Implementation of handwritten text recognition using density value of Delauney tessellation

Adithya Ravindran
Abstract

Implementation of handwritten text recognition using density value of Delauney tessellation

Adithya Ravindran

This paper presents a novel Word spotting technique for handwritten documents using density value of Delaunay triangulation. Delaunay tessellation is constructed from a set of data points on a query image and the density value is computed for each data point. This information is either directly used for training in a feed-forward neural network or used to compute the probability estimates of a class from Delaunay Tessellation Field Estimation and classification follows using naive Bayesian classifier. This paper discusses the performance of a Delaunay tessellation field estimation model and neural network model.
Acknowledgements

This thesis has been highly educational and deepened my understanding of the concepts from both Statistical machine learning and Image Processing field.

I would like to thank my supervisor Anders Hast for supporting me this thesis period and for taking part in discussions with the doubt regarding the image processing part of the thesis.

I would also like to thank my reviewer Michael Ashcroft for his invaluable insights regarding the concepts from Statistical Machine Learning and for letting me take part in the machine learning group discussion every Friday.

Finally, I would also like to thank my thesis coordinator Justin Pearson for giving me a table to work for my thesis. This had helped me work more efficiently and helped me finish my thesis in time.
This thesis is dedicated to my parents
Mr. Vasu Ravindran
Mrs. Beena Ravindran
my sweet younger brother Anindiya ravindran
and to the person whom i love the most
Contents

1 Introduction .................................................. 17
  1.1 Project Purpose and Goal ............................... 17
  1.2 Scope .................................................. 17
  1.3 Structure of the Report ............................... 17

2 Background ................................................. 19
  2.1 Word Spotting ......................................... 19
  2.2 Delauney Triangulation ............................... 19
  2.3 Density of Delaunay Tessellation ..................... 20
  2.4 Delaunay Tessellation Field Estimation ............... 21
    2.4.1 Linear Regression ................................ 21
    2.4.2 Walking in a Triangulation ....................... 21
    2.4.3 Density Estimate ................................ 24
    2.4.4 Probability Estimates ............................ 24
    2.4.5 Classifier ........................................ 24
  2.5 Neural Network ......................................... 25
  2.6 Learning of neural network ........................... 26
    2.6.1 Performance Function .............................. 27
    2.6.2 Learning Algorithm ............................... 28

3 Input Data and Classes ..................................... 30
  3.1 Feature Points ........................................ 30
  3.2 Non-Maximal Suppression ............................. 30
  3.3 Descriptors .......................................... 32
    3.3.1 Descriptor 1 .................................... 32
    3.3.2 Descriptor 2 .................................... 32
    3.3.3 Descriptor 3 .................................... 32
    3.3.4 Descriptor 4 .................................... 32
    3.3.5 Descriptor 5 .................................... 32
    3.3.6 Descriptor 6 .................................... 33
    3.3.7 Descriptor 7 .................................... 33
3.4 Preprocessing .......................... 33
3.5 Delaunay Tessellation ................. 33
3.6 Computing Density of Delaunay Tessellation ................. 33
3.7 Classes .................................. 35
3.8 Removed Files .......................... 36

4 Neural Network Model .................. 38
  4.1 Experiment 1 .......................... 38
     4.1.1 Network Setting ................... 38
     4.1.2 Initial testing and result .......... 38
     4.1.3 Tuning of Network Parameters ...... 39
     4.1.4 Result ............................ 39
  4.2 Experiment 2 .......................... 40
     4.2.1 Input data with dummy points ...... 40
     4.2.2 Initial testing and result .......... 42
     4.2.3 Tuning of Network Parameters ...... 42
     4.2.4 Result ............................ 43

5 Delauney Tessellation Field Estimation Model ............. 45
  5.1 PreProcessing ........................ 45
  5.2 Linear-Regression ...................... 45
     5.2.1 Ridge Regression .................. 45
  5.3 Remembering Stochastic Walk .......... 46
  5.4 Density Estimates ...................... 46
  5.5 Classifier ............................ 46
  5.6 Result ............................... 48

6 Discussion ................................ 49

7 Conclusion ................................ 51

Bibliography ................................ 52
List of Figures

2.1 Example of Delaunay Tessellation .................................. 20
2.2 Artificial Neuron ...................................................... 25
2.3 A Simple Feed-Forward Neural Network ............................ 26

3.1 The pixel neighbourhood .............................................. 31
# List of Tables

3.1 Delaunay Triangulation of A Query Word With Different Sets of Feature Points ........................................ 34
3.2 Classes from the Data Set ........................................ 35
3.3 Table of Images Removed from the Dataset ...................... 37
4.1 Image Without and With Dummy Points Respectively .......... 40
4.2 Initial Testing and Result for Experiment 1 .................... 41
4.3 concatenated data of Descriptor 1 and 6 .......................... 41
4.4 Training of Neural Network with Descriptor 7 .................. 41
4.5 Tuning of neural network with Descriptor 7 ..................... 41
4.6 Tuning of neural network of Descriptor 1 and 6 with different learning rate ........................................... 42
4.7 Tuning of neural network with Descriptor 4 ....................... 42
4.8 Initial testing and result for Experiment 2 ....................... 43
4.9 Tuning of neural network of Descriptor 1 and 6 with different learning rate ........................................... 44
4.10 Tuning of neural network with Descriptor 1 and 6 with momentum value .............................................. 44
4.11 concatenated data of Descriptor 1 and 6 with Bayesian regularization .................................................. 44
5.1 Table representing 2d triangulation and its corresponding irregular polygon .............................................. 47
5.2 Result of dtfe .................................................... 48
Chapter 1

Introduction

1.1 Project Purpose and Goal

The purpose of this project is to evaluate the performance of Delaunay tessellation density estimation in the area of word-spotting. The evaluation will be based on the classification of handwritten words and comparing their result.

