Multiclass Density Estimation Analysis in N-Dimensional Space

featuring Delaunay Tessellation Field Estimation

Yuhao Li
Abstract

Multiclass Density Estimation Analysis in N-Dimensional Space

Yuhao Li

Multiclass density estimation is a method that can both estimate the density of a field and classify a given point to its targeted class. Delaunay Tessellation Field Estimation is a tessellation based multiclass density estimation technique that has recently been resurfaced and has been applied in the field of astronomy and computer science. In this paper Delaunay Tessellation Field Estimation is compared with other traditional density estimation techniques such as Kernel Density Estimation, k-Nearest Neighbour Density, Local Reachability Density and histogram to deliver a detailed performance analysis. One of the main conclusions is that Delaunay Tessellation Field Estimation scales in the number of data points but not the dimensions.
Acknowledgements

I would like to thank my supervisor, Michael Ashcroft, for all the help, guidance and feedback provided during the course of the project.
## Contents

1 Introduction ........................................... 3

2 General Concept .......................................... 3
   2.1 Tessellation ........................................ 3
      2.1.1 Voronoi Diagram ................................... 4
      2.1.2 Delaunay Triangulation .......................... 4
   2.2 Neighbors ........................................... 5
      2.2.1 Natural Neighbor ................................. 6

3 Target Method ............................................ 6
   3.1 Delaunay Tessellation Field Estimation .............. 6
      3.1.1 Point Density Estimation .......................... 6
      3.1.2 Interpolation using Linear Regression ............ 7
      3.1.3 Interpolation using Ridge Regression ............. 7

4 Comparison Methods ....................................... 8
   4.1 Histogram ........................................... 8
   4.2 Kernel Density Estimation ........................... 8
   4.3 K-Nearest Neighbor Density Estimation ............... 9
   4.4 Local Reachability Density ......................... 10

5 Test Explanation ........................................ 10
   5.1 Hyperparameters ..................................... 10
      5.1.1 Hyperparameter Validation ........................ 10
      5.1.2 Time Complexity of Hyperparameter Validation ... 11
   5.2 Classification ....................................... 12
   5.3 Scoring using Log Probability ........................ 12

6 Experiments .............................................. 12
   6.1 Data Used ........................................... 12
   6.2 Tools Used .......................................... 13
   6.3 Experimental Setup .................................. 13
      6.3.1 Histogram Setup .................................. 13
      6.3.2 Delaunay Tessellation Setup ..................... 14
   6.4 Result and Time Complexity .......................... 15
      6.4.1 Delaunay Tessellation Field Estimation .......... 15
      6.4.2 Histogram Density Estimation .................... 16
      6.4.3 Kernel Density Estimation ........................ 17
      6.4.4 K-Nearest Neighbor Density Estimation ........... 18
      6.4.5 Local Reachability Density Estimation ........... 18
      6.4.6 Compared Result .................................. 19

7 Discussion ............................................... 20

8 Conclusion ............................................... 21

9 Future Work ............................................. 21
1 Introduction

Multiclass Density Estimation (MDE) is a combination of machine learning and statistical learning techniques that can both estimate the density of a field and classify a given point to its targeted class. Most classifiers used in statistical learning are in fact MDE in its core. One specific technique that draws our interest is the Delaunay Tessellation Field Estimator (DTFE) first introduced by Brown[3] and Ord[6] and since then applied in different fields. In astrophysics, it is used to study the large scale of the universe[10]. In computer science it has been implemented for facial recognition[7]. The hypothesis behind this algorithm is that without any difficult hyperparameters to adjust, it can very efficiently discover regions in the input space dominated by a class of interest[9], where efficiency is measured by both accuracy and execution speed. In this paper a number of different MDE techniques are explained and experimented with to give a comparison on their performance to the contrast of DTFE. The performances that are discussed are the level of accuracy at which a certain method estimates the density, the accuracy of predicting the class and the time complexity of that method.

2 General Concept

Some general concepts that are needed to be explained for a better understanding are tessellations formed in both Voronoi diagram and its dual; Delaunay Triangulation and natural neighbors associated with these tessellations.

2.1 Tessellation

Tessellations are geometric shapes that in one dimension cover a line, two dimensions an area, three dimensions a volume and in multi-dimensions cover a hyper volume. Tessellations can also be described as a set of non overlapping polytopes filling the D-dimensional space or a subset of that space.

Three simple rules/equations can define a tessellation[9, 4], \( \mathcal{T} \), in the D-dimensional space, \( \mathbb{R}^D \), where \( \mathcal{T} \) is a set of polytopes, \( \{X_1, X_2, X_3 \ldots X_n\} \). These rules are as follow:

\[
\hat{X}_i \cap \hat{X}_j \neq \emptyset \quad \text{for} \quad i \neq j
\]  

(1)

where \( \hat{X}_i \) denotes the interior of the polytope \( X_i \). The first equation ensures that the polytopes are not overlapping.

\[
\bigcup_i X_i = \mathbb{R}^D
\]  

(2)

The second equation makes the polytopes \( X_i \) fill the D-dimensional space, \( \mathbb{R}^D \).

\[
|\{X \in \mathcal{T} | X \cap A \neq \emptyset\}| < \infty \quad \forall \quad \text{bounded} \quad A \subset \mathbb{R}^D
\]  

(3)

The third and final equation makes sure that the number of polytopes does not exceed infinity and is always countable.

