Machine Learning Algorithms and Applications

DISSERTATION

Submitted in partial fulfilment of the requirement
for the award of the degree of

MASTER OF TECHNOLOGY

IN

ELECTRONICS ENGINEERING

(Electronic Circuits and System Design)

Submitted by
Mohd Arqam Khan

Under the Supervision of

Prof. Syed Atiqur Rahman

DEPARTMENT OF ELECTRONICS ENGINEERING
ZAKIR HUSAIN COLLEGE OF ENGINEERING & TECHNOLOGY
ALIGARH MUSLIM UNIVERSITY ALIGARH
ALIGARH-202002 (INDIA)
2021

DEPARTMENT OF ELECTRONICS ENGINEERING
ZAKIR HUSAIN COLLEGE OF ENGINEERING & TECHNOLOGY
ALIGARH MUSLIM UNIVERSITY ALIGARH
ALIGARH-202002 (INDIA)
2021

CERTIFICATE

This is to certify that the work of dissertation which is being presented in this report entitled as **"Machine Learning Algorithms and Applications"** is submitted by **Mr. Mohd Arqam Khan** in partial fulfilment of the requirement for the award of the Degree of **M.Tech. (Electronic Circuits and System Design)**, Department of Electronics Engineering, Aligarh Muslim University, Aligarh is an authentic record of his own work carried out throughout M. Tech. under my guidance and supervision.

Dr. Syed Atiqur Rahman

Dated:

ACKNOWLEDGEMENT

I am thankful to God for guiding me in completing this report and my course work in-time. With utmost sincerity, I express my heartfelt thanks to my supervisor Prof. Syed Atiqur Rahman, his astute and critical inputs helped me in every step of the way. He was instrumental for his erudite guidance and suggestions and without his critical evaluation and moral support it would not have been possible for me to successfully complete my Dissertation.

I am also indebted to the commendable department and the exceptional facilities that are provided to complete my work. At last I would like to thank my friends and family for their moral support and encouragement.

Z.H.C.E.T **Mohd Arqam Khan**
AMU, Aligarh

ABSTRACT

Machine Learning (ML) is one of the most dominant emerging field in engineering for the past decade. The evolution of technology especially in computing capabilities and capacity to generate, store, access enormous amounts of data are few of the key factors that results in the outburst of ML. Practically almost every industry utilizes ML in one way or another. Due to the high versatility of ML, the potential applications are wide-ranging and substantial. One of the important application of ML is recognizing handwritten digits. Nowadays handwritten digits are utilized in processing bank checks, smart calculators, sorting of postal zip codes etc. Roughly, a complete handwritten digit's recognition system constitutes a scanning system, a training model and a classifier.

In this report a review of popular supervised machine learning algorithms is presented and implemented on three different handwritten digits' datasets. Logic gates by Perceptron Pocket Learning Algorithm is implemented. The curve approximation on the dataset of chance of admit of students into a university by using linear regression (pseudo-inverse) and gradient descent is implemented. Four datasets are used in this work; one is used for curve approximation and three are used for classification. The best accuracy on MNIST dataset is obtained by k-nearest neighbour algorithm with k=6 of 96.6% with a training time of 30 minutes, on Arabic Dataset it is obtained by k-NN with k=6 giving 97.94% with training time of 31 minutes and on self-generated dataset by k-NN giving 74%.

The best training time with acceptable accuracy is obtained by 3-layered artificial neural network using mini-batch gradient descent for training on both MNIST and Arabic datasets. An accuracy of 96.2% under 2.2 minutes of training is achieved on MNIST and 97.1% under 2.4 minutes on Arabic dataset. A binary logistic regression based approach is developed for 10-class Classification; the testing accuracy obtained on Arabic dataset is 96% and on the self-made dataset is 60%.

Keywords- Machine learning, handwritten digit recognition system, logic gates, curve approximation, perceptron pocket learning algorithm, pseudo-inverse linear regression, gradient descent, mini-batch gradient descent, MNIST handwritten digits, k-nearest neighbours.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 General Introduction

In layman's term, machine learning is the process by which a machine learns and implement a certain task like human beings learn. Technically, a system that learns, analyse and adapt without being explicitly programmed by using different statistical models and extract certain form of pattern or trend is known as machine learning. Machine learning uses computational algorithms on empirical data, extracting some form of trend and thereby converting it into usable and meaningful data. The first ML model was introduced in 1949 by Donald Hebb [1] in the booked titled as "The Organization of Behaviour", this book summarizes the theories by Hebb on neuron excitement and communication between different neurons. In 1957, the first perceptron model was introduced by Frank Rosenblatt at Cornell Aeronautical Laboratory [2] This perceptron was intended as a machine rather than a software, which was constructed for image recognition. In the 1960s, the discovery of multilayers added numerous possibilities for implementation of non-linear functions. By utilizing more layers instead of only one layer, more processing power could be harnessed. Major conglomerate such as Google, Facebook, Microsoft, Amazon, etc., utilize machine learning in almost every major disciplines. A plethora of data is generated and collected by these technological super giants. This is possible due to the intricate connectivity of peoples through internet and technology. Due to this, researchers are exposed to substantial opportunities by which more advanced statistical/computational approaches for auto-generation of models from empirical data are possible. Initially ML was utilized as a training program for artificial intelligence but in late 1970s and 1980s, AI researchers focused primarily on logical and knowledge-based systems rather than the algorithms. This led to the separation of machine learning from artificial intelligence. Now ML is considered as a subset field of AI and in some particular scenarios a subset field of computer science. Because of this it is considered the most versatile field as it is used in multiple disciplines and in inter-disciplines. The key difference between ML and statistical programming is that former develops dynamic algorithms by analysing and inferring from data while the latter only follows specific predefined rules to develop

relationships between variables. “ML domain is about developing an algorithm to produce an outcome based on previous experience and data” (Suciu, Marginean, Kaya, Daume, & Dumitras, 2018).

“A machine learning task aims to identify (to learn) on learning a function $f : X \rightarrow Y$ that maps the input domain X (of data) onto output domain Y (of possible predictions)” [3]. The choice of function f depends on the type of learning algorithm and hypothesis of training model. Mitchell (1997) defines "learning" as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [3]. The performance measure P is the quantitative measure of the quality of model being trained that is how well is the model learning. Generally, accuracy of model is chosen as the performance parameter for classification tasks where accuracy is defined as the correct classifications out of the total classifications. Experience E depicts the datasets through which the model undergo training and testing.

Now the question is why do we even need ML or why it is important? The primary difference between humans and computers is that latter cannot improvise new decisions on their own without being explicitly programmed. They cannot “think” like humans can. Thus, all computer decisions are actually translations of a predefined decisions made elsewhere and stored into a format of machine codes and computer only executed them. Problem with this type of approach is that not all problems cannot be explicitly coded in a computer friendly manner or are at least extremely difficult and tedious. Scenarios involving vague, ambiguous or abstract decision making cannot be effectively coded but ML provisions such cases. Nowadays, decisions based on data mark the difference between whether an organization is keeping up or falling behind the competition. Machine learning is one of the key-factors that differentiate between the quality of feedback decisions between different customer based companies. Domains which are harmful to human beings can utilized models based on ML which are as closest as we can go currently. Factors like growing volumes and varieties of available data, availability of large data storage facilities, increase in computational processing which is now cheaper and more powerful than ever before, all of these things are bringing about the possibility to quickly, automatically and effectively produce models, which can analyse and extract numerous trends from colossal amount of data. Thus, by

building these type of precise models, any organization can take advantage to increase their profit and minimize the risk to great extent.

1.2 Applications of Machine Learning

The following section roughly summarize the major application areas of machine learning. In Fig. 1.1, three levels of applications are shown; next level being the sub-domain of previous broader applications. The last levels are the most specialized applications and generally are implemented by researchers.

