PATTERN RECOGNITION REPORT

Ahmad Alzeitoun - MR.Nr 2413759

University of Bonn, Germany

ABSTRACT

In this report, we discuss in details some techniques from Pattern Recognition which are widely used. We present their compact explanation, performance evaluation, their complexity, implementation details and advantages over each other.
We describe in this report: First, a few techniques about data distribution where we predict and make an analysis about the nature of the data. Second, a few approaches regarding data structure, similarity searching as well as data classification e.g. Least Square,KD-tree and KNN etc. We not only explain these techniques but also provide their detailed and in-depth performance analysis.
Thirdly, we also elaborate how to perform perfect clustering for compact and non-convex data e.g K-means and Spectral Clustering. Furthermore, we also illustrate that how can we reduce the dimensionality of the data and can make our algorithm fast enough without losing any significant information.

***Index Terms*—** Outlier, Squared Error, Normal Distribution, Wei-bull fit, L_p Norm, Spectral Clustering, Aitchison Geometry, Least Square Regression, K-Mean, Conditional Expectation, K-Nearest Neighbor Classifier, KD-Tree, Clustering, Lloyd, Hartigan, Numerical Instabilities, Mac-Queen, Dimensionality Reduction, PCA, Multiclass LDA.

1. INTRODUCTION

We first introduce the data distribution as well as a few key points about unit circle. We will show that how to remove the outliers from the given data, and fit the Normal or Gaussian distribution. We then demonstrate Weibull distribution to fit into one-dimensional data. Furthermore, we will draw a unit circle using L_p norms with a specific value of p and also will draw a circle in $\mathbb{S}^3$ by using Aitchison Geometry.All these methods will be explained in Section 2.

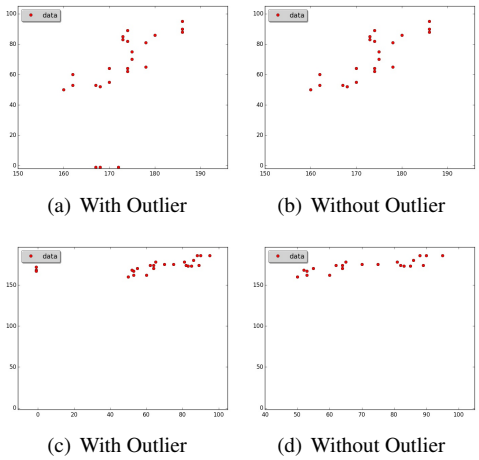
In Section 3, Least Square Regression and Nearest Neighbor Classifiers have been explored where we will predict the missing values in the dataset using different techniques e.g. Least Square Regression, Conditional Expectation and Bayesian Regression. We will also classify the given data using some of the most popular classification techniques e.g. K-Nearest Neighbor and KD-Trees.

In section 4, we will focus on how to perform clustering on a given dataset using various well-known clustering techniques e.g. Spectral Clustering and K-Mean clustering using Lloyd, Hartigan and MacQueen algorithms. Moreover, we will also discuss Principal Component Analysis (PCA) and Multi-Class Linear Discriminant Analysis in term of dimensionality reduction for a given dataset. Finally before the overall conclusion, we will check the numerical instabilities in the given piece of code.

2. DATA DISTRIBUTION AND UNIT CIRCLE

2.1. Outlier Removal

Mostly, there exist data objects which do not comply with the general behaviour or model of the data. such data object, which is grossly different from or inconsistent with the remaining set of data are called an outlier. An outlier is a point far apart from the normal data. It may be an error or wrong distribution. The given size and weight data set of students contains some negative values which are basically outlier as some students do not want to disclose their data.



(a) With Outlier (b) Without Outlier

(c) With Outlier (d) Without Outlier

Fig. 1. Outlier Removal

we can see outlier in Figure 1(a) and Figure 1(c) while Figure 1(d) and Figure 1(b) are outlier free. The Figure

1(a) and Figure 1(c) shows both positive and negative results where negative results lead to outlier while Figure 1(d) and Figure 1(b) show only positive results and, therefore, are without noisy data.

2.2. Fitting a Normal Distribution to 1D data

Normal or Gaussian Distribution approximates many natural phenomena and is the standard of reference for many probability problems. It is symmetric, bell-shaped and continuous for all values X between $-\infty$ to $+\infty$. It depends upon μ and σ which determine the shape of distribution. Therefore, the rule of normal distribution is as under:

$$f(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\left(e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) \tag{1}$$

where
μ is mean or expectation. The average value of some function $f(x)$ under a distribution $p(x)$ is called expectation of $f(x)$

$$\mathbb{E}\left[f\left(x\right)\right] = \int f\left(x\right) p\left(x\right) dx$$

SpecialCase : expectation of a random variable X

$$\mathbb{E}\left[X\right] \equiv \mu$$

We have fitted the Normal Distribution over one-dimensional given body sizes data as under:

Algorithm 1 Fitting Normal distribution over 1D data

1: Consider 1D data array containing body size information
2: Calculate mean μ
3: Calculate variance σ^2
4: Calculate Density function
5: Plot data and Normal Distribution characterizing its Density

The resulting plot of normal distribution is as under:

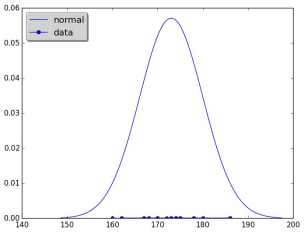


Fig. 2. Normal Distribution

Bell curve or Gaussian curve above is achieved by plotting a line using the data points for body sizes that meet the criteria of the Normal Distribution. We can see that the centre (highest point on the arc of the line) contains the largest number of a value and is called mean of data. The curve decreases on either side and is concentrated in the centre. It shows that the data is symmetrical which leads to creating reasonable expectations with a possibility that output will lie within the range to the left or right of the centre after measuring the deviation.

