MDL-based Map Segmentation for Trajectory Mining

Lincen Yang (s1882511)

Internal Supervisor: Prof. Dr. Peter Grünwald
External Supervisor: Dr. Matthijs van Leeuwen
External Supervisor: Dr. Mitra Baratchi

MASTER THESIS

Defended on Month Day, Year

Specialization: Data Science

STATISTICAL SCIENCE
FOR THE LIFE AND BEHAVIOURAL SCIENCES

Abstract

Discretization is a key issue in urban trajectory pattern mining tasks. By assuming that regions with different functions will probably have different densities of visiting people, we propose to segment the city map-and hence discretize trajectory data-by finding region boundaries based on strong density changes. We solve the map segmentation problem as a model selection problem, using the existing MDL-histogram approach. We also propose a heuristic algorithm so that we can apply MDL-histogram on 2-dimensional data (longitude and latitude). Finally, we validate our approach and algorithm by simulation studies and on taxi trajectories from New York City.

Acknowledgements

I want to firstly thank my daily supervisor Dr. Matthijs van Leeuwen and Dr. Mitra Baratchi, for letting me explore my own interests and encouraging me to keep pursuing what I think is important. The weekly discussions between us were always inspiring and helpful for me. Moreover, I want to thank my internal supervisor, Prof. Peter Gr*ü*nwald, for his patient guidance, encouragement and support.
Finally, I want to thank my parents for their financial support during my two-year study at Leiden.
The accomplishment of the thesis would never be possible without any of you.

Contents

1 Introduction

As the prevalence of location-acquisition technologies, a huge amount of data of people movements and trajectories is produced and collected every day: taxi trajectories recorded by GPS devices, car trajectories recorded by navigation software on smartphones of drivers, check-in data on location-based social networks, only to name a few. All of these carry enormous information about human movement flows within a city. A lot of related research has been going on in recent years, trying to mine meaningful patterns from people's movements and gain insights into people's activities [19].

One key challenge in pattern mining from movement data is discretization [14]. Human trajectories are usually recorded as sequences of GPS coordinates (longitudes and latitudes); however, GPS coordinates are continuous but most pattern mining algorithms only work for discrete data [20]. In other words, similar trajectories should be grouped together. In rare cases, the trajectories are represented as sequences of names of venues (e.g., Central Station, University Library), and seem to be discrete. Still, it is necessary to group places based on spatial locations and semantics of places to get meaningful patterns [14], which also involves discretization of longitudes and latitudes of venues.

Ad-hoc methods for the discretization are widely used, such as making a grid on the map with plausible granularity. More advanced methods usually involve dense region detection, but they only consider trajectories leaving from or heading to the dense regions, and do not distinguish trajectories within one (even very big) dense region [4,17].

Three requirements for grouping trajectories were proposed by Zhang *et al.* [14], which are *Spatial compactness, Semantic consistency, and Temporal continuity*, but their algorithm only works for datasets with information about the semantic meanings and categories of venues, which are usually not accessible in the real world. Informally speaking, the trajectories are required to be spatially and temporally close enough, and the trajectories should leave from and arrive at places that have similar semantic meanings.

Is it also possible to discretize GPS coordinates and group trajectories in a way that satisfies *Semantic consistency*, without any external information about the semantic meanings of venues? While it is not possible to group trajectories with specific visiting purposes (e.g., go to dinner at restaurants) merely with sequences of GPS coordinates, we argue that it is indeed feasible to group trajectories that are semantically consistent in a broader sense, such that they leave from or are heading to a region with similar functions.

Suppose we ask a local expert to segment the map of a city into regions with different functions (e.g., residential area, restaurants area, shopping street, business district), and we ask all the people on their way to their next destinations which region they left and which region they are heading to; then, we record the frequencies of people leaving from different regions (or heading to different regions), and divide the frequency of people leaving or heading to each region by the area of each region. That ratio, representing the density of trajectories' starting points (or destinations) in each region are very unlikely to be precisely equal.
To detect boundaries where the density of data points change rapidly, we firstly make some assumptions about the probability distribution of trajectories. Under such circumstances, the probability distributions of origins and destinations are also fixed, since they are the marginal distributions of probability distribution of trajectories. We then regard each possible map segmentation (a set of boundaries within the map) as a model, and finally we solve this model selection problem by *the Minimum Description Length Principle (MDL).*
Our work is very similar and indeed directly motivated by MDL-histogram [1,2,8,9], which segments 1-dimensional data into bins with irregular widths. The purpose of the MDL-histogram approach is to construct an irregular histogram that captures more detail when the density is locally relatively high, which is also a desirable property for our task since we want a fine granularity when many people are leaving from or heading to some places.
In summary, the main contributions of the thesis are:

- We propose to use map segmentation to discretize trajectory data for trajectory pattern mining tasks. We propose to segment the map by detecting boundaries where the densities of origins and destinations of trajectories rapidly change.
- We regard the map segmentation task as a model selection problem and use the Minimum Description Length Principle (MDL) framework to solve it.
- Based on the MDL-histogram optimization algorithm [2], we propose a heuristic approach to segment 2-dimensional maps.
- We investigate our method and algorithm by simulation studies and real-world taxi trajectory analysis.

In Section 2, we will discuss the map segmentation problem in detail. Then, we will briefly review some important concepts of MDL as well as the algorithm of MDL-histogram as preliminaries in Section 3. The description of our algorithms and heuristics is in Section 4. Simulation studies and real-world data analysis are in

Section 5 and 6. Finally, a brief review of related works and our conclusions and discussion are given at Section 7 and 8.

2 Problem Description

In this section, we discuss a probabilistic model for trajectories data and our assumptions, as well as some properties of human traffic flows within a city.

2.1 A Probabilistic Model for Trajectories

Since our map segmentation is based on densities of origins and destinations as 2-dimensional points on a map, we consider trajectories in the form of origin-destination pairs.
The collection of origin-destination pairs with size n is denoted as

$$x^n = [(s_1^x, s_1^y, d_1^x, d_1^y), (s_2^x, s_2^y, d_2^x, d_2^y), ..., (s_n^x, s_n^y, d_n^x, d_n^y)] \in \{\mathscr{T} \times \mathscr{T}\}^n$$

where $\mathscr{T}$ denotes the map, which is the set of all possible positions of the data points, s^x, s^y are longitudes and latitudes of origins (**s**tarting points) of a trajectory, and d^x, d^y are longitudes and latitudes of a **d**estination of a trajectory.
Formally, a map $\mathscr{T}$ is defined as a closed and bounded subset of R^2. A map segmentation $F = \{F_1, F_2, ..., F_K\}$ is a collection of subsets of the map $\mathscr{T}$, such that $\bigcup F = \mathscr{T}$ and $F_i \cap F_j = \emptyset$ for any $i, j \in \{1, ..., K\}$.
We assume that

1. the probability of a person who goes to region j from region i follows a multinomial distribution (with parameter θ);
2. the locations where a person starts and stops to move follow a 2-dimensional uniform distribution within region i and j respectively;
3. the probability of each trajectory as an origin-destination pair is independent.

Given a certain map segmentation, as well as the assumptions just given, we define the *probabilistic source* [3] of trajectories to be

$$f(x^n|\theta, C_1, C_2) = \prod_{i,j=1}^{K_1,K_2} (\frac{\epsilon^4 \cdot \theta_{ij}}{S_i^s \cdot S_j^d})^{h_{ij}},$$

where C_1, C_2 are two different segmentations, for the starting points of trajectories and the destinations of trajectories respectively. We will discuss the necessity of different segmentations for starting points and destinations later. h_{ij} is the number of trajectories starting from region i of the map segmentation for starting points and arriving at region j of the map segmentation for destinations. K_1, K_2 are respectively the number of regions after segmentation for the map of starting points and destinations. S_i^s, S_j^d are respectively the areas of region i of the map for starting points and of region j of the map for destinations. ϵ is the precision of longitudes and latitudes. Finally the $\theta_{i,j}$ are the multinomial distribution parameters.
It is possible that the starting points and destinations of a region are not uniformly distributed. In other words, although the functions of venues within one region are much more similar than those in neighboring regions, people may not use the venues uniformly. We will defer this potential problem to the discussion section.

2.2 Problem Formulation: A Model Selection Problem

Given a map segmentations for starting points and destinations, the *probability source* of the whole dataset now only depends on the multinomial distribution parameter θ. In other words, a certain segmentation for starting points and destinations consists of a class of probability sources, which can be regarded as a *model* [3].
The *model class* is defined as all possible map segmentations, of which the boundaries of regions are step line segments. That is, all boundaries are composed of vertical and horizontal line segments. The model class is also constrained by the number of regions after the segmentations, which are bounded by hyperparameters K_1^{max} and K_2^{max} respectively.
Now our map segmentation problem is converted to a model selection problem. Namely, we want to choose a map segmentation that detects the boundaries within a map where densities rapidly change, and after the segmentation, the starting points and destinations are uniformly distributed within each region. The model selection can be formalized in several ways. Section 4 will describe our approach based on the MDL principle.

2.3 Properties of Human Traffic Flows

2.3.1 Functions of Regions Change over Time

The functions of some regions are complex. Specifically, it is common that one region has one function during the daytime and another function during the night time. Thus, the segmentation of timeline is necessary; otherwise, a region with some

specific function might be concealed in a bigger region with different functions.
For instance, suppose there is a student bar on a university's campus. If trajectories after 10 pm are separated from the trajectories before 10 pm, it is possible that we can detect the bar, since perhaps the number of students who go to places near the bar after 10 pm is much bigger than the number of students who go to other places on campus during that time. In contrast, if time is not segmented, students who go to the bar at night are just a very small part of all the students who go to the university during this 24 hours, so the density of trajectories around the bar would probably be very similar to the density of trajectories around the whole campus. Under such circumstances, it is unlikely that we can detect the bar.
However, the algorithm we use to segment the map cannot be used to segment the timeline, because the algorithm is based on one of our assumptions that starting points and destinations of trajectories within a region are uniformly distributed. Apparently, no matter how we segment the timeline, it is unlikely that trajectories are uniformly distributed on time. For example, during the morning, the number of people who leave their home and go to work would probably increase from very early and started to decrease after the peak hour.
Therefore, for now we will segment the timeline by hand, and consider trajectories within each time period separately.