The major application domains of machine learning applications are summarized as follows:

- Computer Vision: This does not strictly falls under the subset of machine learning but rather it's also a subset of artificial intelligence just like ML. It is the process by which computers can perceive high level understanding of images and videos similar to how human beings perceive. To achieve this ML is utilized.
- Prediction: This refers to the correct projection or forecasting of results based on the training of model on the historical data of that particular area. The models that are utilized for analysis, classification and recommendation purposes falls under this category of application of machine learning.
- Semantic Analysis: This refers to the extraction of meaningful information from unstructured text. Through this analysis computers are able to understand and interpret large paragraphs, documents by checking for the grammatical and linguistic structures. In a nutshell it's the way for machines to identify relationships between individual words.
- Natural Language Processing: This is the domain through which machines are able to analyse and understand not only written but also spoken human language. This is possible by performing various processing task like speech recognition, sentiment analysis and text summarization. All the virtual assistants like Alexa, Siri, Google assistant uses NLP to decode and give answers to humans.
- Information Retrieval: This field enables a user to retrieve document or any other data by matching the specific query or description. The searching mechanism used by different search engines is a prime example. To refine the retrieval mechanism, ML is utilized by taking the feedback for different users; their way of searching, style of description provided.

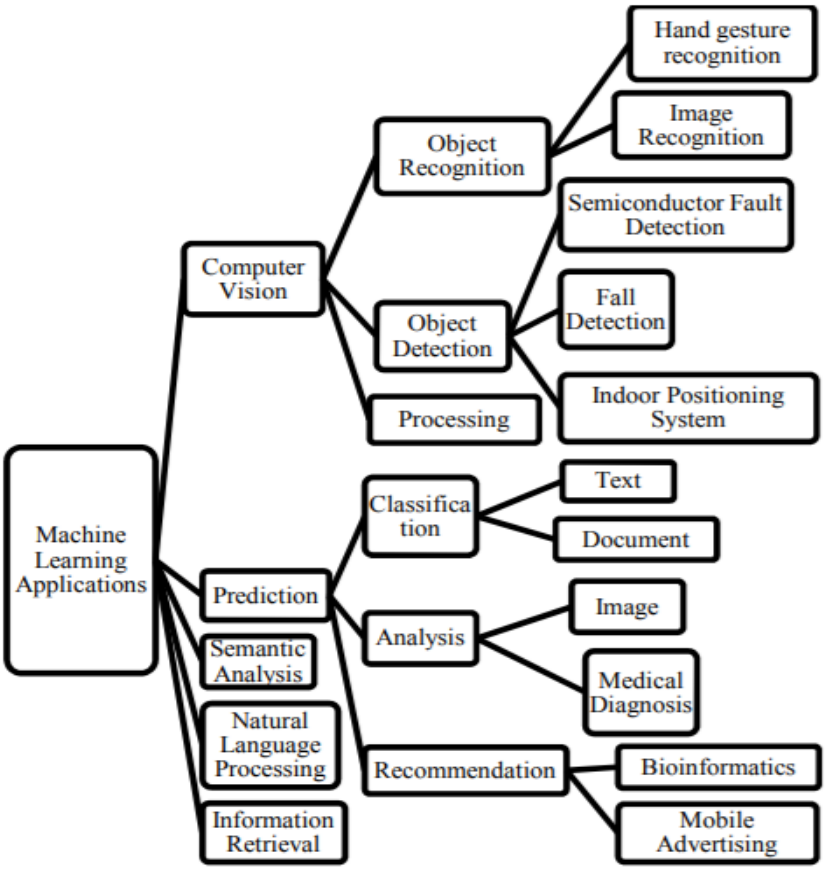


Fig. 1.1 Applications of Machine Learning

- Handwritten recognition: It refers to method of analysing and understanding the natural written characters, numeric digits of human beings by machines through different data acquisition tools like cameras, scanners, tablets/smartphones.

1.3 Literature Survey

In 1990, Y.Le cunn [4], applied a large backpropagation network to a real image processing problem and showed that not very complex pre-processing of data is required. Instead of using feature vectors, direct images were used as inputs to the network. The pre-processing of data is performed; first binarization of data and then proper segmentation which was very difficult because the size of digits was varying in nature. Therefore, normalization was applied before segmentation by linear transformation. The connections in the neural network were all adaptive in nature. The network used four hidden layers, two of them were weight feature extractors and two of them were subsampling layers. The network designed was not

fully connected neural network instead it was highly constrained specific network to reduce the number of adjustable parameters. The error rate on the whole test set was obtained as 3.4% and MSE of 0.024.

In 1998, Daniel Cruces Alverez [5], proposed a clustered and multilayer neural network in which training algorithm used was backpropagation for the recognition of handwritten and printed numerals. For each image of the digit the feature used is the presence/absence of the line segment and the orientation was used. The dataset consists of around 9300 handwritten digits by 90 different writers. The extracted data was normalized into image size of 16x16. To obtain the position and orientation of line segments, a first order differential axis detector was used known as Kirsch Detector. The network was made up of one hidden layer and output layer of 10 units. The input unit have 5 subnets have 4x4 units. This network is fully connected in each layer. The network obtains the best accuracy of 96.2% for classification. A classification error of 3.8% with no rejection and 1% with a rejection rate of 9% was obtained.

In 2010, Dewi Nasian [6], proposed a method for recognition of handwritten isolated Latin characters using freeman chain codes representation. The method was trained and tested on NIST handwritten dataset. Initially thinning was applied to the database in order to remove extra redundant information and to compress the images to obtain the characteristic features. This paper employs random algorithm to generate freeman chain codes as the primary feature extraction method. A total of 69 features were extracted to feed the neural network having one hidden layer (69 units). The overall accuracy of classification on 81179 samples containing upper and lower case letters+digits (0-9) was 97.48%.

In 2012, P.Pandi Selvi [7] utilized feedforward neural network for the recognition of handwritten numerals. For pre-processing they followed several steps; firstly, they performed filtering to remove noise both horizontally and vertically by using a 3x3 Gaussian filter. Secondly, the coloured images were transformed into binary images by using OTSU's thresholding method. Thirdly, labelling or segmentation of binary connected components is performed. Finally, the segmented binary images were rescaled by bilinear interpolation and was fed into multi-layer neural network. In the network two layer were used; one hidden layer and one output layer. The training algorithm used here was backpropagation along with

delta parameter update rule. This whole method was coded on MATLAB and an accuracy of 95-100% is obtained.

In 2013, SuthasineeIamsa-at and Punyaphol Horata [8], two different classifiers are compared for their recognition capability viz. 'Deep Learning Feedfoward Backpropagation Neural Network' (DFBNN) and 'Extreme Learning Machine' (ELM). Three datasets were used; Thai handwritten characters, Bangla handwritten numerals and Devanagri handwritten numerals. Images was resized to 32x32 pixels and feature extraction was done by Histograms of oriented gradients (HOG). The dataset was categorized into raw extracted features (original) and HOG extracted features. The method utilized 10-fold cross validation. DFBNN used 80 hidden layer units and logistic activation function. The results showed that the HOG extracted features performed better i.e gave better recognition rates than original features.

In 2014, Rajiv Kumar and Kiran Kumar [9], a study on recognition accuracies on Devnagiri numerals by using different feature extraction methods were shown. Simple features such as pixel values, slightly computationally demanding profiles and complex gradient features were utilized and compared their recognition accuracies. The data is extracte from two different forms, they named them as form-1 and form-2. Form-1 was used to extract the isolated digits, characters and form-2 was used to extract the constrained and unconstrained words for panagram. An automatic skew correction (random transform) was performed before applying image segmentation (OTSU's method) on form-1 followed the noise filtering. Four different classifier was used to compare the results viz. feed forward neural network, fitness function, cascade neural network and k-nearest neighbour of which first three are neural network based and the last one is statistical based. Finally, the paper combines different classification models and obtained an average accuracy of 97.87%.

In 2015, Saeed AL-Mansoori [10], implemented and tested a multi-layer neural network on MNIST handwritten digit dataset. The observation of system accuracy was done by varying the number of hidden units one of the hyper-parameters in neural networks. The feature used was pixel values (20x20=400). The network used one hidden layer and one output layer having 10 units each representing the corresponding digits. For training, gradient descent coupled with backpropagation was used. A total of 5000 samples were used out of which 4000 were utilized in training and 1000 were in testing; the overall training accuracy of 99.32% and testing accuracy of 100% were attained.

In 2016, Haider Al-Wzwazy [11], the design of deep convolutional neural network (DCNN) on Arabic handwritten digits was proposed by careful selection of critical hyper-parameters. The dataset used have 46000 digits from 840 different subjects from different grades and institutions. The network has 5 layers; first layer has 6 feature maps; second layer is subsampling layer (6 feature maps); third and fourth are max-pooling layers (16 feature maps); and one final Softmax layer for classification. For every iteration a different mini-batch of 128 samples and a gradual decreasing learning rate was used. The overall accuracy of 95.7% was obtained.