Tessellations created from a given set of data points can be extremely useful when trying to estimate the density of a field. There are many different types of
tessellations that can be created from a set of data points and the most common ones are Voronoi diagrams and Delaunay triangulation.

2.1.1 Voronoi Diagram

Voronoi diagrams also known as Dirichlet tessellations are a type of tessellation that provides proximity information which proves to be useful when interpreting questions like:

- Which WiFi-spot has the highest signal from any given point?
- Where is the nearest restaurant from my location?

Formally, for a given set of finite sites, \( \mathcal{P} \in \mathbb{R}^D \), Voronoi diagram can be defined as the partition of the space, \( \mathbb{R}^D \), into none-overlapping polytopes, called Voronoi cells, \( \mathcal{V}_i \). Each \( \mathcal{V}_i \) consists of all points that are closer to the site, \( \mathcal{P}_i \), than any other sites and can be generated by the equation below:

\[
\mathcal{V}_i = \{ x \mid d(x, \mathcal{P}_i) < d(x, \mathcal{P}_j) \forall j \neq i \}
\]

where \( d(x, \mathcal{P}) \) is the distance between a point to an existing site. From figure 1 below, we can easily differentiate where the point \( x \) has its closest site, by the edges of the Voronoi cell, \( p \).

![Voronoi Tessellation created by RStudio.](image)

2.1.2 Delaunay Triangulation

The polytopes of a Delaunay triangulation in \( \mathbb{R}^D \) have the property of always containing \( D + 1 \) points, \( \mathcal{P}_i : \{x_{i,1}, x_{i,2}, \ldots, x_{i,D+1}\} \), when connected to form a Delaunay simplex, \( \mathcal{D}_i \). In \( \mathbb{R}^2 \) the simplex is a triangle with 3 points and in \( \mathbb{R}^3 \) is a tetrahedron connected by 4 points. The circumscribed sphere surrounding \( \mathcal{D}_i \) does not contain any other points beside the set of points, \( \mathcal{P}_i \). Unlike the Voronoi diagram a set of Delaunay simplexes that together form the Delaunay
A Delaunay triangulation is heavily related to a Voronoi diagram in the sense that a vertex of the Voronoi diagram in $\mathbb{R}^D$ is the point equidistant to the points, $P_i$, that form the Delaunay simplex. This also implies that Delaunay triangulation is simply a dual graph to a Voronoi diagrams, i.e. for each cell of the Voronoi diagram there is a vertex on the Delaunay triangulation that is connected through the edges of each Voronoi cell as shown in figure 3.

2.2 Neighbors

The term neighbors in Machine Learning usually refers to the nearest neighbors, meaning a certain number, $k$, of data points surrounding one specific point.
hence k-Nearest Neighbors (kNN). However when dealing with tessellations, using natural neighbors is a more frequent approach.

### 2.2.1 Natural Neighbor

In Voronoi diagram natural neighbors are a pair or a set of points whose Voronoi cells have a facet in common. This in Delaunay triangulation is the set of points that are connected by the edges. From figure 4 below, point, \( x \), has the natural neighbors of all \( n_i \).

When a new point, \( x_* \), is added to the existing dataset, a new Voronoi tessellation can be created around \( x_* \) using formula 4. The natural neighbors of \( x_* \) can also be described as the points that would lose area to \( x_* \). This concept is important to note since it can be used in natural neighbor interpolation.

![Figure 4: Natural Neighbors in respective tessellations](image)

### 3 Target Method

#### 3.1 Delaunay Tessellation Field Estimation

DTFE is one of the tessellation based methods of estimating densities. The algorithm is mainly divided into two steps. Let \( \mathcal{P} : \{x_1, x_2, ..., x_n\} \) be a set of random variables. The first step is to estimate the point density in each \( x_i \in \mathcal{P} \). When every point density is calculated, an interpolation on these points using any arbitrary regression technique is performed to form a volume-covered field. This field will have the probability density of any given new point within this field.

