and then strong positive correlations to past intervals of the Dow Jones index. A possible explanation for the lower correlation might be that the Dow Jones index does not only consist of tech-companies where as the NASDAQ Composite index is a composite of tech-company stocks. The development of the tech companies might influence the Bitcoin course more since it is more directly connected to them.



(a) BTCE vs DowJones - Detrend (b) Mt Gox vs DowJones - Detrend

Figure 4.15: Detrend - Correlation between the Bitcoin markets and the DowJones

4.3.4 Correlation between the Bitcoin Markets and Google Trends

The last correlation that is measured is the correlation between the Bitcoin markets and the weekly Google Trends values for the search term "Bitcoin". A strong correlation could indicate that the value of the Bitcoins is strongly dependent on the interest of the people in Bitcoins but it could also mean that the interest of the people changes according to the value of Bitcoins.

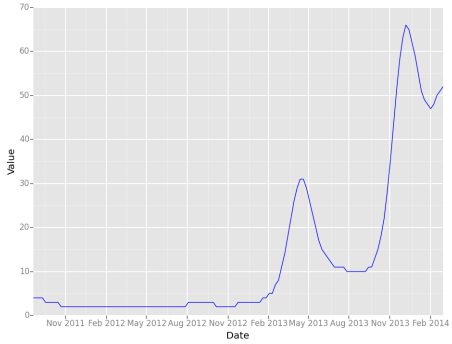
Since the weekly data of the Bitcoin markets and the Google trends data is time-aligned, only the time-aligned correlation value is available. The time-aligned time series show very large correlations. The correlation of the BTCE and Mt Gox market with the Google Trend data is about **0.95** for both markets. This means that the exchange course of the Bitcoins and the Google search for Bitcoins is almost perfectly correlated. This clearly shows that there is a strong connection between the interest of people in Bitcoins and the value of Bitcoins.

4.4 ARMA

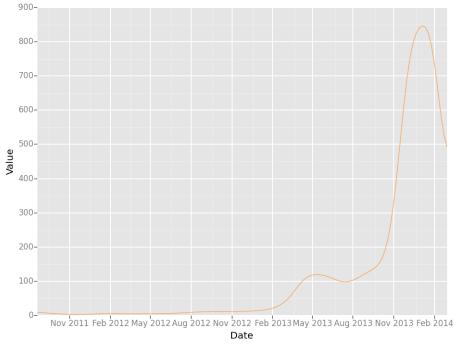
ARMA models [20] are a common tool in time series Analysis and used for forecasting. They are mixtures of autoregressive and moving average models.

4.4.1 Theoretical Background

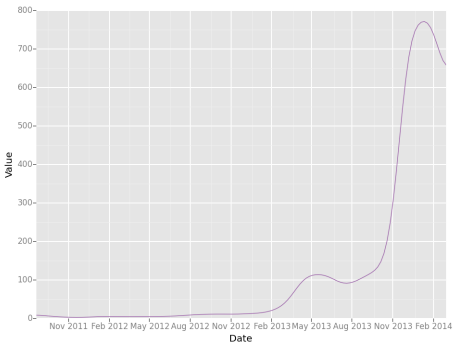
This Section gives a short introduction to ARMA Models. To introduce the ARMA model it is of help to introduce some theory of time series analysis. A discrete-time time series



(a) Google Trends for "Bitcoin"



(b) Mt Gox - Weekly Course



(c) BTCE - Weekly Course

Figure 4.16: Weekly Market Courses and Google Trends search for 'Bitcoin'

can be defined as a sequence of random variables $\{x_t\}_{t=-\infty}^{\infty}$ on a probability space.

Stationarity and White Noise

The term **strict stationarity** describes the case, that the distribution of the random variables $(x_{t_1}, x_{t_2}, ..., x_{t_k})$ does not change compared to $(x_{t_1+\tau}, x_{t_2+\tau}, ..., x_{t_k+\tau})$ for any time points t_i or any integer τ.

Weak stationarity is given when a constant $E[x_t] = \mu$ exists for all t, the variance $var[x_t] = \sigma_x^2 < \infty$ and the autocovariance $E[(x_t - \mu)(x_{t-r} - \mu)] = c_r$ depends only on r.

Due to its definition via mean and variance, in a Gaussian time series weak stationarity and strong stationarity is the same. The assumption that $E[x_t] = 0$ yields $c_r = E[x_t x_{t-r}]$ and an autocorrelation of $\rho_r = E[x_t x_{t-r}]/\sqrt{var(x_t)var(x_{t-r})} = c_r/c_0$ for lag r where c_r is estimated by $\hat{c_r} = \frac{1}{n}\sum_{t=r+1}^{n} x_t x_{t-r}$ and thus $\hat{\rho}_r$ is estimated by $\hat{\rho}_r = \hat{c}_r/\hat{c}_0$.

This leads to a definition of **white noise**. A process ϵ_t that is weakly stationary and contains only uncorrelated ϵ_t is called **white noise**.

Autoregressive Processes

The first component of ARMA models are autoregressive processes. Let $a_1, a_2, ..., a_p$ be constants and ϵ_t zero-mean white noise. If for the process x_t, ϵ_t is uncorrelated to the $x_{t-1}, x_{t-2}, ...$ of the process and

$$x_t = \sum_{k-1}^{p} a_k x_{t-k} + \epsilon_t,$$

then x_t is an autoregressive of order p. The existence of an autoregressive of order p is equivalent to all roots of the polynomial $P(z) = 1 - a_1 z \cdots - a_p z^p$ being values outside of the unit circle in the complex plane. Then the autoregressive is also weakly stationary.

Moving Average Model

The second component of the ARMA models are Moving Average Models. x_t is a moving average of order q if there are constants $b_1, b_2, ..., b_q$ and $b_0 = 1$ for which

$$x_t = \sum_{k=0}^{q} b_k \epsilon_{t-k}$$

holds.

Autoregressive Moving Average Models

Put together the models yield an **autoregressive moving average process** of order p, q, also called ARMA process. An ARMA(p,q) process exists if there are constants $a_1, a_2, ..., a_p$ and $b_1, b_2, ..., b_q$ and zero mean white noise ϵ_t, such that

$$x_t = \sum_{k-1}^{p} a_k x_{t-k} + \epsilon_t \sum_{k=0}^{q} b_k \epsilon_{t-k}$$

and ϵ_t is uncorrelated to $x_{t-1}, x_{t-2}, ...$ of the process.

Autoregressive integrated moving average (ARIMA) models are a generalization of ARMA models which utilize the derivations of time series, if the original time series is non-stationary.

4.4.2 Application and Evaluation

In this subsection the application of the ARIMA model on the Bitcoin exchange rate time series is discussed. The model has been applied to the original time series as well as to the first and second derivative and the detrended version of the data.

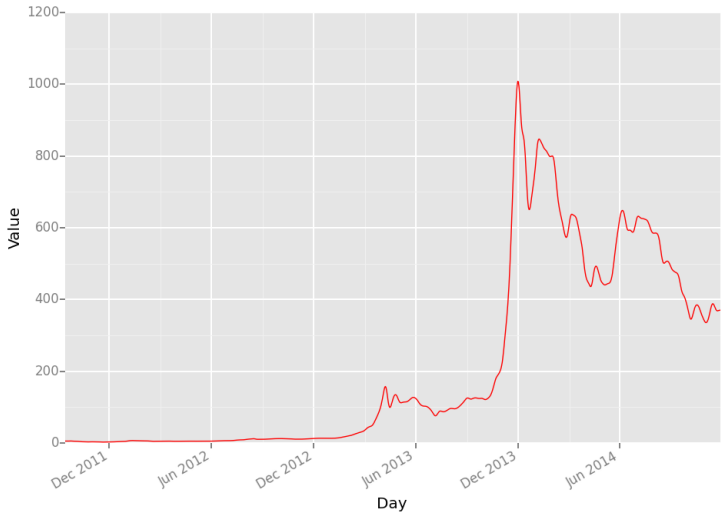


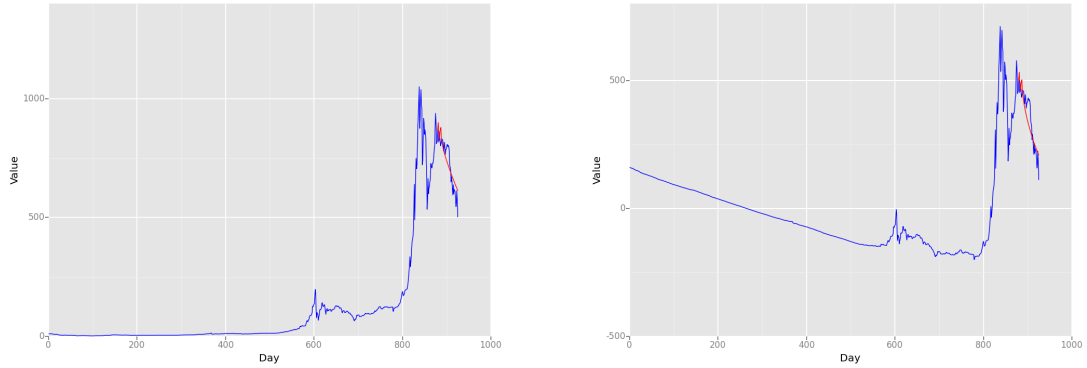
Figure 4.17: Smoothed Bitstamp time series

To evaluate the ARIMA model the p and q parameter space has been tested for values from 0 to 19. The ARIMA model has been applied to the Mt Gox and BTCE data as

well as the Bitstamp data. The Bitstamp data includes transactions from September 13th 2011 up till November 28th 2014 and is hence larger than the Mt Gox and BTCe data sets. It is included in this evaluation since the data from March till November 2014 is slightly less volatile than the months before.

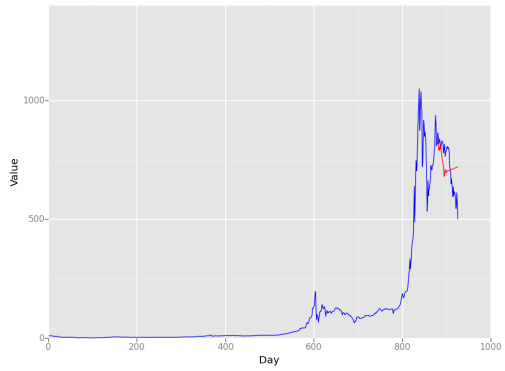
BTCE and Mt Gox

The application of the ARIMA model on the Mt Gox and BTCE time series is evaluated by letting the model learn on the first 95% of the data and then predict the last 5%. Such a large training set is chosen because the amount of data is very small and approximately the last 30% of the time series deviate significantly from the time periods before.