In 2019 Saqib Ali and other researchers [12], proposed convolutional neural network based classifier of MNIST handwritten digits by using the DL4J framework for recognition. The network consists of two feature extraction layers followed by two feature classification layers. The overall accuracy of 99.26% was obtained. Also the study of change in accuracies with number of CNN layers was given and in their case the overall accuracy was decreasing with increasing CNN layers.

1.4 Objective and motivation

The problem of handwritten digit classification is of the most coveted application of machine learning and computer vision primarily because of its application in the real world. There are multiple challenges that are associated with this problem which if remains unresolved would affect gravely. Some of them are; the style of writing of different peoples are different; the variation in shape and size, thickness, or orientation of the digits; the degree of similarity between different digits like 3 and 8,1 and 7, 9 and 8, 5 and 6 etc.

In this work first some basic logic gates are implemented using discrete perceptron algorithm, then the approximation for predicting admission into a university based on different parameters using linear regression and gradient descent. Different supervised machine learning algorithms are implemented on three different handwritten digits dataset; MNIST English numerals, Arabic numerals and self-generated English numerals. Finally, an attempt to develop a simple and fast method based on logistic regression is presented. In order to do this a small handwritten dataset is generated.

1.5 Report structure

This report is structured into five chapters. Chapter 2 deals with the relevant background necessary on dissertation topic; basic steps in machine learning, how datasets are divided. Chapter 3 provide insight into various machine learning algorithms that are implemented in this work in context of handwritten character recognition. Chapter 4 shows the work done and results of this work; dataset generation, pre-processing, algorithms used and results. Finally, in Chapter 5, conclusion and discussion on the various difficulties and shortcomings of implemented algorithms along with the scope of improvement is given. At the end, the proper references and appendix is given for the various MATLAB codes that are used in writing the algorithms.

Chapter 2

Background

2.1 Classification of Machine Learning

Machine learning can be largely classified into three categories by the type of datasets that are used as experience namely supervised learning, unsupervised learning and reinforcement learning. In Supervised learning systems, the target values or labels are provided while training that is for each x in dataset there is a label y. The goal here is to obtain a deterministic function which defines the relation between input and output data-points which in-turn is utilized to predict the future data-points, this whole process is achieved by minimizing a cost/error function. In Unsupervised learning systems, the target values or labels are not provided or specifically speaking these systems applied on those areas which doesn't have labels to work with. The goal here is to analyse the unlabelled data for similarity (clustering) and grouping without the intervention of human beings. In Reinforcement learning systems, the machine finds the solution to a problem by hit and trial without any prior information. An external agent allocates a reward based system; depending upon the solution given by the machine, the agent either penalize or reward the machine. The objective here is to maximize the reward. This is analogous to giving hints regarding how to solve a problem rather than giving complete answers. These systems are feedback based systems. Another sub-category is semi-supervised machine learning; this is the in-between method in which some of the datasets are labelled and some are unlabelled, the division of datasets depends on the application area and availability of labels.

Two of the most important supervised domain of machine learning that are implemented in this work are regression and classification. In regression, the machine predicts the actual value rather than the label of the data. The machine analyses the datasets (x,y) where y is the actual value corresponding to x and attempt to predict the y for different x by fitting the pre-defined datasets. Some examples could be the prediction of stock prices, prediction of salaries, prediction of credit lines for customers etc. In classification, the machine attempts to predict the class or label of a data-point. This is mostly used in recognition problems. Handwriting and object recognition are the most common examples.

2.2 Artificial Neural Network

ANN are networks that are modeled after biological human brain. They are made up of multiple units called artificial neurons interconnected with each other. Figure 2.1 shows the basic structure of artificial neuron. It has multiple inputs and a weight associated with every input and a threshold term. If the output is above the specified threshold value, the neuron is activated. To simplify the calculations a bias term is introduced which is equal to the negative of threshold and is the weight associated with input having a value of one.

$$net = (\sum_{i=1}^{N} w_i x_i) \tag{1}$$

$$o = \varphi(net+b) \tag{2}$$

where x is input, w is weight, b is bias and equal to -θ and φ is called activation function which is applied to obtain different functionalities or to introduce non-linearity.

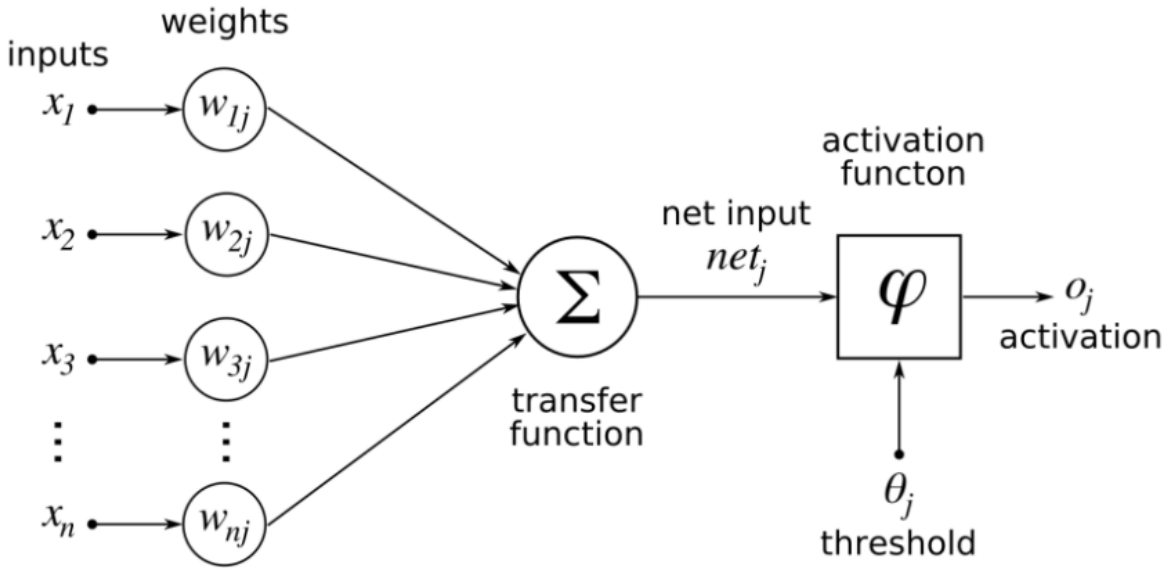


Fig2 1 Artificial Neuron

ANNs have multiple layers; each layer has multiple neuron units. Each layer is connected with next layer as shown in figure 2.2. The first layer is called input layer and the last layer is called output layer, the in-between layers are called hidden layers.