2.3. Fitting a Weibull Distribution to 1D data

Weibull Distribution has a great ability to fit data. It is a continuous probability distribution. It is extensively used in reliability and lifetime modeling fatigue, survival analysis and to model an extensive range of failure rates because of its flexible shape.

$$f(x|k,\alpha) = \frac{k}{\alpha}(\frac{x}{\alpha})^{k-1}e^{-(\frac{k}{\alpha})^k}; k,\alpha \in (0,\infty) \tag{2}$$

where,
k: is the shape parameter which also known as the Weibull slope.It has great a effect on the behavior of the Weibull distribution
α: is scale parameter.
For k=1, The equation 2 becomes the cumulative distribution function of exponential distribution as under:

$$f(x|k,\alpha) = 1 - e^{-(\frac{x}{\alpha})^k} \tag{3}$$

For a given data sample D the log-likelihood for the parameter of Weibull distribution is as under:

$$L(\alpha,k|D) = N(logk - klog\alpha) + (k-1)\sum_i logd_i - \sum_i (\frac{d_i}{\alpha})^k \tag{4}$$

deriving L w.r.t α and k is leads to a coupled system of partial differential equation for which we introduce. Newton's method for the simultaneously equation:

$$\begin{bmatrix} K_{new} \\ \alpha_{new} \end{bmatrix} = \begin{bmatrix} k \\ \alpha \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 L}{\partial K^2} & \frac{\partial^2 L}{\partial k \partial \alpha} \\ \frac{\partial^2 L}{\partial k \partial \alpha} & \frac{\partial^2 L}{\partial \alpha^2} \end{bmatrix}^{-1} \begin{bmatrix} -\frac{\partial L}{\partial K} \\ -\frac{\partial L}{\partial \alpha} \end{bmatrix} \tag{5}$$

where the entries of the gradient decent vector and the Hessian matrix are as under:
$\frac{\partial L}{\partial k} = \frac{N}{k} - Nlog\alpha + \sum_i logd_i - \sum_i (\frac{d_i}{\alpha}^k)log(\frac{d_i}{\alpha})$

$\frac{\partial L}{\partial \alpha} = \frac{k}{\alpha}(\sum_i (\frac{d_i}{\alpha})^k - N)$

$\frac{\partial^2 L}{\partial k^2} = -\frac{N}{k^2} - \sum_i (\frac{d_i}{\alpha})^k (log(\frac{d_i}{\alpha}))^2$

$\frac{\partial^2 L}{\partial \alpha^2} = \frac{k}{\alpha^2}(N - (k+1)\sum_i (\frac{d_i}{\alpha})^k)$

$\frac{\partial^2 L}{\partial K \partial L} = \frac{1}{\alpha}\sum_i (\frac{d_i}{\alpha})^k + \frac{k}{\alpha}\sum_i (\frac{d_i}{\alpha})^k log(\frac{d_i}{\alpha}) - \frac{N}{\alpha}$

HessianMatrix :is a square matrix of second-order partial derivatives of a scalar-valued function

2.3.1. *Implementation*

We have fitted the Weibull distribution on Google Trends data which shows how global interest in the query term myspace evolved over time:

Algorithm 2 Fitting Weibull Distribution over 1D data

1: Given the histogram h for a vector X e.g

x	h(x)
1	3
2	1
3	2

2: From this, the individual observations **d** are derived:$[1, 1, 1, 2, 3, 3]$

3: We made use of the following equivalencies:

$$\sum_i log(d_i) = \sum_j log(x_j)h(x_j)$$
$$sum_i(\frac{d_i}{\alpha})^k = \sum_j (\frac{x_j}{\alpha})^k h(x_j)$$
$$\sum_i (\frac{d_i}{\alpha})^k log(\frac{d_i}{\alpha}) = \sum_j (\frac{x_j}{\alpha})^k log(\frac{x_j}{\alpha})h(x_j)$$
$$\sum_i (\frac{d_i}{\alpha})^k (log(\frac{d_i}{\alpha}))^2 = \sum_j (\frac{x_j}{\alpha})^k (log(\frac{x_j}{\alpha})^2 h(x_j)$$

4: Set the Hessian Matrix and does the arithmetic in a user defined function UpdateParam.

5: Result obtained from updateParam are fed into the formula and results are plotted accordingly.

The results obtained are as under:

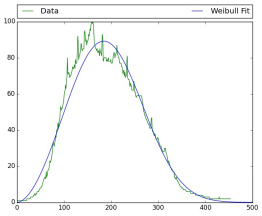


Fig. 3. Weibull Fit k=1, $\alpha = 1$,iterations =20

Weibull Distribution behaves like Exponential Distribution at $k = 1$ and like Rayleigh distribution at $k = 2$. The shape of the distribution changes with different values of k and increasing k reduces the skewness of Weibull Distribution. It becomes Normal distribution for k = 3.48.

2.4. L_p Norm Unit Circle for $p = 0.5$

2.4.1. L_p *norm*

We know that inner product is used to induced a norm. for $x \in \mathbb{R}^m$ and $p \in \mathbb{R}$, $p \geqslant 1$, the $L_p - norm$ of x is given by

$$\|x\| = (\sum_{i=1}^{m} |x_i|^p)^{\frac{1}{p}} \quad (6)$$

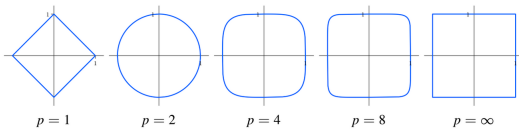


Fig. 4. L_p Norm Unit Circle examples

2.4.2. *Implementation*

Now we have to draw the unit circle with $P = 0.5$. We know that the unit circle means drawing unit vectors.We have solved this in the following way.

$$\sum_i^d (|x_i|^p)^{\frac{1}{p}} = 1$$
for p= 0.5 and d= 2, we have.
$$\sum_1^2 (|x_i|^0 .5)^2 = 1$$
$$|x_1|^0 .5 + |x_2|^0 .5 = 1$$
$$1 - |x_1|^0 .5 = |x_2|^0 .5$$
$$(1 - |x_1|^0 .5)^2 = |x_2|$$
$$\pm(1 - |x_1|^0 .5)^2 = x_2 \text{s.t} x_1 \in [-1.0, 1.0]$$

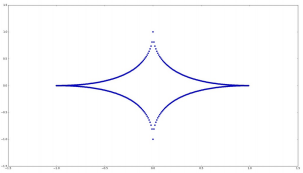


Fig. 5. Unit Circle with $p = 0.5$

2.5. Aitchison Geometry and Circle in $\mathbb{S}^3$

2.5.1. *Compositional Data*

The sample space of compositional data is the simplex and is defined as

$$\mathbb{S}^m = x \in \mathbb{R}^m | x_i > 0 \wedge \sum_i^m x_i = 1 = \Delta^{m-1} - \delta\Delta^{m-1} \quad (7)$$

