AMRITA SCHOOL OF ENGINEERING, COIMBATORE

COMPUTATIONAL ENGINEERING AND NETWORKING

15^{th} February, 2022

COMPUTATIONAL LINEAR ALGEBRA AND OPTIMIZATION TECHNIQUES FOR DATA SCIENCE

Hindi Varnamala(Alphabet) recognition using Random Kitchen Sink Algorithm

---

By:

Nishitkumar Prajapati CB.EN.P2DSC21018

CONTENTS

Abstract

Optical character recognition is a well known problem in computer vision and NLP. Wherever there is handwritten text processing is used, It has wide range of important applications. Here I have presented report on hand written characters' recognition of one of the most used language in the world, Hindi. I have used data set from kaggle. I have applied various pre-processing techniques and Random kitchen sink algorithm to recognise the hand written characters.

I. Introduction

Computer vision and NLP remains as top application fields of AI. Optical character recognition binds both computer vision and NLP together as it tries to recognise hand written data. This has massive application in various industries like Robotics, Industrial Production, Text automation, etc. Blind people can be helped if we combine this solution to speech synthesis as they would have gadget that speaks whatever is handwritten. So I have chosen one of the most spoken language in the world, Hindi. Hindi contains 52 alphabets, which is known as *Varnamaalaa*. It contains 33 consonants known as *Vyanjana*. It contains 11 vowels known as *Svara*. There are 8 special characters. In data 0 to 9 Hindi digits are also present. I have used kaggle data set to train and test the models.

On this data set, I have performed various pre-processing techniques to make it more suitable for processing, like standard scaling, MinMax scaling. I have performed principal component analysis to know how many features can be reduced to lower the computational cost of the data processing. After performing pre-procesing, I have applied Random Kitchen Sink algorithm before doing classification. Several neural networks and deep learning techniques have been used to solve this problem [8], [9], [10], [11], [14], which are computationally costly. To make this task simple and computationally less costly, I have used very simple, basic Machine learning method that can be performed on normal computer.

II. Literature

I have downloaded data set from kaggle website. I have reviewed previous works of optical character recognition [1], [2], [3], especially MNIST data set based research papers [5]. MNIST data set is standard data set for handwritten English digits 0 to 9. several neural networks have been trained for that. Researchers from Amrita Vishwa Vidyapeetham have performed multiple feature extraction operations on MNIST data set [5]. Several neural networks have been trained on Hindi hand written characters [9], [10]. Several deep learning techniques have been applied on this problem [8]. I have used various pre-processing techniques, which can be found in this book [6]. I have used basic machine learning algorithm, which can be found in this book [6].

III. Objectives

Objective of this project is, to apply principal component analysis to reduce data features and to apply random kitchen sink algorithm to recognise hand written Hindi characters.

IV. THEORETICAL BACKGROUND

I have used standard scaling, MinMax scaling and principal component analysis to pre-process data. I have used Random kitchen sink algorithm to classify data. Topics are discussed here in brief.

A. *Standard scaling*

It is done to make the mean of data to 0 and to make standard deviation of data to 1. In Python, command called *StandardScaler()* can be used to perform this operation. It is done using below given equation:

$$Z = \frac{X - \mu}{S} \tag{1}$$

where :

Z = standard score

X = data point

μ = mean

S = standard deviation(default = 1)

B. *MinMax scaling*

It is performed to normalize data into range [0,1]. After performing MinMax scaling, maximum value in the data can be 1 and minimum value in the data can be 0. In Python, command called *MinMaxScaler()* can be used to perform this operation. It is performed using below given equation :

$$X(std) = \frac{X - X.min(axis = 0)}{X.max(axis = 0) - X.min(axis = 0)} \tag{2}$$

$$X(scaled) = X(std) * (max - min) + min \tag{3}$$

Where :