#### 3.1.1 Point Density Estimation

Point density of \( x_i \in \mathcal{P} \) is estimated by the inverse of the contiguous Voronoi cell[12]. Let \( x_i \) be one of the vertices that form the union, \( N \), of all surrounding Delaunay simplexes. A contiguous Voronoi cell is then defined as \( D : \{d_1 \cup d_2 \cup \cdots \cup d_n\} \in N(x_i) \). If points located on the convex hull, \( CH(\mathcal{P}) \) is estimated as 0, the density of a point, \( x_i \) can therefore be calculated as:
\[ \hat{p}(x_i) = \begin{cases} \left( \sum_{d_j \in N(x_i)} \text{vol}(d_j) \right)^{-1} & \text{if } x_i \notin CH(P) \\ 0 & \text{otherwise} \end{cases} \] (5)

where \( \text{vol}(d_j) \) is the volume of one of the Delaunay simplexes having \( x_i \) as one of its vertices.

### 3.1.2 Interpolation using Linear Regression

Linear regression is a technique implemented to model the relationship between the independent variables, \( X : \{x_1, x_2, x_3, \ldots, x_n\} \), where \( x_i \) is a vector in its dimension and the dependent variable, \( Y \). It assumes the relationship between \( X \) and \( Y \) is linear, i.e. in \( \mathbb{R}^2 \) a line with function \( y = \alpha + \beta x \) and in \( \mathbb{R}^D \) a hyperplane such that:

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n = X^T \beta \] (6)

where \( \beta \) is the linear regression coefficient.

Consider a Delaunay tessellation, \( D : \{d_1, d_2, x_3, \ldots, d_n\} \), built with a set of points, \( P \) with \( d_i \) as its simplex. Each \( d_i \) consists of \( D + 1 \) vertices denoted as \( \mathcal{X} \). For each \( d_i \) a linear regression coefficient, \( \beta \) can be generated using ordinary least square method:

\[ \beta = (\mathcal{X}^T \mathcal{X})^{-1} (\mathcal{X}^T \mathcal{Y}) \] (7)

where \( \mathcal{Y} \) is the estimated density of each vertex, \( \mathcal{X} \). This minimizes the sum of mean square errors of the estimated density, \( \hat{Y} \), to the actual density, \( Y \).

\[ \min \sum (\hat{Y} - Y)^2 \Rightarrow \min \sum (\mathcal{X}^T \beta - Y)^2 \] (8)

To estimate a new point \( x_* \notin P \), a Delaunay simplex, \( d_* \), that contains the point \( x_* \) needs to be located and \( \beta_* \) is calculated accordingly. If \( x_* \) is on a boundary between Delaunay simplexes, any adjacent simplex can be chosen. If \( x_* \) is outside the convex hull, \( CH(P) \), the density is enforced to \( 0 \). The estimated density of \( x_* \) is then:

\[ \hat{p}(x_* \notin P) = \begin{cases} \beta_*^T \cdot x_*^{-1} & \text{if } \exists d_i \in D(P)x_* \in d_i \\ 0 & \text{otherwise} \end{cases} \] (9)

### 3.1.3 Interpolation using Ridge Regression

Ridge regression is similar to linear regression where the difference lies in the way to calculate the regression coefficient. By adding a \( \lambda \) parameter to formula 7, the new regression coefficient can be calculated as:

\[ \beta = (\mathcal{X}^T \mathcal{X} + \lambda I)^{-1} (\mathcal{X}^T \mathcal{Y}) \] (10)

where \( I \) is an identity matrix together with the \( \lambda \) parameter forming an identity matrix with \( \lambda \) as its diagonal. It is now minimizing the sum of the mean square errors of \( \hat{Y} \) and \( Y \) with a sum of \( \beta^2 \) of \( \lambda \), i.e. \( \min \sum (Y - \hat{Y})^2 + \lambda \sum \beta^2 \). where \( \lambda \sum \beta^2 \) is a penalty term also known as the L2 regularization.

Adding the \( \lambda \) term also ensures that the formula \( \mathcal{X}^T \mathcal{X} + \lambda I \) is always invertible and is very useful when dealing with data points that are closely clustered.
4 Comparison Methods

4.1 Histogram

Given a set of data points, a histogram can visually represent the frequency distribution of these points. To create a histogram, essentially data binning is performed over the range of values to produce bins, (data binning is to divide the range of values into a series of intervals that may or may not be equal sized) and then calculate how many of these points fall into each bin.

In other terms, histograms can also be defined as \( H : \{ m_1, m_2, m_3, \ldots, m_n \} \), where \( m_i \) is the count of points that fall into each disjoint bin, known as the frequency. Let \( n \) be the total number of data points and \( k \) be the total number of bins, the histogram, \( H \), needs to meet the following condition[8]:

\[
   n = \sum_{i=1}^{k} m_i
\]

(11)

In order to estimate the density of any given point, \( p \). Firstly a histogram model of a set of points needs to be constructed following steps previously discussed. Let \( m_i \) be the frequency of each bin, the density of point, \( p \), can be calculated as follows:

\[
   H(p) = \frac{m(p)}{(\sum_{i=1}^{n} m_i) \cdot \text{area}}
\]

(12)

To summarize, the density is simply the frequency at the given point over all other frequencies times the total area of the histogram.