The aim of this project is to implement two machine learning models. First by using the Delaunay tessellation field estimation and second by using neural network. Density value of Delaunay tessellation from the data points generated in each image are used as the input data for each model.

1.2 Scope

This paper will focus on developing a text recognition system using two different machine learning models. Conclusions will be drawn from comparing the accuracy of classification.

We will be using pre-segmented images, and this paper won’t discuss the retrieval of words from original documents. There will be a discussion on pre-processing for each machine learning model and also for computing the density value for each word.

This paper will limit the discussion of Delaunay triangulation to 2 dimensions.

1.3 Structure of the Report

The rest of this paper is written as follows

Chapter 2 discusses the background for the paper. It contains a discussion on the theory part of the paper regarding word spotting, Delaunay tessellation,
the density of Delaunay tessellation and the machine learning models used for classification.

Chapter 3 discusses the preprocessing of each image for computing the density value and a brief summary of classes. It also presents an implementation for computing the density value.

Chapter 4 discusses the first model using neural network. Here we discuss the pre-processing done with the input data. It also presents a brief discussion on the neural network model and results obtained after training and testing.

Chapter 5 discusses the second model using Delaunay tessellation field estimation. It also presents a discussion on the pre-processing done with the input data and implementation for getting the probability estimates and the results of the classification.

Chapter 6 summarizes the result of both machine learning models. Chapter 7 summarizes the conclusion drawn and further improvement on the current model and also proposes a new machine learning model which could be used for word spotting using density value of Delaunay tessellation.
Chapter 2

Background

2.1 Word Spotting

Word spotting can be defined as the retrieval of a class from a query image that has been classified using features which were extracted from it. Word spotting can be grouped into Statistical representation and Structural representation based on the input information used in classification. Statistical representation involves using statistical information of the input image for classification, while structural representation uses information from the geometric characteristics of the input image [1].

2.2 Delauney Triangulation

"for a finite set of points 'S' in a plane, a triangulation of S is a simplicial complex 'T' such that S is a set of vertices in T." [2]

The necessary condition for a Delaunay triangulation is that the circumcircle of each simplex in the simplicial complex T does not contain a point in S other than the three vertices which form the simplex [2].

From the figure 2.1, We can decipher that only the leftmost triangulation satisfies the rule and is a valid Delaunay triangulation.
2.3 Density of Delaunay Tessellation

For a set $S$ of $n$ data points

$$S \in \{x_1, x_2, x_3, \ldots, x_n\}$$

and its corresponding Delaunay triangulation $T$ with $m$ triangles.

$$T \in \{t_1, t_2, t_3, \ldots, t_m\}$$

The density estimate for a point $x_i \in S$ which is not present in the convex hull is the inverse of the contiguous Voronoi polytope$[5]$. The contiguous Voronoi polytope for a point $x_i$ is provided by the union of all the Delaunay polytopes in which $x_i$ is a vertice$[5]$. If $x_i$ is present in the convex hull, then we set the density estimate of $x_i$ to zero.

In our 2d Delaunay triangulation $T$, let’s take the number of Delaunay triangles in which $x_i$ is a vertice of as $V$ and let’s take a set of points $P \in S$ such that all the points in $P$ are present in the convex hull of $T$.

$$\rho_D(x_i) = \begin{cases} 
\sum_{k \in V} \text{area}(k)^{-1} & \text{if } x_i \notin P \\
0 & \text{if } x_i \in P 
\end{cases}$$
2.4 Delaunay Tessellation Field Estimation

"Delaunay Tessellation field estimation estimates the density of a field at a point from the contiguous Voronoi polytope and interpolates these through linear regression within Delaunay triangles." [5]

2.4.1 Linear Regression

We model the linear regression coefficient for density values of points in a query simplex based on the previously estimated density values of vertices of a Delaunay polytope. For a Delaunay polytope, the dependable variable is the density value of each vertex in the polytope, and the explanatory variable is the coordinates itself.[3]

Let $\gamma_i$ be the density value of each of the vertices and $X_i$ be the matrix containing the vertices of the $i^{th}$ polytope. The unique solution of regression coefficient is given by [3]

$$\beta_i = (X_i^T \ast X_i)^{-1} \ast X_i^T \ast \gamma_i$$

2.4.2 Walking in a Triangulation

For computing the density of a new point in a given tessellation, we have to find out the polytope which contains the query point in the tessellation. We will be using remembering stochastic walk algorithm[4]. Algorithm (2) illustrates the pseudo code for the remembering stochastic walk, and the implementation of this algorithm is in section 5.3.

In the remembering stochastic walk algorithm, the input arguments are an initial triangle $q$ and the query point $p$. A total of two variables are used in the initialization part of the algorithm. Variable $\text{previous}$ which stores the previous triangle after traversing to another and a boolean variable $\text{end}$ used here as the stopping criterion for traversing. We initialize $\text{previous}$ to $q$ and $\text{end}$ to $\text{false}$[18].

Three variables are used while traversing a triangulation. The variable $e$ which stores the random edge of $q$, variable $s$ which stores the vertex of triangle $q$ not present in edge $e$ and $\text{nb}$ which stores one neighboring triangle of edge $e$[18].
There are two essential conditions for traversing a triangulation[18].

- If the neighboring triangle of edge $e$ stored here as $nb$ is not equal to previous triangle.

- If the point $s$ and point $q$ are of different orientation with respect to the edge $e$. The orientation of a point with respect to an edge gives whether the point is on the left hand side or the right hand side of the edge. The implementation of the orientation test is presented in 5.3.1.