2.3.2 Different Map Segmentations for Starting Points and Destinations

Sometimes regions have different functions as starting points and as destinations. Thus, it is necessary to segment the map in different ways for starting points and destinations. We illustrate this necessity with an example.
Suppose there is a city whose city center consists of both residence apartments and company buildings. Company buildings are clustered in a few different blocks, while residence apartments are scattered over this whole region. During the morning, the starting points of trajectories here probably represent people who leave their home in the city center and go to the working places at the suburb; however, the destinations in the city center probably represent people who come to the city center to work.
Therefore, when we segment the map for the starting points during the morning, we should probably treat the whole city center as one region of residency. In the contrast, when we segment the map for the destinations, we may want to separate the company areas from nearby residential buildings.

3 Preliminaries

In this section, we will review some important concepts of the *Minimum Description Length Principle (MDL)*, especially one particular encoding strategy, *Normalized Maximum Likelihood (NML)*, as well as its application on *model selection*. Moreover, we will briefly review the MDL-histogram approach.

3.1 The Minimum Description Length Principle

A fundamental part of the MDL framework is to encode data with another dictionary. The key idea of MDL is to find an encoding scheme such that the data can be compressed most. One of the key principles of the MDL is that *to learn more is to compress more*. The degree of compression is measured by the code length [3].
The most fundamental theory in the MDL is the *Kraft's Inequality*, which describes the relationship between probability and code length. It states that, given a dataset with elements from a finite set, if we define a probability P on this set (regard it as the sample space), there always exists a corresponding prefix code for the dataset, with the code length for each element equal to $-\log_2 P$; on the contrary, if we encode the dataset with some prefix code with code word length for each element equal to L, there is also a corresponding probability for each element in the set, defined as $P = 2^{-L}$. In summary, code length is equivalent to probability [3].
Another important theoretical result, the *No Hyper-compression Theorem* [3], states that if the dataset is generated by some probability distribution, the code length corresponding to the true probability distribution is (asymptotically) the shortest over all possible encoding schemes. This shortest code length is called *entropy* in information theory.

3.2 Normalized Maximum Likelihood and Model Selection

Model selection under the MDL framework is to find a (probabilistic) model whose corresponding code length is minimum over a given model class.
We already know that assigning a probability distribution to the elements of the dataset is equivalent to encoding the dataset. We also know that finding an encoding scheme of the dataset with shortest code length is equivalent to finding a probability distribution that somehow fits the dataset best. However, a model is usually a group of probability distributions. Thus, how do we connect a model instead of a single probability distribution to the code length?
Given a model, i.e., a group of probability distributions, we can encode the data

using *universal coding.* A universal code means that if the unknown true probability distribution of the data is included in the model, the code length of the universal code will only be a little bit worse than as if we know the true probability distribution. We call the universal coding scheme to be "encoding with the help of the model" [3]. How much extra code length do we need if we only know the model other than the true probability? The expected extra code length is defined as *regret.* One encoding strategy called *Normalized Maximum Likelihood (NML)* will produce the code length with *minimax* regret [3].
In summary, the model selection problem under the MDL framework is to encode the data by a universal code with the help of each model in the model class, and search for the one with the minimum code length of data with the help of the model plus the code length needed to encode the model.

3.3 K&M's Algorithm for MDL-histogram

In this section, we will briefly review the idea and the algorithm of MDL-histogram by Kontkanen and Myllymaki [2].

3.3.1 A Probabilistic Model

Irregular histogram density estimation, which is a density estimation method based on plotting a histogram with unequal bin-widths, is equivalent to the segmentation of a 1-dimensional line segment. Given a set of cut points C of a histogram, the probability of data points of size n, $x^n \in \mathscr{X}^n$, is defined as:

$$f(x^n|\theta, C) = \prod_{i=1}^{K} (\frac{\epsilon \cdot \theta_i}{L_i})^{h_i}$$

where K is the number of bins of the histogram, L_i is the ith bin's width, h_i is the number of data points within the range of ith bin, ϵ is the precision of the data, and $\mathscr{X}$ is the set of all possible values of x on the line segment to be segmented [2]. In fact, we are assuming that each data point has a probability of θ_i to appear in the ith bin (i.e., a multinomial distribution), and the exact location within the ith bin follows a uniform distribution.
The 1 dimensional segmentation is regarded as a model selection problem in [2]: each segmentation is a model, represented by M, and all the possible segmentations form the model class $\mathscr{M}$. To encode the data with the help of the model (one certain way of segmentation), the Normalized Maximum Likelihood (NML) approach is used.

The *stochastic complexity*, which is the code length of the data with the help of the model by NML, is defined as:

$$SC(x^n|\hat{\theta}, C) = -\log_2(\frac{f(x^n|\hat{\theta}, C)}{\sum_{x^n \in \mathscr{X}^n} f(x^n|\hat{\theta}(x^n), C)})$$

where

$$\hat{\theta} = (\hat{\theta}_i)_{i=1,...,K}, \quad \hat{\theta}_i = \frac{h_i}{n}$$

is the *Maximum Likelihood* estimator for the multinomial distribution parameter. Thus,

$$f(x^n|\hat{\theta}, C) = \prod_{i=1}^{K} (\frac{\epsilon \cdot h_i}{n \cdot L_i})^{h_i}$$

$$R_K^n := \sum_{x^n \in \mathscr{X}^n} f(x^n|\hat{\theta}(x^n), C) = \sum_{x^n \in \mathscr{X}^n} \prod_{i=1}^{K} (\frac{\epsilon \cdot h_i}{n \cdot L_i})^{h_i}$$

where $h_1, ..., h_K$ are at least 1, and the R_K^n is called *parametric complexity.* It is observed that given the segmentation C and the numbers of points in each bin $h_1, ..., h_K$, there are $(\frac{L_i}{\epsilon})^{h_i}$ possible locations of data points within the ith bin. It is also observed that the multinomial coefficient $n!/(h_1!...h_K!)$ counts the number of arrangements of n data points into K bins each containing $h_1, ..., h_K$ objects, respectively. Thus,

$$R_K^n = \sum_{h_1+...+h_K=n} [\frac{n!}{h_1!...h_K!} \prod_{i=1}^{K} [(\frac{L_i}{\epsilon})^{h_i} (\frac{h_i \cdot \epsilon}{n \cdot L_i})^{h_i}]]$$

$$= \sum_{h_1+...+h_K=n} \frac{n!}{h_1!...h_K!} \prod_{i=1}^{K} (\frac{h_i}{n})^{h_i}$$

Interestingly, the parametric complexity R_K^n does not depend on the width of the bins. Thus, for every segmentation with the same cut points, the parametric complexity is the same.
To calculate R_K^n directly involves a exponential sum. However, previous researches [2, 8,9] show that the complexity can be reduced to linear-time by a recursive formula:

$$R_K^n = R_{K-1}^n + \frac{n}{K-2} R_{K-2}^n$$

Finally, the model selection problem is defined as an optimization problem [2]. The goal is to find a set of cut points $\hat{C}$ such that

$$\hat{C} = \arg\min_{C} B(x^n|E, K, C)$$

where the optimization function B is defined as:

$$B(x^n|E, K, C) = SC(x^n) + \log \binom{E}{K-1}$$

$$= \sum_{k=1}^{K} -h_k(log(\epsilon \cdot h_k) - \log(L_k \cdot n)) + \log R_K^n + \log \binom{E}{K-1}$$

where E is the number of all possible cut points. The reason for including $\log \binom{E}{K-1}$ in the optimization function is that we also need to encode the model, i.e., which cut points we choose among $\binom{E}{K-1}$ possibilities [3].

3.3.2 Algorithm

Firstly, the search space C is defined as:

$$C = \bigcup_{i}^{n} \{x_i + \epsilon/2, x_i - \epsilon/2\} \setminus \{\min_i x_i - \epsilon/2, \max_i x_i + \epsilon/2\}$$

Given the number of bins K, $K \in \{1, 2, ..., K_{max}\}$, there are $\binom{E}{K-1}$ of possible irregular histograms. Since we would never know the true number of bins, it is necessary to search starting from $K = 1$ until a predetermined maximum number of bins, $K = K_{max}$. Therefore, $\sum_{K=1}^{K_{max}} \binom{E}{K-1}$ possible cut point sets exist, and brute-force search would normally not be feasible. However, K&M solved this problem by a brilliant dynamic programming algorithm.
Firstly, denote $C = \{c_0, c_1, ..., c_{E+1}\}$, where $c_0 = \min_i x_i - \epsilon/2$ and $c_{E+1} = \max_i x_i + \epsilon/2$, and $c_0 < ... < c_{E+1}$. Given the dataset x^n, it is defined that the constrained datasets $x^{n_e} := \{x : x < c_e\}, \forall e = \{1, ...E + 1\}$, where n_e is the number of data points in the constrained dataset. Denote

$$\hat{B}_{K,e} = \min_{\hat{C} \subset C} B(x^{n_e}|E, K, C)$$

Observe that, given a fixed set of cut points, $C = \{c_{e_1}, ..., c_{e_{K-1}}\}$, for the constrained dataset $x^{n_{e_K}} = \{x : x < c_{e_K}\}$, where $c_{e_1} < ... < c_{e_{K-1}} < c_{e_K}$,

$$\begin{aligned} B(x^{n_{e_K}}|E, K, C) = B(x^{n_{e_{K-1}}}|E, K-1, C') \\ - (n_{e_K} - n_{e_{K-1}})(log(\epsilon(n_{e_K} - n_{e_{K-1}}) \\ - log((c_{e_K} - c_{e_{K-1}})) + log\frac{R_K^{n_{e_K}}}{R_{K-1}^{n_{e_{K-1}}}} + log\frac{E-K-2}{K-1} \end{aligned} \quad (1)$$

where $C' = \{c_{e_1}, ..., c_{e_{K-2}}\}$.
Then, based on that, the recursive formula for dynamic programming is

$$\begin{aligned} \hat{B}_{K,e} = \min_{e'}\{\hat{B}_{K-1,e'} - (n_e - n_{e'})(log(\epsilon(n_e - n_{e'})) - (log(c_e - c_{e'})n)) \\ + log\frac{R_K^{n_{e_K}}}{R_{K-1}^{n_{e_{K-1}}}} + log\frac{E-K-2}{K-1}\} \end{aligned} \quad (2)$$

where $e' = K-1, ..., e-1$, and $e = 1, 2, ..., E$. The recursion is initialized with

$$\hat{B}_{1,e} = -n_e(log(\epsilon \cdot n_e) - log(c_e - c_0)n)$$

for $e = 1, 2, ..., E$. The number of bins increase by one each time until K_{max}. Then, the optimal number of bins is chosen to be

$$\hat{K} = arg \min_K \hat{B}_{K,E}$$

Finally, track the e' that minimizes (2) recursively to get the positions of the cut points. The complexity of the whole algorithm is $\mathcal{O}(E^2 K_{max})$ [2].

