TUHH
Hamburg University of Technology

Master's Thesis

Jonas Lüthke

Location Prediction Based on Mobility Patterns in Location Histories

September 29, 2013

supervised by:
Prof. Dr. rer. nat. habil. Ralf Moeller
Prof. Dr. rer. nat. Volker Turau

Hamburg University of Technology (TUHH)
Technische Universität Hamburg-Harburg
Institute for Software Systems
21073 Hamburg

TUHH STS
Software
Technology
Systems

Abstract

Human individuals generally tend to follow several habits during the course of the day. This fact intuitively allows predicting human behavior to a certain degree based on previous observations. This thesis focuses on the mobility of human individuals. To be specific, a generic algorithm that uses *kernel density estimation* and *quadratic optimization* to provide location predictions is proposed. There are several imaginable fields of application for such an algorithm, like for example location based services or commercials. The proposed algorithm was implemented and tested using mobility traces of taxis. The test results clearly indicate that the algorithm can extract and exploit patterns in the data to predict future locations. For instance, the algorithm achieves an accuracy better than 1000m in approximately 32% of the executed tests using a prediction interval of six minutes. Moreover, in 13% of these tests the prediction error is smaller than 500m. In addition, the test results show that the algorithm is able to estimate the reliability of its predictions with an accuracy of up to 98.75%. As expected, the test results also clearly demonstrate that the prediction capability of the algorithm strongly depends on the properties of the given location data and the underlying stochastic process.

Contents

List of Figures

List of Tables

Chapter 1

Motivation

Human individuals generally tend to follow several habits during the course of the day. Thus, many daily actions and processes in human life are a matter of routine. This fact intuitively allows to predict human behavior to a certain degree based on previous observations. Certainly, the performance of this task requires to identify recurring patterns in the observed behavior at first. Then, these patterns can be exploited to make an educated guess about the future. The same approach is basically used in the research field of *data mining* to extract meaningful relations from large amounts of data. Due to remarkable progresses in computer technology regarding computing capacity as well as data storage, interest increased in data mining. Numerous research projects engage in this field and are producing promising results.
This thesis focuses on so called *location time series data* that can be collected by tracing the mobility of a human individual. Particular data mining techniques are applied to find recurrent patterns in location histories and to predict future locations. To be specific, a generic algorithm is proposed that uses *kernel density estimation* and *quadratic optimization* to provide location predictions. There are several possible fields of application for this algorithm, like for example location based services. Since the number of smartphone owners is increasing fast these services become more and more interesting. Location based services are particular functionalities and information that are provided to the user with respect to the geographic position (e.g. [1, 2]). For instance, once a user approaches a subway station, the smartphone could provide him with the timetables of all related subway lines. Generally, the geographic context allows services to adjust to the needs of the user. Therefore, one can also think of location based commercials suggesting nearby local stores or restaurants to the user. Clearly, location based services or commercials would profit by a location prediction engine. This would enable them to provide the user with geographically adapted content even in advance.
This thesis aims at the development and the subsequent verification of a location prediction algorithm using appropriate existing data mining techniques. This goal is further specified in chapter 2 providing a clearly defined hypothesis. Chapter 3 relates this thesis to previous work in the field of human mobility analysis and prediction. In chapter 4 a specific representation for location time series data is described since it serves as a basic principle for the prediction algorithm proposed in this thesis. In addition, the algorithm uses Gaussian mixture models to estimate probability distributions. Therefore, chapter 5 briefly emphasizes some specific properties of such models. Based on this, chapter 6 motivates and describes the actual location prediction algorithm that uses kernel density estimation to learn a probability distribution and quadratic optimization to obtain a prediction. As no location traces of human individuals were available, the developed algorithm was tested using a data set containing location histories of taxi cabs. Chapter 7 describes and discusses the observed test results. Finally, a summary of this thesis with respect to the initially defined goal is given and ideas for further work are described in chapter 8.

Chapter 2

Goal: Location Prediction

The goal of this thesis is to develop a location prediction technique that identifies and exploits patterns in location history data. A generic algorithm for this purpose shall be developed and tested. The algorithm shall base its predictions on a probability density distribution that is derived from a certain amount of previously observed location time series data. In addition, the prediction technique shall preferably rely on non-parametric methods to ensure a certain degree of adaptability. This implies particularly the processing of location data without previous clustering. Hence, the predictor shall not base on the assumption of a limited set of discrete locations. Moreover, the prediction technique is required to resolve in time and space and provide a reliability value for each prediction. Generally, this algorithm should be able to cope with any location data that is generated by some stochastic process. Nevertheless, this thesis is mainly aimed at prediction of human mobility. Summarizing, the central hypothesis to be verified is defined as follows: Recurring patterns in location histories can be extracted by non-parametric learning methods and then be used for prediction.
This hypothesis shall be verified by developing a *proof-of-concept* algorithm. In particular, the developed algorithm shall be tested using a *real-world* data set.
To the best of my knowledge the algorithm that is proposed in this thesis is a new approach that has not been examined so far.

Chapter 3

Related Work

A lot of research has been done in the field of analyzing human mobility and several studies have shown that generally certain patterns can be extracted from human location histories (e.g. [3, 4, 5, 6]). Furthermore, different techniques to predict locations of human individuals based on such patterns have been proposed and examined (e.g. [7, 8, 9, 10]). Here some important results are presented briefly due to their close relation to the approach that is presented in this thesis. Actually, some parts of the prediction algorithm proposed in this thesis are motivated by the work presented in the following.

3.1 Human Mobility Analysis

Human mobility has been analyzed at varying geographic scales. In [3] the authors investigate the location data of 100,000 mobile phone users that have been collected by tracking each person's position for six months. They were able to show that after some corrective preprocessing the individual travel patterns of the test persons can be described by a single spatial probability distribution. Thus the results of this study suggest that humans generally follow simple reproducible patterns.
In [4] the actual predictability provided by these mobility patterns is examined. The authors of this article use the data of 50,000 mobile phone users that have been collected over a period of three months to study the statistical properties of human mobility. They use different entropy measures to estimate the potential predictability of a user's trajectory. Based on their analysis they report to have found that the user mobility provides a potential predictability of 93% taking their whole data base into account. These results are particularly interesting with regard to this thesis. To be specific, the algorithm that is presented here, can be motivated by the notion of *spatiotemporal entropy* as defined in [4]. This will be described in more detail in chapter 6.

3.2 Prediction of Human Mobility

Different techniques have been applied to the problem of predicting locations of human individuals. In [7] several different approaches are examined to predict locations of students based on data collected by Wi-Fi access points. The authors of this paper define a set of discrete locations using the Wi-Fi cells on a university campus. Furthermore, they limit predictions to the next location without taking time into account. Two different kinds of location predictors are tested. Namely, the authors use a k-th order Markov predictor as well as a LZ-based predictor. Based on their test results they report that the second order Markov predictor with a certain fallback feature[1] performed best and provided a median accuracy of 72%[2]. Basically the approach in this thesis also assumes a k-th order Markov process as explained in chapter 6. However, the algorithm proposed in this thesis is more generic since it resolves in time and does not rely on a set of discrete locations.

[1] They used recursively the first order or zeroth order predictor if prediction was not possible.
[2] This holds only for users that provided a sufficient amount of data.

Consequently, its accuracy can be expressed by means of Euclidean distance in meters and depends on the prediction interval. In contrast, the accuracy metric used in [7], only distinguishes between wrong and right predictions and does not take the prediction interval into account.
A similar prediction approach is proposed in [8]. There, a sort of extended Markov predictor is presented. To be specific, the authors use *delay embedding* to extract location sequences of a certain length from a time series. Then these sequences are directly used to predict a users location. Actually, a prediction is obtained by comparing the last observed locations to all embedded location sequences. This technique takes the arrival times and residence times into account. However, it relies on a finite set of distinct locations. Thus, it can only predict the user to be or not to be in one of his most frequented locations[3]. This approach is basically quite similar to what is presented in this thesis. Especially, delay embedding is applied, too. Nevertheless, the algorithm proposed in this thesis does not work with discrete locations but uses coordinate values to estimate a density distribution and provide predictions. This can be considered as a more general approach.

[3] These locations need to be extracted from the data in advance.

Chapter 4

Location Time Series Data

This work is aimed at predicting future locations based on location history data. In this chapter a particular representation for such data is described as it serves as a basic principle for the prediction algorithm. Furthermore, map projection is explained briefly.
Location history data is a particular form of time series data [11]. Generally, a location time series dataset consists of a sequence of data points that have been measured successively. Each data point should at least consist of a location[4] and a corresponding timestamp. Depending on the data source additional information can be related to the data points.

4.1 Delay Embedding

There are different possible mathematical representations of location time series data. Maybe, the most intuitive representation would be a three dimensional vector space with latitude, longitude and time each being one dimension. With respect to the prediction algorithm that is proposed in this thesis, the choice of a particular data representation is crucial. The main question to be considered when choosing a mathematical representation for the data is whether it allows the extraction of the desired information. Regarding the aim of this thesis, the information in demand are mobility patterns. In terms of location time series data, mobility patterns are similar sequences of coordinate pairs that occur repeatedly over time. Hence, to extract mobility patterns from location time series, subsections of the data series could be compared to each other. If there are sufficiently many sections that resemble each other, this could be assumed to be a mobility pattern. However it has to be defined how resemblance can be quantified and what amount of similar sections are needed to assume a pattern.
So the information that need to be extracted from the location time series data are similarities between different subsections of the time series. A common representation for time series that is used in *nonlinear time series analysis*[12] is the *delay vector reconstruction*. This representation is obtained by *delay embedding*:
To embed a time series $(s_1, s_2, ..., s_N)$ in an m-dimensional space, a delay ν is defined and then the delay vector reconstruction is created for the time series value s_n as follows:

$$\boldsymbol{\delta}_n = [s_{n-(m-1)\nu}, s_{n-(m-2)\nu}, ..., s_{n-\nu}, s_n] \tag{4.1}$$

$$n \in \{1, 2, ..., N\}$$

All vectors $\boldsymbol{\delta}_n$ have the dimension m. The m-dimensional space these vectors are defined in is called *embedding space*. In [8] delay embedding is used to build a predictive model directly based on the time series data. Namely, the next location is predicted by averaging over similar previously observed delay vectors. This technique relies on a limited set of possible locations.
In contrast, the algorithm that is described in this thesis uses the delay vectors in the embedding space to learn a continuous probability density function. Predictions are then based on this density

[4] E.g. a geographic coordinate defined by latitude and longitude values.

function. This approach is motivated by the fact that the similarity between two time series sections of length m is mapped to the Euclidean distance between the corresponding vectors in the embedding space. Thus, the density in the embedding space is a measure for the observed frequency of a particular time series subsequence.
Loosely speaking, this means, similar sequences that are observed multiple times will be mapped to points that are close to each other in the delay embedding space. The parameter m defines the number of subsequent time series values to be considered and ν is the time interval between two values. There are techniques that optimize these parameters to exploit redundancies in the data to the best degree [12, p. 36-39]. Such methods are not used or discussed in this thesis as they depend on the particular data source. However, in section 7.6 the impact of ν and m on the prediction quality with regard to the used test data set is analyzed.