If both of the conditions are met, then the algorithm traverses from $q$ to neighbor $nb$. Algorithm(1) illustrates the pseudo code of the condition required for traversing a triangulation.

\[
\text{if (neighbour of edge } e \text{ not equal to previous) and (point } s \text{ and point } p \text{ are of different orientation with respect to the edge } e \text{) then} \\
\quad \text{previous} = q; \\
\quad q = nb; \\
\text{end}
\]

Algorithm 1: Condition for traversing in a triangulation
Algorithm \textit{RememberingStochasticwalk}(q,p)

for a delaunay triangulation \( T \) this algorithm traverses from the triangle \( q \in T \) to the query point \( p \).

\begin{verbatim}
previous = q; end = false;

while not end do
    e = random edge of q;
    s = vertex of q not contained in e;
    nb = neighbor of q over e;
    if (nb not equal to previous) and (p on the other side of e than s) then
        previous = q;
        q = nb;
    end
    else
        e = next edge of q;
        s = vertex of q not contained in e;
        nb = neighbor of q over e;
        if (nb not equal to previous) and (p on the other side of e than s) then
            previous = q;
            q = nb;
        end
        else
            end = true;
        end
    end
end
\end{verbatim}

Algorithm 2: Remembering Stochastic Walk
2.4.3 Density Estimate

Previously we had discussed how to find the Delaunay polytope for a query point. Now we use the regression coefficient from the linear regression model to find the density value associated with a query point.

For a set of data points $S$, let’s consider that the query point $x_*$ is present at the $i^{th}$ polytope of the Delaunay tessellation $T$. Let’s take a set of points $P \subseteq S$ such that all the points in $P$ be present in the convex hull of $T$. Let $\beta_i$ be the coefficients from the linear regression model associated with each polytope. Then the density estimate of $x_*$ is given by

$$
\text{densityestimate}(x_* \notin S) = \begin{cases} 
(\beta_i)^t \times x_* & \text{if } x_* \notin P \\
0 & \text{if } x_i \in P 
\end{cases}
$$

2.4.4 Probability Estimates

The probability estimates for a query feature point $x_*$ is computed by normalizing the density estimate of $x_*$ and then calculating the probability of a class given the feature point $x_*$. 

- Normalization
  
The normalization for each point is done by dividing the density estimates with the volume of the irregular polygon formed by triangulation of each class. The x and y coordinates of the points in a class are the length and width respectively whereas the density value is the height.

- Probability estimate of a class given a feature point
  
  Let prior($class$) be the prior probability of a class and $\rho(x_*)$ be the normalized density estimate of the query point $x_*$. For a $k$ class classification problem, the probability estimate of $x_*$ for class "$i$" is given by

  $$
  \text{probability}(class_i|x_*) = \rho_i(x_*) \times \text{prior}(class_i)/\sum_{j=1}^{k} \rho_j(x_*) \times \text{prior}(class_j)
  $$

2.4.5 Classifier

Classification follows by calculating the maximum of posterior probability. Let prior($class$) be the prior probability of a class and $\text{probability}(x|\text{class})$ be the probability estimate of that class given the feature point $x$. Then the posterior probability of a class with $n$ query feature points is given by
 posterior probability(class|x₁, x₂, ..., xₙ) ∝ prior(class) × \prod_{i=1}^{n} \text{probability}(x_i|\text{class})

The image is classified as the class with the highest posterior probability estimate.

### 2.5 Neural Network

"An artificial neuron implements a nonlinear mapping of x₁, x₂, ..., xₚ input signals to [0, 1] or [−1, 1] or whichever value is preferred by the problem in hand." [6]. There is a weight associated with each input signal, and the strength of the output signal is influenced by the activation function and the threshold value. Figure 2.2 depicts a simple neuron.

The learning part involves adapting the weights and threshold value for a particular problem. In our classification problem, we would be using multilayered feedforward neural network. If the neural network has more than one layer, then it’s a multilayered neural network, and a multilayered neural network with connection towards only one direction is a feed-forward neural network.[6]

Figure 2.3 presents a simple feed-forward neural network.
2.6 Learning of neural network

As explained in the previous section, the learning part of the neural network involves iteratively adapting weights till a performance function is either minimized or maximized. For a layer $j$ with $n$ nodes, let $I$ be a set containing the $n$ input values and $W$ be a set containing the corresponding weights associated with it. Then the summation of the output value of each node is given by [11]

\[
I \in \{x_1, x_2, x_3, \ldots, x_n\}
\]

\[
W \in \{w_1, w_2, w_3, \ldots, w_n\}
\]

\[
S_j = \sum_{j \in n} x_j \times w_j
\]

For a given bias $\theta$, the nonlinear mapping of these output values is provided by [11]

\[
y(x_j) = f(S_j) + \theta
\]

The function $f(S_j)$ non-linearly maps $S_j$. The choice of the function $f()$ is depended on the problem.

The first partial derivative of the nonlinear mapping $f(S_j)$ with respect to $S_j$ is given by

\[
f'(S_j) = \frac{d(f(S_j))}{dS_j}
\]
If \( \hat{y}_j \) is the network response of an output node \( j \) and the target value is \( y_j \), then the error associated with the output node is given by

\[
\delta_j = f'(S_j) \times (y_j - \hat{y}_j)
\]

Now let us consider a hidden node \( j \) with a subsequent hidden layer \( i \). Let \( I_j \) be the input vectors from node \( j \) and let \( w_{ji} \) be the corresponding weight vector. If \( \hat{y}_j \) is the network response of the node \( j \) and \( \delta_i \) be the error associated with layer \( i \), then the error related to the hidden node \( j \) is given by

\[
\delta_j = (f'(S_j) \times \sum_{i \in I_j} \delta_i \times w_{ji})
\]

Let \( \alpha, \eta \) and \( y_k \) be the learning rate, momentum and the activation of the node in layer \( k \) respectively. If \( \delta_j \) depicts the error associated with the node \( j \), then the change of weights from a node in layer \( j \) to a node in layer \( k \) at \( n^{th} \) iteration is given by[11]

\[
\Delta w_{kj}(n) = \alpha \times \delta_j \times y_k + \eta \times \Delta w_{kj}(n - 1)
\]

The adapted weights can be considered as the information used for classification[11].

### 2.6.1 Performance Function

The learning part of a neural network involves iteratively adapting weights for either maximizing or minimizing a performance function. We would be dealing with two types of performance function.