4 Map Segmentation by NML

In this section, we will propose to use *Normalized Maximum Likelihood* described in Section 2.2 to solve the map segmentation task as a model selection problem. Since the algorithm of MDL-histogram is not directly applicable to higher dimensions, we propose a heuristic algorithm based on the MDL-histogram algorithm to handle 2-dimensional data points (starting points or destinations of trajectories). Finally, we discuss how to deal with 4-dimensional data points, which are the trajectories as origin-destination pairs.

4.1 A Model Selection Problem

The model class is defined as all possible map segmentations for starting points and destinations of trajectories, of which the boundaries of regions are step line segments. The model class is also constrained by the number of regions after the segmentations, which are bounded by K_1^{max}, K_2^{max} respectively for starting points and destinations. Given the model, the dataset is encoded with the help of the model by Normalized Maximum Likelihood. The *Stochastic Complexity (SC)* is given by

$$SC(x^n|\hat{\theta}, C_1, C_2) = -log_2(\frac{f(x^n|\hat{\theta}, C)}{\sum_{x^n \in \mathscr{X}^n} f(x^n|\hat{\theta}(x^n), C)})$$

where

$$f(x^n|\hat{\theta}, C_1, C_2) = \prod_{i,j=1}^{K_1,K_2} (\frac{\epsilon^4 \cdot h_{ij}}{S_i^s \cdot S_j^d \cdot n})^{h_{ij}}$$

$$R_{K_1,K_2}^n := \sum_{x^n \in \mathscr{X}^n} f(x^n|\hat{\theta}(x^n), C_1, C_2) = \sum_{x^n \in \mathscr{X}^n} \prod_{i,j=1}^{K_1,K_2} (\frac{\epsilon^4 \cdot h_{ij}}{S_i^s \cdot S_j^d \cdot n})^{h_{ij}}$$

$$= \sum_{\sum h_{ij}=n} [\frac{n!}{\prod h_{ij}!} \prod_{i=1}^{K_1,K_2} [(\frac{S_i^s \cdot S_j^d}{\epsilon^4})^{h_{ij}} (\frac{h_{ij} \cdot \epsilon^4}{n \cdot S_i^s \cdot S_j^d})^{h_{ij}}]]$$

$$= \sum_{\sum h_{ij}=n} \frac{n!}{\prod h_{ij}!} \prod_{i,j=1}^{K_1,K_2} (\frac{h_{ij}}{n})^{h_{ij}}$$

where $h_1, ..., h_K$ are at least 1. K_1 and K_2 denote the number of resulting regions after the segmentation, for starting points and destinations respectively. Similar to the case of MDL-histogram, the parametric complexity does not depend on the area of each region after segmentation.
Therefore, the map segmentation problem, defined as a model selection problem, is to find $\hat{C}_1, \hat{C}_2$ such that

$$(\hat{C}_1, \hat{C}_2) = \arg\min_{C_1,C_2} [SC(x^n|\hat{\theta}, C_1, C_2) + CodeLength(C_1, C_2)]$$

4.2 How to Encode Every Segmentation as a *Model*?

Since we constrain the maximum number of regions after map segmentation, and the data has a precision ϵ, the number of models within the model class is actually finite.

The standard encoding scheme in this case is to use the *uniform code* [3]. Suppose the number of models in the model class is M, then the code length for every model is $\log(M)$.
However, it is very difficult to calculate the exact number of all possible segmentations in the model class. To segment the map into K regions is actually to partition a grid with granularity ϵ, and calculating the number of possible partitions of a grid is very complex. We will illustrate a strongly simplified example.
Suppose we have a 2-dimensional rectangle, with E_1 possible cut positions on the horizontal edge, and E_2 on the vertical edge. Suppose we make a grid based on all the cut positions on the edges, and we want to segment the rectangle into 2 parts, with a step-line composed of inner edges of the grid that satisfies:

- The step-line must start from the left edge and end at the right edge. Starting from or ending at the horizontal edges is not allowed.
- The sum of the lengths of all the horizontal line-segments that consist of the step-line must be equal to the length of the horizontal edge of the rectangle. In other words, if we draw the step line from left to right, horizontally we must always draw towards the right.

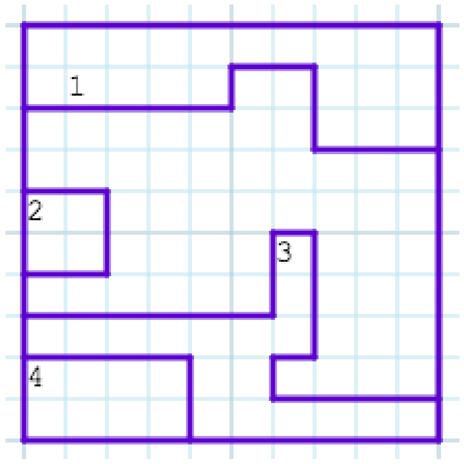


Figure 1: Several examples of map segmentation line-segments. Only line-segments 1 satisfy the constraints; line-segments 3 violate the second constraint; line-segments 4 violate the first one; line-segments 2 violate both.

It is obvious that ${E_2}^{E_1}$ possible step-lines exist. In real map segmentation tasks, the number of possible models is even bigger than ${E_2}^{E_1}$, because we have to segment the map into more than two regions, and because the two above constraints do not exist. Therefore, the standard uniform code seems not to be working here. Luckily, this problem is automatically solved when we introduce our heuristics.

4.3 A Heuristic for 2-dimensional Data

The dynamic programming algorithm for the MDL-histogram on 1-dimensional data is not applicable to 2 and higher dimensions. One key step of the MDL-histogram algorithm is to construct constrained datasets, which is not applicable to 2-dimensional data because only 1-dimensional data has a natural order that allows constructing such constrained datasets.
According to our definition of trajectories, each trajectory is a 4-dimensional point in the Cartesian product of two identical maps, as close and bounded subsets of R^2. If we consider the starting points and destinations separately, the data points become 2-dimensional. We will now discuss our heuristic to extend K&M's algorithm for 2-dimensional data, and then discuss how to deal with 4-dimensional data in the next section.
We propose a heuristic called the "split-and-merge" approach. Given a map, we firstly construct a Cartesian coordinate system and fix the directions of the x-axis and the y-axis (usually along the directions of longitude and latitude lines).
Then, we constrain the model class to be all possible segmentations by at most K_{max} straight lines parallel to the y-axis. Then we search for a set of straight lines parallel to the y-axis, such that the code length of the data with the help of the model (i.e., the set of straight lines parallel to the y-axis) plus the code length of describing the model is minimized. This step is indeed a model selection problem almost equivalent to the MDL-histogram problem. The only difference is that, for 2-dimensional $x^n \in \mathscr{X}^n$, the optimization function is defined as

$$B(x^n|E_1, K_1, C_1) = SC(x^n) + log\binom{E_1}{K_1 - 1}$$

$$= \sum_{k=1}^{K_1} -h_k(log(\epsilon^2 \cdot h_k) - log(S_k \cdot n)) + logR^n_{K_1} + log\binom{E_1}{K_1 - 1}$$

The parametric complexity is defined as

$$R^n_{K_1} := \sum_{x^n \in \mathscr{X}^n} f(x^n|\hat{\theta}(x^n), C_1) = \sum_{x^n \in \mathscr{X}^n} \prod_{k=1}^{K_1} (\frac{\epsilon^2 \cdot h_k}{n \cdot S_k})^{h_k}$$

$$= \sum_{h_1+...+h_K=n} [\frac{n!}{h_1!...h_K!} \prod_{k=1}^{K_1} [(\frac{S_k}{\epsilon^2})^{h_k} (\frac{h_k \cdot \epsilon^2}{n \cdot S_k})^{h_k}]]$$

$$= \sum_{h_1+...+h_K=n} \frac{n!}{h_1!...h_K!} \prod_{k=1}^{K_1} (\frac{h_k}{n})^{h_k}$$

E_1 is the number of possible cut lines parallel to the y-axis of the map, K_1 is the number of regions after the segmentation, h_k is the number of data points fallen into region k, S_k is the area of region k, $R^n_{K_1}$ is defined and calculated the same as MDL-histogram, and finally ϵ is the precision of data. The search procedure is the same as the K&M's MDL-histogram algorithm.
Suppose each data point $x^n \in \mathscr{X}^n$ is in the form of $(x_{1i}, x_{2i})_{(i=1,2,...,n)}$ given the x-axis and y-axis. The search space of the cut lines can be denoted as

$$\{(x,y)|x \in \bigcup_i^n \{x_{1i} + \epsilon/2, x_{1i} - \epsilon/2\} \setminus \{\min_i x_{1i} - \epsilon/2, \max_i x_{1i} + \epsilon/2\}\} \bigcap M$$

where M is the map, as a closed and bounded subset of R^2.
Suppose we produced $\hat{K}_1$ regions after this first step of segmentation. Now we consider each of those $\hat{K}_1$ regions separately. Namely, for each of these regions, we search for a set of straight lines parallel to the x-axis (at most K_{max} lines) that minimizes the code length of data with the help of the model plus the code length needed to encode the model.
We repeat the procedure, alternating the direction of the cut lines in each iteration, until the algorithm stops to segment any region. Figure 3 shows an illustrative example of a possible split procedure, and Algorithm 1 gives it pseudo code.