4.2 Map Projection

Location data is usually given in terms of latitude and longitude value pairs. These values refer to angles that define a location on the surface of the earth using a polar coordinate system. To draw a map using such data, they can be projected to a two-dimensional Cartesian coordinate system with equally scaled axis. This transformation is called map projection and there are several different techniques that can be used for this purpose. Clearly, such a projection leads to a loss of information since not all metric properties can be preserved. However, the algorithm as described in this thesis, assumes equally scaled coordinates. Therefore, the use of a map projection is necessary. In the implementation that is part of this thesis the Universal Transverse Mercator (UTM) system is applied. For a detailed description of the UTM it is referred to the literature (e.g. [13]). This projection preserves angles and approximates shape but distorts distance and area measures. However, within a UTM zone the distance measured using the projection does not deviate more than one meter from the true distance. This is a sufficient accuracy regarding the algorithm presented in this thesis.

Chapter 5

Gaussian Mixture Models

The prediction technique described in chapter 6 is based on Gaussian mixture models (e.g. [14]) and relies on some particular properties of such models. Therefore, a short description of these models and the relevant properties is given in the following.
A Gaussian mixture model is defined as a weighted sum of Gaussian mixture components. Each component m is a Gaussian distribution with a dedicated mean $\boldsymbol{\mu}_m$ and covariance $\boldsymbol{\Sigma}_m$. The model describes the probability density distribution of a random vector $\mathbf{X}$:

$$p(\mathbf{x}) = \sum_{m \in \mathcal{M}} \omega_m \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{5.1}$$
$$\mathbf{x} \in \mathbb{R}^d,\ \boldsymbol{\mu}_m \in \mathbb{R}^d,\ \boldsymbol{\Sigma}_m \in \mathbb{R}^{(d,d)},\ \mathcal{M} = \{1, 2, ..., M\},\ \sum_{m \in \mathcal{M}} \omega_m = 1,\ \omega_m \geq 0.$$

5.1 Conditional Distribution of a GMM

It is possible to derive the conditional and marginal distributions from the joint distribution of a Gaussian mixture model[15, 16]. The following equations are based on [17].
Let $p(\mathbf{x})$ be a multivariate mixture of Gaussians with $\mathbf{x}$ being an d-dimensional vector. Then the conditional distribution of $p(\mathbf{x})$ with respect to the component x_k, $k \in \{0, ..., d-1\}$, of $\mathbf{x}$ is defined as

$$p(\hat{\mathbf{x}}|x_k) = \frac{p(\hat{\mathbf{x}}, x_k)}{p(x_k)},\ \hat{\mathbf{x}} = (x_0, ..., x_{k-1}, x_{k+1}, ..., x_{d-1})^T, \tag{5.2}$$

$$p(x_k) = \int p(\mathbf{x}) dx_k. \tag{5.3}$$

This is directly derived from the definition of conditional probability and can easily be extended to depend on multiple components of $\mathbf{x}$. $p(x_k)$ is called the *marginal distribution* of $p(\mathbf{x})$.
Furthermore, it is possible to express the marginal and conditional distributions of a mixture of Gaussians in terms of the mixture components:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix},\ \boldsymbol{\mu}_m = \begin{pmatrix} \boldsymbol{\mu}_{m1} \\ \boldsymbol{\mu}_{m2} \end{pmatrix},$$

$$\boldsymbol{\Sigma}_m = \begin{pmatrix} \boldsymbol{\Sigma}_{m11} & \boldsymbol{\Sigma}_{m12} \\ \boldsymbol{\Sigma}_{m21} & \boldsymbol{\Sigma}_{m22} \end{pmatrix},\ \boldsymbol{\Lambda}_m = \boldsymbol{\Sigma}_m^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{m11} & \boldsymbol{\Lambda}_{m12} \\ \boldsymbol{\Lambda}_{m21} & \boldsymbol{\Lambda}_{m22} \end{pmatrix}.$$

Then the marginal distribution is given by

$$p_m(\mathbf{x}_1) = \int p_m(\mathbf{x}) d\mathbf{x}_2 = \mathcal{N}(\mathbf{x}_1|\boldsymbol{\mu}_{m1}, \boldsymbol{\Sigma}_{m11}). \tag{5.4}$$

The conditional distribution for each component m is derived as

$$p_m(\mathbf{x}_1|\mathbf{x}_2) = \frac{p_m(\mathbf{x}_1, \mathbf{x}_2)}{p_m(\mathbf{x}_2)} = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{m1|2}, \boldsymbol{\Lambda}_{m11}^{-1}), \tag{5.5}$$
$$\boldsymbol{\mu}_{m1|2} = \boldsymbol{\mu}_{m1} - \boldsymbol{\Lambda}_{m11}^{-1}\boldsymbol{\Lambda}_{m12}(\mathbf{x}_2 - \boldsymbol{\mu}_{m2}).$$

For the entire Gaussian mixture model the conditional distribution is given by

$$p(\mathbf{x}_1|\mathbf{x}_2) = \sum_{m\in\mathcal{M}} \omega'_m p_m(\mathbf{x}_1|\mathbf{x}_2), \quad \omega'_m = \frac{\omega_m \mathcal{N}(\mathbf{x}_2)|\boldsymbol{\mu}_{m2}, \boldsymbol{\Sigma}_{m22})}{\sum_{m\in\mathcal{M}} \omega_m \mathcal{N}(\mathbf{x}_2)|\boldsymbol{\mu}_{m2}, \boldsymbol{\Sigma}_{m22})}. \tag{5.6}$$

5.2 The Mahalanobis Distance

The Mahalanobis distance is a distance measure between two points in a multidimensional vector space. It is defined as

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})}. \tag{5.7}$$

The Mahalanobis distance between the mean of a particular Gaussian component m and a point $\mathbf{x}$ gives a measure of how much the component contributes to the density $p(\mathbf{x})$ at $\mathbf{x}$. This is helpful to decide whether a particular component has to be considered during the evaluation of $p(\mathbf{x})$. Thus, it can save computing time during the evaluation.

Chapter 6

Prediction of Future Locations

The prediction technique that is presented in this thesis consists of two main parts. The first part is a machine learning method that estimates a probability density distribution from given sample data. The second part uses the probability distribution and a set of lastly observed locations to predict a future location. Each of these parts consists of a few subcomponents. In the following, at first a motivation for this prediction technique is given and then the technique is described in more detail.

6.1 Motivation of the Proposed Technique

Assume that the actual probability distribution that underlies the mobility of some human individual is known. Let it be described by the conditional probability $Pr[X_t = x|h_{t-1}]$, where X_t is the location at time t and $h_{t-1} = \{X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, ..., X_1 = x_1\}$ is the location history consisting of $t-1$ previously visited locations. This notation suggests a statistical dependence of the future location X_t on the location history h_{t-1}. This property is called *serial dependence*. The presence of serial dependence in a location time series generated by human mobility is the main prerequisite for the algorithm that is described in this thesis. As presented in [4] by means of an empirical analysis it is legitimate to assume this prerequisite to be fulfilled.
Intuitively, an optimal prediction algorithm that is based on the distribution mentioned above, would always predict the location that maximizes the probability distribution:

$$\hat{x}_{ML} = \arg\max_x \{Pr[X_t = x|h_{t-1}]\} \tag{6.1}$$

This means it would predict the person to be in its most likely location. Actually it is shown in [18, p. 12] that any prediction algorithm based on an observed location history can not do better than this.
Hence, to design a powerful prediction technique, the distribution $P(X_t = x|h_{t-1})$ needs to be estimated from the given location history data. As described in section 4.1, the statistical dependence of the future location X_t on the location history h_{t-1} can be extracted from the data by means of recurring patterns. This approach is also encouraged by the notion of spatiotemporal entropy in [4]. Entropy is a useful measure to quantify the potential predictability of a time series. In [4] the entropy of a location time series is defined as

$$S = -\sum_{X' \subset X} P(X') \log_2[P(X')], \tag{6.2}$$

where $X = \{X_1, X_2, ..., X_{t-1}\}$ is the observed sequence of locations and $P(X')$ is the relative frequency of the time-ordered subsequence X' in X. This definition of spatiotemporal entropy refers to a limited set of discrete locations. The recurrence of a subsequence of locations in the time series decreases the entropy and thereby increases the predictability. To extend this definition of entropy the locations are now assumed to be defined in a two dimensional continuous space by their Cartesian coordinates. Furthermore, the definition of $P(X')$ is extended to be the relative

frequency of all time-ordered subsequences in X weighted by their similarity to X'. This matches well with the idea of extracting mobility patterns by examining similar subsequences in a location time series as described in section 4.1.
Given this motivation, the prediction of future locations should be based on the entire observed location history. Hence, a possible prediction algorithm could work like this:

Algorithm 1 : A possible location prediction algorithm that considers all serial correlations in a location history

1: $t \leftarrow X.length + 1$
2: **for** $m = 3$ to $X.length$ **do**
3: $X' \leftarrow [X_{t-1}, ..., X_{t-(m-1)}]$
4: **for all** time-ordered subsequences $X_{m,i}$ of length m in X **do**
5: Compare X' to $X_{m,i}$ ignoring the m-th element:
$\omega_{m,i} \leftarrow sim(X', X_{m,i}[1 : m-1])$
6: Estimate the next location by taking the weighted sum of the m-th elements:
$X_{pred} \leftarrow X_{pred} + \omega_{m,i} X_{m,i}[m]$
7: **end for**
8: **end for**

6.2 Markov Assumption

The above algorithm uses all available information and considers every possible correlation to predict a location. Thus, it should yield reasonable results. However, the time complexity of this algorithm depends quadratically on the number of observed locations ($O(n^2)$). As the number of observed locations is increasing and needs to be large to provide sufficient statistics, this is not a usable algorithm in practice. To derive an algorithm with a significantly smaller complexity it can be assumed that the next location only depends on a limited number of previously visited locations:

$$Pr[X_t = x|h_{t-1}] = Pr[X_t = x|h_{(t-1):(t-m)}], \tag{6.3}$$

where $h_{(t-1):(t-m)} = \{X_{t-1}, X_{t-2}, ..., X_{t-m}\}$ are the m previously visited locations. This property of a stochastic process is called the *Markov property* and the related process is then called a *Markov process*. Assuming this property and the knowledge of the probability distribution $Pr[X_t = x|h_{(t-1):(t-m)}]$, a prediction algorithm with complexity O(n)[5] can be defined as follows:

Algorithm 2 : A location prediction algorithm that considers only correlations between m subsequent locations in a time series

1: $X' \leftarrow [X_{t-1}, ..., X_{t-m}]$
2: $X_{pred} = \arg\max_x \{Pr[X = x|X']\}$

The prediction is derived directly from the last m observations by maximizing $Pr[X_t = x|h_{(t-1):(t-m)}]$. Yet, the distribution $Pr[X_t = x|h_{(t-1):(t-m)}]$ needs to be known in advance. In algorithm 1 this distribution is obtained by assigning a weight to each subsequence in the entire location history with regard to similarity. Basically, this is what kernel density estimation does when being applied to the embedding space.
Considering the technique that is proposed in this thesis, an m-th order Markov process is assumed. Furthermore, kernel density estimation is used to estimate the unknown distribution $Pr[X_t = x|h_{(t-1):(t-m)}]$. The details of this algorithm are explained in the following sections.