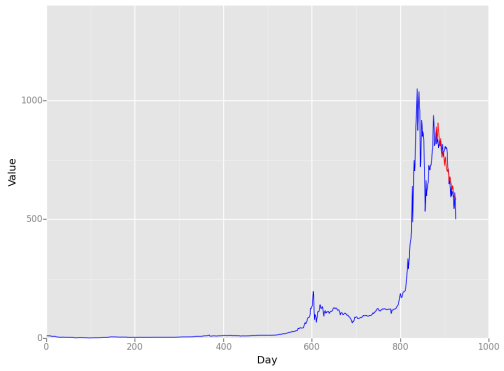


(a) BTCE Original - error: 0.14

(b) BTCE Detrend - error: 0.14



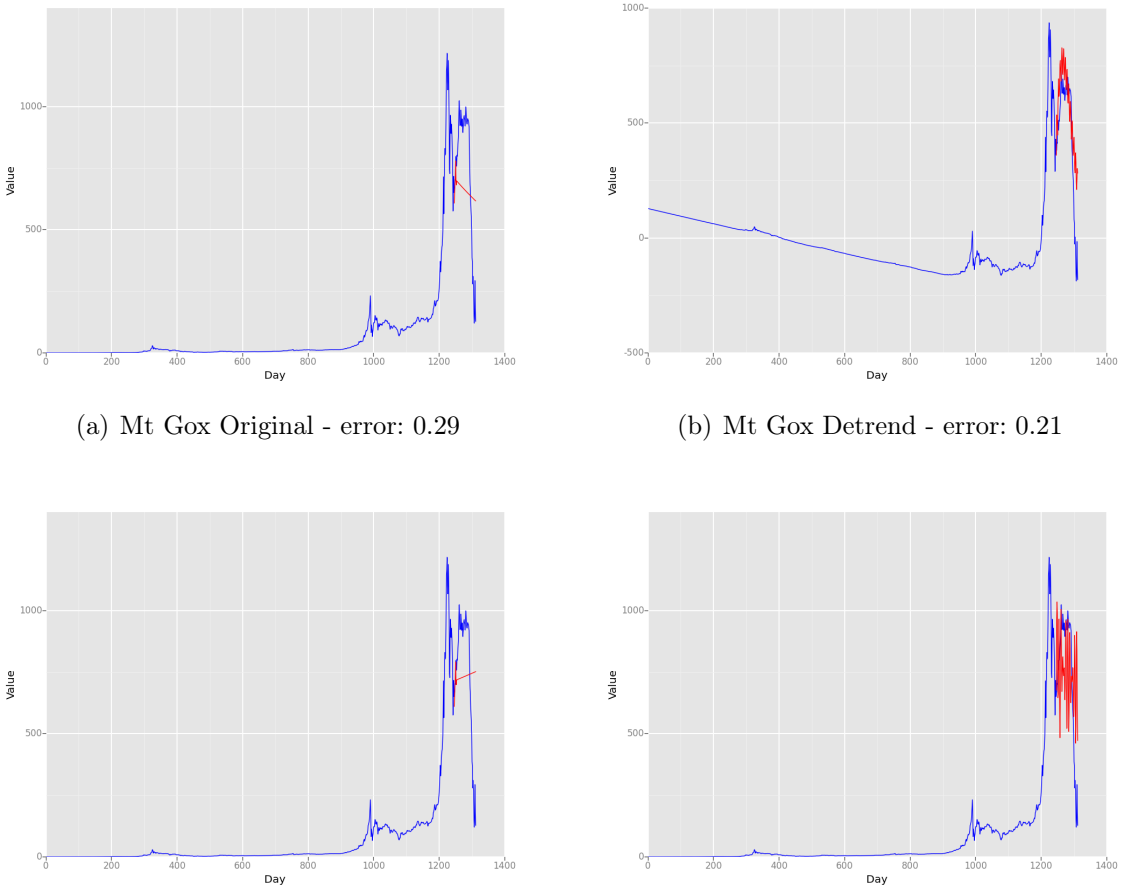
(c) BTCE 1st Derivative - error: 0.25



(d) BTCE 2nd Derivative - error: 0.11

Figure 4.18: ARIMA Models for BTCE - red: Prediction

A look at the results in Figure 4.18 and Figure 4.19 for either the plain data, the derivatives or the detrended data shows that the ARIMA model does not fit the time series very well



(a) Mt Gox Original - error: 0.29

(b) Mt Gox Detrend - error: 0.21

(c) Mt Gox 1st Derivative - error: 0.30

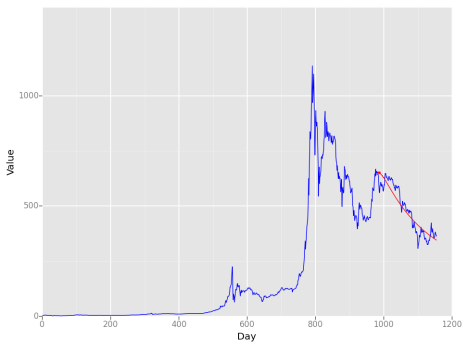
(d) Mt Gox 2nd Derivative - error: 0.30

Figure 4.19: ARIMA Models for MTGox - red: Prediction

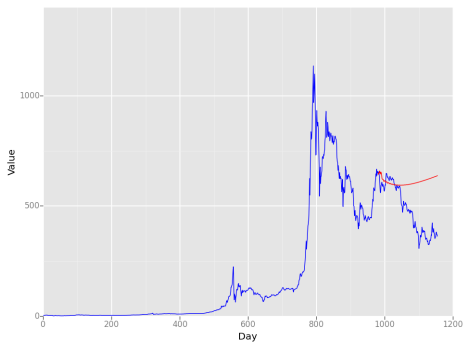
regarding short term developments. Either the volatility of the time series is not modeled or the values are clearly off. The best results considering the mean squared error show a convergence to the mean of the data almost ignoring any volatility. While this describes the long term development quite well it is not suited for describing short term developments of the time series. Yet the ARIMA model works a lot better for the BTCE time series. This is because this time series shows more stationarity. Recall that Mt Gox had to handle some exchange rate shocks due to theft.

Bitstamp

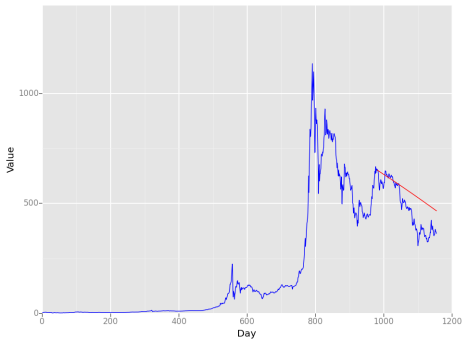
Because the Bitstamp data is significantly longer, here 85% of the data is used as training data and 15% for the actual prediction.



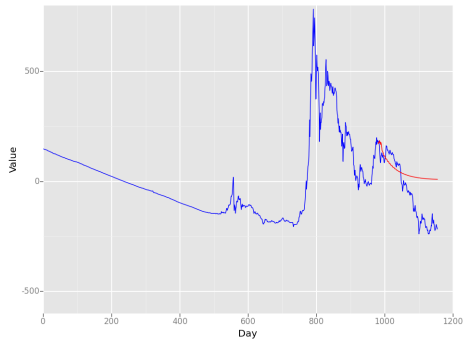
(a) Bistamp Original - error: 0.11



(b) Bitstamp 1st Derivative - error: 0.48



(c) BTCE 2nd Derivative - error: 0.28



(d) Bitstamp Detrend - error: 0.32

Figure 4.20: ARIMA Models for Bitstamp - red: Prediction

The ARIMA Model shows almost the same performance as on the Mt Gox and BTCE data. Again either the volatility or the actual long-term development is described but not

	Mt Gox	BTCE	Bitstamp
Original	0.29 (p=1,q=8)	0.14 (p=2,q=12)	**0.11 (p=14,q=0)**
1st Derivative	0.30 (p=0,q=8)	0.25 (p=2,q=17)	0.49 (p=7,q=18)
2nd Derivative	0.30 (p=14,q=3)	**0.11** (p=11,q=9)	0.28 (p=2,q=0)
Detrend	**0.21** (p=9,q=4)	0.14 (p=2,q=12)	0.32 (p=6, q=17)

Figure 4.21: Lowest NRMSE for each data set when comparing the predicted values with the actual time series

both at once. It reinforces the assumption that the ARIMA Model tempts to converge toward the mean.

Discussion

The search space shows a lot of sparsity because many of the $p - q$-combinations result in no stationarity and hence no ARIMA models. The $p - q$-combinations which produce ARIMA models either fail to model the volatility or to estimate the development of the values. The ARIMA models do not seem to be suitable to model the Bitcoin exchange rate time series properly. The relatively low NRMSE are a product of ARIMA estimating future values by converging to the mean of the data.

Chapter 5

Extracting/Predicting Prototypical Trends – Exchange Rate Segmentation

In this chapter a method for the segmentation of time series data is presented. The goal is to extract prototypical segments of the time series that can be used to describe the whole time series. This is to be regarded as a reduction of the whole time series data to substantially fewer prototypes, followed by an assignment of the time series data to those prototypes.

At first the preprocessing of the data is described. Various transformations of the data are compared and applied. The next step is the clustering of the extracted segments and the acquirement of the different prototypical segments. In the last step the segments of the time series are assigned to their corresponding prototypes.

5.1 Preprocessing

The preprocessing of the data is a very important part of the procedure since it drastically changes the outcome of the prototype finding and the segmentation process. It is the goal to find prototypical trends of the time series that can be used to describe the whole data. Those trends should not cover the time span of the whole time series, but short intervals. First the extraction of the intervals has to be done.

5.1.1 Interval-Extraction

Recall that the data that is worked with represents the mean exchange rate of the markets per day. This means that it is necessary to first extract the intervals. Given a time series T of length n consisting of a sequence of n observations $T = (t_1, t_2, ..., t_n)$ two different

sequences of intervals are extracted. The intervals will all be of the same length $k << n$. One interval thus consists of k consecutive observations. The first interval for example, contains the first k observations: $I_1 = (t_1, t_2, ..., t_k)$.

The first sequence of intervals I_{Ov} contains overlapping observations. It contains $n - k$ interval sequences. Consecutive intervals are all overlapping in $k - 1$ observations.

$$I_{Ov} = (I_1, I_2, ..., I_{n-k+1})$$

$$\begin{aligned} I_1 &= (t_1, t_2, ..., t_k) \\ I_2 &= (t_2, t_3, ..., t_{k+1}) \\ &\vdots \\ I_{n-k} &= (t_{n-k+1}, t_{n-k+2}, ..., t_n) \end{aligned}$$

This sequence contains all subsequent overlapping intervals in the time series. It can be obtained by a sliding window of size k over the whole time series.

The second sequence I_{Dj} contains disjoint intervals. It partitions T into n/k segments. If $n \ mod \ k \neq 0$ the last observations are omitted. The sequence is built up as follows

$$I_{Dj} = (I_1, I_2, ..., I_{n/k})$$

$$\begin{aligned} I_1 &= (t_1, t_2, ..., t_k) \\ I_2 &= (t_{k+1}, t_{k+2}, ..., t_{k+k}) \\ &\vdots \\ I_{n/k} &= (t_{m+1}, t_{m+2}, ..., t_{m+k}) \end{aligned}$$

$$m = \begin{cases} n - k & if n \ mod \ k = 0 \\ n - k - n \ mod \ k & otherwise \end{cases}$$

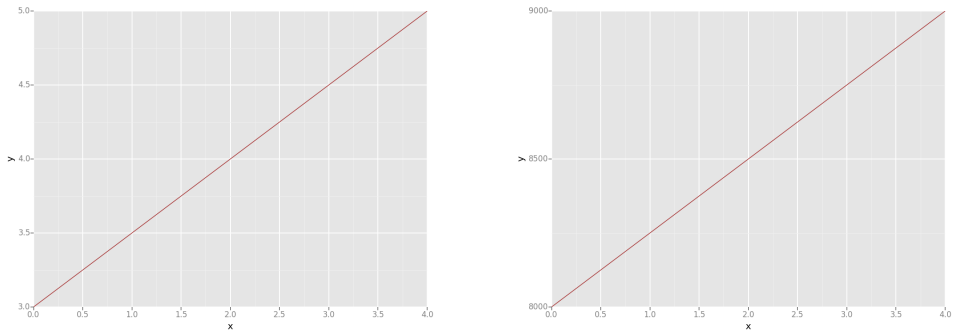
This sequence splits the time series into uninterrupted disjoint segments.

The sequence I_{Ov} will be of importance for the finding of prototypes, while the sequence I_{Dj} plays a bigger role in the prediction of the trend development.

5.1.2 Transformation of the Intervals

As of now the Intervals contain the plain unmodified data. This means that an interval contains the exact daily exchange rates for the time-segment the interval corresponds to.

Here are two example intervals that show similar course developments but very different exchange rates:



(a) Rising Low Values (b) Rising High Values

Figure 5.1: Two Rising intervals with very different scale

When trying to extract prototypical segments, or trends, this is problematic. Both intervals clearly show the same rising trend but on a very different scale.

To make the problem obvious the intervals can be defined in a different way. Instead of as intervals, they can be regarded as multidimensional points. An interval with k elements is a point $p_i \in \mathbb{R}^k$. Every observation t_j in that interval corresponds to the coordinate in the $j-th$ dimension with $j \in \{1, 2, ...k\}$.

In this space both intervals that describe a similar development are far apart when the given plain data is used. If clustering would be applied now to find prototypical intervals that summarize both as 'uprising' trends, the algorithm would almost certainly not cluster those intervals together. Especially if a distant metric is used as similarity such as in K-Means. Even worse than not clustering both rising intervals in the same cluster is that the sinking interval in figure 5.2 would almost certainly be clustered together with the interval in Figure 5.1(b). Both intervals would represent relatively close points in the k-dimensional space (Figure 5.3). This means that two opposing trends would be regarded as similar and fails the purpose.

Thus, two transformations will be used to transform the intervals in a way that they emphasize the development of the data, no matter how large the exchange rate values are.