Algorithm 1: Split and Merge Approach: Split Part

```
1  Input: 1. The location coordinates given the direction of the x-axis and the
     y-axis; 2. (Optional) A given outer boundary of the map. If missing, the
     outer boundary will be constructed based on the data. That is, the outer
     boundary will be the smallest rectangle that contains all the data points;
2  Output: A list of coordinates of the vertices of each region after segmentation;
3  Regions = list(M), M is the map;
4  k = 1;
5  while the algorithm continues to segment do
6      for i in 1:length(Regions) do
7          if k = 1 then
8              search cut lines parallel to y-axis for Regions[i], such that the
                 function B is minimized, based on the data within Regions[i];
9          end
10         if k = 2 then
11             search cut lines parallel to x-axis for Regions[i], such that the
                 function B is minimized, based on the data within Regions[i];
12         end
13     end
14     Regions = list(regions after segmentation);
15     $k = (k + 1) \bmod 2$
16 end
```

In the second phase, the merge phase, we check each pair of neighboring regions and decide whether to merge them. Whether to merge is also a model selection problem, of which the model class consists of only two models: “merge” and “not merge”. We solve this model selection problem by NML again.
The code length needed to encode the model, either “merge” or “not merge”, is $\log_2 2 = 1$. Therefore, we only need to compare the code length of the data with the help of the model.
For all the data within a pair of neighboring regions, we compare

$$B(x^{n'}|\text{merge}) = n'(log(\epsilon^2 \cdot n') - log((S_1 + S_2) \cdot n') + logR_2^{n'}$$

and

$$B(x^{n'}|\text{not merge}) = B(x^{n'_1}) + B(x^{n'_2})$$

where

$$B(x^{n'_1}) = n'_1(log(\epsilon^2 \cdot n'_1) - log(S_1 \cdot n'_1) + logR_1^{n'_1}$$

$$B(x^{n'_2}) = n'_2(log(\epsilon^2 \cdot n'_2) - log(S_2 \cdot n'_2) + logR_1^{n'_2}$$

where n'_1 and n'_2 are the number of data points within these 2 neighboring regions, $n' = n'_1 + n'_2$, and S_1 and S_2 are the areas of these 2 neighboring regions.
If $B(x^{n'}|\text{merge}) < B(x^{n'}|\text{not merge})$, then we merge these two neighboring regions and form a new region, and then continue to check other neighboring regions.

4.4 How to Deal with 4-dimensional Data?

The most straightforward approach is to consider starting points and destinations separately. In this case, the "split-and-merge" heuristic for 2-dimensional data can be applied directly.
It turns out that this approach is not a bad one at all. To do segmentation based solely on the starting points (or destinations) is actually to detect the boundaries where the *marginal* probability of trajectories (i.e., the probability of starting points or destinations) rapidly changes. On the contrary, if we do the segmentation based on trajectories as origin-destination pairs, we are actually detecting the boundary where the probability of trajectories rapidly changes. The only difference lies in the interpretation. For instance, during the late afternoon, the probability that a trajectory's destination is in a residential area is probably different than the probability that a trajectory's destination is in a nearby shopping street. Also, during the late afternoon, the probability of people leaving an office buildings region and heading to a restaurants region is probably different than the probability of people who leave the same office building region and heading to a residential area.
However, the map segmentation based solely on the marginal distribution can go wrong in very rare cases, when trajectory probabilities are different but coincidentally the marginal probabilities are the same.
For example, suppose the "true" map segmentation for starting points and destinations both segments the starting point map and the destination map into two parts, denoted as A_1, A_2 and B_1, B_2 respectively. Suppose the probabilities of trajectories are

$$P_{A_1,B_1} = 0.1; P_{A_1,B_2} = 0.4;$$

$$P_{A_2,B_1} = 0.4; P_{A_2,B_2} = 0.1;$$

Then, the marginal probabilities are

$$P_{A_1} = P_{A_2} = P_{B_1} = P_{B_2} = 0.5$$

Thus, in this case, if we apply our algorithm separately for starting points and destinations, the algorithm surely cannot detect the boundary of A_1 and A_2, as well as

that of B_1 and B_2.
A similar previous work [10] also tried to extend K&M's MDL-histogram to higher dimensions, in order to construct a 2-dimensional histogram for time series data. Kameya [10] used an alternative approach to achieve the goal, namely *Coordinate Descent*. For our case, *Coordinate Descent* means that we initialize a map segmentation for starting points of trajectories, keep this segmentation fixed, and segment the map for destinations based on the probability of trajectories using *split-and-merge* method. Then we keep the destination map fixed, and segment the map for starting points again. We iteratively repeat this procedure until it converges.
However, *Coordinate Descent* is an optimization method for continuous data with a smooth optimization function. Applying coordinate descent method directly to our discrete optimization problem would be problematic [18].

4.5 Properties of the Split-and-Merge Approach

One key property of our "split-and-merge" approach is that, when we segment the regions produced by the algorithm in previous iterations, we consider each region separately.
As a model selection problem, the set of cut lines that minimize the code length of data within this region plus the code length needed to encode the positions of cut lines, does not necessarily minimize the code length of the whole dataset with the help of all current cut lines plus the code length needed to encode all the cut lines on the map.
However, if we set the optimization goal of the model selection within one region as the code length of the data on the whole map with the help of all the segmentations, plus the code length needed to encode the segmentations, the order of doing the segmentations for different regions on the map will have an effect on the result, which is very undesirable.
However, can we segment all the regions produced by the previous segmentation step simultaneously? Under this circumstance, the optimization is based on the code length of the whole dataset plus the code length of the whole segmentations. However, suppose our map is already segmented into K_1 regions, and now we alter the direction of the cut lines, and we want to segment the map into K_2 regions ($K_2 >= K_1$). Suppose we segment each of the K_1 regions into $K_{2,1}, ..., K_{2,K_1-1}$ regions, then the optimization involves a searching of over all possible values of $K_{2,1}, ..., K_{2,K_1-1}$, for any $K_{2,1} > 0, ..., K_{2,K_1-1} > 0, K_{2,1} + ... + K_{2,K_1-1} = K_2$, which is exponential.
Surprisingly, considering separately each region that was produced in previous itera-

tions of segmentation might not be a bad idea at all. We will show its rationale later in the simulation study section.

5 Simulation Studies

In this section, we will show the performance of our *split-and-merge* heuristics on simulation data. Moreover, since the *split-and-merge* approach considers more and more "local" regions when we recursively keep splitting regions produced in previous iterations, we will show for the 1-dimensional case that this property does not pose problems in practice.

5.1 Recursive Binary Search for MDL-histogram

This is a byproduct of the simulation study of this thesis that surprisingly justifies the "split-and-merge" approach. We set the maximum number of bins to be 2. Then, we consider each of these two bins separately, and recursively run the MDL-histogram algorithm within each bin, with again the maximum number of bins to be 2. We repeat this procedure until the algorithm stops to segment all the current bins.
The simulation result shows that this "recursive binary search" approach produces very similar results as the original K&M MDL-histogram algorithm.

5.1.1 Experiment Settings

Set the data range to be within $[0, 1] \subset R$. Set $\epsilon = 0.01$.
Firstly, sample uniformly 4 (chosen arbitrarily) cut points from

$$\{0 + \epsilon/2 + k \cdot \epsilon \,|\, k = 1, 2, ..., 99\} = \{0.005, 0.015, ..., 0.995\}$$

Then uniformly draw 5 numbers from $[0, 1]$, normalize them to have a sum of 1, and assign them to θ as the multinomial distribution parameter.
Finally, for each data point, simulate firstly which bin it belongs, and then simulate the location of the data point by a uniform distribution within the bin.
After the data is generated, we apply K&M's algorithm on the dataset but set K_{max} to be 2. Then, for each of the two bins produced, we apply K&M's algorithm again respectively, with $K_{max} = 2$. We repeat this procedure until the algorithm stops to segment all the bins.
The whole procedure is repeated 100 times. The cut points produced by this approach (we call it Recursive Binary Search) are compared with the cut points produced by the original K&M's algorithm.

5.1.2 Loss Function and Results

The "true" cut points, denoted by C, and the cut points produced by the algorithms (either K&M's algorithm or the "recursive binary search"), denoted by C', are both finite sets of numbers. Thus, the "loss" is defined to be the Hausdorff distance.

$$L(C, C') = max\{\max_{c \in C} \min_{c' \in C'} d(c, c'), \max_{c' \in C'} \min_{c \in C} d(c, c')\}$$

where $d(c, c') := |c - c'|$.

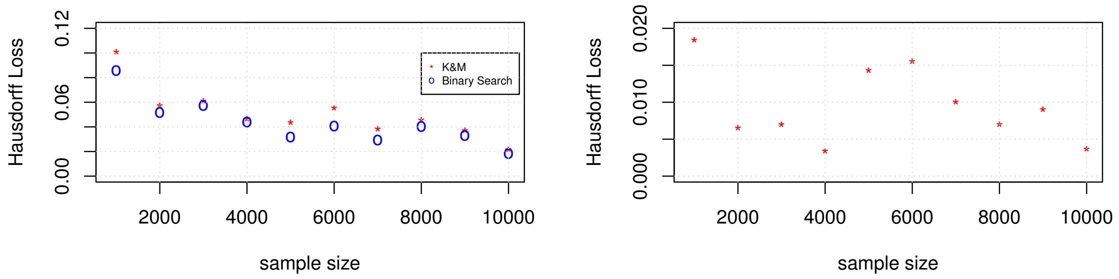


Figure 2: Top: Convergence Rate of K&M's algorithm and our "recursive binary search" algorithm, i.e., we compare the Hausdorff distance between cut points produced by K&M's Algorithm and the "true" cut points, with the Hausdorff distance between the cut points produced by "Recursive Binary Search" and the "true" cut points; Bottom: Hausdorff distance between the cut points produced by K&M's algorithm and the cut points produced by our "Recursive Binary Search" algorithm, for every generated dataset. Note the different scale on the y-axis.

Figure 2 (top) shows that the difference (on average) between cut points produced by K&M's algorithm and the "true" cut points is very similar to the difference between cut points produced by *Recursive Binary Search* method and the "true" cut points. In Figure 2 (bottom), we calculate the Hausdorff distance between the cut points produced by K&M's algorithm and those by our "Recursive Binary Search" algorithm, for every single generated data sample. It can be seen that all the average Hausdorff distances are less than 0.02; since we have 4 cut points, the average Hausdorff distance due to each individual cut point is 0.005, which is less than $\epsilon = 0.01$. Therefore, it shows that, the recursive binary search and K&M's algorithm not only have very similar convergence rates but also produce very similar cut points given the generated data sample.

5.2 Map Segmentation

One goal of this simulation study is to show the difference between the "true boundaries of regions" and "boundaries produced by the algorithm". Another goal is to test the robustness of the algorithm to noise, since GPS record devices are not perfectly accurate.