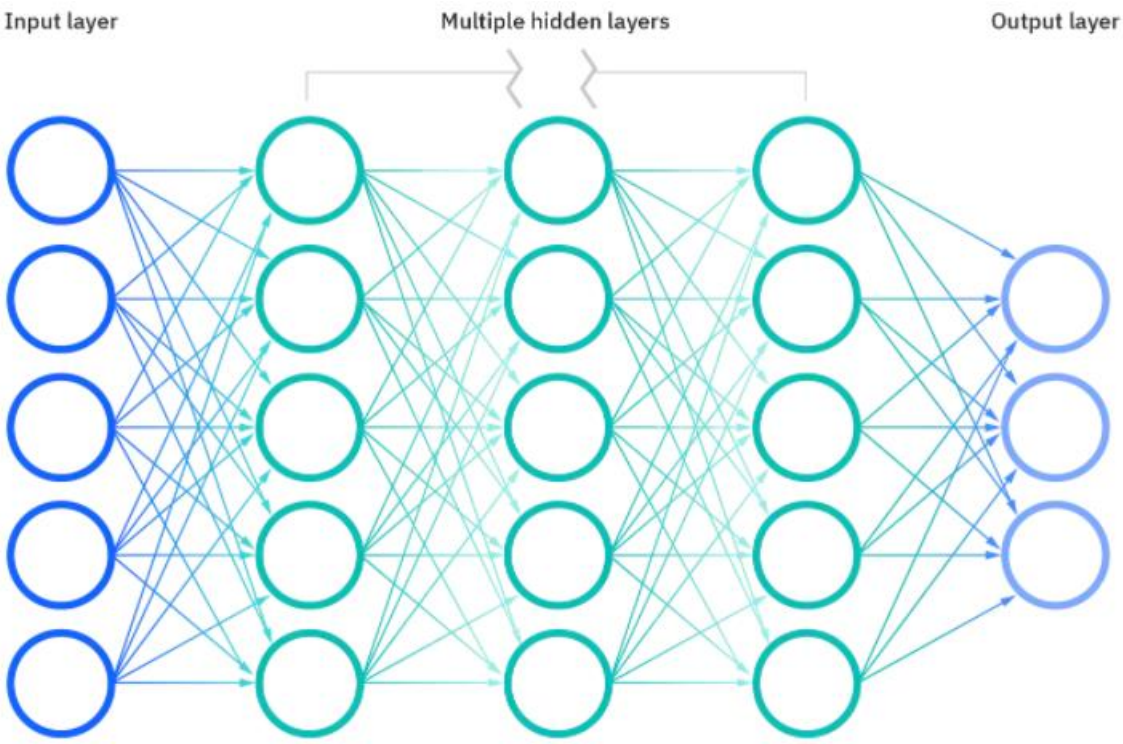


Fig 2.2 Multi-layered artificial neural network

2.3 Perceptron Architecture

The simplest type of neuron is called perceptron. The equation is given as-

$$y = sign(w^T x - \theta) \tag{3}$$

Here the activation function is simply a unit step function that is for positive net sum it will fire and for negative net sum the output becomes zero.

Perceptron's decision surface: Figure 2.3 depicts the decision surface in the two dimensional input space, dividing it into two classes, according to their label. This is an

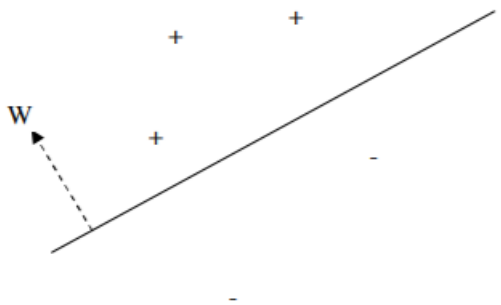


Fig.2.3 Decision plane in input space

example of a two class classification. This decision line can classify dataset into two linearly separable classes only. The weight vector is perpendicular to this decision surface.

2.4 Steps in Machine Learning

The framework of every machine learning system include major seven steps-

1. Data acquisition
2. Data pre-processing
3. Model selection
4. Model training
5. Validation
6. Hyper-parameter tuning
7. Prediction or testing

- **Data Acquisition:** This is the first step which includes the collection of raw data and converting it in ordered data. It also includes the generation of dataset from scratch or using standard datasets available on internet that will be used in the machine learning system. The quality and quantity of available data is one of the factors affecting the accuracy of the model. Quality means whether the data have sufficient features or not for a particular successful model training and prediction. Quantity implies that the data should contain sufficient samples for the incorporation of statistical advantage. Moreover,

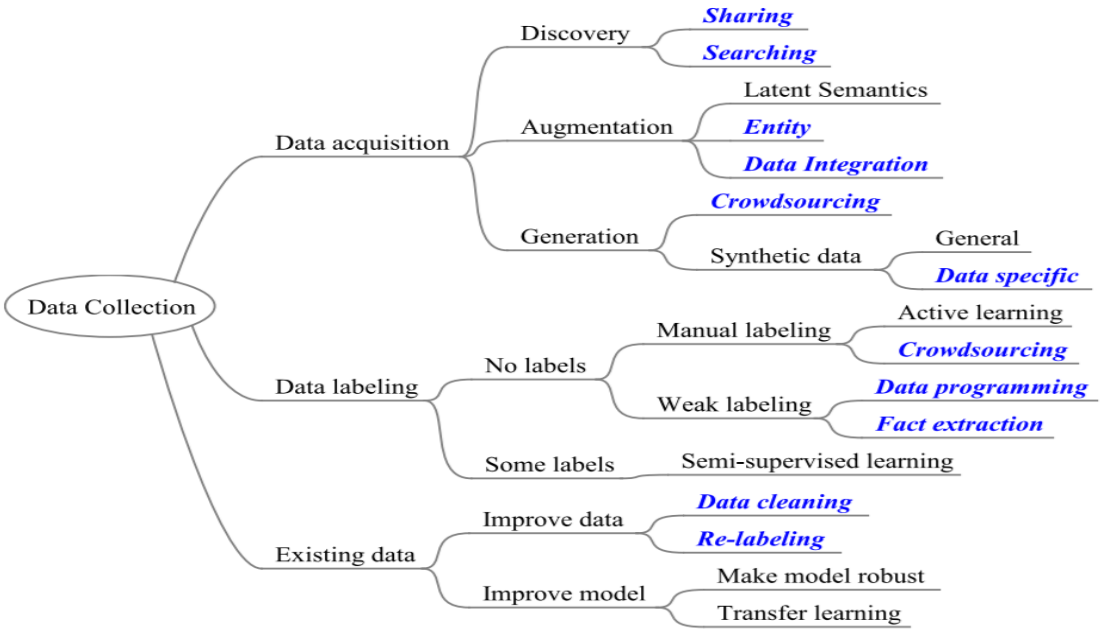


Fig.2.4 A high level research landscape of data collection for machine learning [15]

the data should be accurate and up to date for the successful application on real life problems.

Figure 2.4 summarizes all the type of work done in Data Collection step. In data acquisition three main approaches are followed; Discovery, Augmentation and Generation. Data discovery is the visual and explorative analysis to find trends and patterns in raw data. Data augmentation is the method by which an additional modified or new data is added to the existing dataset in order to improve its quality or quantity or any other attribute. Data generation is used when there is no data found on that particular domain or for the experimental purposes. Artificial data is constructed from the scratch by following data generation protocols.

- **Data pre-processing:** It is the fundamental and crucial step of machine learning that is implemented to improve the quality of data for better performance in terms of prediction accuracy, training time etc. Raw data cannot be feed directly to the machine for training and other purposes. Generally, it is not in the suitable form for machine and contains uncleaned data. Data pre-processing refers to the process of preparing, cleaning the data in order to make it usable for the models. Typically, the raw data contains noise, redundant information, missing values, different format than required. There are different methods for processing of data prior to use it the model, feature extraction techniques, interpolation techniques, normalization techniques, pruning and so on. Major five methods of pre-processing are summarized as follows-

    1. Data quality assessment
    2. Feature aggregation
    3. Feature sampling
    4. Dimensionality reduction
    5. Feature encoding

  Data quality assessment: Refers to gauging of data for the defined standards of quality of the features. This include assessment of data for consistency (whether the dataset is accurate and is generated using standard protocols or not), completeness (whether the data have any missing values or not), timeliness (the expected time for accessibility and availability of data), duplicate or corrupted values (whether the data contains redundant or unusable values or not).

Feature aggregation: It refers to the generation of new features by utilizing local features of data. This is done in order to reduce the memory consumption, processing time and complexity. As the local features are aggregated to form more generalized and stable features, the resulting features provide a higher level view of data.

Feature sampling: It is the method of downsizing the original dataset into a smaller data in terms of features or samples. The original dataset could be very large which results in large processing time and higher computation complexity. The down sampling is done considering that the original statistical or any characteristics must approximately remain the same that is it should be a good representation of original dataset.

Dimensionality reduction: It refers to the reduction of features of dataset. These new principle features are extracted from the original redundant features by combining or other set of defined rules. Dimension refers to the features of dataset; with increasing dimensionality complexity of dataset increases.

Feature Encoding: It is the method of transformation of categorical feature into the continues numerical feature. This is done in order to make the categorical feature usable in the ML model. Some of the encoding techniques are one-hot encoding, target-mean encoding, frequency encoding.

- **Model Selection:** It is the process of selecting the proper model for training and testing for the particular problem. Whether the regression or classification type model should be selected or not. In classification type which type of model is selected like MLN, CNN, SVM etc. Overall there are two general approaches for model selection:
    1. Probabilistic approach
    2. Resampling approach

  Probabilistic approach is the method of model selection using the probabilistic statistical measure taking in consideration of the score calculated by performance on training dataset and complexity of model.
    - Akaike Information Criterion (AIC).
    - Bayesian Information Criterion (BIC).
    - Minimum Description Length (MDL).
    - Structural Risk Minimization (SRM).