4.2 Kernel Density Estimation

Kernel Density Estimation (KDE) is one of the most common used density estimation techniques and it is closely related to histogram. Like the name suggest, KDE uses kernels in order to estimate the density of a point. Kernels are functions that have the properties that are continuous, symmetric and integrate to one. A range of different kernel function can be used: uniform, triangular, Epanechnikov, logistic and many more. However the most common one is the Gaussian kernel and has the formula:

\[
   K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}
\]

(13)

where \( u \) is the distance function between two points. A graphic representation of different kernel functions can be seen in figure 5.

Now let \( P : \{ x_1, x_2, x_3, \ldots, x_n \} \) be a set of randomly distributed points in the \( \mathbb{R}^D \). For a new point, \( x_\ast \), the density of that point can be estimated with the formula:

\[
   \hat{f}_h(x_\ast) = \frac{1}{n} \sum_{i=1}^{n} K_h(x_\ast - x_i) = \frac{1}{nh} \sum_{i=1}^{n} K\left( \frac{x_\ast - x_i}{h} \right)
\]

(14)

where \( K \) is the kernel function and \( h \) is the bandwidth for smoothing. The bandwidth decides the height and width of the kernel function and needs to be
fine-tuned for optimal results. A difference in bandwidth of a Gaussian Kernel is illustrated in figure 6 below. This function, in other words, interpolates over the given points, \( P \), to form a density graph when enough new points are drawn.

![Different types of Kernel](image)

(a) Triangular Kernel  (b) Gaussian Kernel  (c) Epanechnikov Kernel

Figure 5: Different types of Kernel

![Gaussian Kernel with different bandwidth](image)

(a) Bandwidth: 0.2  (b) Bandwidth: 0.5  (c) Bandwidth: 1

Figure 6: Gaussian Kernel with different bandwidth

### 4.3 K-Nearest Neighbor Density Estimation

The use of kNN to estimate the density of a given point, \( x \) in \( \mathbb{R}^D \) is to first determine the k-distance, \( d_k \), i.e. the distance between the point \( x \) and the \( k \)-th closest neighbor. A surrounding volume is then created using the \( d_k \) as its radius to form a hyper sphere. With the help of the hyper sphere, the density, \( K(x) \), can be estimated as follows\[11\]:

\[
K(x) = \frac{k}{N \cdot v_D \cdot d_k^D(x)}
\]

where \( N \) is the total number of points, \( d_k^D(x) \) is the k-distance of point \( x \) and \( v_D \) is the volume of a unit sphere in \( \mathbb{R}^D \) which is calculated as:

\[
v_D = \frac{\pi^{D/2}}{\Gamma(D/2 + 1)}
\]
4.4 Local Reachability Density

Local Reachability Density method is partially based on K-Nearest Neighbor Density Estimation method and is often used to determine whether a point is a local outlier. Similar to kNN density estimation a distance between a given point $A$ and an existing point $B$, called reachability distance, needs to be calculated in before hand as such:

$$reachability.distance_k(A, B) = \max[d_k(B), d(A, B)]$$

where the reachability distance between point $A$ and $B$ is the maximum value of $k$-distance of $B$ and the actual distance from $A$ to $B$. Once the reachability distance is given the density of any given point can be found using the equation below:

$$lrd(A) := 1/\left(\sum_{B \in N_k(A)} reachability.distance_k(A, B) / |N_k(A)|\right)$$

The local reachability density in short is the inverse of sum of the reachability distance from a new point to all its $k$-nearest neighbors, over the total number of neighbors.

5 Test Explanation

5.1 Hyperparameters

Hyperparameters also referred to as meta-parameters and free parameters are a set of adjustable variables that fine-tunes the statistical learning models. The prefix "hyper-" or "meta-" is explicitly pointing out that the variables are not a primary parameter, such as the training data, that the models learns from during the training phase, but rather parameters of the model generation algorithm.

Different hyperparameters are treated differently in the each of the methods that are experimented on. In KDE both the kernel and the bandwidth are the hyperparameters. After testing various kinds of kernels such as: logistic, Epanechnikov and Gaussian, the difference was not significant and the more traditional Gaussian kernel was chosen. In kNN based methods, i.e. kNN density estimation and local reachability density, the hyperparameter is the $k$-nearest neighbor where the number of neighbors selected is a dominate factor when determine the density of a point. The histogram is an algorithm that has the number of bins and bin size as its hyperparameter. In my implementation of histogram, the size of the bin is determined by dividing the number of bins over the range of the data. Therefore, only the number of bins needs to be tuned.

5.1.1 Hyperparameter Validation

In order to find the optimal values as hyperparameters, a combination of k-fold cross-validation and grid search is performed on each of the algorithms.

K-fold cross-validation is a common technique used to validate a performance of a certain model. By dividing the dataset, $D$, into $k$ parts such that $\{s_1, s_2, \ldots, s_k\} \in D$ where $s_i$ is a subset of $D$. For each subset $s_i$, use all
but $s_i$ as training data and test on subset $s_i$. The average error over these $k$
experiments scoring the performance can then be calculated.