[5] The complexity depends on the *argmax*-step.

6.3 Stationary Process

A necessary assumption for the prediction technique that is presented in this thesis is stationarity. A stationary process is a stochastic process whose probability distribution does not change over time. Relating to algorithm 2, this means that the distribution $Pr[X_t = x|h_{(t-1):(t-m)}]$ remains the same over time. Thus, the prediction algorithm described in this thesis relies on patterns in location time series data that do not change over time.

6.4 The Prediction Algorithm

Taking the above motivation into account, the prediction algorithm can be designed as shown in the following diagram:

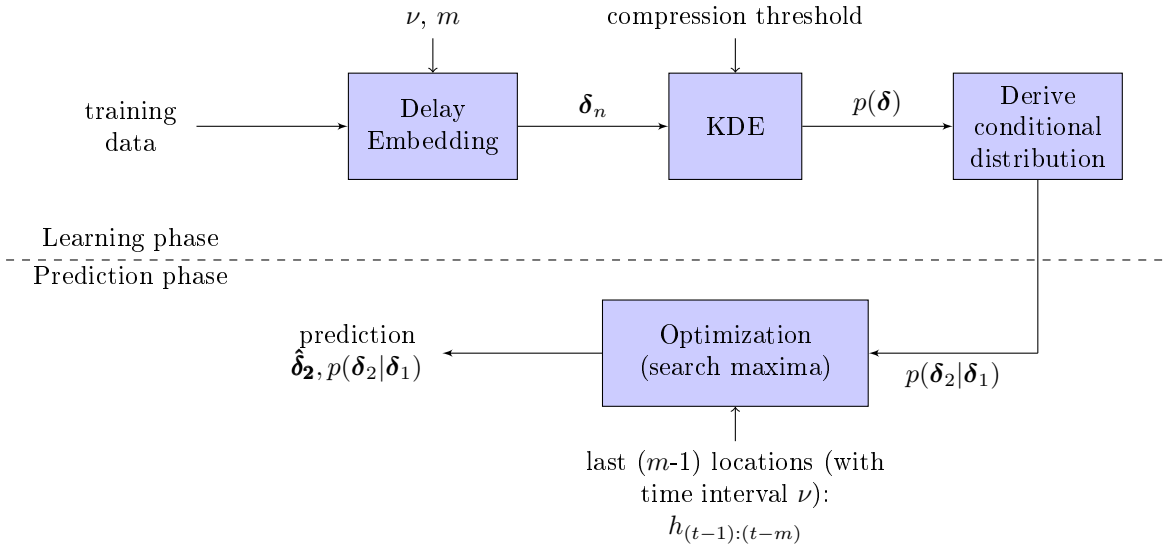


Figure 6.1: Block diagram of the prediction algorithm

As depicted in fig. 6.1, the prediction algorithm consists of four components that are arranged successively. During the learning phase, at first the given sample data is embedded in an m-dimensional space using the delay ν. Then a kernel density estimator is used to learn the joint probability density function $p(\boldsymbol{\delta})$ of the data in the embedding space. Finally the conditional density function is derived from $p(\boldsymbol{\delta})$ as described in section 5.1. After the learning phase the density distribution can be used for prediction. To be specific, the prediction is then the optimization of the conditional density function $p(\boldsymbol{\delta}_2|\boldsymbol{\delta}_1)$ with $\boldsymbol{\delta}_1$ ={delay embedded last (m-1) locations}, assuming an (m-1)-th order Markov process. Each of these steps is based on existing data mining and optimization techniques. In the following, the working principle of each step is explained[6] and particular properties or modifications with regard to this thesis are emphasized.

6.4.1 Delay Embedding

In section 4.1 delay embedding as a specific data representation for time series was explained. In the algorithm that is presented here, delay embedding is applied to location histories. More precisely, a location history $h_{t-1} = \{\mathbf{x_{t-1}}, \mathbf{x_{t-2}}, ..., \mathbf{x_1}\}$ with $(t-1)$ two-dimensional location coordinate vectors is given. Each coordinate vector $\mathbf{x_n} = (x_n^1, x_n^2)^T$ refers to a location that was observed at time $n \in \{1, 2, ..., t-1\}$. This history is embedded in the embedding space by

$$\boldsymbol{\delta}_n = [x^1_{n-(m-1)\nu}, x^2_{n-(m-1)\nu}, x^1_{n-(m-2)\nu}, x^2_{n-(m-2)\nu}, ..., x^1_{n-\nu}, x^2_{n-\nu}, x^1_n, x^2_n] \tag{6.4}$$

[6] For further details it is referred to the cited works.

with m being the embedding dimension and ν being the time delay. Thus, the embedding vector $\boldsymbol{\delta}_n$ contains the coordinates of m locations that were observed using a time interval ν. As eq. (6.4) shows, the coordinate values of each location are two subsequent elements in the embedding vector. This can be regarded as a slight modification of the common delay embedding technique where each element in the embedding vector is related to a different point in time.
In other words, the delay embedding vector is obtained by taking any m subsequent locations that have a time-lag of ν and writing them successively into a single vector of length $2m$. As mentioned in section 4.1, this data representation is used to map spatiotemporal similarity between two subsequences of a time series to Euclidean distance. In this way, it is possible to identify patterns by evaluating density in the delay embedding space.

6.4.2 Estimating a Probability Density Function

The prediction is based on a probability density function estimated from the observed location data. Hence, the extent to what the density function reflects the model that underlies the data is crucial for the final prediction quality. In this section the choice of the estimation algorithm is motivated and the chosen algorithm is described.

Requirements on the Estimation Algorithm

There are different requirements that were considered when choosing an appropriate algorithm for density estimation:

Continuous distribution: A continuous density function is obtained to avoid the discretization of locations.

Non-parametric: The algorithm does not rely on any assumptions concerning the probability distribution of the observed data.

Incremental: The distribution can be updated as new observations are made and updates don't need access to previous observations.

Constant memory footprint: The amount of memory used by the algorithm remains the same as new updates are made.

Gaussian Mixtures and Kernel Density Estimation

A well known approach to estimate a probability density function from given sample data is to assume a Gaussian mixture model as described in chapter 5. There are several different algorithms to approximate the parameters of a mixture of Gaussians (e.g. [19, 20]). However, the number of mixture components is a parameter that has to be defined in advance. Regarding location time series data, it can be very hard to predefine this parameter since it is proportional to the number of dominant patterns in the time series. Actually, this is something that needs to be estimated directly from the data. Several techniques have been proposed to learn the number of mixture components from the observed data (e.g. [21, 22]).
Another approach to fit a density function to given sample data is kernel density estimation (KDE) [23, 24]. KDE is a non-parametric approach as it does not make use of any predefined assumptions about the data distribution. The basic idea is to define a kernel function for each data point in the sample data. Then the KDE is the weighted sum of all kernel functions smoothed by a so called bandwidth. This way every data point is incorporated in the distribution and no prior information is needed. More formally the KDE is defined as

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} \Phi_H(x, x_i), \tag{6.5}$$

where $\Phi_H(x, x_i)$ is a kernel function and H is the related bandwidth matrix. A common choice for the kernel function is the Gaussian distribution. The only parameter that needs to be determined is the bandwidth H. The estimation of H is an optimization problem as it can be obtained by

minimizing the distance between the kernel density estimation and the true distribution of the data. However, in this case the actual data distribution is unknown. A common approach to specify the distance between a KDE and the unknown underlying distribution is the *asymptotic mean integrated squared error* (AMISE) [25, pp. 94-99]. Hence, the AMISE is minimized to obtain the optimal bandwidth.

Online Kernel Density Estimation

In [26] an online variant of kernel density estimation (oKDE) is proposed. This algorithm matches very well with the requirements specified above. It is an incremental approach that maintains a non-parametric model of the observed data and allows updating the model as new data-points arrive. Based on [26] a short description of oKDE is given in the following. For more details see the original paper.
The main idea of oKDE is to distinguish between the sample distribution $p_s(x)$ and the distribution estimated by KDE $p_{KDE}(x)$. The sample distribution is defined as

$$p_s(x) = \sum_{i=1}^{N} \omega_i \Phi_{\Sigma_{si}}(x - x_i), \tag{6.6}$$

where $p_s(x)$ is the distribution of the sample data. In the standard KDE approach the covariance matrices Σ_{si} would all be zero and the weights ω_i would all be $\frac{1}{N}$. In this case $p_s(x)$ consists of N Dirac-delta distributions and each Dirac-delta represents one data-point. Given $p_s(x)$, the KDE can be obtained by the convolution with the kernel function as

$$p_{KDE} = p_s(x) * \Phi_H(x), \tag{6.7}$$

where $\Phi_H(x)$ is a kernel function with the bandwidth H. Note that this equation is the same as eq. (6.5) if all $\Sigma_{si} = 0$ and all $\omega_i = \frac{1}{N}$. So this is only a different notion of eq. (6.5). Yet, it allows to compress the KDE. To be specific, the oKDE method maintains a compressed version of $p_s(x)$ and updates it as new data-points are added. Compression is achieved by approximating clusters of data points by single Gaussian distributions. The result of the compression is a new sample distribution $p_s(x)$ with not all Σ_{si} necessarily being zero.
The two important parameters for the oKDE algorithm are the compression threshold D_{th} and the forgetting factor F. The compression threshold defines the maximum error that is accepted during compression. To measure this error the unscented Hellinger distance is used. The forgetting factor allows to deal with non-stationary distributions. The factor defines the weight of new data-points compared to old ones. Regarding the prediction algorithm presented in this thesis, a stationary process is assumed and the forgetting factor is ignored. In fig. 6.2 the oKDE algorithm is summarized.