Resampling methods. employ the model selection method based on the performance of model on the out-of-sample data. Three common resampling model selection methods include:

- Random train/test splits.
- Cross-Validation (k-fold, LOOCV, etc.).
- Bootstrap.

- **Model training:** A machine learning algorithm, also called model, is a mathematical expression that represents data in the context of a problem. In context of supervised learning, a training dataset is utilized in model training and output is calculated. The correlation of this output is checked with the actual or desired output by using a loss function. By using a training algorithm, this loss function is minimized. This whole process is called model training.

- **Validation:** After model training, the model is tested on a validation set of data in order to test the generalization ability. The validation is performed to gauge the performance of the model on the new dataset and modify the hyper-parameters accordingly. This validation set is separate portion of same dataset of which training set is part of. Some of the validation techniques are-

    1. Hold-out
    2. K-fold cross-validation
    3. Leave-one-out-cross-validation (LOOCV)
    4. Random subsampling
    5. Bootstrapping

  Hold-out: The dataset is partitioned into training set and testing set. The model is trained on training set and tested for performance or accuracy on the testing set. The partition typically used are 60-40, 70-30, 80-20, 90-10 percent of training-testing respectively. In order to avoid the uneven distribution of classes of data in training and testing sets a technique called stratification is applied on whole dataset.

  K-fold cross-validation: In this technique, the dataset is partitioned into k subsets of equal or nearly equal length. For each run, k-1 subsets are used for training and one subset for testing and the performance is recorded. The runs are arranged in such a way that each subset has an opportunity for performing as testing set.

Leave-One-Out Cross-Validation (LOOCV): It is the most extreme case of k-fold cross validation, in which k is taken as number of samples of dataset. It is computationally expensive and should be used on small datasets.

Random Subsampling: Multiple sub sets of datasets are selected to be used as testing sets and the remaining are used as training set.

Bootstrapping: In this technique, the training dataset is randomly selected with replacement. The remaining examples that were not selected for training are used for testing. Unlike K-fold cross-validation, the value is likely to change from fold-to-fold. The error rate of the model is average of the error rate of each iteration.

- **Hyper-parameter tuning:** The parameters through which the architecture of model is defined are called hyper-parameters. These parameters do not affect the output of model directly. Hyper-parameters, in contrast to model parameters, are set by the machine learning engineer before training. There is no concrete standard formula to set these hyper-parameters which makes it the most difficult problem of machine learning. These are set by hit and trial, intuition or experience of the ML scientists or engineers. Some examples of model hyper-parameters include: The learning rate for training a neural network, the **C** and $\boldsymbol{\gamma}$ hyper-parameters for support vector machines, the **k** in k-nearest neighbours. The hyper-parameter tuning refers to the process of selection of model parameters so as to maximize the performance of model. Some of the common strategies for optimizing hyper-parameters are:

    1. Manual hyper-parameter tuning
    2. Grid search
    3. Random search
    4. Bayesian optimization
    5. Gradient-based optimization
    6. Evolutionary optimization

    Manual hyper-parameter tuning: The use of hit and trial is utilized in this method. The more experienced engineers can guess and set intuitively which values give the better performance.

    Grid search: The model is trained on each possible value of hyper-parameter and the performance is calculated. The value giving the best performance is selected.

Random Search: Instead of each and every value, a random set of values are chosen and model is evaluated. This method is useful when the values are very large to apply grid-search. One can choose those values which have more probability for better performance to minimize time consumption.

Bayesian optimization: This method keeps track of past performances of the model and build a Bayesian probabilistic model to evaluate the probability of score for a particular hyper-parameter. The more the model is evaluated the more probably this algorithm chooses the good hyper-parameter. In a nutshell, this method builds a probability model based on objective function, find best hyper-parameter on probability model, test those hyper-parameters on objective function and then update the probability model based on performance.

Gradient-based optimization is specially used in the case of Neural Networks. It computes the gradient with respect to hyper-parameters and optimizes them using the gradient descent algorithm.

- **Prediction or testing:** It refers to the actual implementation of model which has undergone the training and validation tests. Now the model is applied on the actual problem of unseen dataset and predict the outcome whether it is classification or regression problem.

2.5 Partition of dataset

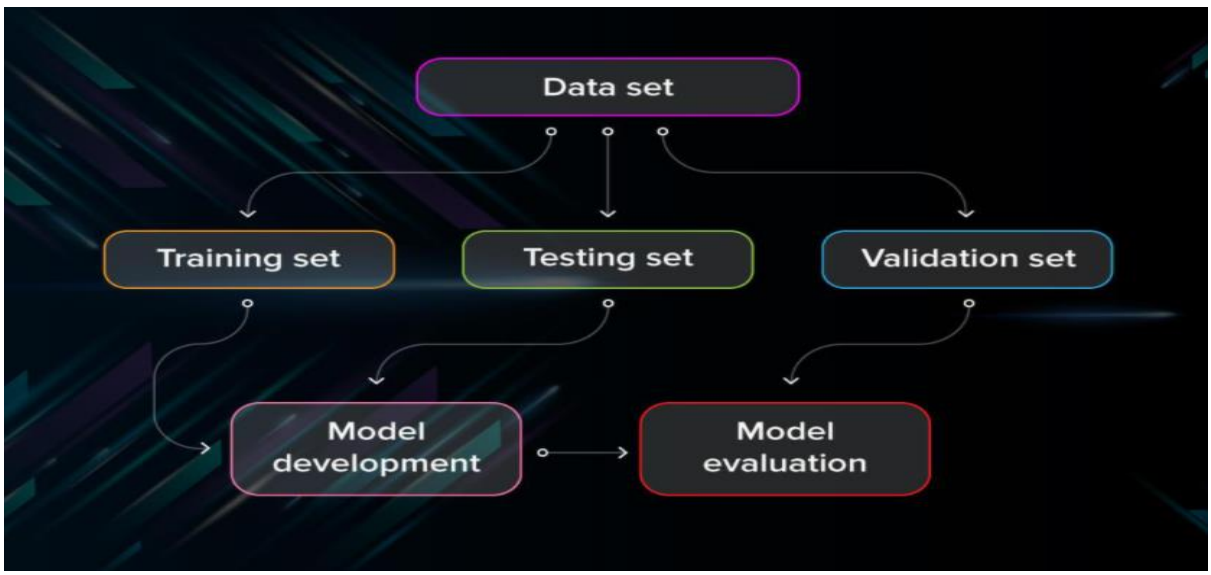


Fig.2.5 Partition of dataset for Machine Learning

To increase the quality of ML model the dataset is divided into three independent non-overlapping subsets; Training set, Validation set, Testing set as shown in figure 2.5.

- **Training set**

  This is subset is used to train the model by using some training algorithm. Majority of dataset is partitioned into this subset.
- **Validation set**

  This subset is used to tune the hyper-parameters of the model to maximize the model performance and in order to avoid the overfitting of model on training set. Overfitting can result in bad generalization of model resulting in bad performance on the unseen test dataset.
- **Testing set**

  This subset is used for final model evaluation. It is the assessment of generalization error on the final chosen model. The test set should be large enough to provide statistically meaningful results and be representative of the data set as a whole.

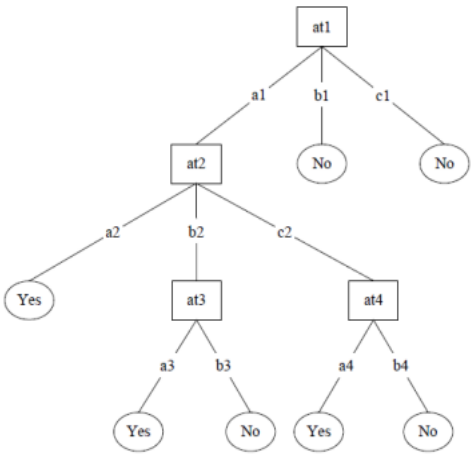
Chapter 3

Machine Learning Algorithms

In this chapter, the basic introduction of four ML algorithm is summarized: Decision Trees, Gradient Descent and Backpropagation, Support Vector Machines and k-Nearest-Neighbour. Neural network is another popular algorithm and is interesting in particular, as it out-performs other algorithms in classification domain and is discussed in previous chapter. Although the training algorithm primarily used in ANN (Gradient descent and Backpropagation) is discussed here. The theory behind each algorithm is explained.