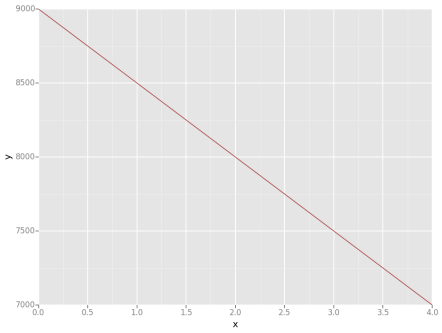


Figure 5.2: A sinking interval with high values

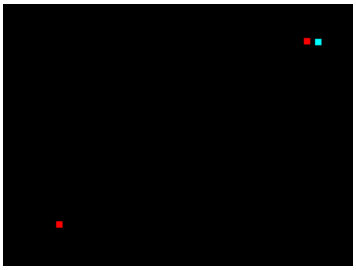


Figure 5.3: The rising (red) and sinking (blue) trends can not be differentiated in the k-dimensional space

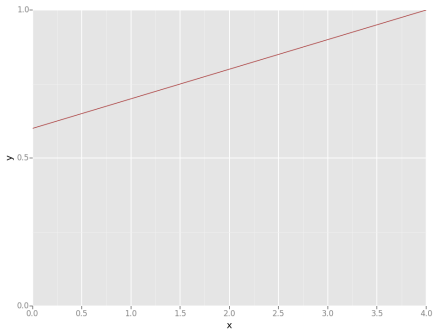
Division By Maximum

The first transformation aims at adjusting the intervals in a way that they are all in similar value ranges. The transformation is defined as follows

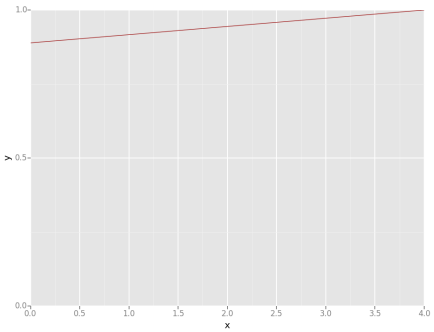
$$T_{divmax}(I) = \frac{I}{I_{max}} \tag{5.1}$$

where I is the interval that is to be transformed and I_{max} is the largest value in this interval.

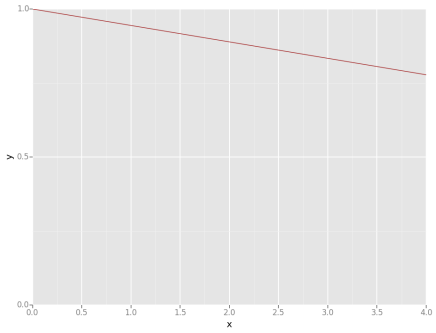
Figure 5.4 shows that the transformation has the desired effect of bringing the rising intervals closer together spatially and clearly separating them from the sinking interval. This gets clearer when taking a look at figure 5.5 which visualizes the proximity of the data points in the k-dimensional space after the transformation. The points of the rising intervals are now close together while the sinking interval is far away from them. This allows a clear separation and would enable clustering them into two different clusters, while keeping the rising intervals together in one cluster.



(a) Rising Low Values - Divmax



(b) Rising High Values - Divmax



(c) Sinking Values - Divmax

Figure 5.4: Two rising and a sinking interval, transformed with divmax

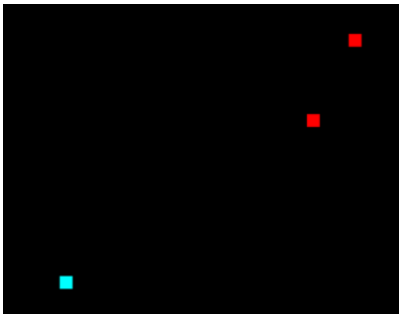
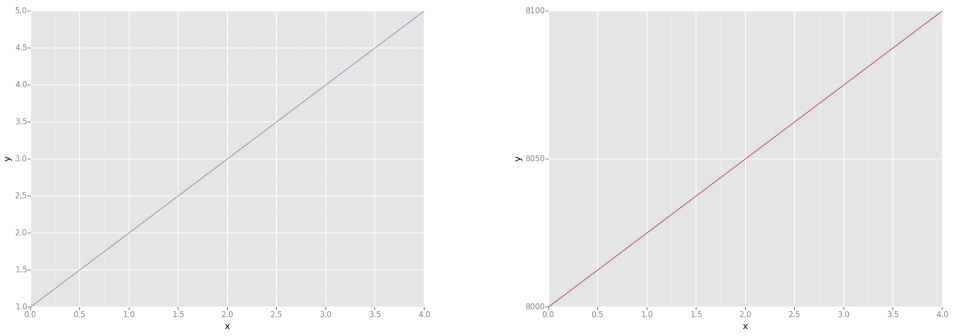


Figure 5.5: After the transformation, the rising (red) and sinking (blue) trends are separated

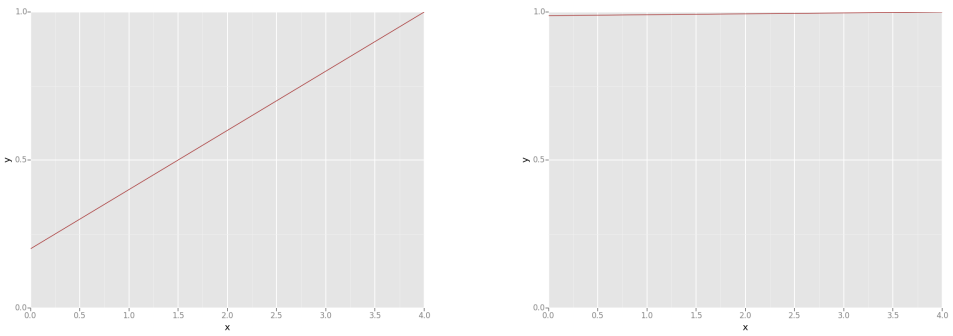
A problem that arises with this approach is easy to understand with the example in figure 5.6. Both intervals represent a rising trend, the absolute value in the second interval rises



(a) Rising Low Values (b) Rising High Values

Figure 5.6: Two rising intervals, second rising by larger margin

drastically by about 100, while the absolute value in the first interval rises by only 4. If the T_{divmax} operator is applied now, it yields the following intervals:



(a) Rising Low Values - Divmax (b) Rising High Values - Divmax

Figure 5.7: Both rising intervals, transformed with divmax

The uprising trend is clearly recognizable for the first interval, but the effect has vanished for the second interval and it seems like the value stays almost at the same level. Considering absolute values this is false but relatively it is correct. This allows the statement that the T_{divmax} highlights changes relative to the current exchange rate value.

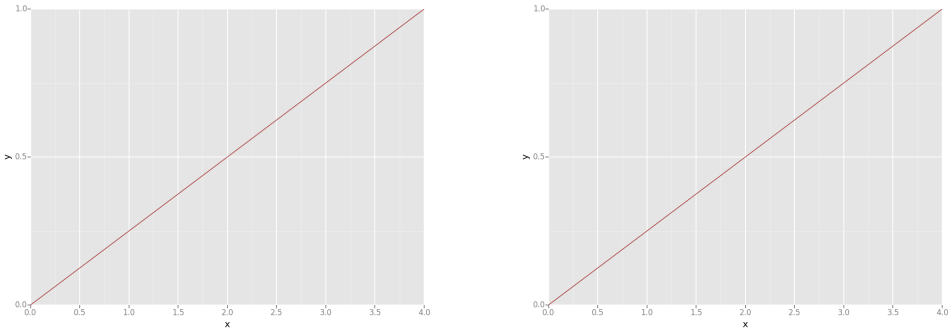
Normalization between 0 and 1

The second transformation aims at improving on the issue of the T_{divmax} operator. It utilizes a normalization of the values between 0 and 1 and stretches all of the intervals to this margin. It is a common tool in Image Processing and used to stretch the histogram values of an image. It is defined as

$$T_{norm}(I) = \left[\frac{t - I_{min}}{I_{max} - I_{min}} \middle| t \in I \right] \tag{5.2}$$

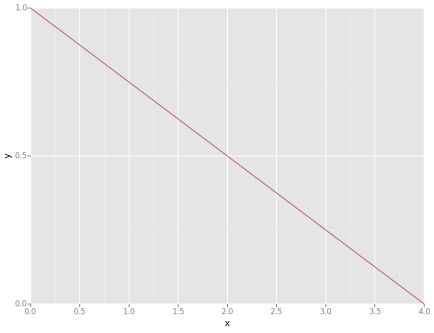
T_{norm} is very similar to T_{divmax}. It differs in a shift of every value by the minimum value in an interval. This has a big effect on the outcome, because it stretches the minimum value to be at 0 and the maximum value to be at 1.

Applying T_{norm} to the same intervals as before yields the results which can be seen in 5.8



(a) Rising Low Values - Norm

(b) Rising High Values - Norm



(c) Sinking Values - Norm

Figure 5.8: Two rising and a sinking interval, transformed with norm

Now the development of the series in the interval is clearly highlighted. This transformation emphasizes the trend in the interval and does not put it into relation with the magnitude of the exchange rate value at the time.

5.2 Extracting Prototypical Trends

The extracted intervals in all three versions, the plain data and both transformed versions, are now used to find prototypical trends. Such prototypical trends summarize intervals that show a similar development of the time series. Examples could be a strong rising trend, a strong falling trend or a non-changing trend.

To find prototypical trends clustering can be used. The assumption is, that the intervals, which describe similar trends, share similarities. Such similarities could for example be expressed in spatial proximity. As discussed in Subsection 5.1 the data, which is transformed by T_{divmax} and T_{norm}, should be especially suited to find trends of the time series development. The performance of various clustering algorithms will be evaluated, considering finding prototypes, segmenting the time series with them and predicting the development of the time series.

K-Means

K-means, as described in Section 3.1.1, utilizes the euclidean distance of observations as criterion for the search for clusters. It aims at finding clusters such that the distance of each observation, in this case the intervals, to the cluster centers is minimized.

It is important to recall that K-Means will not choose given intervals as cluster centers. The cluster centers, which in this application will be the prototypes of the intervals, are centroids of the intervals which belong to them. This means that they represent the mean of all intervals which are assigned to them. The number of prototypes to be found can be adjusted at will.

The means which are found this way will represent the prototypical trends of the time series. The number of clusters will significantly change the type of trends which are found. While only few clusters could result in very vague trends like 'rising course' or 'falling course', a higher number of clusters will imply finer trends.

Affinity Propagation

Affinity Propagation, described in Section 3.1.2, can be used with various similarity measures. For this application the euclidean distance is chosen as measure, as Affinity Prop-

agation still offers crucial differences to K-Means. The first big difference is that Affinity Propagation will choose actual observations as prototypical trends.

Another crucial difference is that the number of clusters is not pre-set. Affinity propagation will choose as many exemplars as necessary to reach a stable configuration. This could imply very accurate trends.

HMM

Hidden Markov Models are useful for finding hidden processes that underlie the sequences of observations. In this case the observations are the intervals. It is important that those intervals are inputted in their actual order since the HMM does not only consider the values of the observations, but also the order of them.

A slightly modified version of the traditional HMM described in Section 3.1.3, the Gaussian HMM will be used. A Gaussian HMM assumes that the intervals are Gaussian distributed. It will try to find a sequence of hidden states and their associated observations. The observations which are associated to a state i will then be defined by a mean μ_i and a variance σ_i^2. The similarity measure is again the distance in the $\mathbb{R}^k$ space. Like in K-Means the number of prototypes to be found can be preset in the HMM.

The states which the HMM discovers and their associated μ_i will serve as the prototypical trends. While the similarity measure is similar to the measure in K-Means and Affinity Propagation, the HMM incorporates the order of the observations.

In addition to the prototypes the HMM will yield a transition probability matrix which represents the probability of transitioning from one trend to another in the time series. These probabilities can be regarded as likelihoods that describe in what way the time series will develop given the current trend it is in.

DESICOM-HMM

The last clustering approach utilizes the HMM and DESICOM (Section 3.1.4). First the HMM is run to find k trends based on the observation sequence. The HMM will return the k trends and an asymmetric probability matrix M_{kk} which describes the transition probabilities from one trend to another.

These asymmetric transition probabilities can be used as input for DESICOM. In contrast to the former described clustering methods, DESICOM will search for transitive affinities in the transition matrix and reduce the number of trends k found by the HMM to $l < k$ time and space dependent trends.

Similar to the HMM, DESICOM yields an affinity matrix that describes the affinities between the trends and can also be used as transition probability matrix, that describes the

probability of a transition from one trend to another and hence the possible development of the time series.

5.3 Segmentation of the Time Series

After extracting the prototypical trends it is now the goal to describe the whole time series as a sequence of the trends. The clustering methods which were fit to the data in order to extract the prototypes can also be used to predict the associated trend for every observation. All of the models offer affinities $af(I, p)$ which assign the intervals I to their prototypical trends $p \in P$, where P is the set of all found prototypes.