5.2.1 Experiment Settings

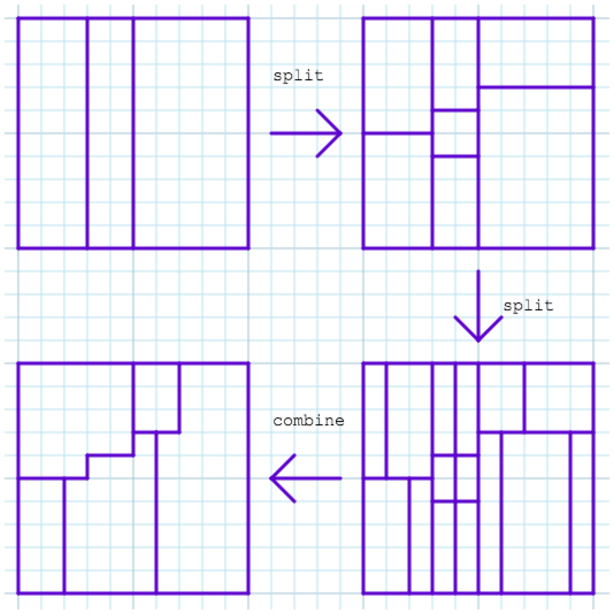


Figure 3: A possible instance of the procedure of map generation in the simulation studies; also a possible instance of the procedure of the map segmentation by the algorithm.

Set the map as the Cartesian Product of $[0, 1] \subset R$, i.e., $[0, 1] \times [0, 1]$. Set the precision of data to be 0.01.
We firstly randomly draw a number b from {1,2,3} (chosen arbitrarily), then we draw b cut lines parallel to the y-axis, the positions of which are randomly drawn from

$$\{0 + \epsilon/2 + k \cdot \epsilon | k = 1, 2, ..., 99\} = \{0.005, 0.015, ..., 0.995\}$$

For each region produced, we randomly draw a number b from {1,2,3} again, then draw b cut lines parallel to the x-axis, also with random positions.
For the regions produced, we repeat the procedure one more time, with cut lines parallel to y-axis again. In summary, three iterations of the drawing are implemented, and we alternate the direction of drawing each iteration.
Now we check every pair of regions that share common edges, and by a chance of 10% (a hyperparameter) we remove the common edge of them to merge these two regions.
Theoretically, this drawing and merging procedure can produce any possible map segmentation if we set $b \in \mathcal{Z}^+$. The number of iterations of drawing (3 iterations), the maximum number of b ($\max_b = 3$), and the chance of merging (10%) are both chosen arbitrarily. In practice, these settings can produce seemingly very irregular map segmentations (see Appendix).
Now, if we produced K regions, labeled as {1,2,...,K}, we generate the multinomial parameter θ to be a vector of length K,

$$\theta = (\theta_1, ..., \theta_K), \theta_1 + ... + \theta_K = 1$$

Finally, to generate the data, we firstly generate the region to which each data point belongs according to θ, and then generate the data points uniformly within each region.
To examine the goodness of segmentation by our algorithm, we do the simulation study with a sample size of 1000 to 10000 (increased by 1000), and of 50000, of 100000, of 500000 without any noise.
To examine the robustness, we do the simulation study with sample size 50000, with Gaussian noise $N(0, 0.1 \cdot \epsilon), N(0, 0.2 \cdot \epsilon), ..., N(0, 1 \cdot \epsilon)$.

5.2.2 Loss Function and Results

Our loss functions are defined based on the idea of Hausdorff distance.
Suppose the ith data point in the dataset is in the form of $x = (x_1, x_2)$, and also suppose we use K_1 regions, labeled as $\{R_1, R_2, ..., R_{K_1}\}$, to generate the data, and the algorithm segments the map into K_2 regions, labeled as $\{R'_1, R'_2, ..., R'_{K_2}\}$.
Denote the data points that fall into regions $R_1, ..., R_{K_1}$ as $D_1, ..., D_{K_1}$ respectively. Similarly, denote the data points that fall into regions $R'_1, ..., R'_{K_2}$ as $D'_1, ..., D'_{K_2}$ respectively.
Define $(l_1, r_1), ..., (l_{K_1}, r_{K_1})$ as the x-coordinates of the *leftmost and rightmost* data points within $D_1, ..., D_{K_1}$ respectively. Also define $(d_1, u_1), ..., (d_{K_1}, u_{K_1})$ as the y-coordinates of the *down-most and up-most* data points within $D_1, ..., D_{K_1}$ respectively. $(l'_1, r'_1), ..., (l'_{K_1}, r'_{K_2})$ and $(d'_1, u'_1), ..., (d'_{K_1}, u'_{K_2})$ are defined similarly.

Define the distance between R_i and R'_j as

$$d_{ij} = |l_i - l'_j| + |r_i - r'_j| + |d_i - d'_j| + |u_i - u'_j|$$

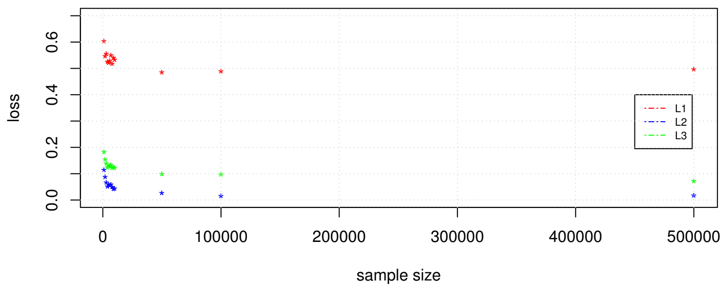
Finally we define three loss functions between the true segmentations and segmentation by algorithm as

$$L_1 = \frac{1}{K_2} \sum_{j=1}^{K_2} \min_i d_{ij}$$

$$L_2 = median\{\min_i d_{ij}\}, j = 1, 2, ..., K_2$$

$$L_3 = max\{\min_i d_{ij}\}, j = 1, 2, ..., K_2$$

To estimate the loss, the simulation is repeated 100 times for each setting. It can be seen from the simulation result that all three loss function values do not change very much after the sample size is larger than 50000.



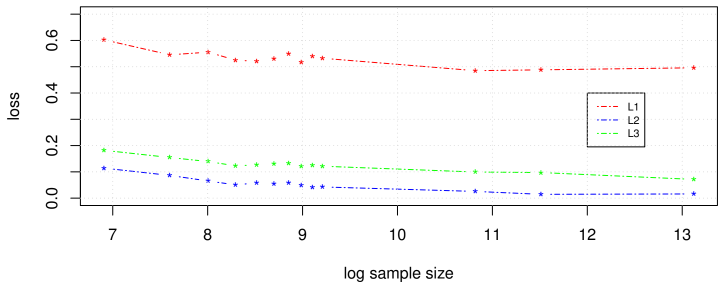


Figure 4: Sample Size vs Loss: The loss between the boundaries produced by our algorithm and the "true" boundaries

In figure 4, L_2 gets almost 0 as the sample size increases, indicating that, on average, at least half of the regions produced by our algorithm match very well with the "true" boundaries. L_3 shows that, on average, each boundary differs by 0.025 ($= 2.5 \cdot \epsilon$).
However, L_1 shows that there is always at least 1 bad segmentation. As discussed earlier, each region produced by previous segmentation lines is considered separately. Therefore, the sample size within each region becomes smaller and smaller as we keep dividing the regions, so the algorithm becomes less and less stable locally if the sample size within some regions become too small.
Combining L_1, L_2 and L_3, the general picture of the boundaries produced by our algorithm is that (also see the Appendix):

1. The boundaries of at least half of the regions match almost exactly with the true boundaries.

2. Usually, there exist few bad boundaries, either because the algorithm becomes very local, or because the generated "true" probability densities of some neighboring regions are very similar.

3. Given that L_3 is also small and L_1 is big, most boundaries should match very well with the "true" boundaries.

We show a few simulation examples in the Appendix. It can be seen that the algorithm easily makes a mistake either when the density of the region is small, or when the densities of neighboring regions are very similar.

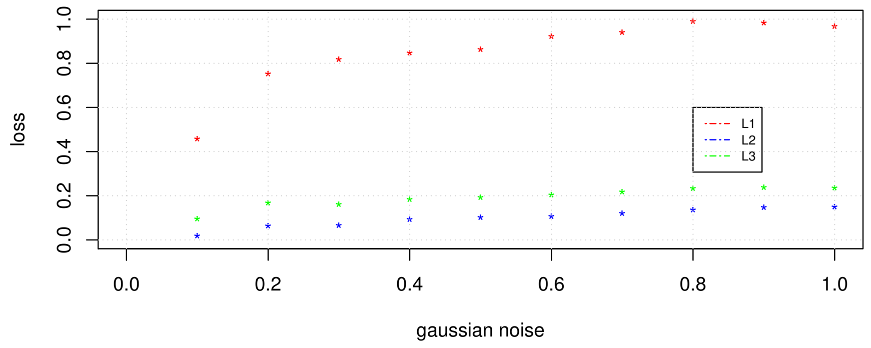
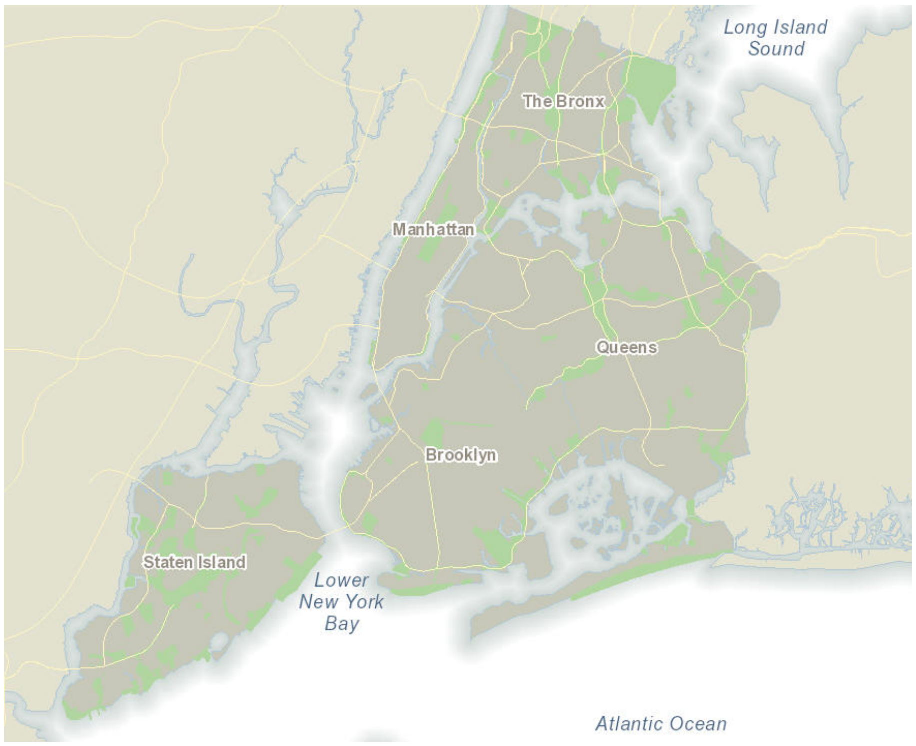


Figure 5: Gaussian Noise $N(0, k \cdot \epsilon)$ vs Loss, $k = 0.1, 0.2, ...1$

Figure 5 shows that the algorithm is not robust to noise. The loss is only acceptable for noise with a standard deviation of $0.1 \cdot \epsilon$. When the standard deviation gets $0.2 \cdot \epsilon$, the median loss become about $0.08 = 8 \cdot \epsilon$. The accuracy of GPS devices, although it varies due to the environment, can be generally considered as 10 meters [20,21]. Therefore, $8 \cdot \epsilon = 80$meters can be problematic for many applications.
It can be seen from the graphs in the appendix that, when we increase the noise, the areas near boundaries will be easily recognized as many different regions with different densities. Thus, in practice, it might be a good idea to set ϵ as large as possible, to avoid the messiness near boundaries.