3.1 Decision Trees

They are the flowchart resembling structures which can either be used for classification as well as regression. A decision tree is illustrated in Figure 3.1. Its primary building blocks are called nodes; the topmost node(at1) is the starting point known as root node which has no incoming edges. The feature that has the capability of splitting the dataset most broadly is chosen as root node. The overall function of a decision tree summarizes as the recursive partition of dataset. All the remaining in-between nodes that have an outgoing edge are known as internal or test nodes (at2, at3, at4). The nodes without any outgoing edges are known as leaf nodes. At each internal node the incoming instance space is partitioned into two or more sub-spaces according to certain discrete function of input attributes that is the



at1	at2	at3	at4	Class
a1	a2	a3	a4	Yes
a1	a2	a3	b4	Yes
a1	b2	a3	a4	Yes
a1	b2	b3	b4	No
a1	c2	a3	a4	Yes
a1	c2	a3	b4	No
b1	b2	b3	b4	No
c1	b2	b3	b4	No

Fig 3. 1 Decision tree for the training set with classes Yes and No. The instance [at1=a1, at2=b2, at3=a3, at4=a4] sorts to the nodes at1, at2 and at3, classifying the instance as positive represented by the value "Yes" [16].

decisions are taken at internal nodes. Each leaf node is assigned to a class representing the most appropriate target. There are several techniques that are implemented for sorting and selection of features into nodes such as Information Gain, Gini Index, or ReliefF Algorithm.

A data compression technique called pruning is utilized in decision trees to remove the sections or nodes that are redundant and do not provide critical or useful to classify instances. It also reduces the overfitting and complexity of model and hence improve the performance of model.

Algorithm:

1. Initiate root node containing the complete dataset.
2. Find the best attribute according to the selection metric using Attribute Selection Measure.
3. Divide the dataset into subsets containing the possible values of best attributes
4. Split the root node into decision nodes containing the best attribute.
5. Repeat step 3 for the existing subsets.
6. Repeat step 2-5 until further classification is not possible.

3.2 Support Vector Machines

Support Vector Machines (SVM) is a discriminative classification technique. The object here is to differentiate the classes with an optimally placed decision surface. One of the methods to optimize the decision surface (hyperplane) is to maximize the margin between the two classes. An illustration of this technique is given in figure 3.2.

By finding the hyperplane with maximum margins the tolerance for noise increases and the generalization ability of the model increases. This results in the best classification performance for both the training data as well as future data. In order to find maximum margin hyper-planes, SVM aims to maximize function in relation to $\vec{w}$ and b:

$$L_P = \frac{1}{2}\left|\left|\vec{w}\right|\right| - \sum_{i=1}^{t} \alpha_i \gamma_i (\vec{w}_i . \vec{x}_i + b) + \sum_{i=1}^{t} \alpha_i \quad (4)$$

Here, t represents training point quantity, α_i stands for Lagrangian multipliers. Vector $\vec{w}$ and

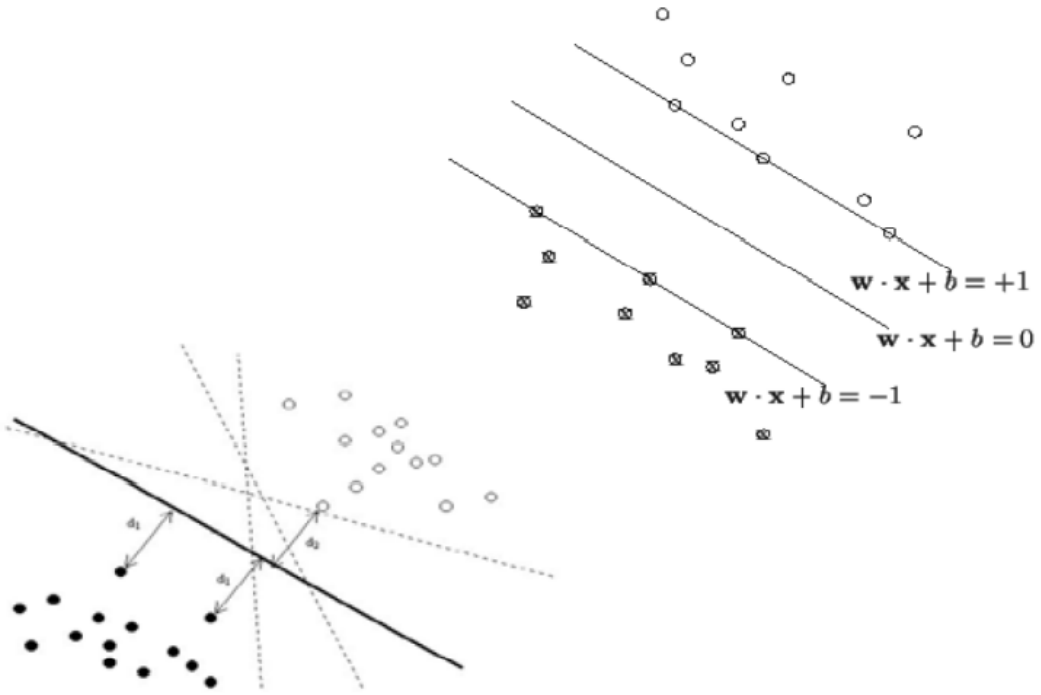


Fig 3. 2 Creating the optimal separating hyper-plane with the use of support vectors.

constant b characterize the hyperplane. The data points that lie on the hyperplane are called support vectors. For the solution, all the remaining data points are irrelevant since only the support vectors and their linear combinations are utilized. This implies that SVMs don't depend on the features of dataset and hence are suitable candidates for problems where features are large in number with smaller samples. This algorithm does not work directly for non-linear datasets and unfortunately typical real world problems are nonlinear nature, therefore no hyper-plane could be found for correct classification. The solution for this problem is transformation of feature space into another higher dimensional space, often some mathematical combination of original features. These transformations are called kernels and could be computationally expensive for certain problems.

3.3 k-Nearest neighbor

The fundamental premise of this algorithm is that similar things exist in close proximity. That is, it classifies into classes based on some distance based metric typically Euclidian distance. K-NN calculates the similarity between incoming or test data with existing or available data and classify into that category which has the maximum degree of similarity with available data. Three crucial components of algorithm are: labelled objects; a metric used for

calculating proximity; and k, the number of nearest neighbours [24]. A typical proximity metric is Euclidian distance given by the formula:

$$D(x, y) = (\sum_{i=1}^{m} |x_i - y_i|^2)^{1/2} \quad (5)$$

When k nearest neighbours are obtained then the test data is classified into that class which has the majority points in nearest neighbours:

$$Majority\ Voting: Y' = argmax \sum_{(x_i, y_i) \in D_Z} I(v = y_i) \quad (6)$$

Here, v represents the class label, yi represents the ith nearest neighbor class label, and I(·) is the indicator functions that returns value one for a valid argument or zero for an invalid argument.

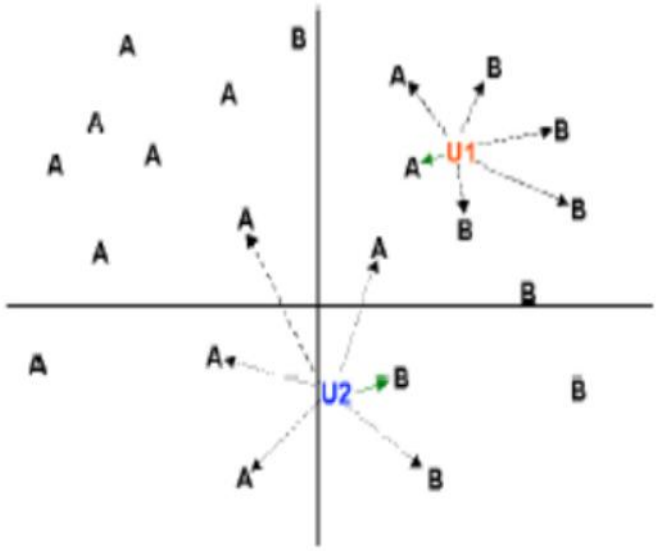


Fig 3. 3Representation of a k-Nearest-Neighbour graph.