K-Means uses the euclidean distance to determine the closest prototype for every interval. Affinity Propagation calculates the affinity of each observation to the given exemplars and chooses the exemplar with the largest affinity. The Gaussian HMM calculates the most likely state, and thus prototype, of the observation given the Gaussian distribution of the prototypes. And finally DESICOM by the nature of the algorithm produces a matrix A which assigns the observations to their clusters and a matrix R which shows the affinities among the clusters. In the case of DESICOM the observations are the previously found clusters with the HMM.

To segment the time series the affinity measure $af(l, p)$ are applied to each of the intervals in either the overlapping sequence I_{OV} or the disjoint sequence I_{Dj}

$$\forall I \in I_{OV} : p(I) = argmax\{af(I, p)|p \in P\} \tag{5.3}$$

$$\forall I \in I_{Dj} : p(I) = argmax\{af(I, p)|p \in P\} \tag{5.4}$$

The result will be a sequence of prototypes $S_P = (p(I_1), p(I_2), ...p(I_n))$ which describes the time series in terms of the underlying trends.

It is important to understand that this prediction does not need to be applied only to the time series from which the prototypes was extracted. In fact it can be applied to any arbitrary time series. If the prototypes are general enough they will be well suited to describe other time series too. Though the error should increase compared to the time series they were extracted from, since they are designed to fit this time series best.

To segment other time series the data has to be prepared the same way as the data in the time series from which the prototypes were extracted. This means that the intervals of same length have to be extracted and then the same transformation has to be applied to the intervals. The last step is applying the affinity measure $av(I, p)$ to the newly extracted interval sequences as shown in equation 5.3 and 5.4.

5.4 Prediction / Forecasting

Now that the time series have been segmented into sequences of trends, these sequences can be used to infer statistics about the development of the time series. The goal is to try and learn the transition behavior of the trends and then apply the learned knowledge to predict the future development of the time series.

More precisely: Given a history of trends $H = (p(I_1), p(I_2), ...p(I_{t-1}))$, which trend p_t is most likely to be next? The next subsections discuss a baseline for this prediction and 3 methods that can be applied to any of the clustering methods.

5.4.1 Baseline

The Baseline algorithm is very simple. Given a trend p_{t-1} it always assumes that at time-point t the same trend will be seen again. This means that $p_t = p_{t-1}$. Although this heuristic seems overly simple it produces some correct predictions depending on the number of prototypes. Assuming that there are only 3 prototypes, they could represent a rising, a sinking and a constant trend. For short intervals it is very likely that e.g. a rising trend is succeeded by yet another rising trend. This heuristic does not utilize any domain knowledge and will fail at every change of trends.

5.4.2 Transition Matrix and Cluster Affinities

The HMM and DESICOM produce a transition probability matrix or respectively a matrix that shows the affinities between the prototypical trends. These matrices can be used for domain knowledge based prediction. For each observed trend the most likely following trend is picked according to the likelihoods in the matrices.

A similar transition matrix can be inferred by the sequences of prototypes produced with K-Means and Affinity Propagation. For every prototype p_i count the transitions to p_j with $j \in 1, 2, ..., k$ where k is equal to the number of extracted prototypes. The matrix is of size $k \times k$ and the entries are $p_{ij} = transition\_count(p_i, p_j)$.

If not all trends are most likely to be followed by the same trend, this should be a substantial improvement over the Baseline algorithm. The actual relationship among the clusters is considered and used to predict the development of the time series. This method is equal to considering only the current state as relevant for the next state and suits the Markov Assumption which was introduced in Section 3.1.3.

5.4.3 High-Order Markov Chain

The High-Order Markov Chain extends the transition probabilities by considering not only the current trend, but a history of trends, to decide about the development of the time series.

As input the High-Order Markov Chains takes a sequence of trends from which it should learn. It then counts the occurrences of subsequences of trends. The length l of those subsequences can be adjusted to best suit the application domain. By counting the occurrences of those subsequences of lengths up to l, a frequency distribution can be generated. While larger values of l can be useful, they also bring the problem of overfitting the training data. Longer chains of prototypes could be coincidence or specific to the data on which the Markov Chain was trained.

5.4.4 Language Model

The last method for prediction is slightly more sophisticated to apply, but very similar to the Markov Chain Model. Again it is the aim to learn transition probabilities considering larger histories of trends. N-Gram-models as described in Section 3.2.2 learn probabilities of sentences or word sequences. **N**-Gram models utilize sequences of lengths of up to n words to calculate the probabilities.

A sequence of k trends with labels $L = 0, 1, 2, ...k$ can be considered as sequence of words describing the trends. Recall that an N-Gram model uses counts for sequences of up to length n and it computes the probability of such sequences. It is important to note that usually N-Gram Models also use a start symbol for the beginning of every sentence and an end symbol for their endings. This yields counts, that display the probability of a word starting or ending a sentence. What is of interest now, is finding the most likely trend that follows $n-1$ previous trends. Or in language model terms: Which word is most probable to end a sequence of words of length n given the $n-1$ previous words.

In order to train this model the sequence is split into every possible subsequence of length n. These subsequences represent the sentences in the training data. Kneser-Ney smoothing is used so that the model does not backup to the most common trend in every case, but rather to the trends which occur in context with most other trends. These might be transitional trends which not occur very often, but in combination with many other trends.

After learning the model the perplexity measure can be used to calculate the most probable following trend. The perplexity measures the likelihood of a sentence. Lower perplexities correspond to higher probabilities. In order to predict the trend at a time point t, the $n-1$ word long sequence $S_{sub} = (t_{t-n+1}, t_{t-n+2}, ..., t_{t-1})$ is appended by all possible trends. For each of the sentences with the possible endings, the perplexity is calculated and the

sentence with the lowest perplexity is chosen. This sentence should correspond to the most likely sequence and its last element is the predicted trend.

Chapter 6

Evaluation

This chapter evaluates the system presented for extracting the prototypes and segmenting the time series based upon the prototypes. The system is evaluated with respect to the different preprocessing methods and the different clustering algorithms. For the evaluation the overlapping interval subsequence has been chosen because the disjoint interval subsequence is a subset of it. Nevertheless the disjoint subsequence is used for displaying the results of the clustering for ease of interpretation.

After the evaluation of the segmentation the prediction of future trends of the time series is evaluated. All 4 presented methods for prediction are evaluated and this time the results for the disjoint as well as the overlapping interval subsequences of trends are compared.

Implementation

The software for this thesis was completely coded in python. The pandas library was used to handle the data which was given in csv format. Scipy and Sklearn were used for the Gaussian Filter, Derivatives, Fourier Transform, K-Means, Affinity Propagation and the HMM. The DESICOM source code was provided by Prof. Dr. Christian Bauckhage and can be reviewed in [16]. The High-Order Markov Chain Model was implemented by Philipp Singer and is available online at `http://www.philippsinger.info/?p=44`. For the language model OPENGrm was used which is developed and maintained by New York University.

6.1 Prototype Extraction and Segmentation

In this Section the extraction of the prototypical trends and the segmentation of the time series into a sequence of trends is evaluated. The data was evaluated for all possible interval lengths $I_L \in \{4, 5, 7, 10, 14, 30\}$ and all possible number of prototypes

$P \in \{3, 5, 7, 10, 15, 20, 40\}$ on the '$\sigma = 3$'-smoothed time series. For the sake of simplicity and ease of interpretation for the reader, most of the figures show the case of interval length 5 and 5 prototypes.

The clustering and the obtainment of the trends has been performed on the Mt Gox data set because it is larger than the BTCE data set. To evaluate the quality of the trends for the segmentation of the time series, the mean squared error between the chosen trend for an interval and the actual interval has been calculated

$$E_{MSE}(p_{I_i}, I_i) = \frac{\sqrt{\sum_{t=1}^{k}(p_{I_i}(t) - I_i(t))^2}}{k} \tag{6.1}$$

where p_{I_i} is the prototype for the interval I_i and $p(t)$ represents the the $t - th$ value of the prototype and $I_i(t)$ is the t-th value of the interval I_i. The sum of the errors for all intervals is calculated and then divided by the number of intervals. The overall error over all intervals is called $E(I)$. To understand the error better, $E(I)$ is normalized with the range of values in the data set to obtain an error percentage. This yields

$$E^*(I) = \frac{E(I)}{I_{max} - I_{min}} \tag{6.2}$$

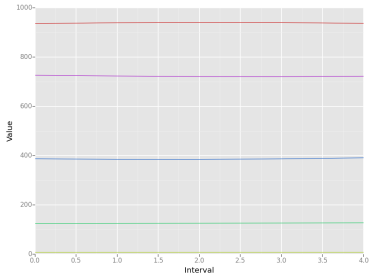
where I_{max} and I_{min} represent the maximum and minimum value in the corresponding data set.

6.1.1 Plain Data

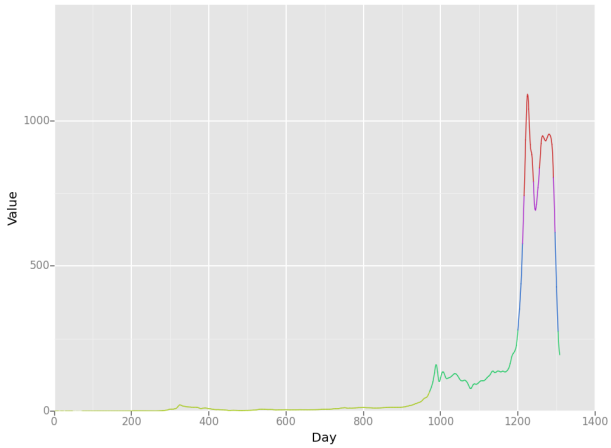
The first data set for which the prototype extraction is evaluated is the unprocessed data set, containing intervals with the plain exchange rate values for the corresponding periods.

As expected K-Means (1000 iterations), which uses the euclidean distance as similarity measure, and the HMM (1000 iterations), which groups the observations by their means in the high-dimensional space, cluster intervals of similar magnitude together (Figure 6.1 and Figure 6.2). These clusters result in a vertical segmentation of the time series. Intervals of similar height are clustered together.

DESICOM (200 iterations) is initialized with the transition matrix of an HMM with 50 clusters. It behaves slightly different than K-Means and the HMM (figure 6.5). Although it does group intervals of similar altitude together it also introduces some kind of continuation, meaning that clusters which follow each other in the HMM clustering are clustered

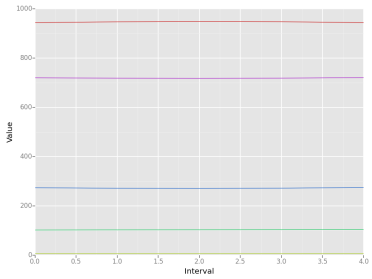


(a) K-Means Prototypes

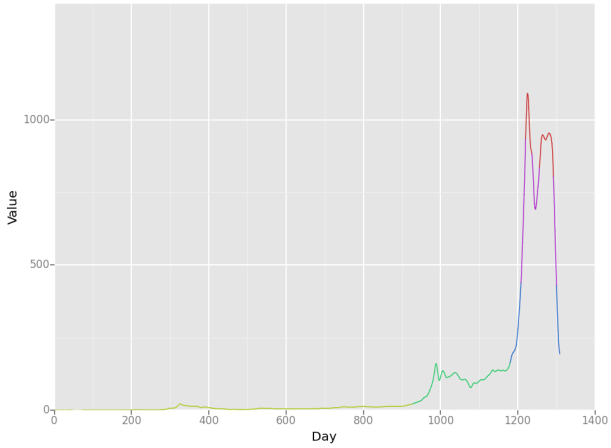


(b) K-Means Segmentation

Figure 6.1: Plain Data: Prototypes and Segmentation for K-Means - 5 Prototypes - Interval Length 5

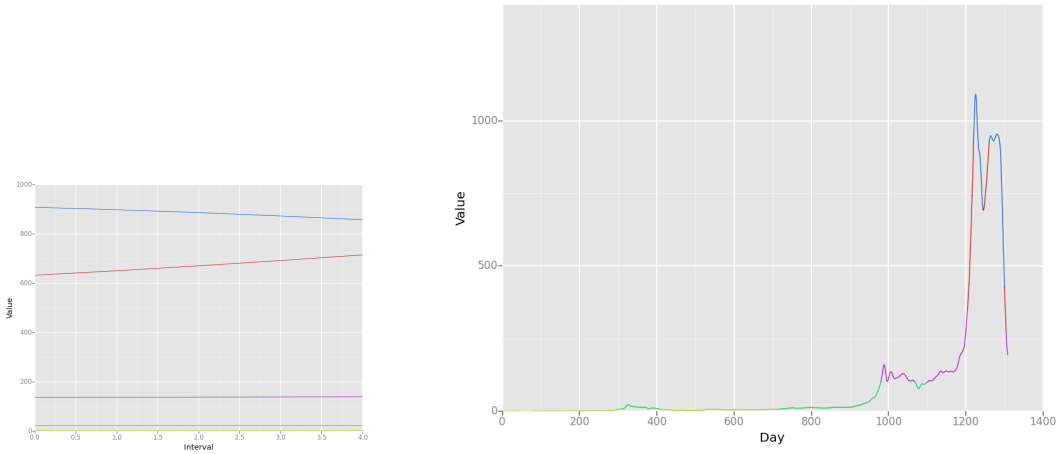


(a) HMM Prototypes



(b) HMM Segmentation

Figure 6.2: Plain Data: Prototypes and Segmentation for HMM - 5 Prototypes - Interval Length 5



(a) DESICOM Prototypes (b) DESICOM Segmentation

Figure 6.3: Plain Data: Prototypes and Segmentation for DESICOM - 5 Prototypes - Interval Length 5

together. This leads to a mixed vertical and horizontal segmentation and prototypes which represent a mixture of altitude and gradient.