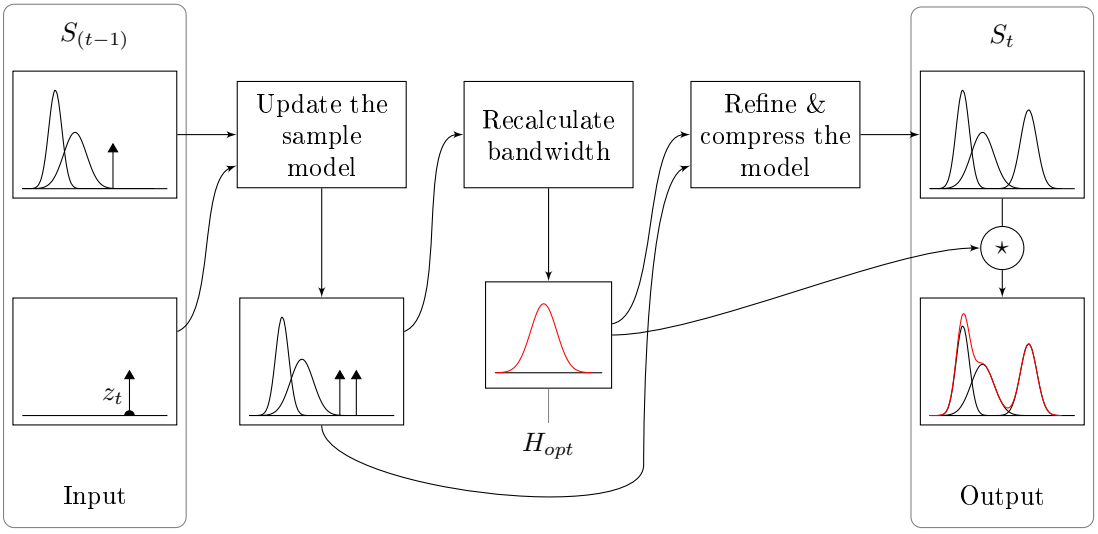


Figure 6.2: A summary of the oKDE algorithm based on a diagram in [26]

6.4.3 Prediction by Optimization

The output of the KDE is the probability density function of location time series data in the embedding space. It defines a probability value for each embedding vector. Thus, it provides information about the probability of observing an m-dimensional sequence of locations with ν being the time difference between two subsequent locations. In addition to the KDE, the prediction algorithm is provided with the current context in form of the (m-1) last observed locations. Based on this information it has to predict the next (m-th) location. As described in section 6.1, the most likely position is used for prediction. So the predicted location $\hat{x}_m$ is obtained by maximizing the probability of the location x_m given the previously observed locations $h_{(t-1):(t-m)}$ in form of a delay embedding vector $\mathbf{x_{m-1}}$:

$$\hat{x}_m = \arg\max_{x_m} \{p(x_m|\mathbf{x_{m-1}})\} \tag{6.8}$$

As the KDE approximates only the joint probability density distribution, the conditional distribution $p(x_m|\mathbf{x_{m-1}})$ has to be derived from $p(\mathbf{x})$ as explained in section 5.1. The final step is then the maximization of the conditional distribution using nonlinear programming (e.g. [27]). There are several approaches to search maxima of Gaussian mixture distributions based on nonlinear programming [28, 29]. A maximum of a probability distribution is called a *mode*. Gaussian mixture distributions are multimodal distributions since they can have multiple modes. Furthermore, multivariate Gaussian mixtures with M components can have more than M modes [28]. This is a challenging property since the available optimization methods rely on the knowledge of the number of modes. For instance, in [28] a method is proposed that starts a search from each component of a Gaussian mixture distribution to find its modes. The author states that this approach has been tested extensively and usually finds all modes of the distribution if there are at most as many modes as components. Though, regarding the distribution that is used for location prediction in this thesis, the number of modes is not limited by the number of components. Therefore, using the components of the model as starting points may not be a reasonable approach. Actually, there is no meaningful heuristic that helps define the starting points. Hence, a search region is defined and the starting points are chosen using a consistent segmentation of the search region. Moreover, the properties of location time series data allow to derive a meaningful definition of the search region. To be specific, a circular area around the last observed location is used as the search region. This is suggested by the fact that there is a maximum distance that can be covered by a human individual during the prediction interval. Of course, this distance depends on the possible means of transportation and has to be defined with regard to a practical use case. There are several possible segmentations of the resulting circular search region. Regarding the prediction algorithm that is proposed in this thesis, the segmentation is defined as shown in fig. 6.3 since it is simple and provides a quite uniform distribution of the starting points. Using these starting points, a quadratic search as described in [28, p. 3] is used to find local maxima. The greatest found maximum is chosen as prediction.

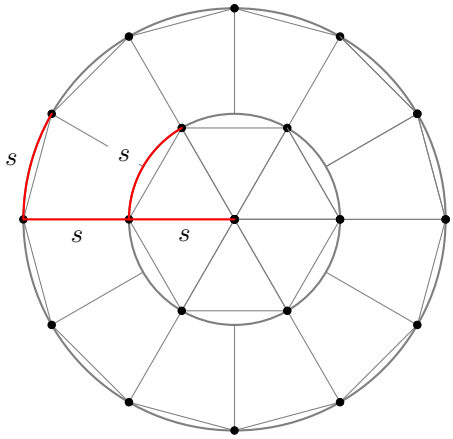


Figure 6.3: Segmentation of a circular search region using the segmentation parameter s that defines a distance

6.4.4 Reliability Measure

Since the algorithm is based on a probability density function, it can easily assign a reliability value to a particular prediction. To be specific, the reliability as implied by the learned density function is just the cumulative probability around the predicted location. Given an accuracy radius r, the algorithm can calculate the probability $\hat{p}_r$ of the actual location being within a radius of r around the predicted location. In the implementation that is part of this thesis, the cumulative probability is approximated using the trapezoidal rule. More precisely, the probability density function is evaluated at the predicted location as well as at a certain number of points on a circle around this location. Then the trapezoidal rule is applied using these points. Therefore, the approximation gets worse as the radius of the circle increases. However, for sufficiently small radii the provided accuracy is acceptably small (see section 7.6.3).

6.4.5 Integration of Additional Attributes

The described prediction algorithm uses a density distribution of the delay embedded location time series data. Therefore, it is not possible to directly integrate additional information apart from location coordinates into the prediction model. As described in section 4.1, similar location sequences are mapped to points that are close in the delay embedding space by means of Euclidean distance. If an additional attribute with a different range and scale is added to the delay vectors, this attribute can distort the distance measure. More specifically, the scale can be regarded as a weight for the attribute regarding the Euclidean distance. Considering the algorithm as presented here, coordinates are scaled using UTM. Thus, a distance of $1 unit$ between two points in the model reflects a distance of approximately 1m in the real world. Now assume that the model shall be modified to incorporate for instance correlations between observed locations and the time of day. This poses the problem how to scale the time of day in order to obtain meaningful prediction results. This is a question that can not be generally answered as it depends strongly on the data. So, to define a scaling, an appropriate heuristic needs to be found by examining the data. As this is beyond the scope of this thesis it will not be further discussed. However, this is an interesting starting point for future investigations.
The algorithm presented and tested in this thesis is solely based on correlations between subsequent values in location time series. Thus, it disregards any additional attributes that may be associated with the location data. Nevertheless, such attributes and correlations are taken into account when analyzing the test results in section 7.6.

Chapter 7

Testing

To verify the proposed location prediction method it shall be tested using real-world data. Therefore, the proposed algorithm has been implemented as a Java program (see appendix B) and several tests have been carried out. In the following, the used testing methodology is described and the observed test results are discussed.
The general approach used for testing the prediction algorithm is cross validation [30]. Thus, the available test data is split into parts for learning and for testing. This is described in detail in section 7.2.

7.1 Testing Metrics

Clearly, the performance of a prediction algorithm can be measured by letting the algorithm predict a known value and then compare the predicted value to the expected outcome. The difference between the predicted and the actual value is referred to as the absolute prediction error. Actually this is the basic measure used to asses the prediction algorithm in the following.
Furthermore, as described in section 6.4.3, the output of the prediction algorithm is a location $\hat{x}$ and an associated reliability p_r. The reliability refers to the accuracy radius r that is defined as an input parameter for the algorithm. Hence $p_r = \hat{p}$ means that the true location lies with a probability of $\hat{p}$ within a radius of r around the predicted location $\hat{x}$. Based on this output, the distribution of the absolute prediction error and the claimed reliability can be statistically verified using a test set containing sufficiently many location time series data. For this purpose, prediction queries with known answers are constructed from the test data. Then the absolute error and the average success rate with regard to the demanded accuracy can be measured and compared to the average reliability claimed by the algorithm.

7.2 Test Data

Because of privacy concerns it is very difficult to gain access to data sets containing location traces of human individuals. Thus, an appropriate alternative needed to be found. To test the proposed prediction algorithm, a location data set with a sufficiently high density of measuring points was needed. In addition, the process that underlies the data should resemble human mobility.
The *cabspotting* data set [31] provides a large amount of well structured location time series data. The location histories in this data set correspond to the whereabouts of taxicabs in San Francisco, USA. To be precise, it contains GPS coordinates of 563 taxis that have been collected in 30 days in San Francisco. On average the interval between two subsequent measurements is less than 60 seconds. The data set consists of 563 files and each file contains data points of a single taxi. Each data point is specified by a GPS coordinate pair and a related timestamp as well as a bit indicating if the taxi is occupied. As no appropriate mobility data set of human individuals is openly available to the best of my knowledge, the *cabspotting* data is a reasonable alternative. Naturally, the taxi data has some particular statistical properties that are unlikely to be observed in mobility traces of

human individuals. For instance, in contrast to human individuals, taxis stay within some bounded geographic area and they only move along roads or highways. Nevertheless, the *cabspotting* data set provides a good basis for testing the proposed prediction algorithm. Actually, the tests that are described in the following were executed using ten particular files of the taxi data set. These files were chosen since they provide a high density of data points over time and an acceptable amount of measurement errors. In order to consume an acceptable amount of computing time, the number of test files was limited to ten.[7]

7.2.1 Recurrence

As described in section 6.4.3, the proposed prediction technique relies on recurring patterns in the given location traces. So, the quality of the test results will reflect the periodic properties of the test data, assuming the correctness of the prediction algorithm. Actually, it seems to be reasonable to assume that the taxi data provides a certain degree of recurrence. Generally, taxis tend to use similar routes to reach certain destinations. Furthermore, taxis customarily frequent a few points very often looking for potential clients. For instance, such locations can be airports, train stations or shopping malls. Indeed, one can identify points that seem to be frequented more often than others by examining the spatial density distribution of the taxi data. In particular, the downtown area, as well as the international airport provide a very high density of data points. This is exemplarily illustrated in fig. 7.1. The degree of recurrence provided by the taxi data set is further examined during the result analysis in the following sections.

7.2.2 Measurement Errors

Another property of the taxi data set that needs to be considered is measurement uncertainty. In order to detect measurement errors in the location data a thresholding method is used. Thus, the filter exploits the fact that speed and acceleration of a taxi are limited. The current average speed of a taxi is calculated using each pair of two subsequent measurements. Then, based on the speed differences, the average accelerations are calculated. Given the speed and acceleration values, a thresholding filter can be applied to remove measurement errors. The results of this filtering method are exemplary shown in fig. 7.4. Actually, there are some taxi files that contain significantly more measurement errors than the average amount[8]. This could be the result of misplaced or faulty GPS equipment and is dealt with during preprocessing (section 7.4).
Unfortunately, the bits indicating the occupation of a taxi also seem to be unreliable as they occasionally flip for a few seconds. However, during the result analysis in the following sections, the average trip time is a useful value. Therefore, a simple filter was applied to the occupation data in order to remove the major part of measurement errors. More specifically, the filter considers a change of the occupation bit to be a measurement error if the average speed at the particular data point is greater than 1m/s (= 3.6km/h). This filtering approach is motivated by the fact that a taxi needs to slow down and finally stop to deposit or to pick up a client. Yet, after filtering there are still a considerable amount of trips with a duration of shorter than 60 seconds in the data. As it is very improbable that a taxi trip takes less than 60 seconds these trip times are most likely incorrect. Therefore, the average trip times (see table 7.2) that were calculated based on this data should still be regarded as approximate values providing a low accuracy.