X = data point

min, max = feature range

C. *PCA (principal component analysis)*

It is performed to analyse the information spread, which leads to reduction of no of data features. First we find covariance matrix of data features. covariance matrix tells us the relationship between data features whether they are dependent on each other or not, as well as nature of dependence, that whether they have negative covariance or positive covariance.

$$covariance = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n - 1} \tag{4}$$

Using covariance formula, we make covariance matrix for given features. if there are n features, then covariance matrix size will be $n \times n$.
example:

$$\begin{array}{c} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} var(x) & cov(x,y) \\ cov(x,y) & var(y) \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{bmatrix} var(x) & cov(x,y) & cov(x,z) \\ cov(x,y) & var(y) & cov(y,z) \\ cov(x,z) & cov(y,z) & var(z) \end{bmatrix} \end{array}$$

Now after finding covariance matrix, we find eigen values and eigen vectors connected to this matrices. biggest eigen value suggest that corresponding eigen vector direction has highest variance and so it has highest information. so now data is projected onto that eigen vector and it becomes principal component. In this manner, we find eigen values that has big values and corresponding eigen vectors becomes axis on which data is projected and we can neglect low eigen value connected eigen vectors. This leads to reduction in data features. maximum spread or information of data is now stored in principal components. so mathematically, for n vectors, of size $N \times 1$, mean $\bar{x}$ is as follows:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix} \qquad \bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{i=n} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix}$$

Let X be the $N \times n$ matrix, which has columns x_1 - $\bar{x}$, x_2 - $\bar{x}$,..... as follows:

$$X = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}$$

Let Q be the Q = XX^T as follows :

$$Q = XX^T = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix}$$

Q is known as covariance mtraix, a scatter matrix. it is symmetric and square matrix. Now we find the eigen values and eigen vectors of this Q matrix. Based upon that, new projected point's equation is as follows:

$$x_j = \bar{x} + \sum_{i=1}^{n} g_{ji} e_i \tag{5}$$

Where e_i are the n eigen vectors of Q with no zero eigen value. Now we sort the eigen values and then select eigen values which has higher values and neglect the lower eigen value and corresponding eigen vectors. This is how dimensionality reduction is achieved through PCA.

D. ***Random kitchen sink Algorithm***

Supervised learning can be considered as a method in obtaining information about the distinction or relation between the data sets at hand. A decision model is built for data prediction. The decision model derived is used to provide class labels (or output values) to the data depending on the various potential features[4].

Support vector machines (SVMs) represent a type of supervised learning algorithms which devises a hyper plane that distinguishes two or multiple classes while trying to achieve a maximum division between the classes[12]. The SVM methods evolved with various developments to be usable in large data sets but limited the functionality of kernel used or vice versa. Storage of the kernel matrix (size of the order of 10^5) for large data sets and its analysis gave heavy burden on RAM which resulted in long processing time. Thus the method became unsuitable for large data sets[15]. This issue was fixed to a large extend by the random feature mapping method developed by Ali Rahimi and Ben Recht which transformed both 'constraints' to 'just another parameter in the analysis[7].

Here, the feature works directly with data rather than on the kernel function. Instead of finding a function curve to separate data as done in prior models[4], the data ($\in R^d$), is mapped onto a higher dimensional space ($\in R^D such that D >> dandisoforderasmentionedin[7]$) and a decision surface is derived in that space. The kernel function can be approximated by taking the inner product of the mapping functions[7]. The relationship is given

$$k(x, y) = \langle \phi(x), \phi(y) \rangle$$

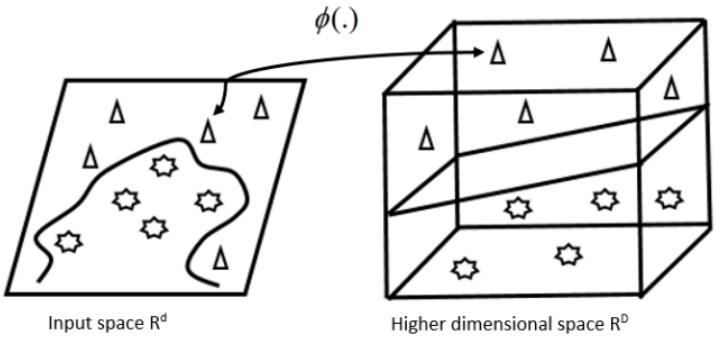
where $k(x, y)$ denotes the positive definite kernel function while $\phi(x), \phi(y)$ represent the mapping function.