Grid search is a search method that systematically locates the optimal value
of a hyperparameter. To start, a grid of ordering values representing possible
hyperparameters is created. In the case where only one hyperparameter is eval-
uated, the grid is a 1-dimensional vector of a minimum of 3 values. A k-fold
cross-validation is then executed on each of the values in the grid to produce a
performance score of each model. The best performed hyperparameter is then
"zoomed in" and a new grid is provided to search for a more optimal value if
any. This procedure is done until the difference in performance is little to none.

The term "zoomed in", in a grid consist of 3 continuous variables can be
explained with 3 simple formula. Let the values $h_1, h_2, h_3$ be the set of hyper-
parameters ordered from minimum to maximum. If $h_1$ gets the best score the
new grid is generated as follow:

\[ |h_1 - \frac{|h_1 - h_2|}{2}|, \quad h_1, \quad h_2 \]

If $h_2$ receives the highest score, the formula would be:

\[ |h_2 - \frac{|h_1 - h_2|}{2}|, \quad h_2, \quad h_2 + \frac{|h_2 - h_3|}{2} \]

and lastly if $h_3$ scores the best, the following formula is used:

\[ h_2, \quad h_3, \quad h_3 + \frac{|h_2 - h_3|}{2} \]

The same concept can be applied to discrete variables where the result is rounded
to a discrete number.

The bandwidth of KDE can be a continuous value and often tends to be
a number between 0 and 2 if the data is scaled and centered. The starting
grid selected is therefore $10^{-5}$, $10^{-4}$ and $10^{-3}$. On the contrary the number
of neighbors and the bin size are discrete variables that depends on the data, thus
the starting grids are 1, 5 and 10 for each of the algorithms respectively.

5.1.2 Time Complexity of Hyperparameter Validation

The time complexity of k-fold cross-validation is considered as linear to the
number of folds as long as the method in evaluation is performing at a constant
time. However the time complexity of a grid search is more difficult to determine.
A k-fold cross-validation is executed for each step of the search. If the cross-
validation has $O(n)$ time where $n$ is the number of folds, the time complexity
of a grid search would be:

\[ s \cdot O(n) \]

where $s$ is the number of steps taken. The steps are non-deterministic and can
vary from search to search, since it heavily depends on both the values of the
starting grid and luck.
5.2 Classification

When dealing with multiple classes, instead of only building one model, a model for each class needs to be constructed. For a new point, \( x_\ast \), the density probability of \( x_\ast \) is evaluated on each of the models. The target class of \( x_\ast \) is then the class model with the highest density.

5.3 Scoring using Log Probability

The most common scoring method used on cross-validations are the sum of the mean square error (MSE) or mean absolute error (MAS). However, these methods are only applicable when the real data are provide to compare the estimated result with the actual result. In the case of density estimation, there are often no actual data to compare with. Instead, it is possible to multiply the probability density at each point to produce the possibility of seeing every point. This also implicates that the product will quickly converge to 0 when a large number of probabilities are in use.

Alternatively, by taking the log of the probability product, each probability density at every point can then be added instead, using the logarithmic law in formula 23. It can then produce a result that can easily be compared even if a huge dataset is given. This also means that the higher the log probability score is, the more accurate the model predicts.

\[
\log(A \cdot B) = \log(A) + \log(B)
\]  

(23)

The log probability scoring method has the downside that if the density of a point is 0, the score of that point will become negative infinity. Thus adjustments are needed for some algorithms for the experiments to be both compatible and comparable.

6 Experiments

6.1 Data Used

The data that are experimented on is a sample of police reports of crimes over The Great Britain. The crime details are provided by the 43 geographic police forces in England and Wales, the British Transport Police, the Police Service of Northern Ireland and the Ministry of Justice[1].

The sample used is from September 2014 given by Avon and Somerset Constabulary. Three main features and 10000 data points were extracted from the dataset and are applied in each model. The longitude as the x-axis, the latitude as the y-axis and the crime type as the classes. Figure 7 illustrates these points of crimes in a 2-dimensional Euclidean space.

To analyze D-dimensional data, no appropriate real-life dataset was found. The data that are used for 3-dimensional and 4-dimensional tests are simply 1000 data points drawn from a multivariate Gaussian distribution and is divided into three different classes. These points of the 3-dimensional dataset can be seen in figure 8 below.
6.2 Tools Used

All implementations and experiments are conducted in R-language using RStudio as IDE. The comparison algorithms as well as the various validation techniques mentioned are implemented from scratch based on the theory explained previously, while the DTFE implementation is done by Mike Ashcroft with the help of geometry library from C++.