[7] One test run using a single file and one particular parameter combination took approximately one hour on average.
[8] Up to 30% of measurement errors have been observed in a taxi file.

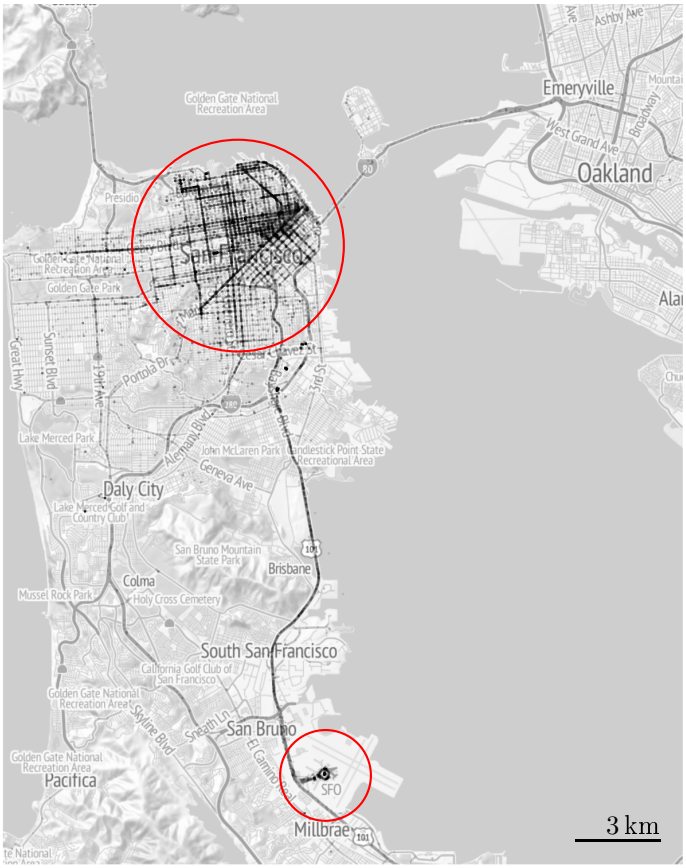
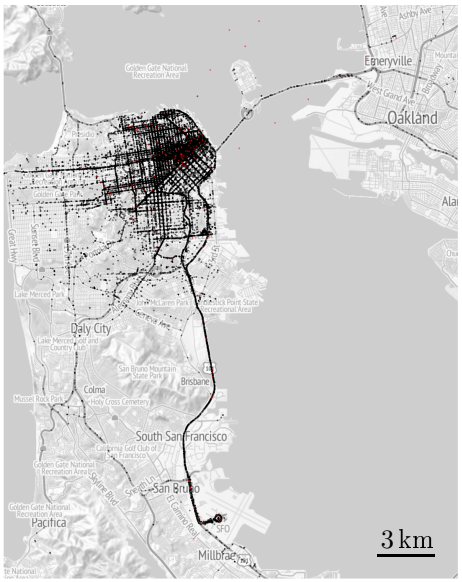
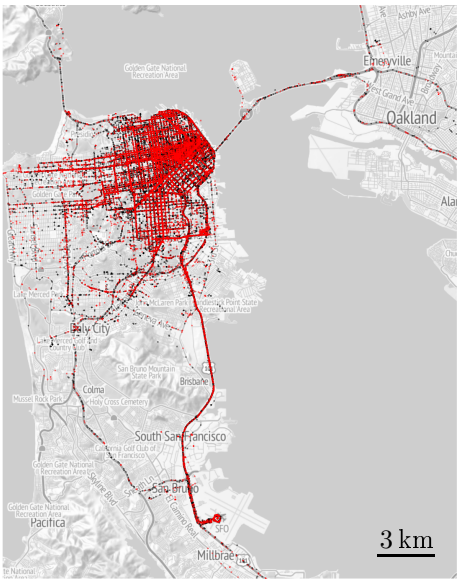


Figure 7.1: This plot of a taxi mobility trace visualizes the density of data points by opacity. The downtown area (upper circle) and the airport (lower circle) provide particular high density values.



(a) 0.6% noise



(b) 33% noise

Figure 7.2: In these spatial plots of taxi mobility traces red points represent measurement errors. The considerably larger amount of errors in the right trace suggests faulty measurement equipment.

7.3 Test Procedure

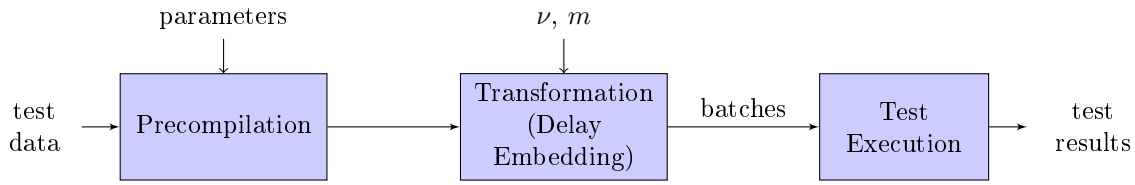
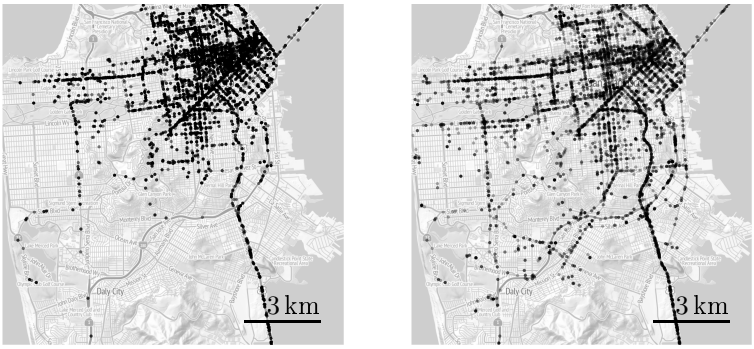


Figure 7.3: Block diagram of the test procedure

The test procedure consists of three main steps (see fig. 7.3). At first, the test data is preprocessed using several filters. Then the preprocessed data is embedded in an m-dimensional space using the delay ν (see section 4.1). The result of the delay embedding are vectors that contain m subsequent locations $\mathbf{x} = \{x^1, ..., x^m\}$. In the following, these vectors will be referred to as *location batches*. One portion of the obtained location batches is used for training and another portion is used for testing. The actual number of batches per taxi used for training is 3000. For testing, 500 batches per taxi are used. During the testing phase the first m-1 locations of each batch are used as input for the algorithm and the m-th value serves as a reference value for the prediction verification.

7.4 Preprocessing

As visible in the geographic plots, the taxis actually move within the downtown area most of the time. Thus, an algorithm that always predicts the taxi to be in the center of this area would generally provide quite good results. So, prediction tests that are directly based on the taxi data may produce insignificant results since prediction can be a trivial task due to low spatial variance. To deal with this fact, the delay embedded location data was filtered before testing in order to increase the spatial variance. Strictly speaking, location batches with a total travel distance being smaller than a certain threshold λ were excluded. Given the value of λ is sufficiently large, the filtered data exhibit a more uniform spatial distribution. In fig. 7.4 the effect of the distance filter is exemplarily shown.
Another part of preprocessing is a filter that removes errors induced at measuring time. Namely, a thresholding filter as described in section 7.2 is used. Particularly, taxi files that contain more than 5% of erroneous measurements are entirely excluded from the test data assuming faulty measurement equipment.



(a) No distance filter used. (b) After applying distance filter.

Figure 7.4: Two samples of the same size from the same taxi trace file. The left sample was extracted without distance-related filtering. The right sample contains only location batches with a total travel distance greater than 7000m, resulting in a higher spatial variance.

7.5 Verification of the Implementation

Before actually testing the prediction capabilities of the proposed algorithm with regard to the given test data, the implementation was generally verified using an *in-sample* verification. For this purpose, the algorithm was given the same data set for the learning phase and for prediction. To be specific, 500 location batches were extracted from each of the ten test files. Using this data, one exemplary test run was executed with the delay embedding parameters set to $m = 3$ and $\nu = 6$min. Thus, for each test file the algorithm was executed once using the extracted location batches as training and prediction test data. As shown by the cumulative error distribution given in fig. 7.5, the results provide very high accuracies. Namely, more than 60% of the predictions achieve an error smaller than 500m. This indicates that the implementation of the algorithm is generally functional. More specifically, the algorithm seems to remember previously observed location sequences and successfully uses them to predict future locations. This is of course no meaningful test for the prediction technique as it does not show if the algorithm can deal with new data. However, it is a useful evidence for the overall correctness of the implementation. In the following, the actual results of the *out-of-sample* tests as described in section 7.3 are presented.

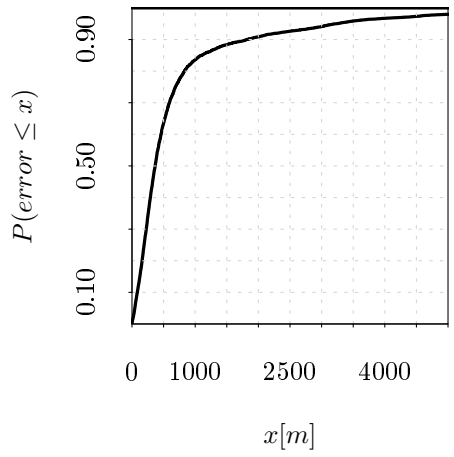


Figure 7.5: In-sample verification: The cumulative distribution of the absolute error observed using $m = 3$ and $\nu = 6$min

7.6 Test Results

As described in chapter 6, the prediction algorithm takes several parameters that need to be defined with respect to a particular use case. The parameters that affect the accuracy of the applied approximate techniques were set to fixed values for all tests. More specifically, they were defined such that the runtime of the algorithm was within an acceptable range. Since the actual values of these parameters are not of much interest regarding the test analysis, they will not be listed here but can be found in appendix A.1. Rather, the test analysis focuses on the impact of the delay embedding parameters m and ν. Furthermore, it is investigated what particular properties of the test data are reflected in the prediction test results.
To examine the predictive capabilities of the algorithm regarding the near as well as the more remote future, different prediction intervals were tested. Actually, the interval ν varies from 3.3 to 60 minutes. The value of m that defines the assumed Markov order of the underlying process is varied between 2 and 5. Regarding the preprocessor, the minimum distances λ used for filtering the input batches are defined with regard to the total time span covered by one batch $t_{total} = m\nu$. However, they are also bounded by the data set characteristics. The same number of samples for learning and testing were used for all tests. Namely, 3000 samples were used for learning and 500

samples were used for testing. These numbers were chosen with regard to some previously executed test runs. In the following, the test results are presented and analyzed with respect to the delay embedding parameters.

7.6.1 Impact of the Time Delay

In this section the impact of the time delay (ν) used for embedding is examined in detail. At first, the relevant test results are presented and then these results are analyzed taking into account the test data.