Affinity Propagation does not find any clustering with less than 40 clusters and the results of this clustering approach for the plain data set have thus be omitted. The search space for Affinity Propagation contained preferences of -50 *to* 0 in increments of 5. Also the minimum and median euclidean difference of all instances was used as preference value.

None of the used clustering algorithms lead to prototypes which represent trends in the development of the time series. Clustering the plain data results in a segmentation into different levels of altitude.

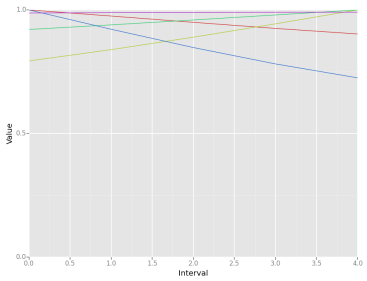
6.1.2 Division By Max

In this data set every interval I was transformed with

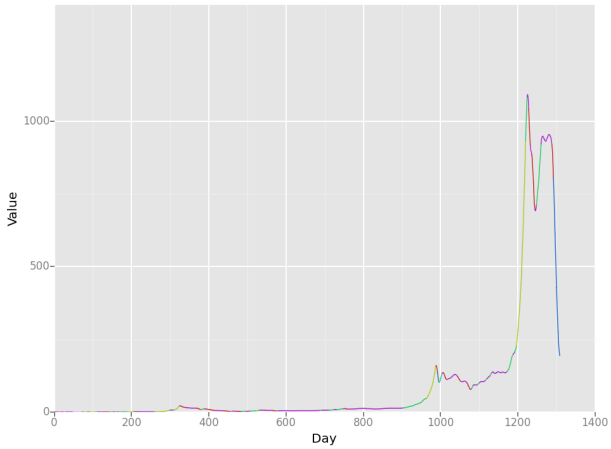
$$T_{divmax}(I) = \frac{I}{i_{max}} \tag{6.3}$$

This causes all values to be in a range from $[0, 1]$.

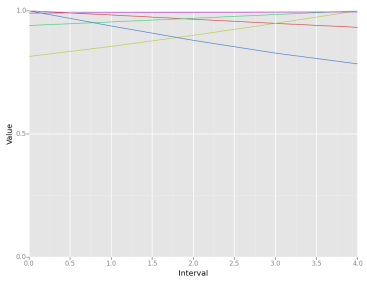
With this data set the K-Means algorithm and HMM actually produce very similar clusters which can indeed be regarded as trends. In the case of intervals of length 5 and clusters



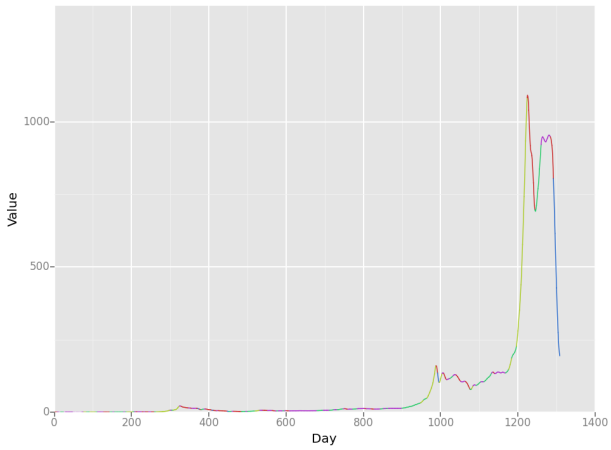
(a) K-Means Prototypes



(b) K-Means Segmentation



(c) HMM Prototypes



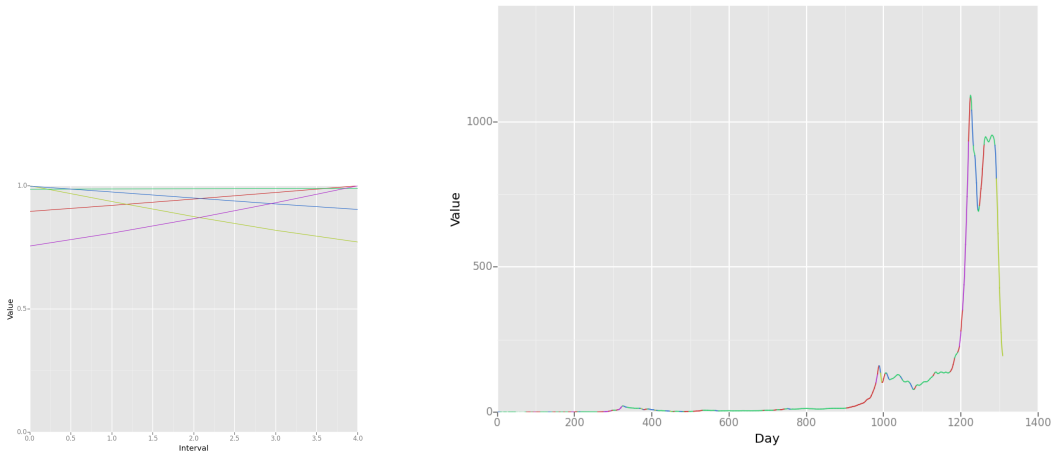
(d) HMM Segmentation

Figure 6.4: Div Data: Prototypes and Segmentation for K-Means and HMM - 5 Prototypes - Interval Length 5

of length 5 the two algorithms produce the clusters and segmentation which can be seen in Figure 6.4.

The clusters can be regarded as "strongly decreasing", "decreasing", "stable", "increasing" and "strongly increasing". The clustering actually produced meaningful trends which are useful to segment the time series into sequences of trends. Note that the trends are almost strictly linear.

When applying DESICOM, again with an initialization of 50 HMM clusters, nearly identical clusters are generated as can be seen in figure 6.5. DESICOM groups the 50 clusters into "strongly decreasing", "decreasing", "stable", "increasing" and "strongly increasing", just as K-Means and the HMM.



(a) DESICOM Prototypes (b) DESICOM Segmentation

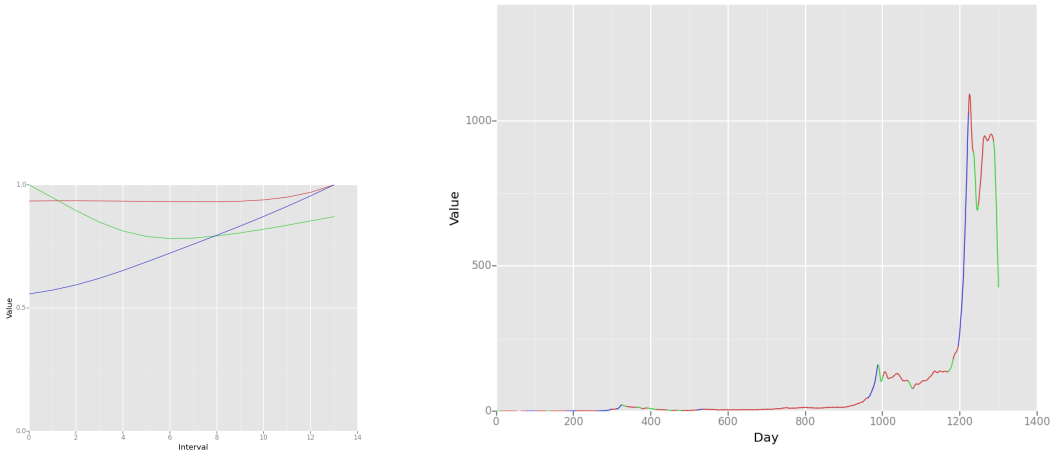
Figure 6.5: Div Data: Prototypes and Segmentation for DESICOM - 5 Prototypes - Interval Length 5

Affinity Propagation does not produce any useful results again. In fact only for intervals of length 14 one stable configuration with 3 clusters has been found which can be seen in figure 6.6.

The 3 clusters represent "relatively stable", "rising" and "relatively sinking" trends.

Problem with the DivMax Approach

Although the divmax-transformation leads to useful trends, their application to segment the time series is not perfect. As discussed in Section 5.1.2 the transformation of the data has some negative side effects. The changes which the trends show are relative to the



(a) Affinity Propagation Prototypes (b) Affinity Propagation Segmentation

Figure 6.6: Div Data: Prototypes and Segmentation for Affinity Propagation - 3 Prototypes - Interval Length 14

actual value of the time series. As visible in the segmentation of the time series in Figure 6.7 this can lead to unwanted effects.

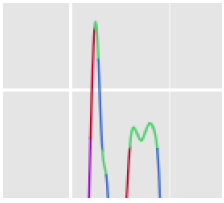


Figure 6.7: Green Areas are classified as "steady trends"

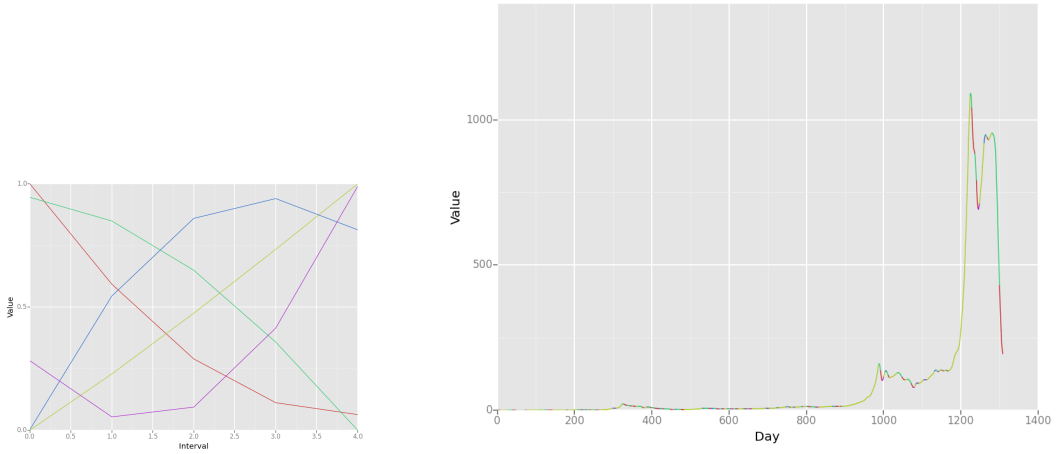
The exchange value clearly rises and sinks in this part of the time series. Because the changes are relatively low in relation to the actual value of the exchange rate, the division with the maximum value leads to almost similar values for all of the time points. Such a transformation can be useful for some use cases, but is not useful if the goal is to predict the development of the time series. Consider a person that speculates with Bitcoins. A decline of the exchange rate of e.g. 40 Dollars per Bitcoin at a course of 800 Dollars might be relatively low, but could lead to significant losses for this person. The divmax-transformation would consider this change as a "stable" course development.

6.1.3 Normalization

The normalized data set was obtained by transforming the plain intervals with

$$T_{norm}(I) = \left[\frac{t - I_{min}}{I_{max} - I_{min}} \middle| t \in I \right] \tag{6.4}$$

Now the values in all intervals are normalized to the range of $[0, 1]$ and also stretched to fully fill this range. This means that the newly obtained intervals emphasize the change of the time series in that interval, without relating it to the actual absolute value of the exchange rate at that time.