The RKS approximates an RBF (Radial Basis Function) kernel of the form,

$$k(x,y) = e^{-\sigma\|x-y\|_2^2} = e^{-\sigma(x-y)^T(x-y)} = e^{-\sigma z^T z}$$
$$e^{-\sigma z^T z} = e^{-z^T(\sigma \mathrm{I})z} = e^{-z^T \Sigma^{-1} z}$$

where x, y denotes a data pair, $z = x - y$ is a single vector variable function, I is a $d \times d$ identity matrix and $\Sigma = \sigma^{-1} I$ represents the covariance matrix given by,

$$\Sigma = \begin{bmatrix} 1/\sigma & & & \\ & 1/\sigma & & \\ & & \ddots & \\ & & & 1/\sigma \end{bmatrix}.$$



RBF follows Gaussian distribution and is shift invariant, ie,

$$k(x,y) = k(x-a, y-a) = k(0, x-y)$$

The inverse of its Fourier transform $p(\Omega)$ is given by,

$$k(x,y) = k(x-y) = \int_\Omega e^{j(x-y)^T\Omega} p(\Omega) d\Omega$$
$$= \int_\Omega e^{jz^T\Omega} p(\Omega) d\Omega$$

This states that inverse Fourier transform of an RBF kernel can be interpreted as the expectation of a random variable $e^{j(x-y)^T\Omega}$[Bochner's theorem].

$$E(e^{-jz^T\Omega}) = \int p(\Omega) e^{jz^T\Omega} d\Omega$$

Also, $P(\Omega)$ follows multivariate normal distribution. The above integration can be considered as sampling several Ω vectors

from multivariate normal distribution and then taking average of the function inside integral over D samples.

$$k(x,y) = \frac{1}{D}\begin{bmatrix} e^{j(x-y)^T\Omega_1} \\ e^{j(x-y)^T\Omega_2} \\ . \\ e^{j(x-y)^T\Omega_D} \end{bmatrix}$$

$$= \frac{1}{D}\left\langle \begin{bmatrix} e^{jx^T\Omega_1} \\ e^{jx^T\Omega_2} \\ . \\ e^{jx^T\Omega_D} \end{bmatrix}, \begin{bmatrix} e^{jy^T\Omega_1} \\ e^{jy^T\Omega_2} \\ . \\ e^{jy^T\Omega_D} \end{bmatrix} \right\rangle$$

$$= \left\langle \begin{bmatrix} \frac{1}{\sqrt{D}}e^{jx^T\Omega_1} \\ \frac{1}{\sqrt{D}}e^{jx^T\Omega_2} \\ . \\ \frac{1}{\sqrt{D}}e^{jx^T\Omega_D} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{D}}e^{jy^T\Omega_1} \\ \frac{1}{\sqrt{D}}e^{jy^T\Omega_2} \\ . \\ \frac{1}{\sqrt{D}}e^{jy^T\Omega_D} \end{bmatrix} \right\rangle$$

Also, $k(x,y) = (\phi(x))^T\phi(y)$. From the above expression we infer that,

$$\phi(x) = \begin{bmatrix} \frac{1}{\sqrt{D}}e^{jx^T\Omega_1} \\ \frac{1}{\sqrt{D}}e^{jx^T\Omega_2} \\ . \\ \frac{1}{\sqrt{D}}e^{jx^T\Omega_D} \end{bmatrix}, \quad \phi(y) = \begin{bmatrix} \frac{1}{\sqrt{D}}e^{jy^T\Omega_1} \\ \frac{1}{\sqrt{D}}e^{jy^T\Omega_2} \\ . \\ \frac{1}{\sqrt{D}}e^{jy^T\Omega_D} \end{bmatrix}$$

Here, finite (D) dimensional $\phi(x)$ are created by taking cosines and sines of the higher dimensional mapped data, thereby avoiding complex numbers. This do not affect the value of $k(x,y)$. Once we have $\phi(x)$ for all data points, any linear classifier can be used for the classification.