Observed Results

A particular group of test sets listed in table 7.1 is considered. These test sets were all executed with a fixed embedding dimension $m = 3$. Only the delay parameter was varied from 3.3 to 60 minutes. Regarding this table, the prediction accuracy obviously decreases as the time delay increases. To be specific, the average error induced by the predictor as well as the variance and the median of the error seem to depend on the time delay. Interestingly, there seems to be some saturation point between $\nu = 10$min and $\nu = 30$min. More precisely, increasing ν beyond 30 minutes does not remarkably worsen the prediction. Similar results are observed using an embedding dimension of $m = 4$ as shown in appendix A.2.

ν [min]	t_{total} [min]	ME[m]	RMSE[m]	Median[m]
3.3	9.9	1130	1585	802
6	18	2252	3153	1561
10	30	4079	5853	2452
30	90	5905	9187	2655
60	180	5911	9314	2776

Table 7.1: Results of five test sets with fixed $m = 3$ and varying ν from 3.3 to 60 minutes.

A more sophisticated perspective on the observed absolute prediction errors in these test sets is provided by the cumulative error distributions given in fig. 7.6. This plot visualizes the portion of predictions providing an absolute error smaller than a certain distance x. Taking this plot into account, the above impressions are confirmed. It shows quite clearly, that the prediction accuracy does not decrease much for delays greater than 10 minutes. However, predictions based on smaller delays provide considerably better results. To understand this effect the spatial distribution of prediction errors is taken into account. Figure 7.7 illustrates the spatial distributions of the absolute prediction errors. In this graphic the actually observed locations of the taxis are plotted. The color of each point indicates the error that was made by the predictor when trying to predict the particular location. Obviously, the good predictions are almost uniformly distributed when using a small delay ($\nu = 3.3$min). In contrast, the predictions providing a small error are more and more clustered around two locations as the delay increases. Actually, in the plots for $\nu \geq 30$min they are clearly clustered around the downtown area and the airport area.

Interpretation

To explain this effect, the properties of the data set as well as the working principle of the prediction algorithm need to be considered. The algorithm assumes successively observed locations to be correlated. The degree of correlation is measured by recurrence. To put it simple, a certain amount of similar observed location sequences are assumed to be a pattern that can be used for prediction. As described above, taxis tend to use the same routes to approach certain locations. Therefore, the mobility trace of a taxi provides a considerable amount of short-term recurrences. However, long-term recurrences are probably very rare since the destinations the taxis are heading for mainly depend on the clients' demands. To be specific, it seems reasonable to assume that subsequently observed locations in the mobility trace of a taxi are only correlated if they are part

of the same trip. Each time a new trip starts a new destination is defined by the new client.[9] Whereas, during a particular trip the destination does not change. Based on this argumentation, a significant amount of recurring similar location sequences can only be observed in the taxi data when limiting the overall time covered by each sequence to the average trip time.
This gives a convenient explanation for the observed clustering effect and matches with the observed test results. More specifically, the observed saturation for $\nu > 10$min can be explained by the average length of a taxi trip. In table 7.2 the average trip times as well as the related variance values are listed for the taxi data files that were used for testing.[10] Considering these values it can be assumed that most taxi trips in the test data take less than half an hour. Thus, in most cases three subsequently observed locations with a time interval of 10 minutes are not part of the same trip. Therefore, they can not be expected to be correlated and to uncover meaningful mobility patterns. Hence, a predictor that uses an embedding dimension of $m = 3$ and a delay time of $\nu > 10$min is not able to make meaningful predictions based on the taxi data set. Actually, it assumes a second order Markov process to underlie the data whereas the true Markov order is lower for the major part of the data. So the predictor falls back to a predictor with $m = 2$ in many cases. This explains the spatial clustering with regard to the prediction accuracy. Assuming a first order Markov process allows considering the current location to predict the next one. This permits a more sophisticated guess than just relying on the spatial frequency distribution[11] but suffers from not being able to recognize trends or directions. Therefore, it seems to be reasonable that a predictor with $m = 2$ will tend to predict exclusively a few very frequently observed locations. So the clustering effect observed when increasing the delay time beyond $\nu = 6$min reflects the fact that the predictor often has to base its predictions on a second order Markov process.

No.	Average of Trip Time[min]	Variance of Trip Time[min]
1	17	8
2	15	6
3	14	6
4	19	10
5	15	7
6	13	9
7	13	8
8	14	6
9	21	842
10	15	10

Table 7.2: The average trip times of the taxis used for testing. The values of taxi no. 9 are definitely unreliable since the variance is implausibly large.

[9] Here it is assumed that the destinations that are demanded by the clients are distributed randomly.
[10] These values were calculated using the occupation bits in the taxi data after applying a filter as described in section 7.2.2. They should be regarded as approximate values as the data seem to be very unreliable.
[11] A predictor that assumes zeroth order Markov process would always predict the single most frequently observed location.

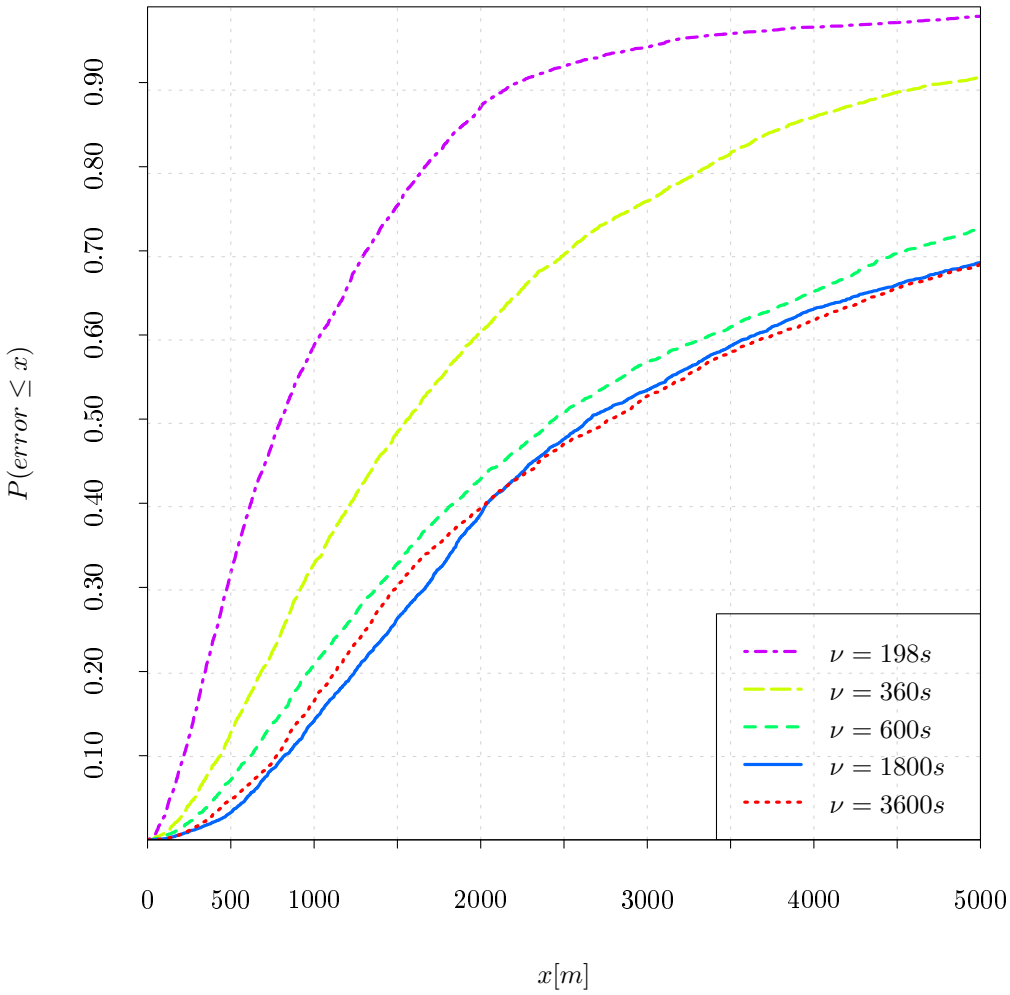
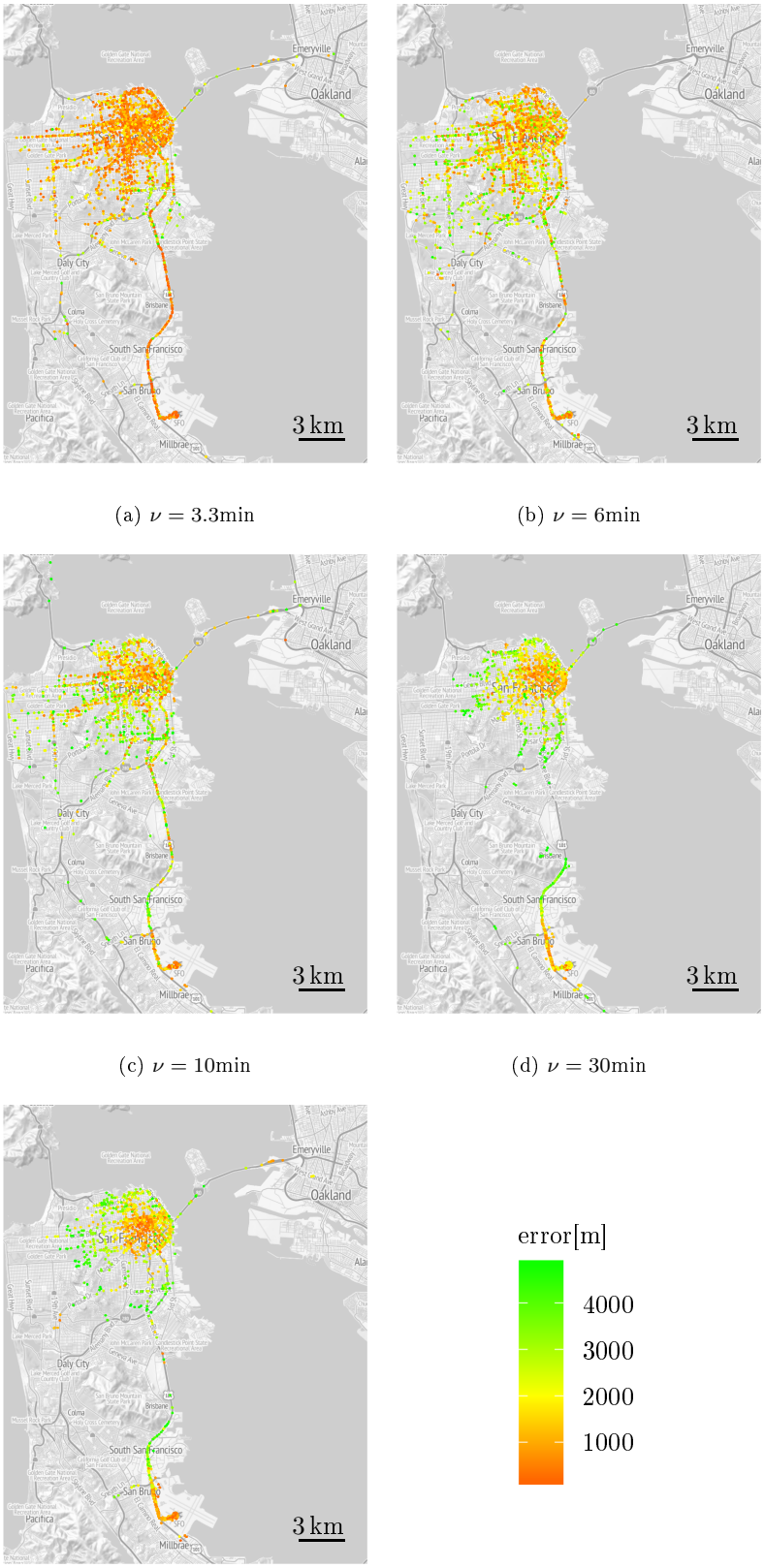


Figure 7.6: The cumulative distributions of the absolute error observed with fixed $m = 3$



(a) $\nu = 3.3$min

(b) $\nu = 6$min

(c) $\nu = 10$min

(d) $\nu = 30$min

(e) $\nu = 60$min

(f) Color legend

Figure 7.7: Spatial distribution plots of the absolute error observed when using a fixed embedding dimension $m = 3$. The more accurate predictions are clustered around two locations as ν increases.