- Mean Squared Error

  For \( n \) inputs and their corresponding weight vector \( \theta \), Let \( \hat{y} \) be the network response of a target \( y \)[13]. Then the mean squared error is given by

  \[
  MSE(\theta) = \frac{1}{n} \sum_{k=1}^{n} (\hat{y}_k - y_k)^2
  \]

- Cross Entropy

  For a network with \( n \) inputs, Let \( \hat{y} \) be the network response of a target \( y \). Then the cross entropy error is given by[15]

  \[
  E = -\left( \sum_{k=1}^{n} y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i) \right)
  \]
2.6.2 Learning Algorithm

The learning algorithm considered in this thesis is Bayesian regularization and Scaled conjugate backpropagation.

- Scaled Conjugate Gradient Backpropagation

  Backpropagation algorithm involves changing the weights of a neural network based on maximizing or minimizing a performance function. Each step in the iterative process is chosen by the direction which minimizes a global error function $E(w)$ where $w$ is the vector containing weights. [11]

  In gradient descent algorithm, the search direction is set to the negative of first derivative of $E(w)$ and a constant step size for changing the weights. The scaled conjugate gradient algorithm draws information about search direction and step size from second order differentiation of $E(w + \Delta y)$ where $\Delta y$ is the change added to the weights.[12]

- Bayesian Regularization

  Backpropagation algorithm is prone to overfitting error. Regularization techniques are used with backpropagation algorithm to get a smoother network response. Bayesian regularization is a training algorithm which includes a probability distribution of network weights[10].

  Let $E(w, D, M)$ be the mean sum of squares of the network error, where $w$ is the weight vector, $D$ is the training set with input target pairs, $M$ is the neural network architecture which contains information about the number of hidden nodes, number of hidden layers and the type of activation function used in each of these layers[10]. Then objective function for Bayesian regularization is given by

  $$F = \beta \times E(w, D, M) + \alpha \times E_w(w, M)$$

  The term $E_w(w, M)$ are the sum of squares of the network weights, and $\alpha$ and $\beta$ are hyperparameters. If $\alpha >> \beta$ then the size of the weights will be reduced which in turn causes an increase in error. If $\alpha << \beta$ then the training algorithm will proceed as normal and makes the error smaller.[10]

  Let the prior probability of network weights be $P(w)$ and $E_w(w, M)$ is the sum of squares of the network weights given an architecture $M$. Then $P(w)$ is given by [20]
\[ P(w) = e^{(-\alpha \times E_w(w,M))} / Z_w(\alpha) \]

here \( Z_w(\alpha) \) is the normalizer given by

\[ Z_w(\alpha) = \int e^{(-\alpha \times E_w(w,M))} dw \]

Let the Likelihood of occurrence be \( P(D|w) \) where D is the training set with input target pairs. Then \( P(D|w) \) is given by [20]

\[ P(D|w) = e^{(-\beta \times E_D(w))} / Z_w(\beta) \]

here \( Z_w(\beta) \) is the normalizer given by

\[ Z_w(\beta) = \int e^{(-\beta \times E_D(w))} dw \]

here \( E_D(w) \) is the error associated with the network response \( \hat{y} \) and the target \( y \). [13] For an n input network, \( E_D(w) \) is given by

\[ E_D(W) = \sum_{k \in 1}^{n} (\hat{y}_k - y_k)^2 \]

Let \( P(w|D) \) be the posterior distribution of weights in a Bayesian framework. Then \( P(w|D) \) is given by [20]

\[ P(w|D) = P(D|w) \times P(w) / P(D) \]

Where \( P(D) \) is the evidence and is given by

\[ P(D) = \int P(D|w) \times P(w) dw \]

An ideal set of weights will maximize the joint posterior probability \( P(w|D) \) and simultaneously minimizes the objective function \( F \). [10]
Chapter 3

Input Data and Classes

3.1 Feature Points

We use seven different edge and corner detectors to get the feature points. The points generated by these detectors are passed for Non-maximal suppression.

3.2 Non-Maximal Suppression

Non-maximal suppression is used to extract best points of interest[19]. In the beginning, this algorithm selects the local maxima of the edges in an image by grey scale morphological dilation operation.

"A morphological operation on an image is the logical transformation of the image based on the comparison of the pixel neighborhood with a pattern"[23]. Figure 3.1 illustrates the neighborhood around a pixel.

The pattern used for comparing the pixel neighborhood is called a structuring element. In our case, the structuring element is a n*n matrix of ones. Let $radius$ be a threshold radius which we will later use to suppress unwanted feature points, then the value of n is given by

\[ n = 2 \times \text{radius} + 1 \]

A morphological dilation operation at pixel location $x$ involves positioning the structural element over $x$ and calculating the sum of each pair of neighboring pixels with the corresponding pixel of structuring element. The maximum value of this amount is then set as the output value at pixel location $x$ for the dilated image.

For an image $I$ and a Structuring element $S$, let $D_I$ and $D_S$ be the domain of all the coordinates of the pixel values. Let $i$ and $j$ be the coordinates of the image.
Let \( x \) and \( y \) be the coordinates of the structuring element. Then the morphological dilation operation is defined as

\[
(I \oplus S)(i, j) = \max\{I(i - x, j - y) + S(x, y) \mid (i - x, j - y) \in D_I, (x, y) \in D_S\}
\]

Dilation operation essentially adds pixel values to the boundaries of the image. The pixel values of the dilated image which are above a threshold pixel value are selected as the local maxima of the edges of the image. The coordinates of local maxima are then compared with that of the feature points. The feature points that match are selected as the best points of interest.[22]

Then this algorithm deletes all the neighboring feature points of the threshold radius around the selected points of interest. This threshold radius was earlier used to create the structuring element. Eventually, this algorithm returns the \( x \) and \( y \) coordinates of the points of interest which best represents the edge and corner detectors.[22]
3.3 Descriptors

We will be using seven different edge and corner detectors. The feature points generated from these descriptors are later used to create Delaunay tessellation.