For any vector of m real positive components $x = [x_1, x_2 \cdots] \in \mathbb{R}_+^m$ and x_i for all $i = 1, 2, \cdots m$, the closure of x is defined as

Closure of $x \in \mathbb{R}_+^m$

$$C(x) = \frac{x}{\sum_i x_i} = \frac{x}{\|x\|_1}$$

First of all, we can define two operations which give the simplex a vector space structure. The first one is the perturbation operation which is analogous to addition in real space. The second one is the power transformation which is analogous to multiplication by a scalar in real space. Both require in their

definition the closure operation; We know that the closure is the projection of a vector with positive components onto the simplex.
Basically, we just add an inner product, a norm and a distance to the previous definitions. With the inner product; we can project compositions onto particular directions as well as can check for orthogonality and determine angles between compositional vectors.
Therefore, with the norm, we can compute the length of a composition. Therefore, the possibilities of a distance should be clear. With all together we can operate in the simplex in the same way as we operate in real space.
Perturbation of $x \in \mathbb{S}^m$ **by** $y \in \mathbb{S}^m$

$$x \oplus y = C([x_1.y_1, x_2.y_2 \cdots x_m.y_m])$$

Powering of $x \in \mathbb{S}^m$ **by** $a \in \mathbb{R}$

$$a \odot x = C([x_1^a, x_2^a \cdots x_m^a])$$

since $(\mathbb{S}^m, \oplus, \odot)$ is a vector space over $\mathbb{R}$
To obtain a linear vector space structure, we take the following inner product with associated norm and distance:
Inner Product of $x, y \in \mathbb{S}^m$

$$< x, y > a = \frac{1}{2m} \sum_{i=1}^{m} \sum_{j=1}^{m} ln\frac{x_i}{x_j} ln\frac{y_i}{y_j}$$

Norm of $x \in \mathbb{S}^m$

$$\|x\|_= \sqrt{< x, x > a}$$

Distance b/w $x, y \in \mathbb{S}^m$

$$d_a(x, y) = \|x - y\|$$

2.6. Estimating the Dimension of Fractal Objects in an Image

In this task we consider a neat practical application of least squares for linear regression. We applied a box counting procedure. First step we binarize a square image in which foreground pixels are set to 1 and background pixels to 0 by using a piece of code provided in the task itself, in the second step we specified the scaling factors and by plotting different values of s and n as shown in Fig.7:

$$S = \left\{ \frac{1}{2^i} \,|\, i \in \{1, 2, ..., L-2\} \right\} \tag{8}$$

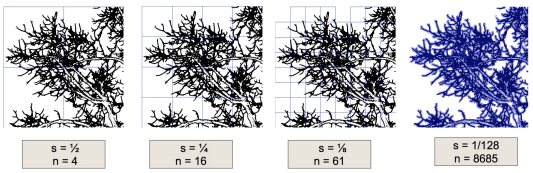


Fig. 6. S is scaling factors n is no. of boxes

In the third step did fitting for tree image and light image, by comparing the plotting between them, so we see in Fig.8 bigger quantities for tree image:

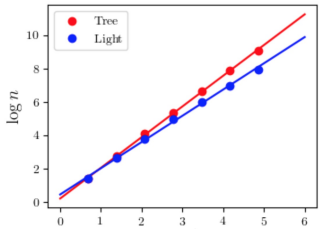


Fig. 7. comparing the plotting between tree and light image

3. LEAST SQUARE REGRESSION AND NEAREST NEIGHBOR CLASSIFIERS

3.1. Least Square Regression

The least square is method is one of the fundamentals methods used for regression. It has two forms linear and nonlinear. We fit a model as following:

Given a sample of n data points $(x_i, y_i) \in \mathbb{R}^2$
and assume they result from

$$y_i = \sum_{j=0}^{d} w_j x_i^j \tag{9}$$

and estimate the coefficients w_j from the data. where,

$$\sum_{j=0}^{d} w_j x^j = w^T x = x^T w \tag{10}$$

where,

$$x = \begin{bmatrix} 1 & x & x^2 & \cdots & x^d \end{bmatrix}^T \text{ and } w = \begin{bmatrix} w_0 & w_1 \cdots w_d \end{bmatrix}^T$$

Basically we minimize the following objective w.r.t w as under:

$$E(w) = \|Xw - y\|^2 \quad = w^T X^T y - 2w^T X^T y + y^T y \tag{11}$$

by taking the partial derivative and solving above equation.

$$\Leftrightarrow w = (X^T X)^{-1} X^T y \tag{12}$$

where,

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots x_1^d \\ 1 & x_2 & x_2^2 & \cdots x_2^d \\ \vdots & & & \\ 1 & x_n & x_n^2 \cdots x_n^d & \end{bmatrix} \in \mathbb{R}^{n \times d+1}$$

$n =$ number of samples
X is a Vandermodnde Matrix where $X_{ij} = x_i^{j-1}$
d= Parameters: $w_0, w_1 \cdots w_d$

3.1.1. Implementation

There are more than one ways for least square regression in python as mentioned in detail in and one have advantages over another. Therefore, the solution exemplified in $lsq - solution - V3$; lstsq is fast and robust while others have either a numerical problem or too slow in different cases for example if dimension m of data exceeds number n of data samples or both m and n are large. Therefore, we have adopted $lsq - solution - V3$ to implement our solution as under:

Algorithm 3 Predict the missing y value in the dataset of $\{x_i, y_i\}$

1: Assume $\exists$ a linear function $f(x) \rightarrow y$
2: Fit the parameter of the linear function to the data.
3: Generate the missing y values for the given x values using the linear function

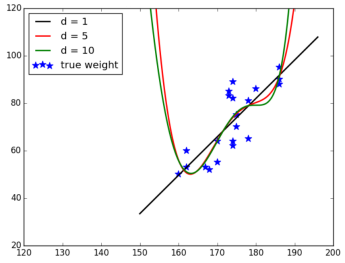


Fig. 8. Weight Prediction with Least square Example

The obtained square error sum with different values of d and predicted outlier weights are as under:

d	Square Error Sum	Predicted Outlier Weights
1	1503.25	[62.5,68.98,60.89]
5	1387.38	[58.58,69.47,55.93]
10	1366.4	[58.04,69.48,55.46]

We can see clearly form above plot, predicted outlier weights and error obtained that the higher degree of polynomial leads to lower the error.