7.6.2 Impact of the Embedding Dimension

The embedding dimension m defines the order of the Markov process that is assumed to underlie the observed location data. In this section the impact of this parameter is examined in detail. Again, at first the relevant test results are presented and then these results are analyzed with regard to the test data.

Observed Results

To examine the impact of the embedding parameter m on the prediction accuracy several test sets were executed. Table 7.3 shows the results of four test runs. During these tests the embedding delay was set to $\nu = 6$min. The embedding dimension m was varied from 2 to 5.[12] Considering these results, a higher embedding dimension generally provides lower average absolute prediction errors and smaller variances. However, the mean error as well as the root mean squared error do not change significantly for $m \geq 3$. Particularly, the test sets with $m = 4$ and $m = 5$ provide very similar results. Moreover, the average error observed with $m = 5$ is even slightly greater than the average error observed with $m = 4$. This indicates that there is a saturation point at $m = 3$ or $m = 4$.

m	t_{total} [min]	ME[m]	RMSE[m]	Median[m]
2	12	4018	6047	2438
3	18	2252	3153	1561
4	24	2102	2883	1535
5	30	2146	2981	1528

Table 7.3: Results of four test sets with fixed $\nu = 6$min and varying m from 2 to 5.

The same development is visible in the cumulative distribution plot given in fig. 7.8. This graphic reveals that the four different error distributions look very similar for 0m $\leq x \leq$ 1000m. However, beyond the point $x = 1000$m, the slope of the $(m = 2)$-distribution considerably decreases. So the prediction with $m = 2$ induced apparently more large errors than the predictions with greater embedding dimensions did. The variance that is indicated by the root mean squared errors in table 7.3 is illustrated in fig. 7.9 by the density plots of the prediction errors. These plots show that the error of the prediction results observed when $m = 2$ has two clearly discernible maxima at approximately $x = 1000$m and $x = 4100$m. In contrast, the density distributions for $m \geq 3$ have a single significant maximum[13] at approximately $x = 1000$m. Furthermore, the spatial error plots shown in fig. 7.10 are considered in the following interpretation of these results. Similar to what was observed when varying the time delay ν, the plot for $m = 2$ indicates a clustering of more accurate predictions around the downtown and the airport area. Whereas, in the plot for $m = 3$ a clustering is hardly visible.[14]

[12] The compression threshold for the oKDE and was slightly varied as well to decrease the computing time (see appendix A.1).

[13] The other local maxima are not taken into account since they are comparatively small.

[14] The spatial plots for $m = 4$ and $m = 5$ are not included here since they look very similar to the plot for $m = 3$. They can be found in appendix A.2

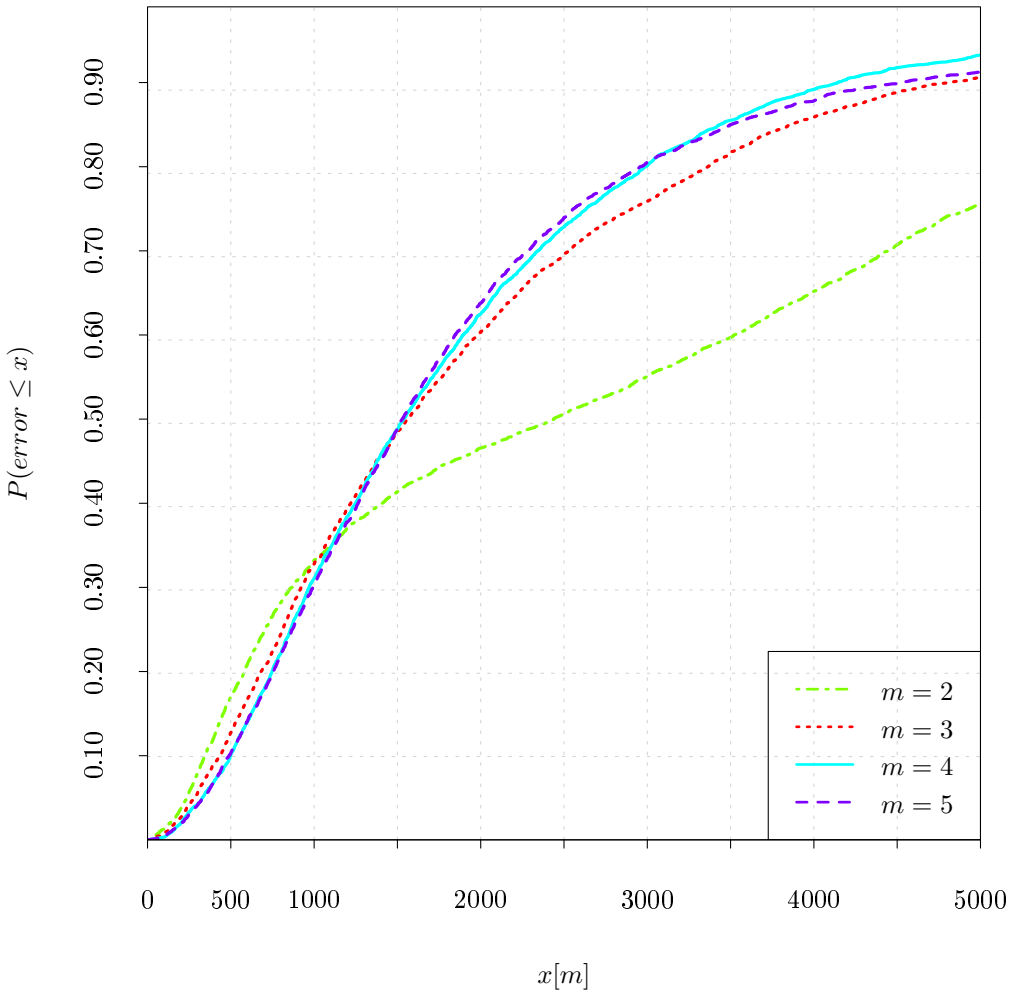
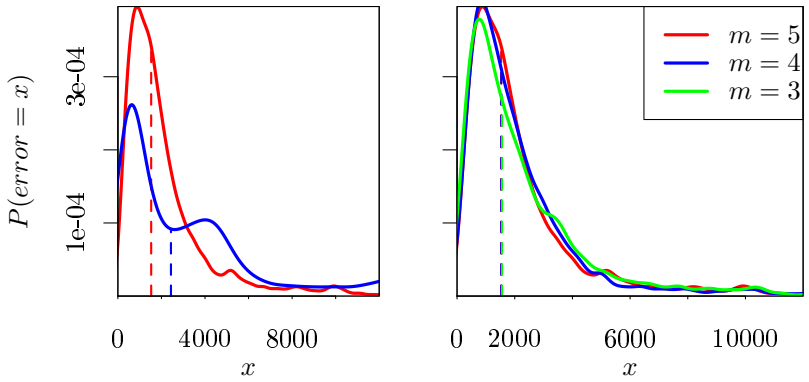


Figure 7.8: The cumulative distributions of the absolute error observed with fixed $\nu = 6\text{min}$



(a) $m \in \{2, 5\}$ (b) $m \in \{3, 4, 5\}$

Figure 7.9: Density distributions of the absolute error observed for $\nu = 6\text{min}$. The dashed lines indicate the medians. The distributions were obtained using kernel density estimation.

(a) $m = 2$ (b) $m = 3$ (c) Color legend

Figure 7.10: Spatial distribution plots of the absolute error observed when using a fixed embedding delay $\nu = 6\text{min}$. More accurate predictions tend to be clustered around two locations when setting $m = 2$.

Interpretation

The results described above match well with the results discussed in the previous section. As also argued above the taxi data provides mostly short-time recurrences. This explains the saturation observed when increasing the embedding dimension m defining the order of the assumed underlying Markov process. Following the above argumentation, increasing m beyond $m = 3$ while setting $\nu = 6\text{min}$ would only provide more accurate predictions if the average trip time in the taxi data were greater than $24\text{min} (= 6\text{min} \cdot 4)$. Else it would fall back to a prediction with $m = 3$ in many cases. Actually, this is observed in the density and cumulative distribution plots of the absolute error.
The clustering effect observed for $m = 2$ was explained in the previous section, too. It results from the fact that the second order predictor uses only the current location to predict the next one and is not able to recognize trends or directions.
Summarizing, the results indicate that increasing the order of the assumed underlying Markov process provides more accurate results. However, the Markov order is certainly limited by the actual stochastic process that underlies the data and depends on the delay time ν.