E. ***Regularised least squares***

Least Square aims at finding an optimum solution of over determined systems, that is, for a set of equations where the number of equations are more than the unknowns. The method uses the concepts of Linear Algebra and Optimization techniques. The sum of the square of the errors is minimized in the overall solution. The method of Least square have been in use since the 18th century[13]. A number of remarkable theories in the astronomical field have been formulated out of the concept of least squares. It is widely applied in the field of statistics[13]. Major application lies in data fitting. In the best fit, we have a minimal sum of squared residuals[12]. Least Square falls into two categories: linear or ordinary least squares and nonlinear least squares

1) ***Least square estimation using first order optimality condition****:* If the function is convex and differentiable, we can obtain the optimum point x* , the location at which the function value is minimum by applying first order condition that

$$\nabla f\left(x^*\right) = 0(vector)$$

Let $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ be the given data points. We have to find slope m and intercept c of the regression line given by y = mx + c. In matrix form the problem is formulated as

$$z^* = \begin{pmatrix} m \\ c \end{pmatrix} = \arg\min_z \left\| Az - y \right\|_2^2$$

where Az = y + e is

$$\underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ x_n & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} m \\ c \end{bmatrix}}_{z} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ . \\ y_n \end{bmatrix}}_{y} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ . \\ e_n \end{bmatrix}}_{e}$$

The objective function is basically,

$$e^T e = \left(Az - y\right)^T \left(Az - y\right) = \left\| Az - y \right\|_2^2$$

$e^T e = e_1^2 + e_2^2 + ... + e_n^2$ is sum of square of errors. This in turn depends on m and c which in compact form is the vector z . The optimal value of m and c should minimize the sum of square of errors. Note that error term $e_1, e_2, e_3, ..., e_n$ can assume positive and negative values. Therefore error sum of square is minimized rather than error sum. By applying first order condition, we have

$$\begin{aligned} f(z) &= \left(Az - y\right)^T \left(Az - y\right) = \left\| Az - y \right\|_2^2 \\ \nabla f(z) = \mathbf{0} &\Rightarrow 2A^T \left(Az - y\right) = \mathbf{0} \\ &\Rightarrow z^* = (A^T A)^{-1} A^T y \end{aligned}$$

If columns of A are independent, $A^T A$ is invertible. If not,

$$z^* = A^+ y$$

A multi-variable classification problem can be formulated as,

$$W^* = \arg\min_{W^{m\times n}} \left\| Y - XW \right\|_F^2$$

In certain cases the data may get over-fitted resulting in a useless prediction. Interpolative prediction for a given X is accomplished by fitting a regression function. The over-fitting usually happens when we try to approximate the function using higher degree polynomials. In many cases, we may not be able to predict the polynomial function which approximates the data. Regularization is a technique used to capture the real trend of data by function and thereby avoids over-fitting. To achieve this, an extra term $\lambda||z||_2^2$ is added to the objective function.

$$\begin{aligned} f(z) &= (Az-y)^T (Az-y) + \lambda \|z\|_2^2 \\ &= \|Az-y\|_2^2 + \lambda \|z\|_2^2 \end{aligned}$$

$$z^* = \arg\min_z \|Az-y\|_2^2 + \lambda \|z\|_2^2$$

λ is a control parameter which the modeler has to specify. This term will try to reduce the absolute value of model parameters. The net effect is the reduction in over-fitting. By changing λ , we can control the amount of over-fitting. A suitable λ can be obtained by trial and error method.