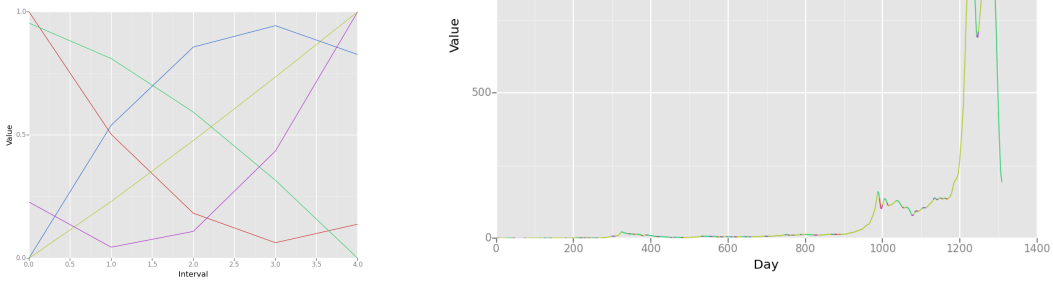
(a) K-Means Prototypes

(b) K-Means Segmentation

Figure 6.8: Norm Data: Prototypes and Segmentation for K-Means and HMM - 5 Prototypes - Interval Length 5

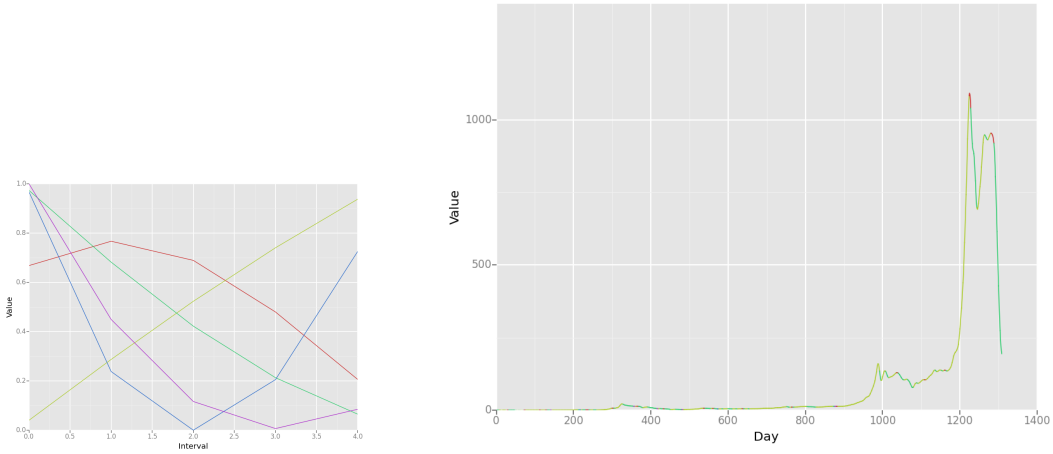
Again the HMM and K-Means produce nearly identical prototypes as can be seen in Figure 6.8 and Figure 6.9. This time the obtained trends actually describe general developments of time series. The trends differ a lot from the divmax-data set. They emphasize the increase and decrease in value. The trends in the images can be described as "increasingly rising", "constant rising", "decreasingly rising" , "increasingly sinking" and "decreasingly sinking". While there is no case of a steady trend, such trends can be observed for other interval lengths or trend numbers. They have the form of first rising and then falling or first falling and then rising trends. With the normalization an almost completely equal trend is only achievable if most of the values in the interval are the same. This is rarely the case and thus prevents the occurrence of such a prototype.

The DESICOM algorithm produces different clusters. While there are 2 different "sinking" trends, there are 2 trends that are "rising and falling"/"falling and rising" and only one "rising" trend. Since the "rising" trends are often followed by one another it groups them together. The overall segmentation is good.



(a) HMM Prototypes (b) HMM Segmentation

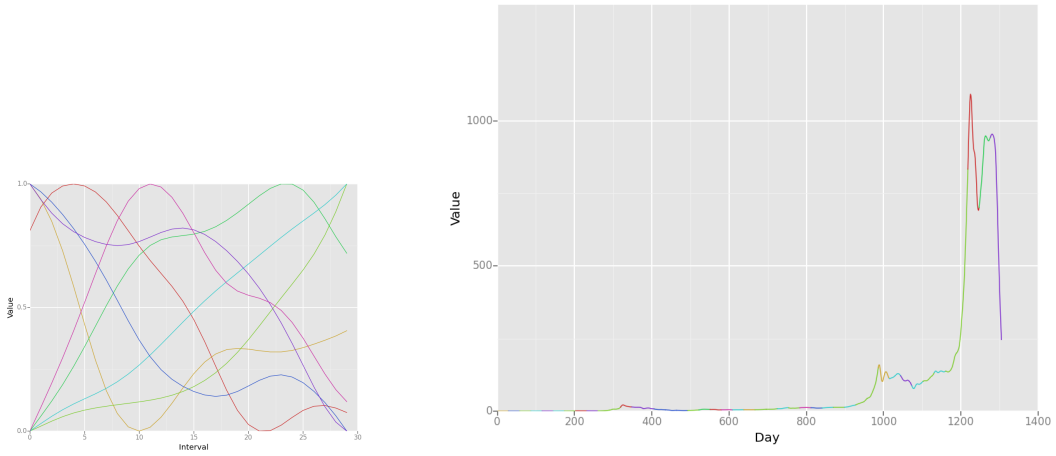
Figure 6.9: Norm Data: Prototypes and Segmentation for K-Means and HMM - 5 Prototypes - Interval Length 5



(a) DESICOM Prototypes (b) DESICOM Segmentation

Figure 6.10: Norm Data: Prototypes and Segmentation for DESICOM - 5 Prototypes - Interval Length 5

Affinity Propagation again does not produce any useful solutions for the prototypes. Only in 11 cases clusterings with less than 50 clusters are found. All of those clusters are for an interval length of 30.



(a) Affinity Propagation Prototypes (b) Affinity Propagation Segmentation

Figure 6.11: Norm Data: Prototypes and Segmentation for Affinity Propagation - 8 Prototypes - Interval Length 30

The so found exemplars are not easy to interpret and do not represent trends rather than long term developments of the time series as can be seen in figure 6.11. This results in an error of 0.023 on the train set and 0.027 on the test set. Compared to this the error of the K-Means segmentation with interval length 5 and 5 prototypes leads to a training error of only 0.010 and 0.011 test error. Although both errors are very low the prototypes of the K-Means clustering are significantly better suited to describe the trends of arbitrary time series.

6.1.4 Discussion

The plain data set is not suited to extract trends. It produces a segmentation of the time series into the different magnitudes of the exchange rate values. This does not represent trends that describe the time series.

The divmax data set is better suited to extract trends. It produces trends which represent different falling and rising gradients as well as a trend that represents a steady development. The problem of this approach is that it produces trends which describe the development relative to the current exchange rate values.

The normalized-data set produces the most useful trends. Especially when used with K-Means and HMM. Rather than producing a steady trend and different linear gradients, the trends produced with this data set are curves that represent different stages of rising or sinking intervals or intervals which are first rising and then sinking. This approach does not falsely classify sinking or rising intervals as steady.

DESICOM reduces the amount of rising prototypes and diversifies the sinking clusters. Affinity Propagation does not produce any useful clusterings. The results of Affinity Propagation are thus omitted for the next evaluations.

The following sections primarily focus on the normalized-data set because it produces trends which are considered as most useful and focus on the error values of the segmentation.

6.1.5 Different Interval Lengths

This Subsection considers how different interval lengths influence the result of the segmentation and prototype extraction. Recall that the interval length corresponds to the length in days of the trends. The amount of clusters will now be fixed to 5 and it will be tested how the interval lengths $I_L \in \{4, 5, 7, 10, 14, 30\}$ affect the result of the segmentation.

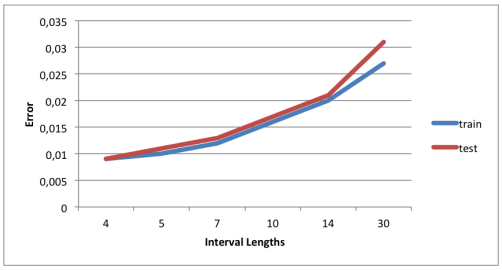


Figure 6.12: K-Means Segmentation - Train and Test Error with 5 Prototypes and different interval lengths

Obviously the segmentation is better for clusters of shorter length. This is easy to explain. Longer intervals are more specific, while shorter intervals describe more general developments of the time series. Hence the shorter trends are better suited to recreate the intervals of time series.

The graph also shows that clusters of all length seem to adapt very good to the train and test set. Nevertheless the error margin rises with longer trends because they are more specific to the train set.

When comparing the performance of the different clustering methods, HMM and K-Means show similar results. DESICOM has overall higher errors but they are not constantly rising

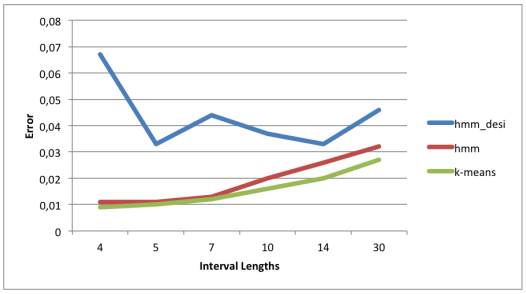


Figure 6.13: All Methods - Segmentation - Train Error with 5 Prototypes and different interval lengths

and more fluctuating. The DESICOM clusters were hereby generated with 50 initial HMM clusters. K-Means outperforms the HMM and DESICOM.

6.1.6 Different Cluster Numbers

Now the effect of different amounts of prototypes is examined on the segmentation. Different numbers of prototypes $P \in \{3, 5, 7, 10, 15, 20, 40\}$ are tested for a fixed interval length of 5.

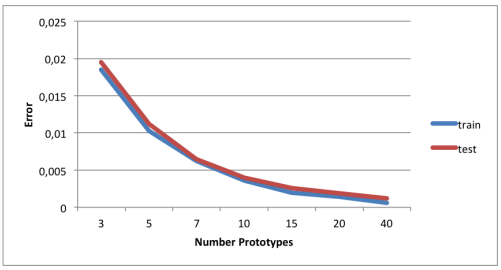


Figure 6.14: K-Means - Segmentation - Train and Test Error with interval length 5 and different prototype numbers

With an increasing number of clusters the error decreases. This can be explained with the fact that more prototypes are able to represent more intervals. This means that it is more likely to find a suiting prototype for each interval if the number of prototypes increases.

Again K-Means outperforms both HMM and DESICOM. HMM and K-Means are almost alike in the quality of the segmentation while DESICOM produces clearly worse segmentations. For K-Means and HMM the error is significantly small for any of the tested numbers of prototypes. For 5 prototypes and more the error is less or equal to 0.01 on the train and test set.

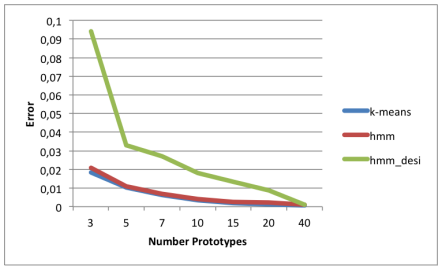
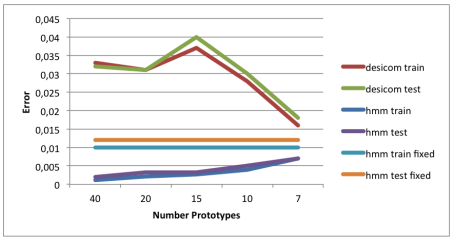


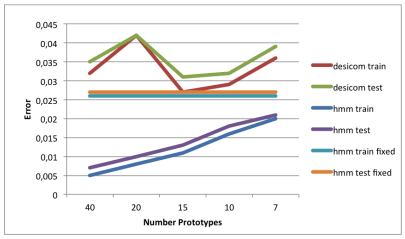
Figure 6.15: All Methods - Segmentation - Train Error with interval length 5 and different prototype numbers

6.1.7 DESICOM with different initializations

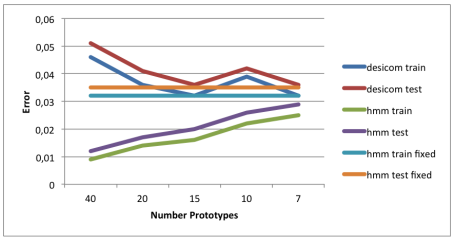
Now the effect that different initializations have on DESICOM is examined. The results show how different numbers of HMM clusters change the error for 5 DESICOM clusters with an interval lengths of 5, 14 and 30.