7.6.3 Reliability Verification

As explained above, the prediction algorithm provides a reliability value for each prediction output. This value defines the approximate cumulative probability that is calculated using a circle with radius r around the predicted location. Hence, the algorithm claims that the predicted location is with a certain probability $\hat{p}_r$ within the radius r. To examine the correctness of this claimed reliability different test sets were executed. At first, the reliability radius was set to $r = 1000\text{m}$ and different embedding parameters were used. Secondly, the radius was varied using fixed embedding parameters $\nu = 6\text{min}$ and $m = 3$. Table 7.4 shows the average claimed and the statistically evaluated reliability values provided by the test results. The latter is referred to as the *actual reliability*. Obviously, for $m \geq 3$ and $r = 1000\text{m}$ the reliability claimed by the algorithm only slightly differs from the actual reliability. Actually, the maximum difference is 15% with regard to the actual reliability. When using an embedding dimension of $m = 2$, the claimed reliability deviates at least by 34% from the actually observed reliability. This matches with the above observations regarding the prediction accuracy. For the reasons explained above, the density estimated using $m = 2$ provides a considerably worse approximation of the true underlying distribution than density estimations based on greater embedding dimensions. Consequently, the estimated reliability is less accurate, too. In addition, when the accuracy radius is increased, the algorithm tends to

overestimate the reliability. For $r \geq 1500$m the deviation is greater than 26%. The main reason for this is most likely the trapezoid rule that is used to calculate the reliability. As explained in section 6.4.4 this approximation gets worse when increasing the radius. So, a different numerical integration technique or different approximation parameters would probably yield more accurate results for radii greater than 1000m.
In summary, these test results indicate that the prediction algorithm provides useful reliability values with an accuracy of up to 98.75% for radii smaller than 1000m. This confirms the presumption that the proposed algorithm is able to properly reflect the actual stochastic process that is underlying the given mobility data.

m	Claimed reliability	Actual reliability	Percentaged Deviation
2	20.80%	33.20%	37.35%
3	33.22%	32.81%	1.25%
4	35.66%	31.05%	14.85%
5	33.95%	30.30%	12.05%

(a) $\nu = 6$min, varying m

m	Claimed reliability	Actual reliability	Percentaged Deviation
2	13.98%	21.44%	34.79%
3	18.49%	20.83%	11.23%
4	18.77%	18.07%	3.87%

(b) $\nu = 10$min, varying m

r	Claimed reliability	Actual reliability	Percentaged Deviation
500	12.58%	12.54%	0.32%
1000	33.22%	32.81%	1.25%
1500	60.23%	47.59%	26.56%
2000	96.88%	59.37%	63.18%

(c) $\nu = 6$min, varying r

Table 7.4: Comparison of claimed and actual prediction reliabilities for varying embedding dimensions (a, b) and varying radii (c). The percentaged deviation in the last column was calculated in terms of the actual reliability.

Chapter 8

Conclusion and Future Work

The central hypothesis of this thesis as specified in chapter 2 states that an appropriate algorithm is able to extract recurring patterns from location histories and use them to predict future locations. The proposed algorithm was shown to underpin this hypothesis. Summarizing, the algorithm uses an online kernel density estimation technique [26] to obtain a probability distribution of the delay embedded location time series. The resulting mixture of Gaussians is transformed into a conditional distribution and then quadratic optimization [28] is used to provide a prediction. This algorithm was tested using the *cabspotting data set*[31] and the tests provided meaningful results. To be specific, the test results indicate that the prediction algorithm proposed in this thesis is able to extract existing patterns from location time series data by means of a continuous probability density function. Furthermore, it is shown that the algorithm can exploit these patterns in order to predict future locations. For instance, the algorithm achieves an accuracy better than 1000m in approximately 32% of the executed tests using a prediction interval of 6 minutes. Moreover, in 13% of these tests the prediction error is smaller than 500m. As expected, the test results clearly demonstrate that the prediction capability of the algorithm strongly depends on the properties of the used location data. Since the algorithm bases its prediction on recurring patterns the degree of recurrence in the data limits the algorithm in its ability to provide accurate predictions. Moreover, the statistic evaluation of the tests showed that the reliability of the predicted locations as claimed by the algorithm generally match well with the actual reliability. This confirms the proposed algorithm in its ability to properly reflect the actual stochastic process that is underlying the given mobility data. Compared to previous work, this thesis uses a more generic approach. In particular, the prediction interval is varied and the algorithm does not assume a finite set of discrete locations.

As the results presented in this thesis are quite promising, a further analysis of the proposed prediction technique using different data sets is expected to provide interesting results. In addition, there are several imaginable approaches to further improve the prediction algorithm. Currently, the algorithm is not able to directly integrate additional attributes that are associated with the coordinate pairs in a location time series into the prediction model. Though, often several such attributes are available. To add an additional attribute to the delay embedding space used by the algorithm, clearly it needs to be mapped to one or several scalar values. This can be a trivial task. However, as described in section 6.4.5, it is crucial to define an appropriate scale for the attribute as the scale assigns a weight to it.

Another approach to obtain an improved prediction technique based on the proposed algorithm is to combine multiple predictors each using different embedding parameters.[15] Then, the prediction providing the highest reliability or a weighted combination of all predictions can be used. This approach takes into account that recurrent patterns in location time series data may generally vary in scale. Thus, multiple predictors using varied embedding parameters could exploit different patterns in the data. Furthermore, one can think of techniques to iteratively optimize the embedding parameters with regard to the data. Yet, this would probably require a considerable amount of additional training data.

[15] In the field of classification this is called *boosting*.

Appendices

Appendix A

Additional Test Parameters and Results

Here, a detailed listing of the parameter values used for testing is given. Moreover, some additional results are presented.

A.1 Test Parameters

Component	**Parameter**	**Value(s)**
Density estimation (oKDE)	forgetting factor	1 ($\hat{=}$ disabled)
	compression threshold	0.03 (for $m \leq 4$) 0.05 (for $m = 5$)
Optimization (Prediction)	search region radius	20000m
	segmentation parameter s	500m
	no. of segments for reliability estimation	10

Table A.1: Values used for test parameters that determine the accuracy of approximations during density estimation and optimization

A.2 Test Results

The following tables show some additional test results characterized by mean error, root mean squared error and median error in meters.

m	t_{total} [min]	ME[m]	RMSE[m]	Median[m]
2	20	6507	9292	3885
3	30	4079	5853	2452
4	40	3724	5287	2373

Table A.2: Test results for $\nu = 10$min

m	t_{total} [min]	ME[m]	RMSE[m]	Median[m]
3	90	5905	9187	2655
4	120	6381	9728	3080
5	150	6541	10061	3238

Table A.3: Test results for $\nu = 30$min

m	t_{total} [min]	ME[m]	RMSE[m]	Median[m]
6	24	2102	2883	1535
7.5[16]	30	2984	4313	2041
10	40	3724	5287	2373
30	120	6381	9728	3080

Table A.4: Test results for $m = 4$

(a) $m = 4$ (b) $m = 5$ (c) Color legend

Figure A.1: Spatial distribution plots of the absolute error observed when using a fixed embedding delay $\nu = 6$min. For increasing m, less clustering is observed(compare fig. 7.10).

[16] In this case only *four* test files were used. All others were executed using ten test files as described in section 7.2.

Appendix B

Contents of the Attached DVD

The DVD that was added to this thesis contains this thesis as a pdf-file as well as the Java source code of the implemented prediction algorithm and the related Javadoc-documentation. Furthermore, all output files containing the discussed test results were added. The folder structure on the DVD is as follows:

- Thesis: this thesis as a pdf-file
- Implementation

  SourceCode: Java source code (two separate projects: *oKDE*, *PrePos*)

  Executables: executable jar-files and test scripts

  Documentation: compiled Javadoc files
- Testing: test output files
- README.txt: Instructions how to execute the Java files and further information.

Bibliography

[1] B. Greßmann, H. Klimek, and V. Turau, "Towards ubiquitous indoor location based services and indoor navigation," in *Proceedings of the 7th IEEE Workshop on Positioning Navigation and Communication (WPNC2010)*, Dresden, Germany, 2010.

[2] H. Klimek, B. Greßmann, and V. Turau, "Indoor navigation and location based services scenario for airports," Hamburg, Germany, 2011.

[3] M. C. González, C. A. Hidalgo, and A.-L. Barabási, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, Jun. 2008.

[4] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010.

[5] B. Jiang, J. M. Yin, and S. Zhao, "Characterizing the human mobility pattern in a large street network," *Physical Review E*, vol. 80, no. 2, pp. 1095–3787, Aug. 2009.

[6] S.-M. Qin, H. Verkasalo, M. Mohtaschemi, T. Hartonen, and M. Alava, "Patterns, entropy, and predictability of human mobility and life," *PLoS ONE*, vol. 439, no. 7075, pp. 462–465, Dec. 2012.

[7] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive wi-fi mobility data," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, Mar. 2004, pp. 1414–1424 vol.2.

[8] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "NextPlace: a spatio-temporal prediction framework for pervasive systems," in *Proceedings of Ninth International Conference on Pervasive Computing*, ser. Pervasive 2011, San Francisco, Jun. 2011, p. 152–169.

[9] S. Akoush and A. Sameh, "The use of bayesian learning of neural networks for mobile user position prediction," in *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications*, 2007, pp. 441–446.

[10] H. Gao, J. Tang, and H. Liu, "Mobile location prediction in a spatio-temporal context," in *Mobile Data Challenge by Nokia Workshop at the Tenth International Conference on Pervasive Computing*, Jun. 2012.

[11] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis.* New Jersey: John Wiley & Sons, 2008.

[12] H. Kantz, *Nonlinear time series analysis*, 2nd ed. Cambridge, UK: Cambridge Univ. Press, 2004.

[13] V. Defense Mapping Agency, Fairfax, "The universal grids: Universal transverse mercator (utm) and universal polar stereographic (ups)," DMA Technical Manual, Sep. 1989.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. Wiley-Interscience, 2000.

[15] H. G. Sung, "Gaussian mixture regression and classification," Ph.D. dissertation, Dept. of Stat., Rice Univ., Houston, TX, 2004.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2006.

[17] J. P. Petersen. (2010) Python pattern recognition, gaussian mixture models. [Online]. Available: http://pypr.sourceforge.net/mog.html [Accessed: 2013-09-29]

[18] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. (2010, Feb.) Supporting online material for [4]. [Online]. Available: http://www.sciencemag.org/content/suppl/2010/02/18/327.5968.1018.DC1 [Accessed: 2013-08-27]

[19] G. McLachlan and D. Peel, *Finite Mixture Models*, 1st ed. Wiley-Interscience, Oct. 2000.

[20] Z. Zivkovic and F. v. d. Heijden, "Recursive unsupervised learning of finite mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 5, May 2004.

[21] J. Novovicova, P. Pudil, P. Somol, and F. Ferri, "Initializing normal mixtures of densities," *Pattern Recognition*, vol. 1, pp. 886–890, Aug. 1998.

[22] S. Solla, T. K. Leen, and K.-R. Müller, "The infinite gaussian mixture model," in *Neural Information Processing Systems*, 2000, pp. 554–560.

[23] E. Parzen, "On estimation of a probability density function and mode," *Annals of Math. Statistics*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962.

[24] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Annals of Math. Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[25] M. P. Wand and M. C. Jones, *Kernel Smoothing*, 1st ed. Chapman & Hall, 1995.

[26] M. Kristan, A. Leonardis, and D. Skočaj, "Multivariate online kernel density estimation with gaussian kernels," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2630–2642, Jan. 2011.

[27] D. P. Bertsekas, *Nonlinear programming*, 2nd ed. Belmont: Athena Scientific, 1999.

[28] M. Á. Carreira-Perpiñán, "Mode-finding for mixtures of gaussian distributions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1318–1323, 2000.

[29] S. Pulkkinen, M. M. Mäkelä, and N. Karmitsa, "A continuation approach to mode-finding of multivariate gaussian mixtures and kernel density estimates," *Journal of Global Optimization*, vol. 56, no. 2, pp. 459–487, Jun. 2013.

[30] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. New Jersey: Prentice Hall, 2003.

[31] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/-mobility (v. 2009-02-24)," Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, Feb. 2009.

Declaration of Authorship

Hereby I, Jonas Lüthke, declare that this work is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other University. Formulations and ideas taken from other sources are cited as such. This thesis has not been published.

Hamburg, September 29, 2013