3.2. Conditional Expectation for Missing Value Prediction

For the bivariate Gaussian $N(h, w|\mu, \sum)$, we have

$$\mu = \begin{bmatrix} \mu_h \\ \mu_w \end{bmatrix} \text{ and } \sum = \begin{bmatrix} \sigma_h^2 & \rho\sigma_h\sigma_w \\ \rho\sigma_h\sigma_w & \sigma_w^2 \end{bmatrix}$$

where $\rho = \frac{cov(h,w)}{\sigma_h\sigma_w}$
is the corelation of h and w
therefore,
The Bivariate Gaussian Density:

$$N(h, w) = \frac{1}{2\pi\sigma_h\sigma_w\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}\left[M_{wh}\right]} \tag{13}$$

where
$M_{wh} = \frac{(h-\mu_h)^2}{\sigma_h^2} + \frac{(w-\mu_w)^2}{\sigma_w^2} - 2\rho\frac{(h-\mu_h)(w-\mu_w)}{\sigma_h\sigma_w}$
therefore,
The Conditional Density for w, given that $h = h_0$ is then

$$N(w|h = h_0, \mu, \sum) = N(w|\mu_{w|h=h_0}, \sigma^2_{w|h=h_0}) \tag{14}$$

where,
$\mu_{w|h=h_0} = \mu_w + \rho\frac{\sigma_w}{\sigma_h}(h_0 - \mu_h)$
$\sigma^2_{w|h=h_0} = \sigma_w^2(1-\rho^2)$
The Conditional Expectation for w given that $h = h_0$is then

$$\begin{aligned} \mathbb{E}\left[w|h = h_0\right] &= \int wN(w|\mu_{w|h=h_0,\sigma^2_{w|h=h_0}})dw \\ &= \mu_w + \rho\frac{\sigma_w}{\sigma_h}(h_0 - \mu_h) \end{aligned} \tag{15}$$

3.2.1. Implementation

The main steps to apply Gaussian distribution and then apply the conditional expectation on the given data are as under:

Algorithm 4 Predict the missing y value in the dataset of $\{x_i, y_i\}$

1: Assume data is distributed according to Gaussian Distribution
2: Fit a multivariate Gaussian Distribution to the data
3: Generate the missing y values for the given x values using Conditional Expectation $\mathbb{E}[w|h_0] = \int wP(w|h_0)dw$

The parameter for multivariate Gaussian Distribution are as under:
Mean: $= \begin{bmatrix} 71.52380952 & 173.57142857 \end{bmatrix}$

Variance: $= \begin{bmatrix} 209.01133787 & 84.84353741 \\ 84.84353741 & 209.01133787 \end{bmatrix}$

We have obtained the obtained the Height vs Weight bivariate Gaussian Distribution as under:

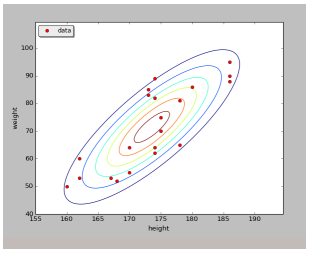


Fig. 9. Height vs Weight Bi-variate Gaussian Distribution

The obtained conditional mean weights are as under:

Conditional Mean Weights	
height	**weight**
168	62.51
172	68.98
167	60.89

we have obtained the following results. where
$w_{MAP} = (X^T X + \frac{\sigma^2}{\sigma_0^2} I)^{-1} X^T y$
$w_{MLE} = (X^T X)^{-1} X^T y$

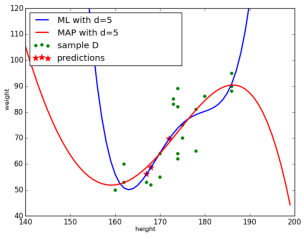


Fig. 10. Bayesian Regression missing values prediction:MLE VS MAP for $d = 5$

Comparison of Models			
Weight Prediction			
Height	MLE	Conditional Expectation	MAP
167	56.01	60.89	60.83
168	58.66	62.51	62.48
172	69.57	68.98	69.08

Squared Error: MLE:1276.2142, MAP:1353.2142
The Error with least square is lower for every d than Bayesian Regression. It explains complete data including noise, therefore, over fitting is more likely with it.

3.3. Boolean functions and the Boolean Fourier transform

This task is related to cellular automata which is consist of an infinite sequence of cells in two possible state -1 is a passive state and +1 is an active sate, as shown in Fig.12 (a) below, we consider the bits section as the design matrix X and the the output in the rules as y and the dead state is -1 and live state is 1, when we minimise w an calculate y, we notice wrong classification. Equations below shows the Boolean series expansion, fig. 12 (b) the prediction is not correct but in Fig. 12 (c) shows the correct result.

$$f : \{-1, +1\}^m \rightarrow \{-1, +1\} \tag{16}$$

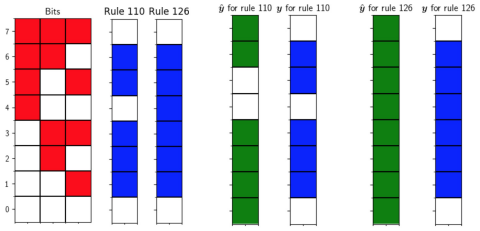
$$f(x) = \sum_{S \in 2^I} \omega_S \prod_{i \in S} x_i = 1 \tag{17}$$

$$\varphi_S = \prod_{i \in S} x_i, \varphi_\phi = \prod_{i \in \phi} x_i \tag{18}$$

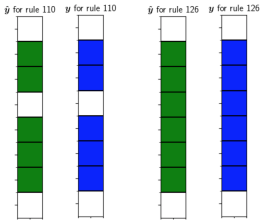
$$f(x) = \sum_{S \in 2^I} \omega_S \varphi_S = \omega^T \varphi \tag{19}$$

$\omega \in \mathbb{R}^{2m}$ and $\varphi \in \{+1, -1\}^{2m}$.

$$f(x) = \sum_{S \in 2^I} \omega_S \prod_{i \in S} x_i \quad , \quad \varphi = \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ X_3 \\ X_1 X_2 \\ X_1 X_3 \\ X_2 X_3 \\ X_1 X_2 X_3 \end{bmatrix} \tag{20}$$