6 Taxi Data Analysis

In this section, we apply our method to a New York taxi dataset to segment the map of New York City, and use a *Point of Interests (POI)* dataset to validate our map segmentation. The datasets are public on the "NYC Open Data" website [1].



[1]https://opendata.cityofnewyork.us/

The taxi trajectory contains information about the longitudes, latitudes and date-time of pick-ups and those of drop-offs. The whole dataset contains more than 150 million taxi trips, all in the year of 2014.
The POI dataset contains more than 10,000 points of interests. All the POIs are already categorized in 13 categories [2]. The dataset also contains the longitudes and latitudes of points of interests.
One of our main assumptions throughout the thesis is that, if we detect boundaries of regions where density changes rapidly, we probably also detect boundaries of regions with different functions. Therefore, the effectiveness of our method highly depends on the validity of our probabilistic assumptions both for trajectories and human traffic flows. Namely, the trajectories (as well as their origins and destinations) are uniformly distributed, and people have different probability to leave from (and arrive at) different regions.
These assumptions might not be totally true in real-world datasets. Thus, after the segmentation, we investigate the POIs within each region, to validate whether our map segmentation methods detect boundaries of regions with different functions.
For each region after the segmentation, we calculate the number of the most frequent POI category within the region, and we divide that number by the total number of POIs within this region. The result shows the "purity" of each region regarding all the POI categories.
Moreover, we construct a vector of length 13, each element of which represent the number of POIs that belong to each category within this region. Then, we calculate the cosine distance of vectors of neighboring regions, which indicates the degree of difference of POIs between neighboring regions.
Suppose we segment the map into K regions, we also make a grid with granularity $g = k \cdot \epsilon$, such that $k \in \mathcal{Z}^+, k = \arg\max_j \{S/(j \cdot \epsilon)^2 \leq K\}$, where S is the total area of the map. We compare the POIs differences and POIs purity between the map segmentation produced by our algorithm and the map segmentation by the simple grid, which can be considered as a naive baseline.
To investigate the POIs after the segmentation, we only use a subsample of the whole dataset, because too big data size will cause the parametric complexity R_K^n to be too big to be stored in the computer. We pick (arbitrarily) all the taxi trajectories on January 10, 2014, which contains more than 3.5 million taxi trajectories. We only use trajectory destinations in the data analysis because the starting points of trajectories have many missing values.

[2]Residential, Education Facility, Cultural Facility, Recreational Facility, Social Services, Transportation Facility, Commercial, Government Facility (non-public safety), Religious Institution, Health Services, Public Safety, Water, Miscellaneous

We segment the timeline into several parts, and we only use destinations points here. K_{max} is set to be 20. The map is set to be a rectangle that contains the true map of the NYC, in order to conveniently calculate the area of regions. ϵ is set to be 0.001 (about 100 meters), because the tall buildings in NYC might strongly reduce the accuracy of GPS devices [22].

time	purity	cosine dist	purity grid	cosine dist grid	data size
20-22	0.3965	0.5757	0.4301	0.4368	409875
22-02	0.4010	0.5767	0.4581	0.3858	562666
02-05	0.3872	0.5390	0.4230	0.4478	190725
05-09	0.4146	0.5719	0.4301	0.4476	397243
09-12	0.3880	0.5617	0.4354	0.4615	493083
12-14	0.4237	0.5877	0.4260	0.4357	364854
14-16	0.4180	0.5755	0.4335	0.4530	381066
16-18	0.4279	0.5695	0.4333	0.4577	360665
18-20	0.4223	0.5763	0.4434	0.4683	467915

Table 1: POI purity and difference. The purity of each region is measured by the ratio of the number of the most frequent POI category to the total number of POIs within this region. The difference between any two neighbouring regions is measured by the cosine distance of POI category vectors.

Table 1 shows that the purity of POIs within each region after the map segmentation by our algorithm is a little bit less than that of the map segmentation by a simple grid. Several possible reasons exist:

1. Most of the taxi destinations are within Manhattan, where the functions of places are super dense and complex.

2. Taxi trajectories are biased, which only reflect part of the traffic flows. It is possible that some POIs are not interesting to taxi passengers.

3. The POIs dataset is also biased, containing only big and famous public venues.

However, the cosine distances between POIs of neighboring regions show that our algorithm does produce boundaries where functions of regions are more different than those of the baseline grid.

Finally, figure 6 shows an example of the map segmentation of NYC. The segmentation in Manhattan is very dense, because of the high population density there. We also see many very small region with the side length equal to ϵ, which is probably because a very tall building within a lot of people is on that location.

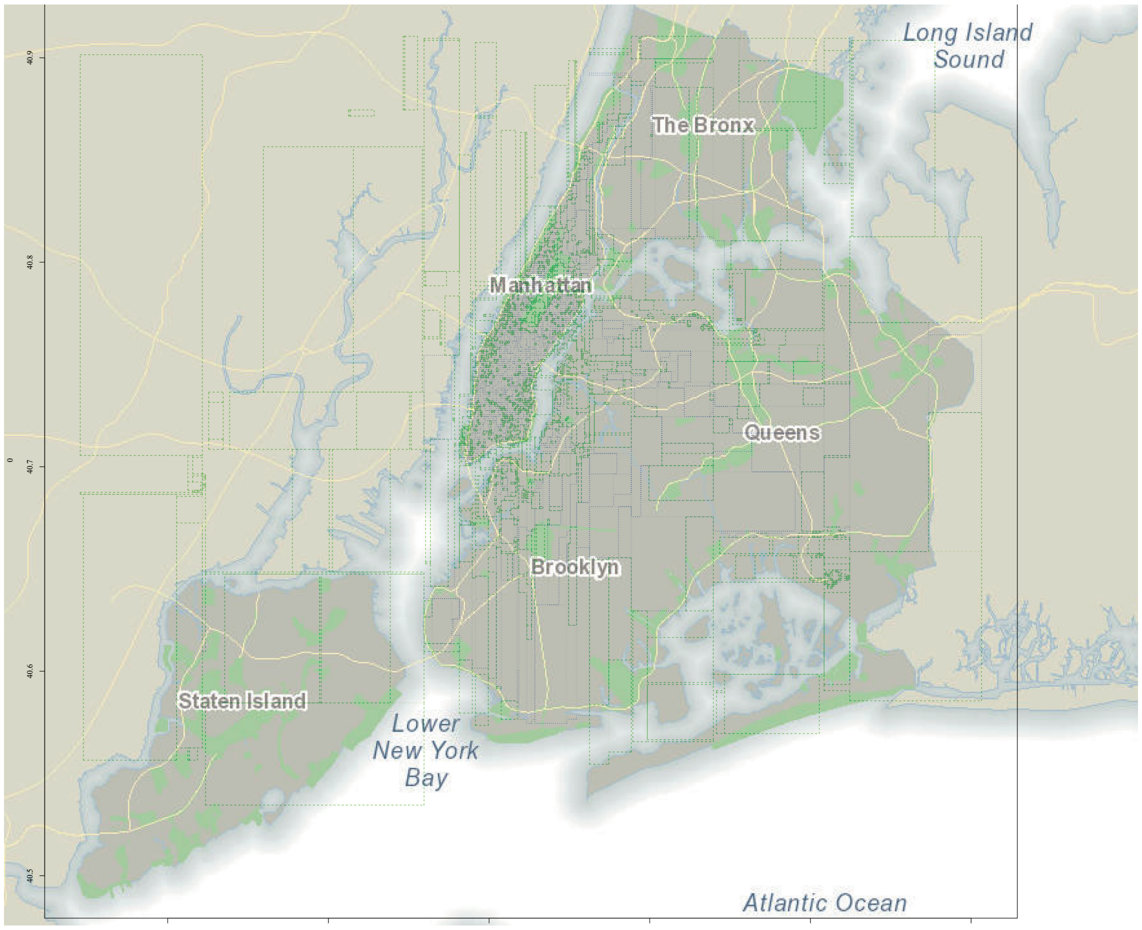


Figure 6: An example of map segmentation based on NYC taxi trajectory destinations. Blue regions are regions with more than 300 data points.

7 Related Work

The cornerstone work of pattern mining for trajectories dataset is [4], which discussed the methodology of using *Kernel Density Estimation (KDE)* to deal with the infinite

patterns problem of continuous pattern mining. A more recent research about sequential pattern mining for semantic trajectories is [14], which tries to group similar places from a totally different perspective from this thesis, as well as proposed an algorithm called *Splitter*.
Moreover, there are currently many researches about semantic trajectories mining, which use external information to annotate events behind the trajectory or the true destination of trajectories [7,12,15]. Our work can be regarded as a preprocessing step for semantic trajectory mining, without using any external information.
Previous researches about Functional Zone Identification usually brought ideas from *Text Analysis*. One typical work is [13], which firstly segments the map into small regions based on main roads and streets, then represents each region as a vector by using *Latent Dirichlet allocation (LDA)*, and finally merges neighboring regions with similar representations. Another work of region representation is [16], which combined the information from so-called spatial-graph and flow-graph.
Finally, as mentioned earlier, this thesis is mainly based on the work of MDL-histogram [1,8,9]. In addition, based on MDL-histogram, [10] extended the algorithm to 2-dimensional case by coordinate descent algorithm, in order to construct a 2-dimensional histogram for Time Series data. It should also be mentioned that our thesis is neither the first piece of work that used the MDL framework on data mining of trajectory dataset, nor the first piece of work that used MDL on data discretization. One typical previous work of MDL trajectory pattern mining is [5], which used MDL to segment the trajectory (as a curve on the plane) into sub-trajectories and group similar sub-trajectories. Also, one previous work about data discretization by the MDL [23] discretizes multivariate continuous data while keeping the correlation among different dimensions, which used a two-part encoding strategy based on the cumulative distributions (cdf) of the data.