6.3 Experimental Setup

Each algorithm is evaluated with the help of k-fold cross-validation using data provided above. For each fold the optimum hyperparameter of the classes is calculated respectively, using the training dataset produced by the k-fold technique. The number of folds is decided depending on the amount of data. The data of the police report have a huge amount of data points and does not require as many evaluation iterations as the 3/4-dimensional test data. Thus, for the 2-dimensional data 2-fold, 3-fold and 4-fold is tested while for 3/4-dimensional data, 5-fold, 7-fold and 10-fold are chosen to be analyzed. The test scenario is experimented 5 times each to provide an average log probability score and run-time speed.

When building a multiclass model, the log probability represents the score of both the accuracy of the point density estimation and the classification prediction. However, as mentioned above log probability will give misleading results for points that have the density of 0. Algorithms that has the possibility of granting such result are histogram and DTFE and needs to be modified.

6.3.1 Histogram Setup

If the estimation point is outside the histogram model, the frequency of that point is then 0. This, according to formula 12 will result a none existing density at that point, thus the log probability score can not be determined correctly.

To solve this problem an outer area is added. Let $B$ denotes the number of bins created in the histogram model. The outer area can be calculated as such:
where $C$ is a constant that decides the size of the outer area and heavily depends on how the estimation points are distributed (in my implementation the $C$ is set to 1000). The outside area is considered as one large bin and the frequency is set to 1. This is justified by treating the histogram as a multinomial distribution and applying basic Bayesian methodologies. This methodology implies that to even this out, each bin in $B$ also needs to increase its frequency count by 1. All points that falls in the outer area will have the density that can be calculated as:

$$\text{Outerdensity} = \frac{1}{\sum_{i=1}^{n} m_i \cdot (\text{Outerarea} \cup \text{Innerarea})}$$

where $m_i$ is the frequency of each bin and the $\text{Innerarea}$ is simply the area of $B$.

### 6.3.2 Delaunay Tessellation Setup

Similar complication affects the DTFE. By looking at equation 9 from the DTFE section, any estimation points on or outside the convex hull is defined as 0. The solution to this shares the similarity with the solution of the histogram.

Let $P$ denotes the given data points. Instead of creating a Delaunay tessellation from $P$, four new dummy points, $x_1, x_2, x_3, x_4$, outside the range of $P$ can be added. The new tessellations is then generated from $\{P, x_1, x_2, x_3, x_4\}$. As
figure 9 shows, the dummy points now create the new convex hull, $\tilde{D}$, and the density of these points are set to 0 by formula 5. This expands the estimation range and estimation points outside the old convex hull (but inside $\tilde{D}$) can now be calculated using any regression technique on the Delaunay simplex that the point falls into.

![Delauny Tessellation with an outer hull.](image)

This indubitably is not a consistent solution as the estimation points can still be located outside $\tilde{D}$. To make this work internal knowledge and prior information is needed to judge the range when creating the dummy points. In my implementation, the range of the dummy points are set as follow:

\[
\begin{align*}
x_1 &= (\min(P_x) - (\max(P_x) - \min(P_x)), \max(P_y) + (\max(P_y) - \min(P_y))) \\
x_2 &= (\max(P_x) + (\max(P_x) - \min(P_x)), \max(P_y) + (\max(P_y) - \min(P_y))) \\
x_3 &= (\min(P_x) - (\max(P_x) - \min(P_x)), \min(P_y) - (\max(P_y) - \min(P_y))) \\
x_4 &= (\max(P_x) + (\max(P_x) - \min(P_x)), \min(P_y) - (\max(P_y) - \min(P_y)))
\end{align*}
\]

where $P_x$ are the points on the x-coordinate of the given data while $P_y$ denotes the points on the y-coordinate.

Also ridge regression is eventually the chosen regression technique oppose to linear regression since the vertex matrix might not be computational invertible due to closely clustered points.

6.4 Result and Time Complexity

6.4.1 Delaunay Tessellation Field Estimation

The DTFE shows promising result using the dataset from the police report. Both the accuracy and especially the run-time speed is exceptionally good. However, when it comes to dealing with the dataset randomly drawn from a probability distribution, it falls off on its performance. One thing to notice is that even though the difference of the accuracy between 3-D data and 4-D data
is considerably small, the run-time speed has noticeably increased, thus does not scale into higher dimensions. A increase in dimensions means it is more computational heavier to create the tessellations and to locate a estimation point to its simplex.