(a) Bit section and rules (b) Output shows wrong classification



(c) Results

Fig. 11. Boolean functions and the Boolean Fourier transform

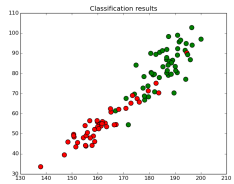
3.4. Nearest Neighbor Classifier

Nearest neighbor classifiers are a class of non-parametric methods that classifies objects based on closest training examples in the feature space. If it classifies dataset X according to $k = 1$ its called **1-Nearest Neighbor**. Therefore, if it determines class label from the majority of $k > 1$ nearest neighbors its called **k-Nearest Neighbor**. is robust against outliers while it losses its performance for irregular classes. For given data $X = x1, ..., xn$ where $x_i \in \mathbb{R}^m$. We have used the following KNN procedure for $k \in \{1, 3, 5\}$.

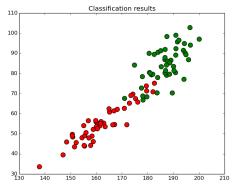
Algorithm 5 Classify test data based on given data using K-Nearest Neighbor Algorithm

1: Specify $k \in 1, 3, 5$, the number of neighbors.
2: For each Datum in test data find the K nearest neighbors in training data.
3: Classify test data based on majority vote of classes from K neighbors

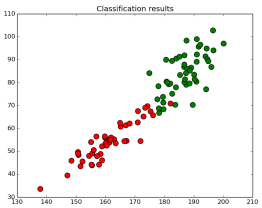
We have obtained following result after classification and we have performed run time evaluation as under:



(a) K = 1, Time elapsed = 0.272 seconds, accuracy = 88.54 %



(b) K = 3, Time elapsed = 0.297 seconds, accuracy = 89.58%



(c) K = 5, Time elapsed = 0.282 seconds, accuracy = 93.75%

Fig. 12. K-Nearest Neighbour Classifier with $K \in 1, 3, 5$

K	Accuracy%	Time Elapsed(sec)
1	88.54	0.272
3	89.58	0.297
5	93.75	0.282
Run-time Evaluation of K Nearest Neighbors		

As we have seen above that higher value of N lead to more accuracy and faster calculation. The complexity of KNN is $O(n)$ per data samples and in total $O(mn)$ where n = training samples and m = test samples. This is only good for small n. For very large n, X has to be queried in a loop which is impractical. Therefore, for large or very large n using KD trees or hash tables, the computational complexity of NN search can move from $O(n)$ to $O(logn)$ or $O(nlogn)$ where X dont change over time and also data pre-processing is required for it.

3.5. Computing a KD-Tree

KD tree is a binary tree for organizing points in a k-dim space. For KD-tree construction, we select the axis for splitting for example based on median or mid point and then we recursively partition the space into 2 parts: left and right tree.
Therefore, its insertion complexity is: $O(nlogn)$ and search complexity is: $O(logn)$ on average where n = number of samples i.e $|X|$
Classification with KD-Tree with alternate and variance dimension splits by mid-point and median and achieved efficiency are as under:

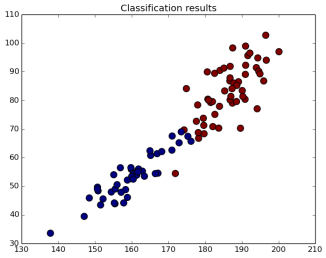


Fig. 13. **KD-Tree Classification** $K = 1$, Time elapsed $= 0.010$ seconds, Accuracy $= 94.97$

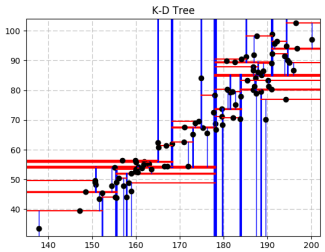


Fig. 14. **KD-Trees with alternate and variance dimension**

Selecting the median as a splitting point and alternating the axis as splitting hyperplane creates the most balanced tree. Therefore,Variance-median method creates the least balanced tree. Furthermore, it effects complexity, but might increase the accuracy in case that data is highly uncorrelated. Consequently, we have achieved run-time efficiency of KD-Tree for all query points: 0.01 seconds and 94.97% accuracy. While, in KNN for $k = 1$ accuracy was 88.54% and run time was 0.272 Seconds which shows that kD-Trees are much faster, efficient and accurate than KNN. Therefore, for the curse of dimensionality KD-Trees do not work well for high dimensional data.

4. CLUSTERING AND DIMENSIONALITY REDUCTION

4.1. Clustering

Clustering is about finding similarities, discovering relationships between members of dataset. Basically, it is the process of organizing objects into groups whose members are similar in some way.

4.1.1. K-Means Clustering

It is a method of vector quantisation. It partition n samples into k clusters. It determines $k << n$ clusters in which each sample belongs to the cluster with the nearest mean, serving as a prototype of the cluster. We will discuss here with given dataset Lloyd, Hartigan and MacQueen K-mean algorithm. We have founded the clusters using scipy.cluster.kmeans2 built-in function of Python Scipy library. In our data we have seen that it tends to fit simple Gaussian mixture model and each cluster corresponds to a voronoi cell;each point in cluster c_i is nearest to the $centroid_i$ among all cluster's centroids. It usually converges quickly but its quality crucially depends on the initialisation of the means $\mu_1, \mu_2 \cdots \mu_k$. The resulting Lloyd k-mean clustering plot for $k = 3$ is as under:

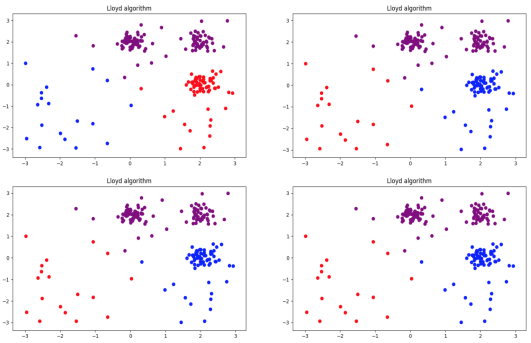


Fig. 15. Lloyd Algorithm K-Mean Clustering

4.1.2. Hartigan K-Mean Clustering

In Hartigan clustering, initial randomness lies in data points assignment rather than random centroids. It is based on a greedy heuristic and our approach is we select a point and assign it optimally in the current iteration then we recompute source clusters mean and destination clusters mean also we recompute the distance of the points in these 2 clusters and we assign it to the cluster that minimises the objective function in its current state.