(a) Interval Length 5



(b) Interval Length 14



(c) Interval Length 30

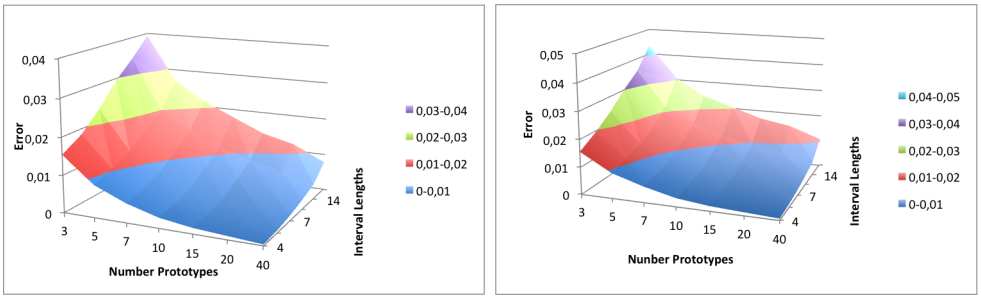
Figure 6.16: Effect on DESICOM when initialized with different numbers of HMM clusters

Although it is not unambiguous the overall result seems to be, that less HMM clusters lead to better DESICOM clusters. This implies that the DESICOM result is closer to the HMM result and points to the assumption that DESICOM does not offer any benefits to the HMM in the given use case. The HMM outperforms DESICOM by a large margin when omitting the reduction with DESICOM and also when fixed to 5 clusters.

Overall these results suggest that the DESICOM clusters are not as suitable as the K-Means and HMM clusters to segment the time series into trends.

6.1.8 Overall Parameter Space

To wrap the evaluation of the segmentation up, the whole parameter space is evaluated and it is checked which combinations of interval lengths and prototype numbers are especially suited to segment the training and test time series.



(a) Performance on Train Set (b) Performance on Test Set

Figure 6.17: Performance Evaluation over the whole parameter space

The maps point at what the previous sections already hinted at. Shorter intervals and more clusters lead to better results. It is still interesting to note that the overall errors are very low for K-Means and that for 5 clusters and interval length 4 the error is under 0.01 for the test and the training set. This is especially interesting because this case represents easy to interpret trends for humans.

6.1.9 Conclusion

The segmentation of the time series with prototypical trends is possible. The divmax-data set as well as the normalize-data set deliver useful trends for segmenting time series and interpreting the results. The prototypes in the divmax-data set highlight changes relative to the current exchange rate value while the normalize-data set displays changes which are independent of the magnitude of the exchange rate value. The normalize-data set segmentation can be considered as more secure since the divmax-data set segmentation classifies significant changes as "steady trend" if the absolute exchange rate value is very high.

K-Means and HMM produce the best results. Affinity Propagation does not produce any results of practical interest. The DESICOM clusters are at best as good as the HMM clusters but never better.

6.2 Prediction

In this Section the evaluation of the prediction methods is discussed. The prediction is performed on the disjoint interval sequence I_{Dj} and the overlapping sequence I_{Ov}. Both are acquired with the segmentation that was realized with the normalized-data set, as this data set proofed to provide the best prototypes and segmentation for the purpose.

As measure for the evaluation three aspects are considered. For any time point of the test time series the prediction model is provided with the history before that time point. Than the amount of times is counted, where the model predicts the correct trend for the given time interval. Also the correct prediction of transitions from one trend to another is counted Lastly E_{MSE}, as introduced in equation 6.1, is used to measure the error of the predicted trend to the actual interval.

In the evaluation process the best prediction model with the numbers of prototypes bound to $P \in \{5, 6, 7, 10\}$ and the interval lengths bound to I_L $in\{4, 5, 7, 10\}$ is sought.

6.2.1 Disjoint Intervals

First the prediction of trends on the sequence of disjoint intervals is evaluated. The Mt Gox time series is used to learn the prediction models and then the prediction model is applied to the Bitstamp time series since this time series covers dates which were not included in the Mt Gox time series, as well as dates which overlap with Mt Gox. The data set I_{Dj} partitions the time series into disjoint intervals. The disjoint intervals are obtained in correct order and the intervals prior to the one the model should predict are given as input. The prediction models are learned on the I_{Dj} of Mt Gox.

K-Means

Considering correct predictions the best result for K-Means is obtained for 5 clusters and interval length 4. 78% of all trends are predicted correctly. The baseline reaches 67% and is clearly beaten. The 78% are achieved using the transition matrix generated by counting the trend transitions in the segmented training data set. While 78% of the overall trends are correctly predicted, only 33% of the transitions between different trends are right. The error of the trends compared to the real intervals is $E_{MSE} = 0.071$.

Baseline	Transition Matrix	MC 1	MC 5	MC 9	2-G	6-G	10-G
0.674	0.778	0.774	0.665	0.353	0.776	0.727	0.677

Table 6.1: Prediction Rate - K-Means

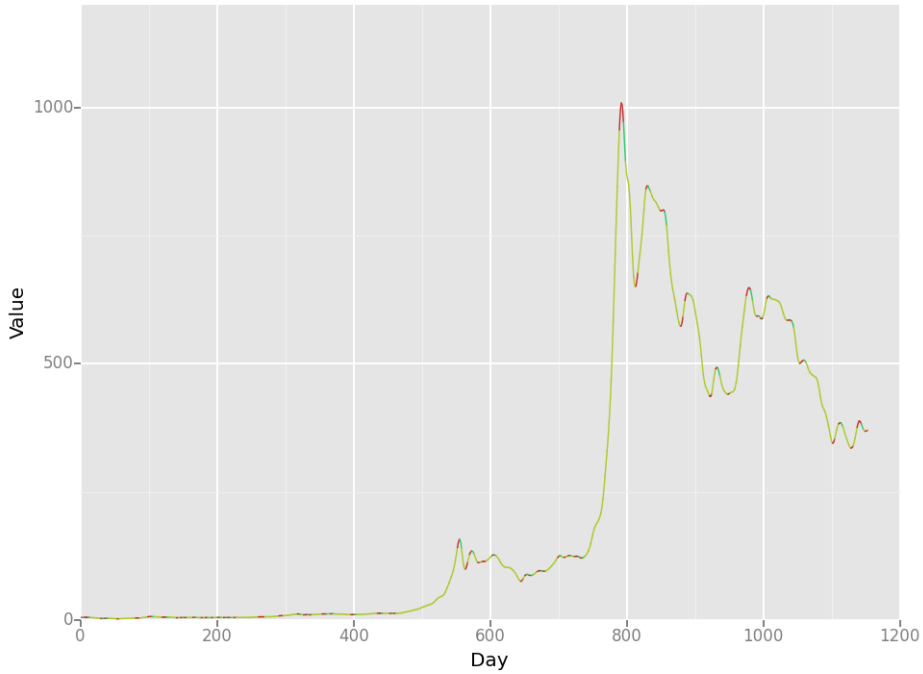


Figure 6.18: **Predictions:** Red: Wrong - Green: Correct - Turquoise: Correct Transition

Table 6.1 shows that the transition matrix gives the best prediction in every case. It seems like the history of the N-Gram model and the Markov Chain do not provide extra information that is useful for the prediction. The N-Gram model outperforms the Markov Chain. This is no surprise since the N-Gram model is build open the same assumptions and further improved with the Kneser-Ney backoff. Higher orders lead to more mistakes indicating that the probabilities obtained by larger history are specific to the training set.

HMM

The best HMM prediction model is again for 5 prototypes and an interval length of 4. 81% of all trends are correctly predicted, compared to a baseline of 68%. Again it is the transition matrix that leads to the best results. The error when comparing the prototypes and the real intervals is $E_{MSE} = 0.069$. 40% of trend transitions to other trends are predicted correctly.

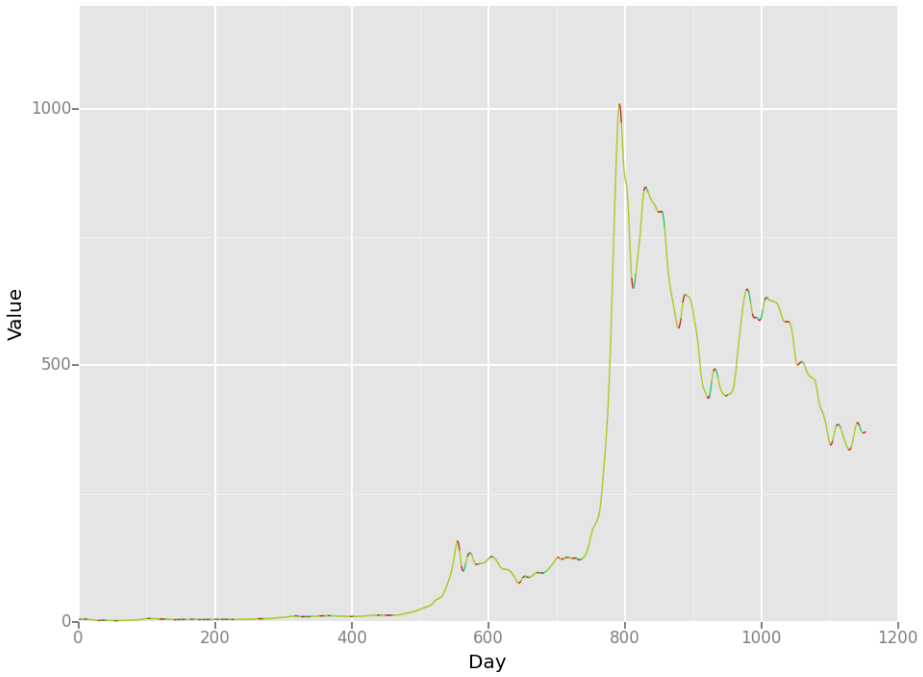


Figure 6.19: **Predictions:** Red: Wrong - Green: Correct - Turquoise: Correct Transition

The N-Gram model outperforms the Markov Chain. Both perform better than the baseline for low orders, but the accuracy drops when using higher orders respectively longer state histories.

Baseline	Transition Matrix	MC 1	MC 5	MC 9	2-G	6-G	10-G
0.684	0.812	0.808	0.608	0.358	0.810	0.781	0.693

Table 6.2: Prediction Rate - HMM

DESICOM

For DESICOM very different results can be observed. The best prediction model is the 2-Gram model predicting the correct trends in 78% of the cases. This is a good value compared to the 64% baseline. 42% of the transitions between different trends are correct. The error when comparing intervals and predicted trends is very low: $E_{MSE} = 0.057$. The transition matrix which is learned with DESICOM only predicts 52% of the trends correctly and is thus even worse than the baseline assumption, that trends are always followed by the same trends.

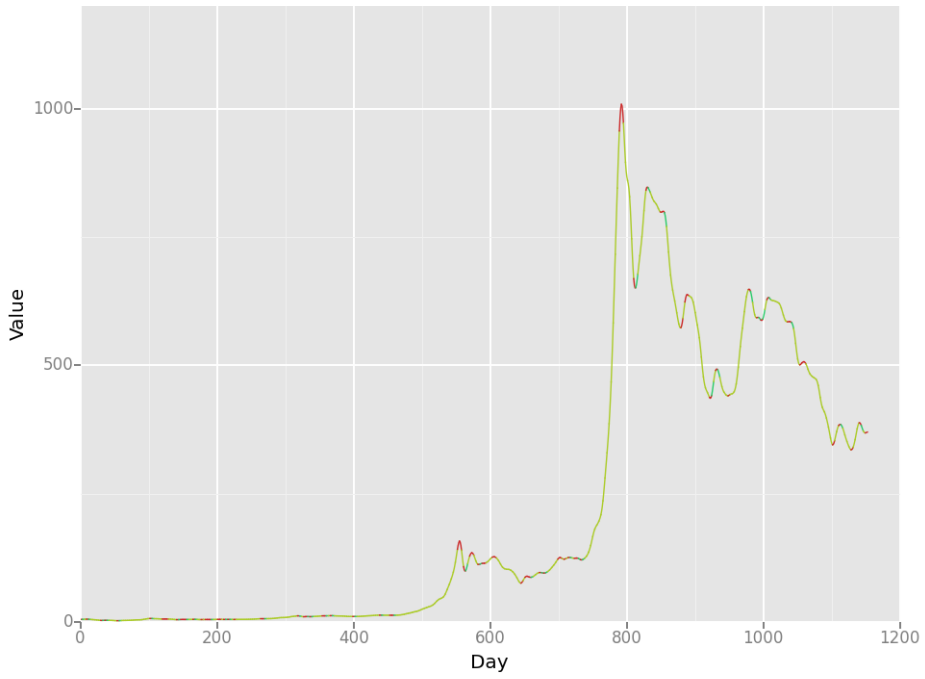


Figure 6.20: **Predictions:** Red: Wrong - Green: Correct - Turquoise: Correct Transition

Baseline	Transition Matrix	MC 1	MC 5	MC 9	2-G	6-G	10-G
0.637	0.522	0.784	0.473	0.281	0.787	0.674	0.643

Table 6.3: Prediction Rate - DESICOM

The N-Gram model outperforms all other prediction models for the DESICOM clustering. But again there seems to be no benefit of using larger histories.