3.3.1 Descriptor 1

If $I_x$ is the first derivative of Image $I$ in the $x$ direction, $I_y$ is the first derivative of Image $I$ in the $y$ direction and $\text{eps}$ is a constant used to avoid division by zero. Then descriptor 1 is the non maximal suppression of $c_1$ where $c_1$ is[16]

$$c_1 = I_y \times I_x - (I_x \times I_y)^2 / (I_x^2 + I_y^2 + \text{eps})$$

3.3.2 Descriptor 2

If $I_{xx}, I_{yy}$ are the second derivatives of Image $I$ in the $x$ and $y$ direction respectively and $I_{xy}$ is the second derivative of the image in both $x$ and $y$ direction. Then descriptor 2 is the non maximal suppression of $c_2$ where $c_2$ is[16]

$$c_2 = I_{xx} \times I_{yy} - I_{xy}^2$$

3.3.3 Descriptor 3

If $I_{xx}, I_{yy}$ are the second derivatives of Image $I$ in the $x$ and $y$ direction and $I_{xy}$ is the second derivative of the image in both $x$ and $y$ direction. Then descriptor 3 is the non maximal suppression of $c_3$ where $c_3$ is[17]

$$c_3 = -(I_{xx} \times I_{yy} - I_{xy}^2)$$

3.3.4 Descriptor 4

If $I_{xx}, I_{yy}$ are the second derivatives of Image $I$ in the $x$ and $y$ direction. Then descriptor 4 is the non maximal suppression of $c_4$ where $c_4$ is[17]

$$c_4 = (I_{xx} + I_{yy})$$

3.3.5 Descriptor 5

If $I_{xx}, I_{yy}$ are the second derivatives of Image $I$ in the $x$ and $y$ direction. Let $\text{eps}$ be a constant used to avoid division by zero, Then descriptor 5 is the non maximal suppression of $c_5$ where $c_5$ is[17]

$$c_5 = (I_{xx} + I_{yy})^2 \times ((I_{xx}^2 \times I_{yy}^2 - (I_x \times I_y)^2)/(I_x^2 + I_y^2 + (I_{xx} + I_{yy})^2) + \text{eps})$$
3.3.6 Descriptor 6

If $I_{xx}, I_{yy}$ are the second derivatives of Image $I$ in the $x$ and $y$ direction and $I_{xy}$ is the second derivative of the image in both $x$ and $y$ direction. Let $\epsilon$ be a constant used to avoid division by zero, Then descriptor 6 is the non maximal suppression of $c_6$ where $c_6$ is[17]

$$c_n = (I_{xx} + I_{yy})^2 \times ((I_x^2 \times I_y^2 - (I_x \times I_y)^2)/(I_x^2 + I_y^2 + (I_{xx} + I_{yy})^2) + \epsilon)$$

$$c_6 = c_n/(2 \times I_{xy}^2 + I_{xx}^2 + I_{yy}^2 + \epsilon)$$

3.3.7 Descriptor 7

If $I_{xx}, I_{yy}$ are the second derivatives of Image $I$ in the $x$ and $y$ direction and $I_{xy}$ is the second derivative of the image in both $x$ and $y$ direction. Then descriptor 3 is the non maximal suppression of $c_7$ where $c_7$ is[17]

$$c_7 = (2 \times I_{xy}^2 + I_{xx}^2 + I_{yy}^2)$$

3.4 Preprocessing

The feature points from seven different detectors are used for the formation of Delaunay tessellation. From the theory, we know that the density value of feature points at the convex hull is zero. So an image with less than or equal to four feature points would produce a tessellation with all its points at the convex hull. To solve this, we would concatenate the feature points from detectors that would not give us a minimum five feature points individually.

3.5 Delaunay Tessellation

Matlab is used for constructing the Delaunay triangulation for a given image. We use `Delaunaytriangulation` function for creating the triangulation and `triplot` for plotting the same.

3.6 Computing Density of Delaunay Tessellation

The method ”`convexhull(tri)`” was used for computing the density of a 2D Delaunay triangulation. The argument tri is essentially a Delaunay triangulation. This method returns all the feature points which are present in the convex hull.
Table 3.1: Delaunay Triangulation of A Query Word With Different Sets of Feature Points

"Points" returns the coordinates and its reference number and "ConnectivityList" returns the triangle list and the reference number of each vertex. These two were used to find the contiguous Voronoi polytope.

Then the area of the contiguous Voronoi polytope is computed using the function "polyarea", and the inverse of this would give us the density of a feature point.
3.7 Classes

A total of 35 classes were taken from 6624 pictures. Classes used are presented in table 3.2.

Table 3.2: Classes from the Data Set

<table>
<thead>
<tr>
<th>rebore</th>
<th>de</th>
<th>fill</th>
<th>filla</th>
<th>pages</th>
</tr>
</thead>
</table>
| dia    | de Bag a | dit | ab | habit
| dFella | Bag a | viuda | viuda | Dijou
| dorjella | disaspe | reberem | dia | i de
| habitat | en Bag a | filla | fill | fill |
| dimar | Dimanres | i fill | fill | fill |
| in page | n Bag a | dimar | gmar | Dit s |

CHAPTER 3. INPUT DATA AND CLASSES
3.8 Removed Files

Images which gave an abnormally high number of feature points were removed from the dataset. Table 3.3 are examples of removed images.
| Table 3.3: Table of Images Removed from the Dataset |
Chapter 4

Neural Network Model

4.1 Experiment 1

In this experiment, we will use density value as input data to a regular feed-forward neural network. A feed-forward Neural network should have the same number of rows for all input data. The Maximum size of rows of input data was found out and padded the input data with 0.

4.1.1 Network Setting

The neural network consists of a single hidden layer. All the network parameters were default settings. The Number of epochs was 1000, the goal of the performance function was set to 0 and value of the gradient was 1.06e-7 by default. The untrained data used as test set comprised of density values from images indexed 6300 to 6625.