Result tables:

### Table 1: Delaunay tessellation model build upon 2-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>67724.51</td>
<td>1.335 minutes</td>
<td>2-Folds</td>
</tr>
<tr>
<td>2</td>
<td>69330.80</td>
<td>1.869 minutes</td>
<td>3-Folds</td>
</tr>
<tr>
<td>3</td>
<td>68891.43</td>
<td>2.064 minutes</td>
<td>4-Folds</td>
</tr>
</tbody>
</table>

### Table 2: Delaunay tessellation model build upon 3-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-8829.52</td>
<td>1.869 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-8739.36</td>
<td>2.295 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-8712.65</td>
<td>2.842 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

### Table 3: Delaunay tessellation model build upon 4-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-9720.57</td>
<td>11.596 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-9669.69</td>
<td>13.675 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-9195.04</td>
<td>15.482 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

Time complexity:

- Creating Delaunay tessellation: $O(n \cdot D \cdot \log(n))$ where $D$ is the number of dimensions
- Locating new point to its simplex: $O(D \cdot \sqrt{n})$
- Interpolation of the vertices: $O(n)$

#### 6.4.2 Histogram Density Estimation

The result of histogram is more or less expected. A histogram is a primitive method of estimating the density and only acts as a baseline when compared to other sophisticated methods. From the result table below, it can be observed that both the accuracy and run-time speed quickly drops off when transitioning from 3-D to 4-D, therefore does not scale into higher dimensions.

Result tables:

### Table 4: Histogram model build upon 2-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-52961.63</td>
<td>15.818 minutes</td>
<td>2-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-53574.88</td>
<td>16.445 minutes</td>
<td>3-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-53340.67</td>
<td>18.923 minutes</td>
<td>4-Folds</td>
</tr>
</tbody>
</table>


Table 5: Histogram model build upon 3-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-15462.76</td>
<td>3.955 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-18412.73</td>
<td>7.439 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-18132.65</td>
<td>9.643 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

Table 6: Histogram model build upon 4-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-22226.61</td>
<td>53.04 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-17831.86</td>
<td>1.695 hours</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-19025.97</td>
<td>2.253 hours</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

Time complexity:

- Creating histogram model: \( O(n^{D-1}) \) where \( D \) is the number of dimensions
- Locating new point to its target bin: \( O(n) \)
- Calculating its density: \( O(n^{D-1}) + O(n) + O(n) = O(n^D) \)

6.4.3 Kernel Density Estimation

KDE has some difficulties when dealing with the police dataset. The accuracy is somewhat awful and on top of that the time complexity of this algorithm is high. On the other hand points that are drawn from a probability distribution can be handled relatively well. The performance using 3-D data and 4-D data is very similar and can safely say it undeniably scales in dimensions.

Result table:

Table 7: KDE model build upon 2-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-49288.44</td>
<td>3.943 hours</td>
<td>2-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-49433.22</td>
<td>4.381 hours</td>
<td>3-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-49301.07</td>
<td>4.829 hours</td>
<td>4-Folds</td>
</tr>
</tbody>
</table>

Table 8: KDE model build upon 3-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3660.44</td>
<td>11.165 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-3630.47</td>
<td>19.102 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-3763.97</td>
<td>30.284 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

Table 9: KDE model build upon 4-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3322.72</td>
<td>12.831 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-3750.49</td>
<td>26.122 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-3770.25</td>
<td>31.404 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>
Time complexity:

- Distance function and Kernel function: $O(n)$
- Creating model and interpolate: $O(n) \cdot O(n) = O(n^2)$

### 6.4.4 K-Nearest Neighbor Density Estimation

KNN method received the best accuracy performance while testing with data from the police report. It seems that KNN is more applicable to datasets that are closely clustered. When it comes to probability distributed data, a worse result was obtained, but makes it up with its fast run-time speed and scalability.

Result tables:

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-6431.79</td>
<td>2.756 hours</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-6523.92</td>
<td>4.802 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-6016.85</td>
<td>7.873 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

### 6.4.5 Local Reachability Density Estimation

The performance of the local reachability density algorithm is the surprise factor of this experiment. It performed comparably well with the 2-D dataset and received a outstanding score using the 3-D and 4-D dataset. Consider that LRD is only used to determine local outlier factors, it is astonishing to see that it can outclass other more common density estimator in use, i.e. KDE and KNN in terms of accuracy. Unfortunately the time complexity of this algorithm is at the same level as KDE, but nonetheless, LRD can/should be explored and experimented with further to give a more detailed explanation of the cause of its performance.
Result tables:

Table 13: LRD model build upon 2-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44465.36</td>
<td>4.063 hours</td>
<td>2-Folds</td>
</tr>
<tr>
<td>2</td>
<td>44130.97</td>
<td>4.938 hours</td>
<td>3-Folds</td>
</tr>
<tr>
<td>3</td>
<td>44201.28</td>
<td>5.443 hours</td>
<td>4-Folds</td>
</tr>
</tbody>
</table>

Table 14: LRD model build upon 3-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2097.41</td>
<td>10.286 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-2127.11</td>
<td>18.496 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-2384.43</td>
<td>29.822 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

Table 15: LRD model build upon 4-D dataset

<table>
<thead>
<tr>
<th>Attempts</th>
<th>Log Probability</th>
<th>Run-time</th>
<th>K-Folds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2608.51</td>
<td>10.542 minutes</td>
<td>5-Folds</td>
</tr>
<tr>
<td>2</td>
<td>-2547.81</td>
<td>19.020 minutes</td>
<td>7-Folds</td>
</tr>
<tr>
<td>3</td>
<td>-2656.68</td>
<td>29.872 minutes</td>
<td>10-Folds</td>
</tr>
</tbody>
</table>