It is easy to understand and implement. Despite being simple the algorithm still performs well for many cases, especially for multi-modal classes. However, there are critical three problems: it is data storage expensive in nature as it stores all the information of data samples, the optimal choice of k is difficult to choose and do not has concrete standard methods and finally it is very sensitive to the proximity metric chosen. Hence this algorithm is not suitable for problems having very large data samples.

Algorithm:

1. Select number of neighbours K and a proximity metric.
2. Calculate proximity of incoming query sample from all the training samples.
3. Select K closest training samples to query sample.
4. Parse the K nearest neighbours into separate classes.
5. Find the class which has maximum number of neighbours.
6. Classify the query point into the previous selected class.
7. Repeat step 2-6 for all query samples

If this method is applied on big data it will be extremely slow process even for MNIST dataset. If the full training and testing sets are utilized, for each query sample 60,000 proximity metric values are calculated and this process is repeated 10,000 times. Although there are various techniques to improve the time and calculations but still this method is used for smaller datasets.

3.4 Backpropagation

This is primarily used in artificial neural networks. It is an efficient and fast method to calculate the gradients in neural network with respect to cost function. This method makes

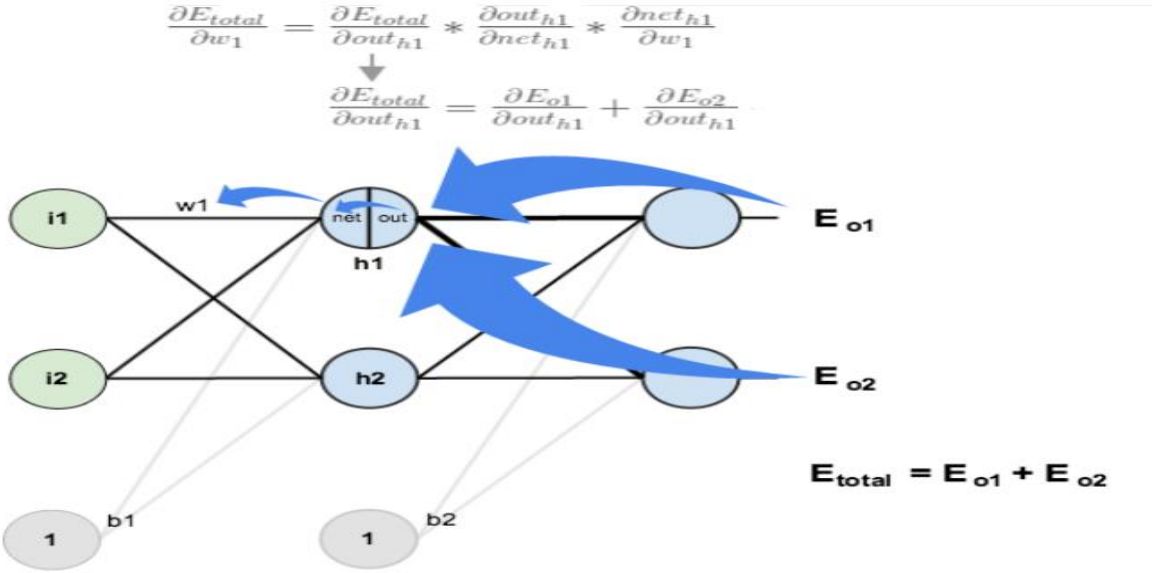


Fig 3.4 Backpropagation in multi-layered neural networks

use of chain rule of partial derivatives. The derivatives are calculated in backward manner starting from the cost function and ending at the parameter of gradient is calculated also known as backward pass as shown in figure 3.4.

$$\frac{\partial E_{o1}}{\partial w_1} = \frac{\partial E_{o1}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1} \quad (7)$$

In order to calculate partial derivative of E with respect to w_1, we have to propagate backward while simultaneously calculating derivatives for each immediate step and combine them by utilizing chain rule of derivatives. Equation 7 gives the proper representation in context of figure 3.4. Before using this algorithm, the learning of neural networks was very difficult and tedious process.

3.5 Gradient Descent

It is the method of optimization of a function which utilizes the concept of gradient. Gradient is the vector form of derivative in the direction of greatest increase of the function. Hence the optimization is done by moving in the direction of negative of the gradient. In machine learning, gradient descent is used to update the parameters of the model. The weight update equation is give as:

$$W^{K+1} = W^K - \eta \nabla \mathrm{E(W)} \quad (8)$$

Where η is learning rate and ∇E(W) represents the gradient and K is the number of iteration. In neural networks it is given as-

$$\nabla \mathrm{E(W)} = \begin{bmatrix} \partial E/\partial w_1 \\ \partial E/\partial w_2 \\ \vdots \\ \partial E/\partial w_n \end{bmatrix} \quad (9)$$

Where w_1, w_2 and w_n are the weight parameters associated with the input of each neuron unit. For square error the gradient is given as:

$$\nabla \vec{E} = [y_d - f(h(x))] f' . \vec{X} \quad (10)$$

Now for linear hypothesis the matrix form of equation (10) is given as:

$$W_{new} = W_{old} - \frac{\eta}{m} [X(H - Y_d)] \quad (11)$$

<u>Algorithm</u>:

1. Initialize the weights.

2. Calculate output and error.
3. Calculate gradient of error with respect to weights.
4. Update the weights by using equation 8.
5. Replace the old weights with new weights.
6. Repeat step 2-5 until the error is minimized.

3.6 Perceptron Pocket Learning Algorithm

The pocket algorithm is same as PLA but with minor update, instead of keeping the weight changed after every modification the best weight is kept in the pocket (weight giving minimum misclassification). For the sake of convenience, pocket algorithm will be referred as PLA in rest of the report.

Algorithm:

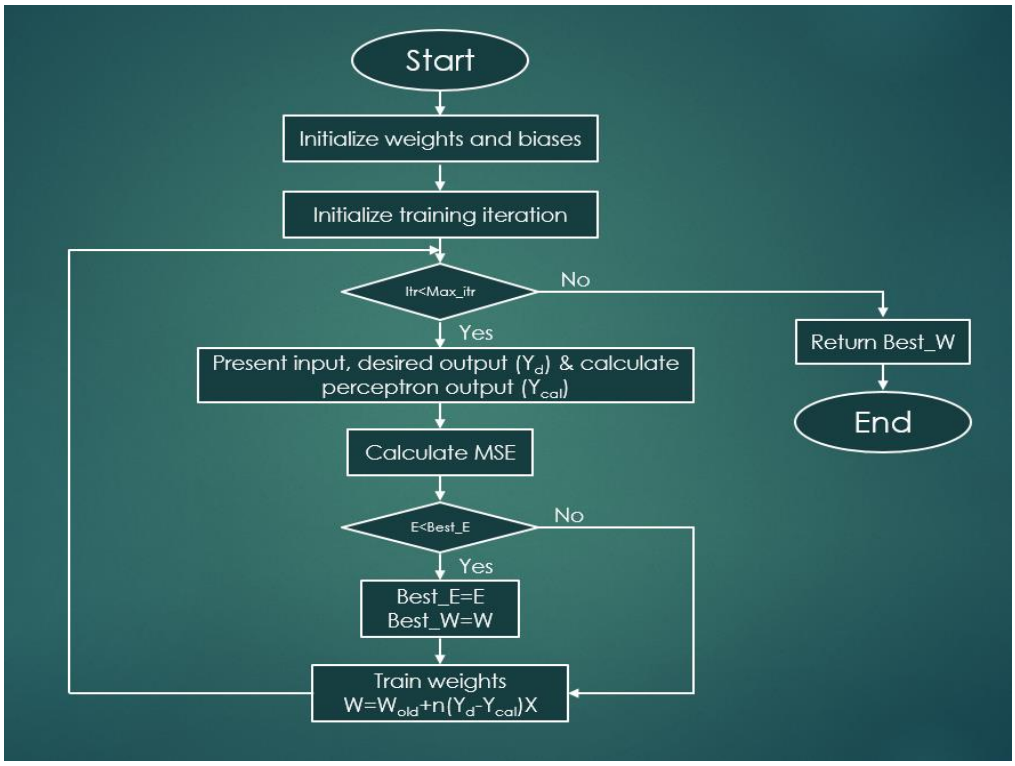


Fig 3.5 Flowchart of Pocket PLA

1. Assume some initial weights W (preferably random generation)
2. Pick an input data sample X

3. Find the output $y=sign(W^TX)$
4. Find error (e)
5. Modify the weights

$$\Delta W=eX$$

$$W_{new}=W_{old}+\Delta W$$

6. If error is minimum, assign $W_{pocket}=W$ and if not then skip
7. Complete steps 2 to 6 for all data samples
8. Repeat steps 2 to 7 till the error is minimized

A more specific flowchart is illustrated in figure 3.5.