The training set, validation set and test set comprised of the ratio 0.7,0.15,0.15 respectively for scaled conjugate training function \textit{trainscg}, while 0.85 for the training set and 0.15 for test set when Bayesian regularization was used.

4.1.2 Initial testing and result

The input data used here are density values from concatenated feature points of descriptor 1 and 6, concatenated feature points of descriptors 2 and 5, descriptor 4 and descriptor 7. A brief discussion of all of the descriptors is presented in section 3.1.

The Scaled conjugate gradient backpropagation was used for initial testing because of faster convergence rate than a regular backpropagation training algo-
A brief discussion of the Scaled conjugate is presented in section 2.6.2.

The performance function used here were cross entropy because it was default performance function in the \texttt{trainscg} function. The performance function Mean squared error was also checked because it’s a basic error function.

Table 4.2 represents the results from initial testing. The Column ”descriptor” represents the name of the descriptor used here is the input data. The column ”training function” and ”Performance-Fn” represents training function and performance function respectively. The column ”training” represents the number of training session’s involved. The column ”hidden nodes” represents the number of hidden nodes and the column ”Accuracy” denotes the percentage of correct classification.

### 4.1.3 Tuning of Network Parameters

After initial testing, the best combination of hidden nodes, the performance function, the training function was found out for each descriptor. Then we change the tuning parameters like learning rate, momentum for each of these combinations. A brief discussion on learning rate and momentum is discussed in section 2.6.1.

The best descriptor was found out to be density value of concatenated feature points from descriptor 1 and 6 with ten hidden layers. The training function Bayesian regularization algorithm was checked because it had produced the lowest mean squared error for function approximation problems[11]. Hence the combination of Bayesian regularization with the mean squared error was used for density data of descriptor 1 and 6.

### 4.1.4 Result

All the final test results of experiment 1 are shown in the table 4.3,4.4,4.5,4.6 and 4.7. The Column ”descriptor”,”training function” and ”Performance Function”,”hidden nodes” is the same as of table 4.2. The column ”Learning rate” and ”momentum” represents the learning rate and momentum of the neural network. The performance was evaluated using column ”Accuracy” which represents the percentage of correct classification.
Table 4.1: Image Without and With Dummy Points Respectively

4.2 Experiment 2

Since we are training the density of a feature point in a tessellation and the density of feature points at the convex hull is zero. If we put dummy points at four corners of the image essentially gives the whole feature points a density value. Table 4.1 illustrates an example of a triangulation with dummy points.

Similar to experiment 1, all of the density data was padded to get the same number of rows. The Maximum size of rows of input data was found out and padded with 0.

4.2.1 Input data with dummy points

Input data consists of the density value of feature points from concatenating descriptors 1 and 6, descriptor 2 and 5, descriptor 4 and descriptor 7 with additional dummy points.

Neural network consists of a single hidden layer. All the network parameters
<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>learning-rate</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>45.092025</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>5</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>5</td>
<td>47.546012</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>44.171779</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>40</td>
<td>5</td>
<td>48.159509</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>40</td>
<td>5</td>
<td>48.159509</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>5</td>
<td>46.932515</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>40</td>
<td>5</td>
<td>45.705521</td>
</tr>
<tr>
<td>2 and 5</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>34.049080</td>
</tr>
<tr>
<td>2 and 5</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>33.435583</td>
</tr>
<tr>
<td>4</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>38.957055</td>
</tr>
<tr>
<td>4</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>38.957055</td>
</tr>
<tr>
<td>4</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>5</td>
<td>37.423313</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>41.411043</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>38.036810</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>5</td>
<td>37.423313</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>37.423313</td>
</tr>
</tbody>
</table>

Table 4.2: Initial Testing and Result for Experiment 1

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>learning-rate</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Bayesian regularization</td>
<td>Mean squared error</td>
<td>10</td>
<td>5</td>
<td>54.2555</td>
</tr>
</tbody>
</table>

Table 4.3: concatenated data of Descriptor 1 and 6

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>Momentum</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>0.8</td>
<td>41.104294</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>0.5</td>
<td>41.104294</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>0.3</td>
<td>41.104294</td>
</tr>
</tbody>
</table>

Table 4.4: Training of Neural Network with Descriptor 7

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>Momentum</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>0.8</td>
<td>41.104294</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>0.5</td>
<td>41.104294</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>0.3</td>
<td>41.104294</td>
</tr>
</tbody>
</table>

Table 4.5: Tuning of neural network with Descriptor 7

CHAPTER 4. NEURAL NETWORK MODEL
Table 4.6: Tuning of neural network of Descriptor 1 and 6 with different learning rate

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden-nodes</th>
<th>learning rate</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.001</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>1e-5</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.1</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.3</td>
<td>48.466258</td>
</tr>
</tbody>
</table>

Table 4.7: Tuning of neural network with Descriptor 4

Table 4.8: Results from initial testing.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>momentum</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.8</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.5</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.1</td>
<td>48.466258</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.3</td>
<td>48.466258</td>
</tr>
</tbody>
</table>

4.2.2 Initial testing and result

The experimental setup was same as experiment 1. Table 4.8 represents the results from initial testing. The column ”descriptor” represents which name of the descriptor used here as the input data. The column ”training function” and ”Performance Function” represents training function and performance function respectively. The column ”training” represents the number of training session involved. The column hidden nodes account for the number of hidden nodes, and the column ”Accuracy” denotes the percentage of correct classification.

4.2.3 Tuning of Network Parameters

In this stage, the best combination of Descriptor, hidden nodes, the performance function, training function was found out. Then we change the tuning parameters like learning rate, momentum for the best combination.