Time complexity:

- Finding the reachability distance: $O(n) + O(n^2) = O(n^2)$
- Creating model and interpolate: $O(n^2) + O(n) = O(n^2)$

6.4.6 Compared Result

Table 16: Comparison result using 2-D dataset under 3-fold crossvalidation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Log Probability</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTFE</td>
<td>69330.80</td>
<td>1.869 minutes</td>
</tr>
<tr>
<td>Histogram</td>
<td>-53574.88</td>
<td>16.445 minutes</td>
</tr>
<tr>
<td>KDE</td>
<td>-49243.22</td>
<td>4.381 hours</td>
</tr>
<tr>
<td>KNN</td>
<td>122066.3</td>
<td>1.764 hours</td>
</tr>
<tr>
<td>LRD</td>
<td>44104.97</td>
<td>4.938 hours</td>
</tr>
</tbody>
</table>

Table 17: Comparison result using 3-D dataset under 10-fold crossvalidation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Log Probability</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTFE</td>
<td>-8712.65</td>
<td>2.842 minutes</td>
</tr>
<tr>
<td>Histogram</td>
<td>-18132.65</td>
<td>9.643 minutes</td>
</tr>
<tr>
<td>KDE</td>
<td>-3703.97</td>
<td>30.284 minutes</td>
</tr>
<tr>
<td>KNN</td>
<td>-6016.85</td>
<td>7.873 minutes</td>
</tr>
<tr>
<td>LRD</td>
<td>-2384.43</td>
<td>29.822 minutes</td>
</tr>
</tbody>
</table>

Table 18: Comparison result using 4-D dataset under 10-fold crossvalidation

19
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Log Probability</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTFF</td>
<td>-9195.04</td>
<td>15.481 minutes</td>
</tr>
<tr>
<td>Histogram</td>
<td>-19025.97</td>
<td>135.18 minutes</td>
</tr>
<tr>
<td>KDE</td>
<td>-3730.25</td>
<td>31.404 minutes</td>
</tr>
<tr>
<td>KNN</td>
<td>-6226.72</td>
<td>7.940 minutes</td>
</tr>
<tr>
<td>LRD</td>
<td>-2656.68</td>
<td>29.872 minutes</td>
</tr>
</tbody>
</table>

Figure 10 to figure 13 is a visual representation of the result from tables above.

![Figure 10: Level of accuracy using data points from police report.](image)

(a) 3-D dataset  
(b) 4-D dataset

Figure 11: Level of accuracy using data point drawn from Gaussian distribution

![Figure 12: Run-time speed using data point from police report.](image)

7 Discussion

A flaw that can be identified within this project is that the method used to validate the hyperparameters. Grid search is indeed a valuable tool for locating the optimum hyperparameter. However, since the steps in the search are non determinstic as previously discussed, the run-time speed can vary a great deal. Another more advantageous approach is to search with the Bayesian Optimization method. The theory of this method has been researched by Snoek, Larochelle and Adams in 2012[5] and proves to be more efficient. The reason not using this technique is that it can be very time consuming to implement in R and therefore chose the simpler procedure, i.e. grid search.
Another issue that might cause confusion is the result of each algorithm. Notice that each algorithm uses log probability to score its performance, which is a combination of both density estimation and classification prediction. Thus, it does not demonstrate which algorithms are superior at only estimating the density. This can easily be solved by only using one class when evaluating the different techniques. As seen in the result, some algorithms take hours to evaluate. To give an average of the result, multiple evaluation is needed. This can cause an immense time consumption and was decided to be left out.

In some cases, the result of certain algorithms scored a positive log probability value. To receive such a value, the probability needs to exceed one. Naturally, in statistics a probability of an event happening can never surpass 100%. This is true when working with actual probabilities. In the case of density estimation, the probability density at each point are proportional to each other, thus can very likely exceeds the 100% mark and result a positive log probability score.

8 Conclusion

It is quite obvious that different technique suits different dataset. From the result, KNN and tessellation based methods tends to perform well when given the huge amount of data that are closely clustered. On the other hand local reachability density and kernel density estimation has the highest performance when dealing with the data drawn from a probability distribution and is scalable in higher dimensions.

DTFE, the method of interest, did not perform badly in any of the dataset categories, nor did it dominate other traditional methods. One thing to take into consideration is that the computational speed in low dimension is unmatched by any others. This is certainly a viable algorithm when creating statistical models with a large amount of data in low dimension.

9 Future Work

One thing that comes to mind that I personally did not experiment with due to time constraint is to evaluate other regression techniques to interpolate the density points of the Delaunay tessellation. As mentioned before Ridge regression is the method used for interpolation, but other methods such as thin plate spline and Gaussian process are surely interesting alternatives to approach this problem.
References