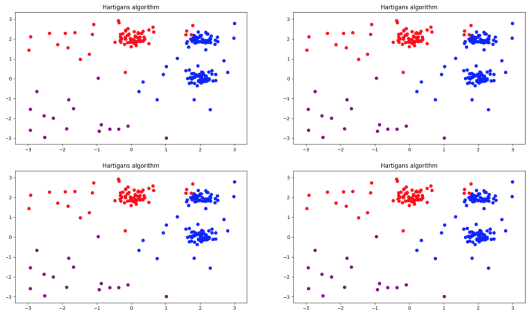


Fig. 16. Hartigan Algorithm K-Mean Clustering

Unlike Lloyd, clusters do not correspond to Voronoi cells. Generally, it is more robust than Lloyd's algorithm and converges quickly.

4.1.3. MacQueen K-Mean Clustering

It is very useful in streaming applications or when $|x| >> 1$. It can also be used for vector quantization and data compression. In MacQueen, we initialize centroids with the first k points that arrive then we assign the next data points to the nearest centroid and update the centroid of the current cluster in a weighted recursive manner as $\mu_n = \frac{n-1}{n}\mu_{n-1} + \frac{1}{n}x_n$.

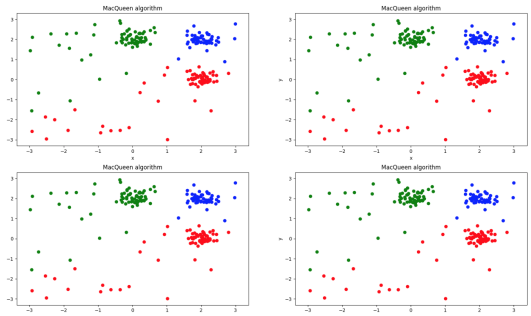


Fig. 17. MacQueen Algorithm K-Mean Clustering

The efficiency of each above mentioned K-means algorithms in terms of time is described in the following table as under:

K=3, Algorithm Runs=10 times	
Algorithm	**Average run time(sec)**
Lloyd	0.0198000192642
Hartigan	6.58339998722
MacQueen	0.0428999900818
K-Mean Clustering Average Run time	

Based on above table it is quite obvious that Lloyds algorithm is faster as compared to other k-mean clustering algorithms.

4.2. Spectral Clustering

Spectral clustering makes the use of eigenvalues (spectrum) of the similarity matrix of data to perform dimensionality reduction before clustering. Before going into detail of spectral clustering we need to know the following terms:

The matrix representation of a graph is called **Laplacian Matrix:** It is also known as admittance matrix, Kirchhoff matrix or discrete Laplacian and is used to find many useful properties of a graph [1]. For undirected graph, it is an $n \times n$ symmetric matrix with one row and column for each node defined by

$$L = D - A \tag{21}$$

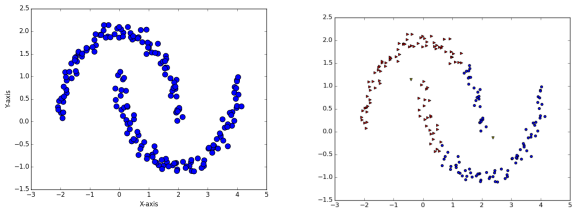
where,
$D = diag(d_1, ..., d_n)$ is the degree matrix, which is a diagonal matrix formed from the vertex degrees.
A is the adjacency matrix. The diagonal elements $l_(ij)$ of L are equal the degree of vertex v_i and off-diagonal elements $l_(ij)$ are -1 if vertex v_i is adjacent to v_j and 0 otherwise.

Furthermore, we can divide Spectral Clustering into three phases.
Pre-processing: In this phase, we construct the graph and the similarity matrix representing the data-set. **Spectral Representation:** In this phase, we form the associated Laplacian matrix and compute eigenvalues and eigenvectors of the Laplacian matrix as well as we map each point to a lower-dimensional representation based on one or more eigenvectors. **Clustering:** In this phase, we assign points to two or more classes, based on the new representation.
Now, we have applied spectral clustering to a non-convex data set and our goal is to find the best value of β which gave us coherent block diagonals in the similarity matrix. We have used SciPy's dendrogram functionality for visualization. therefore, finding a good similarity metric is a key and it models local neighborhood relations between data points. we have adopted the following procedure for Spectral Clustering.

Algorithm 6 Spectral Clustering of Nonconvex Dataset

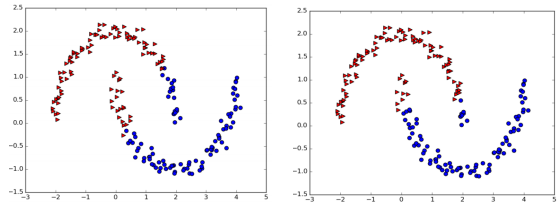
1: Given Dataset:$X = x_1, .., x_n \subset \mathbb{R}^2$
2: Compute the similarity matrix: $S_{i,j} = e^{-\beta \|x_i - x_j\|^2}$
3: Compute the diagonal matrix: $Dij = \begin{cases} \sum_i S_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
4: Compute the Laplacian matrix: $L = D - S$
5: find eigenvalues and eigenvectors of L and sort eigenvalues in descending order
6: If entry i in Fielder vector> 0
7: assign x_i to cluster 1
8: Else
9: assign x_i to cluster 2

Moreover, cluster points using eigenvectors of matrices derived from data; we compute similarity matrix, Laplacian matrix, eigenvalues and vectors and determine clustering based on the second eigenvector. The eigenvector $u2$ that corresponds to the second smallest eigenvalue is called the **Fiedler Vector** and is of significant importance in clustering. It is very useful for highly non-convex structure of the individual clusters. Furthermore, the Parameter β controls the size of the neighborhood of the cluster which shows how rapidly the similarity drops between x_i and x_j. Therefore, in similarity matrix $S_{i,j} = e^{-\beta \|x_i - x_j\|^2}$, smaller the value of β, more rapidly similarity will drop.