After the tuning phase, the best descriptor was found out to be concatenated data of density value from descriptor 1 and 6 with ten hidden nodes. Similar to experiment 1, Bayesian regularization with mean squared error as performance function was used for descriptor 1 and 6.
4.2.4 Result

All the test results are shown in the table 4.9,4.10,4.11. The Column ”descriptor”, ”training function” and ”Performance-Fn”, ”hidden nodes” is the same as the table 4.2. The column ”Learning rate” and ”momentum” represents the learning rate and momentum of the neural network. The performance was evaluated using column ”Accuracy” which represents the percentage of correct classification.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>training</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>57.975460</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>5</td>
<td>58.588957</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>5</td>
<td>56.134969</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>56.441718</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>40</td>
<td>5</td>
<td>57.975460</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>40</td>
<td>5</td>
<td>56.134969</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>5</td>
<td>57.361963</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled conjugate</td>
<td>Mean squared error</td>
<td>50</td>
<td>5</td>
<td>54.601227</td>
</tr>
<tr>
<td>2 and 5</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>34.969325</td>
</tr>
<tr>
<td>2 and 5</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>5</td>
<td>36.503067</td>
</tr>
<tr>
<td>4</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>43.865031</td>
</tr>
<tr>
<td>4</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>5</td>
<td>46.012270</td>
</tr>
<tr>
<td>4</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>5</td>
<td>46.319018</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>10</td>
<td>5</td>
<td>45.092025</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>5</td>
<td>44.478528</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>30</td>
<td>5</td>
<td>43.558282</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>41.104294</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Cross entropy</td>
<td>50</td>
<td>5</td>
<td>43.251534</td>
</tr>
<tr>
<td>7</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>30</td>
<td>5</td>
<td>43.251534</td>
</tr>
</tbody>
</table>

Table 4.8: Initial testing and result for Experiment 2
Table 4.9: Tuning of neural network of Descriptor 1 and 6 with different learning rate

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>learning rate</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>1e⁻⁵</td>
<td>58.588</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.001</td>
<td>58.588</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.1</td>
<td>58.588</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.3</td>
<td>58.588</td>
</tr>
</tbody>
</table>

Table 4.10: Tuning of neural network with Descriptor 1 and 6 with momentum value

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>momentum</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.8</td>
<td>58.588</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.5</td>
<td>58.588</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.3</td>
<td>58.588</td>
</tr>
<tr>
<td>1 and 6</td>
<td>Scaled Conjugate</td>
<td>Mean squared error</td>
<td>10</td>
<td>0.1</td>
<td>58.588</td>
</tr>
</tbody>
</table>

Table 4.11: concatenated data of Descriptor 1 and 6 with Bayesian regularization

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Training Function</th>
<th>Performance-Fn</th>
<th>hidden nodes</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>Bayesian regularization</td>
<td>Mean squared error</td>
<td>10</td>
<td>59.509202</td>
</tr>
</tbody>
</table>

CHAPTER 4. NEURAL NETWORK MODEL
Chapter 5

Delauney Tessellation Field Estimation Model

5.1 PreProcessing

Since we are training the density of a feature point in a tessellation and the density of feature points at the convex hull is zero. To get every feature point a density value, we will put dummy points at four corners of the image.

5.2 Linear-Regression

If \( X_i \) is matrix containing the vertices of the \( i^{th} \) polytope with preceding columns of one’s and \( \gamma_i \) be the density value of each of the vertices, then unique solution of regression coefficient is given by [5]

\[
\beta_i = (X_i^T \ast X_i)^{-1} \ast X_i^T \ast \gamma_i
\]

5.2.1 Ridge Regression

Since some of the triangles had very low area. The matrix was badly scaled and \((X_i^T \ast X_i)^{-1}\) part of the equation was not invertible. Matlab gave a warning that the regression coefficient produced by these triangles were not accurate. To solve this, we used Ridge regression.

Ridge regression adds a penalty to the equation of regression coefficient. In our problem i have used regularization parameter \( \lambda = 0.1 \). So the equation for the regression coefficient for the \( i^{th} \) polytope is given by

\[
\beta_i = (X_i^T \ast X_i + \lambda \ast I)^{-1} \ast X_i^T \ast \gamma_i
\]

This regularization solved the problem of an ill conditioned matrix [7].
5.3 Remembering Stochastic Walk

Remembering stochastic walk algorithm is used to find the polytope enclosing the query point. As explained in section 2.4.2, there are two essential conditions for traversing a triangulation

- The first condition requires us to find the neighbor of a polytope for traversing. The data from Points and ConnectivityList was used for getting the neighbor of a polytope.

- Orientation Test

  The orientation test for a query point and an edge gives us whether the point is on the left-hand side or the right-hand side of the edge[8]. We can get this by verifying the sign of the area of the triangle formed by the query point and the edge. Let \((x_1, y_1), (x_2, y_2)\) be the edge and \((q_x, q_y)\) be the query point then the orientation is given by

  \[
  \text{orientation} = \left| \begin{array}{cc}
  x_2 - x_1 & q_x - x_1 \\
  y_2 - y_1 & q_y - y_1
  \end{array} \right|
  \]

  The orientation is positive if and only if \((x_1, y_1), (x_2, y_2), (q_x, q_y)\) is counter clockwise[8].

5.4 Density Estimates

The density estimates for each point of a given class is obtained by finding the triangle in which this point is inscribed in the class image. The inscribing triangle is found out by using remembering stochastic walk algorithm and then multiplying the point with the corresponding regression coefficient. The equation for density estimate is presented in section 2.4.3.

5.5 Classifier

The Implementation of classifier involves normalizing the density value of the query feature point and then calculating the probability of a class given the feature point. Finally computing the posterior probability of the query image with n feature points given a class.