Prediction on the Training data set

It is interesting to see how the prediction works on the training set itself. Therefore the prediction is run with the HMM on the training set with the best configuration of the test set, consisting of 5 prototypes and an interval length of 4.

Baseline	Transition Matrix	MC 1	MC 5	MC 9	2-G	6-G	10-G
0.667	0.812	0.808	0.822	0.877	0.810	0.824	0.877

Table 6.4: Prediction Rate - HMM on training data

A very different behavior can be observed here. Higher orders lead to significantly better results than the baseline method or the transition matrix. This is because the high order models overfit the training data. They learn the trends of the training data by heart.

Fixing the number of Prototypes

Now the number of prototypes is fixed and the results for different interval lengths are evaluated using the HMM transition matrix as prediction model.

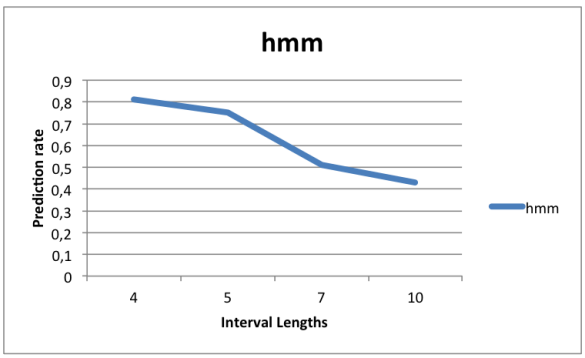


Figure 6.21: Prediction Rate with fixed number of prototypes and different interval lengths

Increasing the interval length leads to a drastically decreased prediction accuracy. This can be explained easily. Increasing the interval length is equivalent to making predictions for events which are further away in the future and increasing the interval length leads to a fewer trends to learn from in the training set. The amount of trends is inversely proportional dependent of the length of the intervals in the I_{Dj} sequence.

Fixing the Interval length

Now the interval length is fixed to 5 and it is tested how different numbers of prototype affect the results. Again with the HMM prediction model.

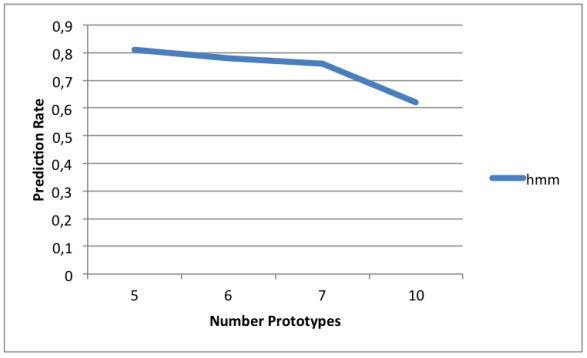


Figure 6.22: Prediction Rate with fixed interval length 5 and different numbers of prototypes

The same effect as for the different interval lengths can be observed. More prototypes lead to a decreased prediction quality. To understand this recall that more prototypes mean a finer segmentation of the time series. More prototypes lead to a more detailed description of intervals. This makes it harder for the models to predict. One example is that in settings with 3 prototypes named "rising", "equal" and "sinking" and with 5 prototypes named "strongly rising", "slightly rising", "equal", "slightly sinking" and "strongly sinking", it is immediately clear that it will be easier for the model to predict whether something will be just "rising" rather than having to differentiate between "strongly" and "slightly" rising. Another important aspect is, that with more prototypes more transitions between trends are introduced. As seen so far the models are not very good in handling such transitions.

Discussion

K-MEANS	HMM	DESICOM
0.778 (0.328)	0.812 (0.405)	0.784 (0.417)

Table 6.5: The best results of the models in comparison, transition prediction rates in brackets

The results so far point at HMM being the best prediction model for the trend based prediction on I_{Dj}. Correct predictions of up to 81% are reached with this model. DESICOM and the K-Means prediction models are just slightly worse, but the K-Means model handles transitions between trends worst. It seems like the time component of HMM and DESICOM have a clear influence here.

The higher order models fail to beat the simple transition matrix models. There seems to be not much additional information in the trend histories. On the contrary higher orders lead to more mistakes on the test set but better performance on the training set. This is because the high order models memorize the structures of the training set. These structures seem to be particular to the training set and of no value for the test set. The bottleneck of the predictions is clearly discovering the correct points for transitions between trends.

6.2.2 Overlapping Intervals

Additionally to the I_{Dj} the models also tested and trained on I_{Ov}. This is only of theoretical interest because predicting trends in this set, given an immediate history is equal to cheating.

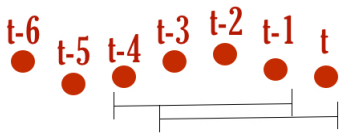


Figure 6.23: t-4 is the last interval that can be utilized for prediction without utilizing unknown values

As can be seen in 6.23 the intervals overlap in k-1 points. If the last time point should be used as trend history, this would correspond to looking $k-1$ points into the future. Without the 'cheated' look-ahead this allows only for a 1 day prediction. Nevertheless it is interesting to see how the prediction model works for this data set and to examine whether the prediction models predict differently. This set is evaluated for the same setting of 5 prototypes and interval length 4.

Evaluation

	Baseline	Transition Matrix	2-GR	6-GR	10-GR
K-MEANS	0.84 (0)	0.89 (0.36)	0.89 (0.36)	0.89 (0.44)	0.88 (0.47)
HMM	0.85 (0)	0.89 (0.58)	0.89 (0.58)	0.91 (0.45)	0.90 (0.45)
DESICOM	0.85 (0)	0.91 (0.56)	0.91 (0.56)	0.91 (0.56)	0.91 (0.48)

Table 6.6: Results on the overlapping intervals, transition prediction rates in brackets

An overall higher prediction value can be observed for all models. This results out of two reasons. The baseline is a lot better for all models. This means that fewer transitions between trends occur. This makes it easier for the prediction model. But also higher orders

lead to better results in contrast to the prediction models for I_{Dj}. This means that more structural information is given in I_{Ov}. Because of the look-ahead of $k-1$ time points, it is clear that the model can predict trend transitions easier. For the HMM model and DESICOM the transitions are now predicted correctly about 56% of the time. For K-Means the prediction of transitions reaches 52% with a 3-Gram model. Interestingly DESICOM now even performs best when considering the 'transition matrix' model. Because in I_{Ov} are also more interval transitions, because there are more intervals than in I_{Dj}, DESICOM gains a lot of accuracy. It seems to benefit of the more accurate transition matrix produced by the HMM.

Discussion

The prediction on I_{Ov} is better but only partially useful in practice. It requires a look-ahead which is not given with real data. If the look-ahead is not provided one can only predict one day ahead with this method.

The better prediction is a result of many factors. First there are more intervals in I_{Ov} to learn from. Especially for N-Gram models large training sets are of benefit, but also for DESICOM which relies on an accurate similarity matrix as input. But another crucial point is that there are larger sequences of trends without transitions, which also results in a better prediction of the simple baseline model. Nevertheless also more correctly predicted transitions can be observed and especially higher order models seem to benefit of the structures in I_{Ov}.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, the exchange rate of Bitcoins was examined with special focus on the Mt Gox and BTCE market. Bitcoin is a virtual currency and, more specific, cryptocurrency that is of rising interest for the general public.

An analysis of the Bitcoin exchange rate with respect to periodicity showed little to no periodic structures in the time series. Large correlations of the Bitcoin exchange rate to the NASDAQ and Dow Jones were discovered. A very large correlation of up to 94% was found when comparing weekly Bitcoin courses to weekly searches of the term "Bitcoin" on Google Trends.

Another result is that ARIMA models seem to be not well suited to predict short-term trends of the exchange rate. They either predicted the volatility or the actual values of the exchange badly.

A method to use clustering for the extraction of prototypical trends was proposed. Those prototypical trends were used to segment time series into trend-intervals. This segmentation offers an alternative reduced representation of the time series. Normalizing the intervals into a range of $[0, 1]$ produced the desired representative trends. K-Means and HMM showed particularly good clusterings, while Affinity Propagation failed. Using DESICOM to reduce the clusters found by an HMM resulted in no benefits over the HMM segmentation.

Lastly methods of using the trend representation of the time series to predict future trends were shown. On a set of disjoint intervals good prediction rates were achieved with models which utilize the Markov property, meaning that they only use the current trend to decide which trend will be next. Transitions between trends were rarely predicted correctly. On an in practice not usable set I_{Dj} better prediction results were achieved and the use of high-order models was of benefit.

Overall the segmentation of the time series and the extraction of trends produced very good results with low errors compared to the actual intervals. The prediction is still not very accurate considering changes of trends (up to 58% correct for I_{Ov} and 41% for I_{Dj}) and needs further refinement. Other than the data itself it could be promising to learn prediction models by incorporating external data.

7.2 Future Work

While the segmentation achieves good results in assigning trends to intervals, it completely ignores the actual value of the exchange rate. It would be of benefit, especially for the prediction, to find a model that shows the trend of the time series as well as a predicted value.

The prediction models seem to be too simple to model transitions between trends. The assumption that such models can be learned solemnly from the data itself might be too simple. This does not mean that these models should be omitted. A way of using them could be, to utilize the probabilities they produce for every potential trend. Other time series like the Google Trend time series for the search term 'Bitcoins', the NASDAQ and Dow Jones could be used to learn further prediction models for the Bitcoin exchange rate because strong correlations with them were found. An ensemble model could be learned that combines the probabilities of the prediction models presented in this theses and the proposed external models.

One more thing that could be regarded for the prediction models, using the data itself, is utilizing variable-order-Markov-Chains [21]. These Markov Chains offer more flexible counts and could improve the prediction of transitions.

List of Figures

List of Tables

Bibliography

[1] J. A. Bergstra and P. Weijland, "Bitcoin: a money-like informational commodity," *CoRR*, vol. abs/1402.4778, 2014.

[2] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security* (A.-R. Sadeghi, ed.), vol. 7859 of *Lecture Notes in Computer Science*, pp. 6–24, Springer Berlin Heidelberg, 2013.

[3] N. Christin, "Traveling the silk road: A measurement analysis of a large anonymous online marketplace," in *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, (Republic and Canton of Geneva, Switzerland), pp. 213–224, International World Wide Web Conferences Steering Committee, 2013.

[4] J. A. Kroll, I. C. Davey, and E. W. Felten, "The economics of bitcoin mining, or bitcoin in the presence of adversaries."

[5] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin." Cryptology ePrint Archive, Report 2014/349, 2014. `http://eprint.iacr.org/`.

[6] S. Zolhavarieh, S. Aghabozorgi, and Y. W. Teh, "A review of subsequence time series clustering," *The Scientific World Journal*, 2014.

[7] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: Implications for previous and future research," *Knowl. Inf. Syst.*, vol. 8, pp. 154–177, Aug. 2005.

[8] J. R. Chen, "Useful clustering outcomes from meaningful time series clustering," in *Proceedings of the Sixth Australasian Conference on Data Mining and Analytics - Volume 70*, AusDM '07, (Darlinghurst, Australia, Australia), pp. 101–109, Australian Computer Society, Inc., 2007.

[9] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, (New York, NY, USA), pp. 2–11, ACM, 2003.

[10] T. Liu, "Application of markov chains to analyze and predict the time series," *Computer and Information Science - Canadian Center of Science and Education*, â€¢.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009.

[12] E. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.

[13] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[14] M. Stamp, "A revealing introduction to hidden markov models," 2004.

[15] H. A. Kiers, "Desicom : Decomposition of asymmetric relationships data into simple components," *Behaviormetrika*, vol. 24, no. 2, pp. 203–217, 1997.

[16] C. Bauckhage, R. Sifa, A. Drachen, C. Thurau, and F. Hadiji, "Beyond heatmaps: Spatio-temporal clustering using behavior-based partitioning of game levels," in *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pp. 1–8, Aug 2014.

[17] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406–413, 7 1955.

[18] R. Cogill, "Introduction to markov chains," 2011.

[19] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009.

[20] C. Hurvich, "The basics of arma models," 2012.

[21] R. Begleiter, R. El-yaniv, and G. Yona, "On prediction using variable order markov models," *Journal of Artificial Intelligence Research*, vol. 22, pp. 385–421, 2004.

Affidavit

Herewith I declare that I have written the present thesis on my own and without the use of other than the indicated support.

Passages that I have adopted, whether in sense or literally from other published or non-published works have been marked as such.

The present thesis has, in the same or in similar form, never been handed in to another board of examination.

Bonn,

(Sven Giesselbach)