8 Conclusions and Discussion

8.1 Conclusions

With some probabilistic assumptions for the trajectories as origin-destination pairs, we solve the map segmentation problem by detecting boundaries of regions where density changes rapidly, which is then solved by the MDL principle.
Our simulation results show that, the MDL-histogram approach works quite well for 2-dimensional data, even with our heuristics. The real world data analysis shows that our algorithm does produce regions with functions that are more different than

just making a grid.

8.2 Discussion

The most worrying assumption of our method is to assume the origins and destinations (or the trajectories) are uniformly distributed within each region. Therefore, if we detect the boundaries where the probability is not uniform but changes rapidly, we detect the boundaries of regions with different functions.
Firstly, for example, even if the density within a shopping street is not uniform, the density differences within the shopping street would probably be much less than the density differences between the shopping street and a nearby residential region.
Secondly, the boundaries of regions with different functions given by local expert can be very subjective. Some experts may segment the map into more regions than others. Actually, if the algorithm splits a region within seemingly very similar functions inside, it can mean that the algorithm thought the size of the data points here is big, so it won't hurt to zoom in and to have a fine granularity here.

8.3 Future Work

As a following work of the MDL-histogram, one of the authors extended the MDL-histogram and proposed a model called *Clustgrams* [6], which can segment 1-dimensional datasets into bins, but unlike MDL-histogram, the probability within each bin can be modeled by a larger group of probabilistic distribution families (other than just uniform distribution). According to the same strategy of this thesis, it might be interesting to also extend *Clustgrams* to 2-dimensional data, so that we can model more complex probabilistic structure of trajectories.
Moreover, the reason of the surprising success of the *Recursive Binary Search* is currently unclear to us. However, if this approach is theoretically solid under the MDL model selection framework, it will significantly reduce the algorithm complexity, both for MDL-histogram and map segmentation problem. It would even allow us to search the proper map segmentations in a much bigger model class. For example, informally speaking, we can "draw" boundaries with different angles, one for each region at one iteration.

9 References

1. Kontkanen, Petri, et al. ”1 An MDL Framework for Data Clustering.” Minimum (2005): 323.
2. Kontkanen, Petri, and Petri MyllymÃ. ”MDL histogram density estimation.” Artificial Intelligence and Statistics. 2007.
3. Grunwald, Peter D. The minimum description length principle. MIT press, 2007.
4. Giannotti, Fosca, et al. ”Trajectory pattern mining.” Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.
5. Lee, Jae-Gil, Jiawei Han, and Kyu-Young Whang. ”Trajectory clustering: a partition-and-group framework.” Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007.
6. Luosto, Panu, and Petri Kontkanen. ”Clustgrams: an extension to histogram densities based on the minimum description length principle.” Open Computer Science 1.4 (2011): 466-481.
7. Wu, Fei, and Zhenhui Li. ”Where did you go: Personalized annotation of mobility records.” Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. ACM, 2016.
8. Kontkanen, Petri, and Petri Myllymäki. ”A linear-time algorithm for computing the multinomial stochastic complexity.” Information Processing Letters 103.6 (2007): 227-233.
9. Kontkanen, Petri, and Petri Myllymäki. ”Analyzing the stochastic complexity via tree polynomials.” Unpublished manuscript (2005).
10. Kameya, Yoshitaka. ”Time Series Discretization via MDL-based Histogram Density Estimation.” Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on. IEEE, 2011.
11. Giannotti, Fosca, Mirco Nanni, and Dino Pedreschi. ”Efficient mining of temporally annotated sequences.” Proceedings of the 2006 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2006.
12. Wu, Fei, et al. ”Semantic annotation of mobility data using social media.” Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015.
13. Yuan, Nicholas Jing, et al. ”Discovering urban functional zones using latent activity trajectories.” IEEE Transactions on Knowledge and Data Engineering 27.3 (2015): 712-725.
14. Zhang, Chao, et al. ”Splitter: Mining fine-grained sequential patterns in semantic trajectories.” Proceedings of the VLDB Endowment 7.9 (2014): 769-780.
15. Ye, Yang, et al. ”Mining individual life pattern based on location history.” Mobile Data Management: Systems, Services and Middleware, 2009. MDM’09. Tenth International Conference on. IEEE, 2009.
16. Wang, Hongjian, and Zhenhui Li. ”Region representation learning via mobility flow.” Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, 2017.
17. Li, Zhenhui, et al. ”Mining periodic behaviors for moving objects.” Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010.
18. Wright, Stephen J. ”Coordinate descent algorithms.” Mathematical Programming 151.1 (2015): 3-34.
19. Zheng, Yu. ”Trajectory data mining: an overview.” ACM Transactions on Intelligent Systems and Technology (TIST) 6.3 (2015): 29.
20. Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.

21. “GPS Accuracy.” GPS.gov: Agricultural Applications, 5 Dec. 2017, ”www.gps.gov/systems/gps/performance/accuracy/”.
22. Modsching, Marko, Ronny Kramer, and Klaus ten Hagen. ”Field trial on GPS Accuracy in a medium size city: The influence of built-up.” 3rd workshop on positioning, navigation and communication. Vol. 2006. 2006.
23. Nguyen, Hoang-Vu, et al. ”Unsupervised interaction-preserving discretization of multivariate data.” Data Mining and Knowledge Discovery 28.5-6 (2014): 1366-1397.

10 Appendix: A Few Examples of Map Segmentation Simulations

We show a few examples of our simulation results. The red lines are "true boundaries", and green lines are boundaries given by our "split-and-merge" algorithm. Faded red lines and faded green lines are boundaries being removed because of regions merging during data generation and during the map segmentation algorithm, respectively.

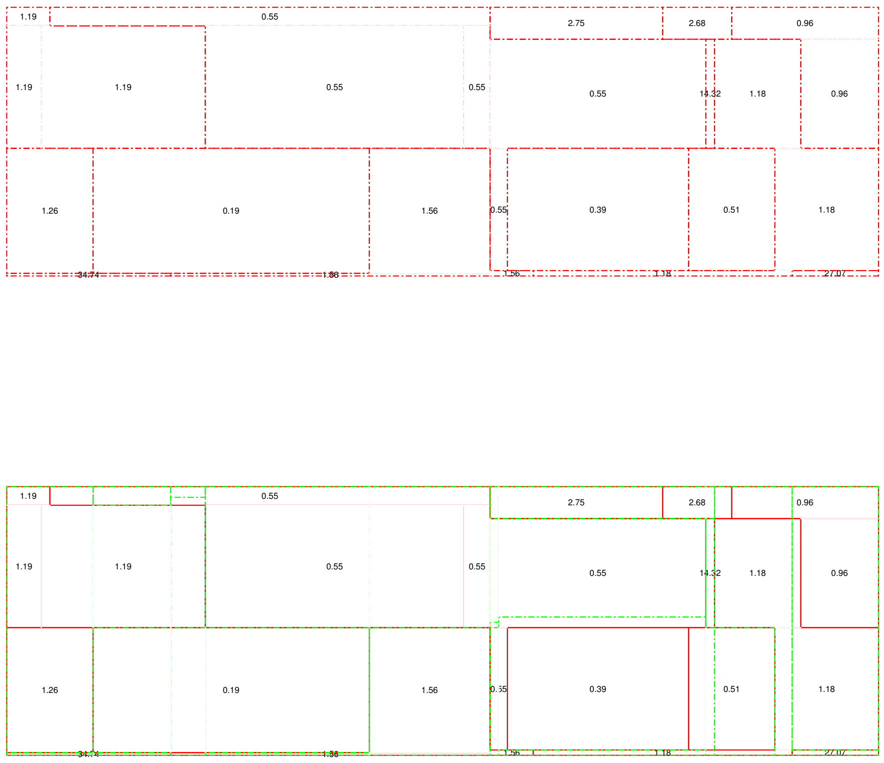
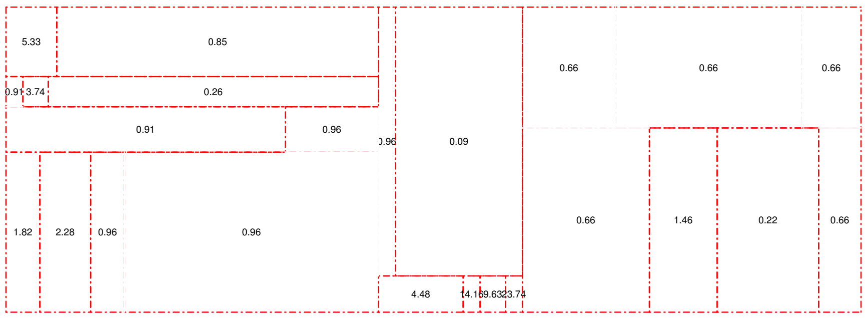


Figure 7: n = 10000, eps = 0.01, noise = 0



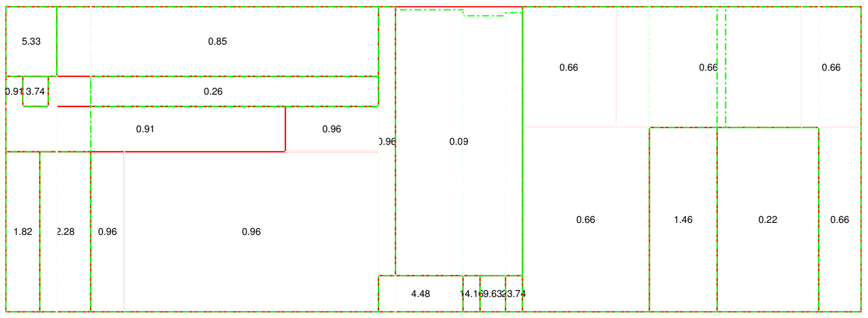
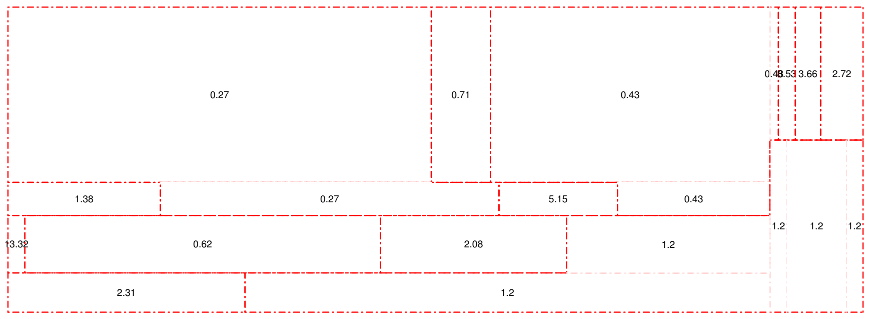