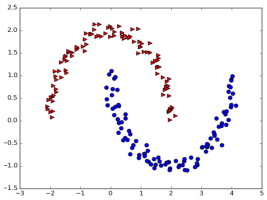


(a) Spectral Clustering Data Plot (b) k-means clustering: $k = 2$

Fig. 18. Data plot and k-means clustering for $k = 2$



(a) Spectral Clustering: $\beta = 1$ (b) Spectral Clustering:$\beta = 3$



(c) Spectral Clustering:$\beta = 6$

Fig. 19. Spectral Clustering and similarity matrix

From the Figure 18, we can see clearly that data is plotted into two clusters. By using k-means algorithm for $k = 2$, we classify the data into two clusters but we can see clearly that similar data is not connected.

While, in Figure 19, we can see that spectral clustering works well with non-convex data and as we increase the value of β, we move toward perfect classification. For $\beta = 6$, we have achieved the perfect classification. [1].

4.3. Dimensionality Reduction

In this section, we have reduced the dimensionality of data through PCA and LDA and compared their efficiency and robustness as under:.observe the differences.

4.3.1. Principal Component Analysis (PCA)

It is also called (discrete) **Karhunen-Loeve Transform (KLT)** or **Hotelling Transform**. It converts high dimensional data set to lower dimension. It represents data in new coordinate system and divides observed space into orthogonal subspaces with maximum variance. It can be used for **lossy data compression** by retaining those characteristics of dataset which contribute most to its variance. It transforms a number of possibly co-related variable into uncorrelated variables known as **principal components**.[2, 3, 4, 5]
Let set X or matrix X of zero mean data

$$X = x_i{}_{i=1}^{n} \subset \mathbb{R}^m \tag{22}$$

and
$X = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^{m \times n}$
we also consider the data covariance matrix as under

$$C = \frac{1}{n} \sum_i x_i X_i^T = \frac{1}{n} X X^T \tag{23}$$

where, C is real ,symmetric and positive semi-definite $m \times m$ matrix. Therefore, PCA is concerned with all solutions to

$$C v_j = \lambda_j v_j \tag{24}$$

and it allows for dimensionality reduction.

$$x_i = \sum_{j=1}^{m} \hat{x}_i^j v_j \tag{25}$$

where

$$\hat{x}_j^i = \langle x_i, v_j \rangle \Longleftrightarrow \hat{x}_i = V^T X_i \tag{26}$$

Now, determine $1 \leqslant k < m$ such that

$$y_i = \sum_{j=1}^{k} \hat{x}_j^i v_j \approx x_i \tag{27}$$

therefore for the average truncate error we have.

$$\epsilon = \frac{1}{n} \sum_i \|x_i - y_i\|^2 = \sum_{j=k+1}^{m} \lambda_j = \text{sum of all eigenvalues} \tag{28}$$

Note: The principal components are the eigenvectors with the largest eigenvalues of the covariance matrix. We only keep eigenvectors with the largest eigenvalues and omit all other eigenvectors corresponding to the smallest eigenvalues.

4.3.2. Implementation

Algorithm 7 Principal Component Analysis:Dimensionaly Reduction:$\mathbb{R}^{500} \longrightarrow \mathbb{R}^2and\mathbb{R}^{500} \longrightarrow \mathbb{R}^3$

1: Given data 500×150 data matrix X that is 150 data vectors $x_i \in \mathbb{R}^{500}$ and corresponding class labels for data vectors are:$y_i \in \{1, 2, 3\}$
2: Normalize data to zero mean
3: Compute data covariance matrix C
4: Compute eigen decomposition of C
5: Sort Eigenvalues in Descending order
6: Select eigenvectors with largest eigenvalues
7: Project the normalized data into 2D and 3D using eigenvectors of the largest eigenvalues accordingly.

We have obtained the following results using PCA by reducing the dimensionality and plotting the data accordingly in 2D and 3D.

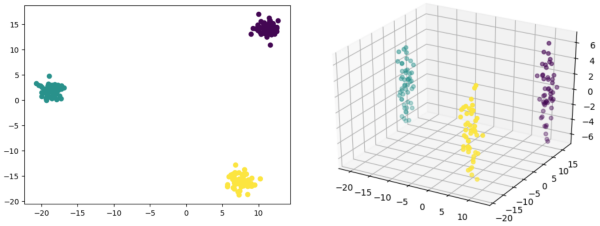


Fig. 20. 2D and 3D Plot for PCA

4.4. Multi-Class LDA

In Multi-ClassLDA If the number of classes is more than two, then a natural extension of Fisher Linear discriminant exists using multiple discriminant analysis. As in the two-class case, the projection is from high dimensional space to a low dimensional space and the transformation suggested still maximise the ratio of intra-class scatter to the inter-class scatter. It is based on the analysis of two scatter matrices: within-class scatter matrix and between-class scatter matrix.

It works as follows. Given a set of samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and their class labels $y_1, \ldots, y_n$: The within-class scatter matrix is defined as:

$$\mathbf{S}_w = \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu}_{y_i})(\mathbf{x}_i - \boldsymbol{\mu}_{y_i})^T \tag{29}$$

where, $\boldsymbol{\mu}_k$ is the sample mean of the k^{th} class. Therefore the between-class scatter matrix is defined as:

$$\mathbf{S}_b = \sum_{k=1}^{m} n_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{30}$$

where,
m is the number of classes
$\boldsymbol{\mu}$ is the Total sample mean
n_k is the number of samples in the k-th class. Then, Multi-Class LDA can be formulated as an optimization problem to find a set of linear combinations (with coefficients $\mathbf{w}$) that maximizes the ratio of the between-class scattering to the within-class scattering, as

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}} \tag{31}$$

The solution is given by the following generalized eigenvalue problem:

$$\mathbf{S}_b \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w} \tag{32}$$

Generally, at most $m - 1$ generalized eigenvectors are useful to discriminate between m classes. Therefore, for Multi-Class discriminant, we have used the following procedure:

Algorithm 8 Multi-class LDA: Dimensionally Reduction: $\mathbb{R}^{500} \longrightarrow \mathbb{R}^2$and $\mathbb{R}^{500} \longrightarrow \mathbb{R}^3$