- Normalization
The normalization for each point is done by dividing the density estimates with the volume of the irregular polygon formed by triangulation of each class. The x and y coordinates of the points in a class are the length and width respectively. Whereas the density value is the height. An example of a triangulation with the class is given by

- **Classifier**

  The theory for computing the probability estimates and posterior probability is found in the section 2.4.4 and 2.4.5. The prior probability used in our case is the word frequency.

```
  a

  b
```

Table 5.1: Table representing 2d triangulation and its corresponding irregular polygon

```
<table>
<thead>
<tr>
<th>length</th>
<th>width</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
</tr>
</tbody>
</table>
```

CHAPTER 5. DELAUNEY TESSELLATION FIELD ESTIMATION MODEL
<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Images</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 6</td>
<td>1 - 600</td>
<td>34.67</td>
</tr>
<tr>
<td>1 and 6</td>
<td>600-1200</td>
<td>30.83</td>
</tr>
<tr>
<td>1 and 6</td>
<td>1200-1800</td>
<td>34.00</td>
</tr>
<tr>
<td>1 and 6</td>
<td>1800 - 2400</td>
<td>36.50</td>
</tr>
<tr>
<td>1 and 6</td>
<td>2400-3000</td>
<td>38.83</td>
</tr>
<tr>
<td>1 and 6</td>
<td>3000-3600</td>
<td>34.00</td>
</tr>
<tr>
<td>1 and 6</td>
<td>3600-4200</td>
<td>31.50</td>
</tr>
<tr>
<td>1 and 6</td>
<td>4200 - 4800</td>
<td>28.67</td>
</tr>
<tr>
<td>1 and 6</td>
<td>4800-5400</td>
<td>31.83</td>
</tr>
<tr>
<td>1 and 6</td>
<td>5400-6000</td>
<td>36.17</td>
</tr>
<tr>
<td>1 and 6</td>
<td>6000-6625</td>
<td>35.33</td>
</tr>
</tbody>
</table>

Table 5.2: Result of dtfe

5.6 Result

The descriptor chosen is the density value from concatenated feature points of descriptor 1 and 6 because it gave the highest accuracy in the neural network. A brief discussion on descriptors is presented in section 3.1. Table 5.3 represents the accuracy of correct classification from Delaunay tessellation density estimation for all preprocessed 6625 images with a batch size of 600.
Chapter 6
Discussion

From the experiments, we can state that the input data from concatenated feature points of descriptor 1 and descriptor 6 gave the best classification. We could consider this data as the best input data for our experiments.

Whereas, the input data from concatenated feature points of descriptor 2 and descriptor 5 gave the lowest accuracy. There was no significant change in accuracy with a different number of hidden nodes. In our case, we could consider data from descriptor 2 and descriptor 5 as the least useful data.

The technique of putting dummy points to get every feature point a density value gave an increased accuracy for the neural network. There was an improvement in performance even in data from descriptor 2 and descriptor 5.

The input data from a combination of descriptor 1 and descriptor 6 gave 38 percentage correct classification for Delaumay tessellation density estimation for a batch of 600 images out of 6625. A feed-forward neural network gave 59 percentage correct classification for 6300 images with the test set comprising of 325 images.

When random values were taken as the test result, a total of 209 images were correctly classified out of 6625. We had used the `randi` function in MATLAB to get random values.

The same dataset was used in another experimental setup using a histogram of oriented gradients for getting the feature points and Support vector machines as the classifier. This setup gave a mean average precision of 58.48 percentage [21].

The experimental setup using graph representation and matching gave an average accuracy of 56.59 percentage [21].

The dataset was used with a Fourier based feature vector and Putative matching analysis for matching the classes. This setup gave a mean average precision of 77.18 percentage[21].
From all of the result, we could conclude that the density of Delaunay tessellation is a good candidate for classification purposes. With different pre-processing techniques, a machine learning model with the density of Delaunay tessellation as the input data has the potential to deliver higher accuracy.
Chapter 7

Conclusion

This project aimed to investigate the performance of density value of Delaunay tessellation in the area of word spotting. Delaunay tessellation was formed using feature points from seven different descriptors. The density value of these feature points was used as the input data for classification of handwritten words in Delaunay tessellation field estimation model and a neural network model.

This study has identified that the density value from the concatenated feature points of descriptor 1 and descriptor 6 gave the best classification. The results of this project also indicate that a machine learning model with the density of Delaunay tessellation as the input data has the potential to deliver higher accuracy for a classification problem.

Further improvements for the models used in this project could be brought by using feature points from better descriptors. Most of the images were poorly segmented, so applying different segmentation techniques at the pre-processing stage to extract the query word from these images is another option for improving the accuracy.

A potential machine learning model for future work is experimentation with a recurrent neural network to classify the sequence of inputs. Each input sequence could comprise of density values from a query image. A similar experiment is demonstrated in the Experiment number 6 of Long Short Term Memory Paper[9].
Bibliography


[8] Rodrigo Silveira *Basic tools: orientation tests* Facultat d’Informatica de Barcelona Universitat Polit’ecnica de Catalunya, computational geometry, slide number 1-8.3


[14] Harry Zhang, The Optimality of Naive Bayes, University of New Brunswick

[15] Peter Sadowski, Notes on Backpropagation, University of California Irvine

[16] Anders Hast, Interest Point Detection Based on the Extended Structure Tensor with a Scale Space Parameter, Uppsala University

[17] Anders Hast, Andrea Marchetti, Invariant Interest Point Detection Based on Variations of the Spinor Tensor, Uppsala University

[18] Ivana Kolingerova, A Small Improvement in the Walking Algorithm for Point Location in a Triangulation, EWCG 2006, Delphi, March 27-29, 2006

[19] Jan Hosang, Rodrigo Benenson, Bernt Schiele, Learning Non Maximal Suppression, Max Planck Institut für Informatik, Saarbrücken, Germany, May 9, 2017

[20] Aaron Courville, Neural Networks for Pattern Recognition, Chapter 10, Carnegie Mellon University


[22] Yoram Tal, Gray scale morphology, School of Mathematical Sciences, Tel Aviv University, Israel

[23] Gabriel Brostow, Morphological operators, Lecture 3, Page 1-75, University College London, United Kingdom

BIBLIOGRAPHY