Figure 8: n = 10000, eps = 0.01, noise = 0



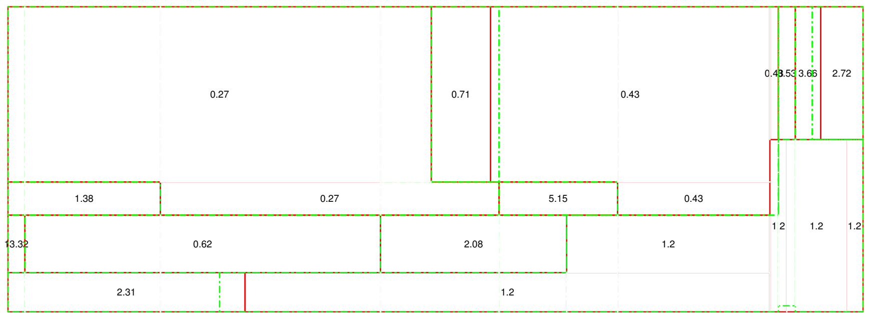
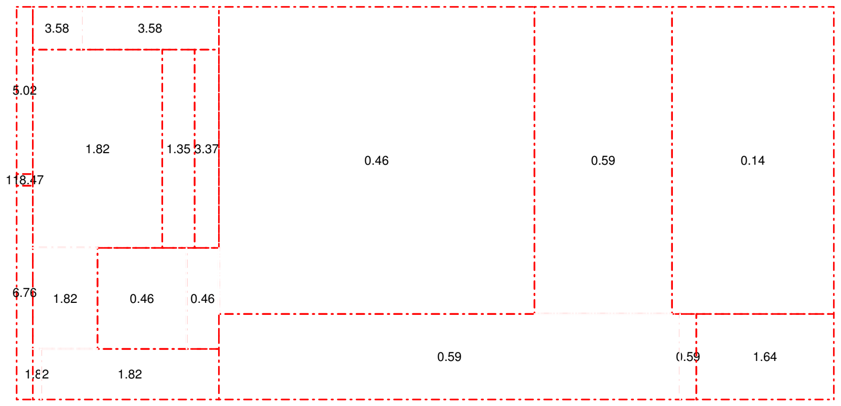


Figure 9: n = 10000, eps = 0.01, noise = 0



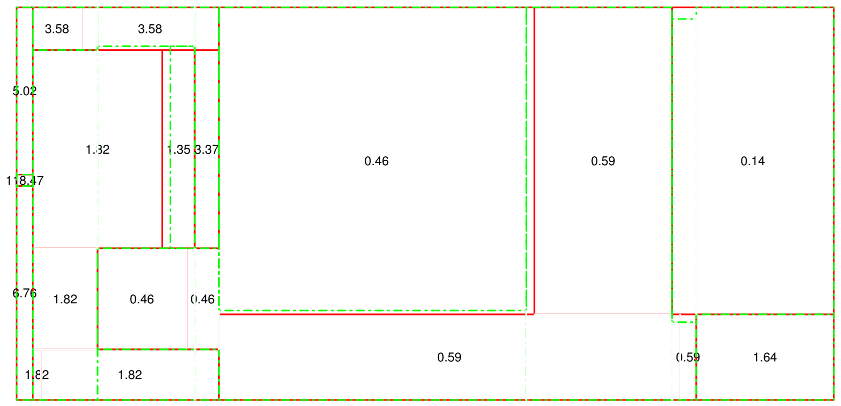
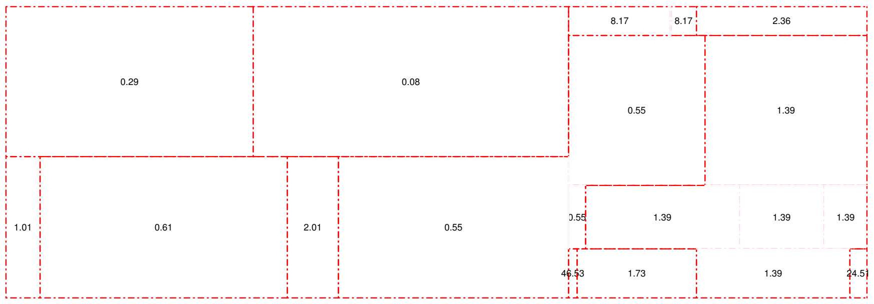


Figure 10: n = 10000, eps = 0.01, noise = $0.1 \cdot \epsilon$



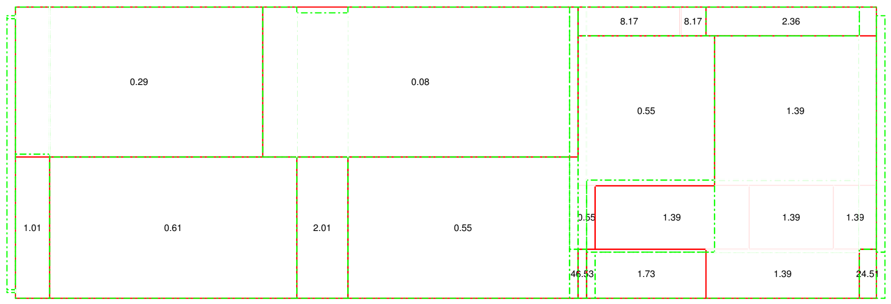
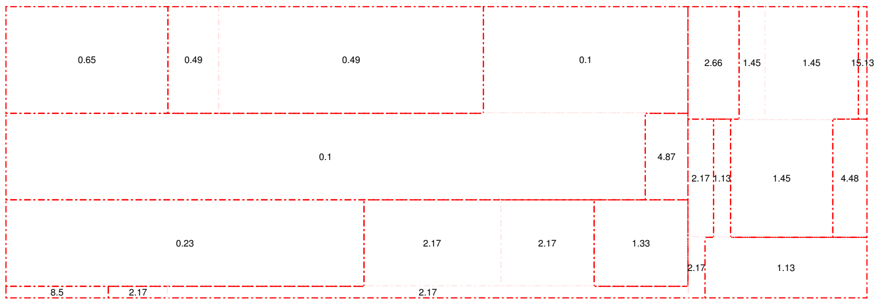


Figure 11: n = 10000, eps = 0.01, noise = $0.3 \cdot \epsilon$



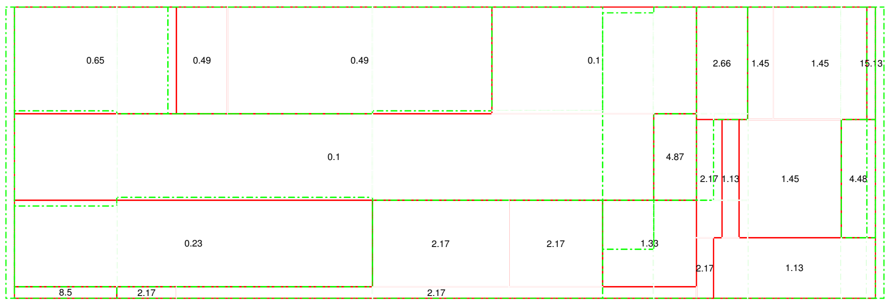


Figure 12: n = 10000, eps = 0.01, noise = $0.3 \cdot \epsilon$

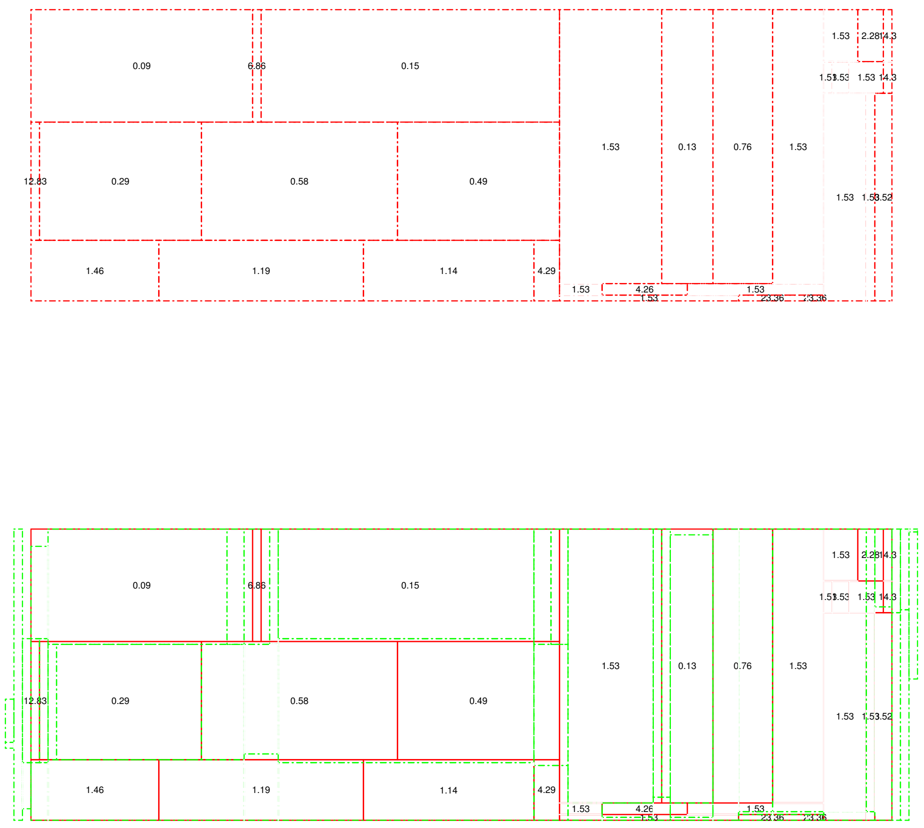


Figure 13: n = 10000, eps = 0.01, noise = $1 \cdot \epsilon$