$$\begin{aligned} \nabla f(z) = \mathbf{0} &\Rightarrow 2A^T (Az-y) + 2\lambda z = \mathbf{0} \\ &\Rightarrow z^* = (A^T A + \lambda I)^{-1} A^T y \\ \frac{\partial f(W)}{\partial W} = 0 &\Rightarrow W^* = \left(X^T X + \lambda I\right)^{-1} X^T Y \end{aligned}$$

*2) **Multiclass learning with Regularized least square**:* Let us consider the data set containing k class of objects. The number of features is n and total number of objects is m . Let a $k-tuple$ represent the class value. The data matrix is of size $m \times n$ where, each data corresponds to an object, is present in rows. Matrix Y hold corresponding label vectors, each of size $m \times k$. A matrix W of size $n \times k$ which map $n-tuple$ data vector to corresponding label vector that contains $n \times k$.

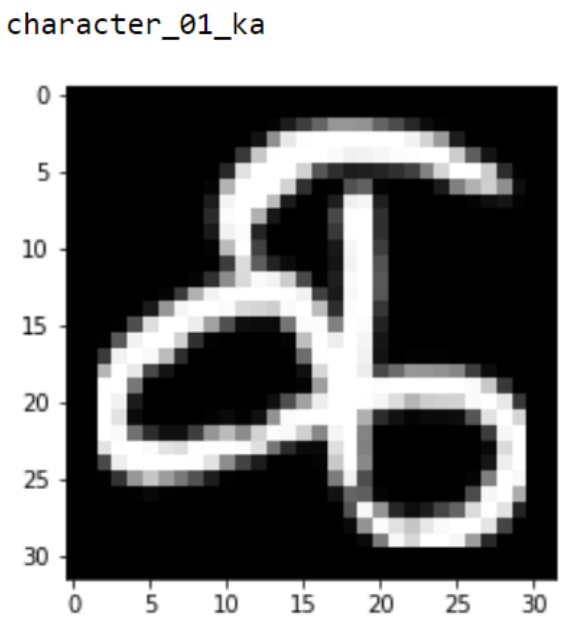
$$W^* = \arg\min_{W^{mxn}} \|Y - XW\|_F^2 + \lambda \|W\|_F^2$$

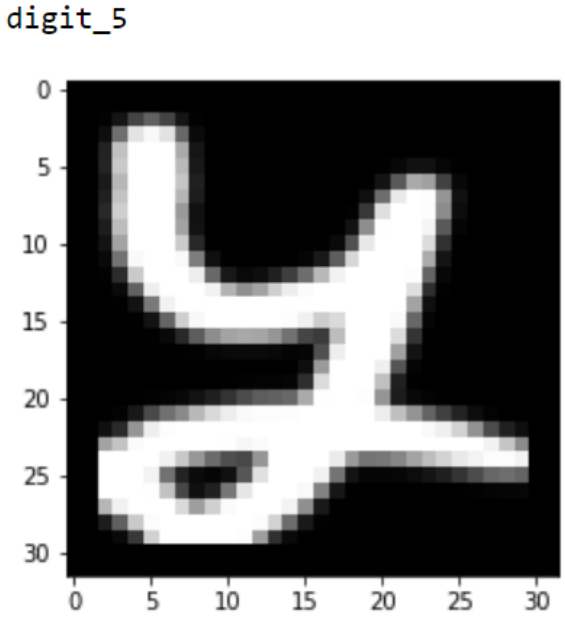
$$\begin{aligned} f(W) &= \|Y - XW\|_F^2 + \lambda \|W\|_F^2 \\ f(W) &= Tr\left(Y^T Y + W^T X^T XW - Y^T XW\right) \\ &\quad - Tr\left(W^T X^T Y\right) + Tr\left(\lambda W^T W\right) \\ \frac{\partial f(W)}{\partial W} = 0 &\Rightarrow W^* = \left(X^T X + \lambda I\right)^{-1} X^T Y \end{aligned}$$