1: Given data 500×150 data matrix X that is 150 data vectors $x_i \in \mathbb{R}^{500}$ and corresponding class labels for data vectors are:$y_i \in \{1, 2, 3\}$
2: Compute the within-class scatter matrix S_w
3: Compute the between-class scatter matrix S_B
4: Project the data into 2D and 3D using eigenvectors corresponding to k largest eigenvalues of the matrix $S_W^{-}1S_B$

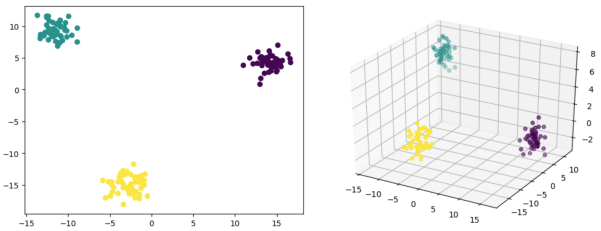


Fig. 21. 2D and 3D Plot for Multi-Class LDA

PCA reduces dimensionality while maximizing variance (i.e. thereby reducing information loss) LDA maximizes inter-class differences while reducing dimensionality.

4.5. Non-Monotonous Neurons

In this task the activation function are monotonous, for training neuron we can use non-monotonous activation function, by calculation gradient decent over the loss function below

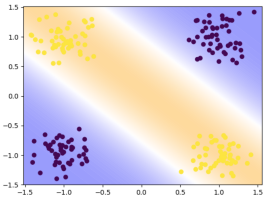
$$E = \frac{1}{2} \sum_{i=1}^{n} (y(x_i) - y_i)^2 \tag{33}$$

Then we derived the equation, after showing the result, we can use support vector machine with another result.

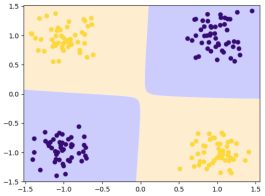
$$\frac{\partial E}{\partial \Theta} = \sum_{i=1}^{n} (\mathbf{x}_i (y(x_i) - y_i) * 2 \exp^{-\frac{1}{2}(\omega^T x_i - \theta)} * (\omega^T x_i - \theta)) \tag{34}$$

$$\frac{\partial E}{\partial \Theta} = \sum_{i=1}^{n} (\mathbf{x}_i (y(x_i) - y_i) * 2 \exp^{-\frac{1}{2}(\omega^T x_i - \omega)} * (\omega^T x_i - \theta) * (-x_i) \tag{35}$$

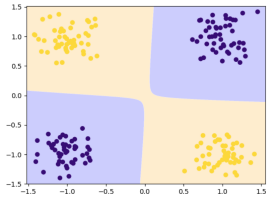
In Fig. 24 we observe how $\omega 20$, $\Theta 20$ influence the behavior and outcome of the training procedure. Also it shows the result of training a kernel SVM on the given data using a polynomial kernel.



(a) Result obtained from $\omega 20$, $\Theta 20$



(b) Apply SVM manually

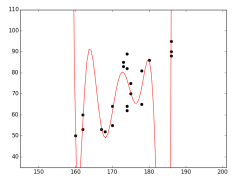


(c) scikit-learn SVM

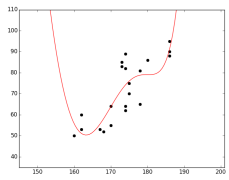
Fig. 22. Non-Monotonous Neurons Results

4.6. Exploring Numerical Instabilities

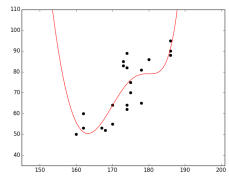
We have run the given piece of code several times. Therefore to identify the numerical instability in given programs we have made following observations:



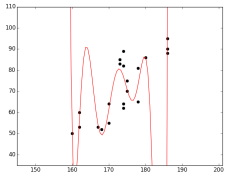
(a) Method1:regression using polyfit



(b) Method2:regression using the Vandermonde matrix and pinv



(c) Method3:regression using the Vandermonde matrix and lstsq



(d) Method4:regression on transformed data using the Vandermonde matrix and either pinv or lstsq

Fig. 23. Different Regression Methods plots

Method 1 and Method 4 are virtually identical, they give us the same output. Similarly, Method 2 and Method 3 are, also, virtually identical. What is handled by the first line of method 1 is being handled by the first two lines of method 2, 4 as well.

5. CONCLUSION

In this report, We have discussed different flavors of data distribution, drawing circles in different spaces, missing value prediction techniques, data clustering and classification as well as dimensionality reduction of data and finally have checked the numerical instabilities in different given programming methods.

Moreover, during practical implementation of all projects we have analyzed that outlier free data leads to better results while variations in value of K lead to variation in shape of Wei-bull distribution. Therefore, a unit circle may have various shapes and drawing a circle in $\mathbb{R}^2$ is required different mechanism than $\mathbb{S}^3$. Moreover, Least Square is more likely to over-fitting than MAP.
Furthermore, we also have analyzed that KD-Trees classifiers are more efficient, accurate and faster than KNN. Spectral Clustering works well with non-convex data and connect similar data points efficiently while k-mean does not. We also have observed that PCA maximizes variance while LDA maximizes inter-class differences to reduce the data dimensionality. Finally, we have observed that chosen programming techniques should be numerically stable for efficiency, correctness and robustness of a program.

6. ACKNOWLEDGMENT

First of All, I am very thankful to the Lord of the universe, without his blessings, I can never do anything. I am also very thankful to my family to motivate and encourage me by sending me motivational gifts. Furthermore, I am thankful to my team who shared their ideas and we enjoyed working together. The most important, I am extremely thankful to **Prof. Dr. Ing. Christian Bauckhage**, without his lectures, exercises, guidance and support along with his positive attitude, we will might not be able to explore the field in a optimal angel.

7. REFERENCES

[1] Ulrike von Luxburg, "A tutorial on spectral clustering," in *Statistics and Computing*, 2007.

[2] Roger Boyle Milan Sonka, Vaclav Hlavac, *Image Processing, Analysis, and Machine Vision(3rd edition))*, Thomson Learning.

[3] Christian Bauckhage, "Pattern recognition (lecture notes)," March 2018.

[4] Richard Szeliski, *Computer Vision: Algorithms and Applications*, Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[5] H. De Raedt K. Michielsen, "Morphological image analysis," *Computer Physics Communications 132 (2000) 94103*, 2000.