V. METHODOLOGY

A. *Data Description*

Hindi character recognition data contains 92000 instances/images. It has 46 classes. 36 consonants and 10 distinct digits from 0 to 9. Each class has 2000 instances. Each instance has 1024 features, which are pixel values of that image. some of the images are shown below:





B. *Pre-Processing Data Set*

Now we apply standard scaling. then we apply min-max scaling. this ensures data is zero centered, with standard deviation 1 and in the range of 0 to 1. Now we perform train test split such that each class has equal no of instances in train and test data. It is achieved through using stratifying parameter in train test split command in python. we stratify data with respect to class labels. we split data into ratio of 80:20. so each class has 1600 training data and 400 testing data. Then we perform PCA on train data. we fit this PCA instance on train and test data and transform data to reduce the features to 335 from 1024. From experiment, it became known that 99 percent of information is contained in 335 features. so we drop 665 features. Then we perform random kitchen sink algorithm to get linear separability.

C. *Classification*

I applied regularised least squares method to classify the data labels.

VI. Conclusion

I applied PCA on data to reduce features, for which there is no major loss of accuracy. At the same time I achieved more than half of the feature reduction and still got 99 percent data information. I applied Random kitchen sink algorithm and got 100% accuracy. In future I would want to apply this method via mobile application to do LIVE recognition on the go. I also want to create data set for other language such as Gujarati, Malayalam, Telugu, etc from scratch and contribute to the world for experimentation. I also want to develop multi-language hand written character recognition.

References

[1] Sonika Dogra and Chandra Prakash. Pehchaan: hindi handwritten character recognition system based on svm. *International Journal on Computer Science and Engineering*, 4(5):718, 2012.

[2] Mohammad Said El-Bashir, Rahmita Wirza OK Rahmat, Fatima Ahmad, and Md Nasir Sulaiman. Principal components analysis for hindi digits recognition. In *2008 International Conference on Computer and Communication Engineering*, pages 738–740. IEEE, 2008.

[3] R Karthieswari, M Sreethivya, D Deva Rajendra Kumar, and KP Soman. Tamil characters classification using random kitchen sink algorithm (rks). In *Proceedings of the 2014 international conference on interdisciplinary advances in applied computing*, pages 1–5, 2014.

[4] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[5] Deepthi Praveenlal Kuttichira, V Sowmya, and KP Soman. Digit recognition using multiple feature extraction. *a, a*, 1(2):1–2.

[6] Andreas C Müller and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[7] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[8] R Vijaya Kumar Reddy and U Ravi Babu. Handwritten hindi character recognition using deep learning techniques. *Dept. of CSE, Acharya Nagarjuna University, Guntur, India, International Journal of Computer Sciences and Engineering*, page 1, 2019.

[9] Dayashankar Singh, Maitreyee Dutta, and Sarvpal H Singh. Neural network based handwritten hindi character recognition system. In *Proceedings of the 2nd Bangalore Annual Compute Conference*, pages 1–4, 2009.

[10] Gunjan Singh and Sushma Lehri. Recognition of handwritten hindi characters using backpropagation neural network. *International Journal of Computer Science and Information Technologies*, 3(4):4892–4895, 2012.

[11] Nikita Singh. An efficient approach for handwritten devanagari character recognition based on artificial neural network. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 894–897. IEEE, 2018.

[12] KP Soman, R Loganathan, and V Ajay. *Machine learning with SVM and other kernel methods*. PHI Learning Pvt. Ltd., 2009.

[13] Stephen M Stigler. Gauss and the invention of least squares. *the Annals of Statistics*, pages 465–474, 1981.

[14] Brijesh K Verma. Handwritten hindi character recognition using multilayer perceptron and radial basis function neural networks. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 2111–2115. IEEE, 1995.

[15] Johan von Tangen Sivertsen. *Scalable learning through linearithmic time kernel approximation techniques*. PhD thesis, Master's thesis, IT University of Copenhagen, 2014.