3.7 Linear Regression

The basic notion of Linear Regression is that it try to approximate a linear hypothesis that best describes the sample data. The variables that are utilized to obtain the hypothesis are called explanatory variables or features and the variables at which approximation is performed are called dependent variables or simply output. In linear regression the output or hypothesis is given by:

$$h(x) = \sum_{i=1}^{d} w_i x_i = \boldsymbol{W^T X} = \boldsymbol{XW} \quad (12)$$

Where **W** is weights associated with input **X**. Square error is given by:

$$E = (h(x) - f(x))^2 \quad (13)$$

Where f(x) is the target function that has to be approximated. Rewriting equation (6) in matrix form:

$$\boldsymbol{E} = \frac{1}{d}||\boldsymbol{XW} - \boldsymbol{Y}||^2 \quad (14)$$

Where "d" is the dimension of input matrix. Minimizing above equation gives:

$$\boldsymbol{X^\dagger} = (\boldsymbol{X^T}.\boldsymbol{X})^{-1}.\boldsymbol{X^T} \quad (15)$$

Where $X^\dagger$ is known as pseudo-inverse of X. now the new weights are given as:

$$\boldsymbol{W} = \boldsymbol{X^\dagger}.\boldsymbol{Y} \quad (16)$$

Where X_{test} are the new data-points, $X^{\dagger}_{train}$ is the pseudo-inverse calculated from training dataset and Y_{target} is the desired output of training dataset.

Algorithm:

1. Select input and target output values
2. Calculate pseudo-inverse matrix of input
3. Calculate weight matrix
4. End

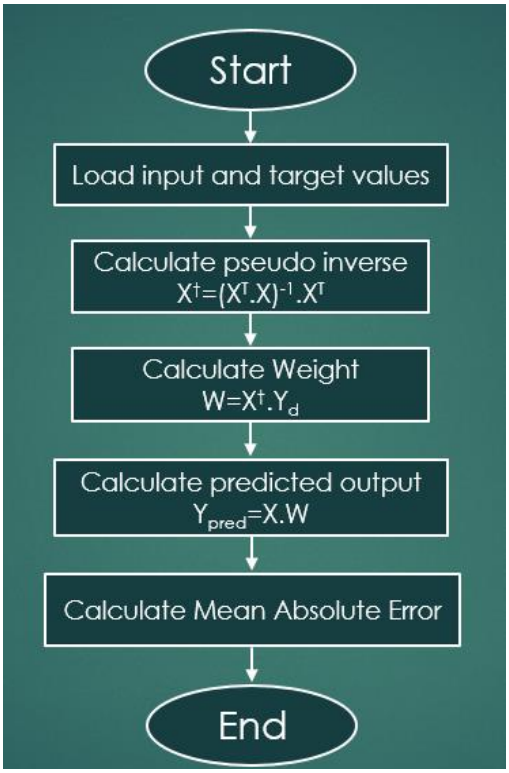


Fig 3.6 Flowchart for one-step linear regression algorithm utilized for obtaining results

3.8 Naïve Bayes

This is the simple yet a useful machine learning algorithm that utilizes the concept of Bayes' theorem. First and foremost concept in this algorithm is Bayes' theorem which is given as-

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{n} P(B|A_j)P(A_j)} \quad (17)$$

Where $P(A_i|B)$ is the conditional probability of an event A that is chance of event A given that B has happened, $P(B|A_i)$ is the conditional probability of event B, $P(A_i)$ is the probability of event A. All these events are mutually exclusive which implies that-

1. $A_i \cap A_j = \emptyset$ for i ≠ j
2. $A_1 \cup A_2 \cup ... \cup A_n = S$
3. $A_j = 0$

Here S is the total sample space. In machine learning this theorem can be utilized to calculate posterior probability of the model's hypothesis as-

$$P(Hypothesis|Data) = \frac{P(Data|Hypothesis)P(Hypothesis)}{P(Data)} \quad (18)$$

Where $P(Data|Hypothesis)$ is known as likelihood of the training data given the hypothesis, $P(Hypothesis)$ is the priori probability of hypothesis and $P(Data)$ is the probability of observed data used without the constraint of hypothesis. For Naive Bayes classifier algorithm equation 18 is modified to simplify the complexity due to the nature of size of datasets used in machine learning. A simplified version is-

$$P(y_j|X_i) = \frac{P(X_i|y_j)P(y_j)}{P(X_i)} \quad (19)$$

Assuming that the individual X_i is independent from each other but this is not a good assumption because the real life datasets does not follow this trend, equation 19 can be modified into-

$$P(y_j|X) = \frac{\prod_{i=1}^{f} P(X_i|y_j)y_j}{P(X)} \quad (20)$$

Where f is the number of features data X contains. Since P(X) does not depend on the class of data it will be constant throughout and hence is removed from the equation. because only the degree of probability of data is required. For continuous data attributes, instead of using frequency of data samples attributes are assumed to follow specific standard statistical distribution like Gaussian distribution. The likelihood now is calculated by using the

probability distribution function. For normal distribution the likelihood of data x is calculated by-

$$P(X|y) = \prod_{i=1}^{f} \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\left(-\frac{1}{2}\frac{(x_i-\mu_i)^2}{\sigma_i^2}\right) \quad (21)$$

Where y is the class of data, σ and μ are standard deviation and mean of data of respective class. Using above value for likelihood probability in equation 20 gives the probability for the hypothesis given data X. The data is classified into that class which gives the maximum probability by equation 20.

Algorithm:

1. Load the training dataset and separate it by class.
2. Calculate the statistical parameters mean and standard deviation for very feature in each class.
3. Calculate the likelihood for every class by equation 21.
4. Calculate the posterior probability of a class by equation 20.
5. Repeat step 3-4 for every testing samples.
6. Classify sample into greatest posterior probability.

3.9 Non-Backpropagation based Multi-class Classifier using Logistic Regression

Logistic Regression is the simplest non-linear binary classification method in machine learning. It utilizes a neuron with an activation function of sigmoid at its output. For training and learning Gradient Descent is used. Since it is a convex optimization problem, GD will always find the global minimum given enough iterations. The output of LG is given as-

$$y = sigmoid(\vec{\boldsymbol{W}}\vec{\boldsymbol{X}}) \quad (22)$$

Where $\vec{\boldsymbol{W}} = [w_0 \quad w_1 \quad \cdots w_f]$ is the weight vector associated with every input and $\vec{\boldsymbol{X}} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_f \end{bmatrix}$ is the input vector and f is the number of features of input. Sigmoid function is given as-

$$sigmoid(x) = \frac{1}{1+\exp(-x)} \quad (23)$$

The loss function is given by-

$$E = -\frac{1}{N}\sum_{i=1}^{N}[t_i \log(y_i) + (1 - t_i)\log(1 - y_i)] \quad (24)$$

Where N is total number of samples, t is the actual target (1 or 0), and y is predicted probability. This loss function is known as Binary Cross-Entropy Loss.

Network Algorithm:

1. Construct One Vs Rest binary classifiers for every digit.
2. Construct binary classifiers of pairs of poorly classified digits by step 1.
3. Construct layer 1 by appending the neuron units by step 1-2.
4. Construct output layer of 10 One vs Rest binary classifiers by utilizing the output of layer 1 as input.

Classifier Algorithm:

1. Initialize weights and biases; select learning rate and number of iterations
2. Calculate output y by equation 22 and error by equation 24
3. Calculate $\partial E/\partial \overrightarrow{\boldsymbol{W}}$
4. Update weights by equation 8
5. Repeat step 2-4 until the error is minimized or number of iterations are completed.

One Vs Rest Approach:

It is the simple method to tackle multi-class classification problem by using multiple binary classifiers. Each binary classifier has a target output of one for a particular class and a target value of zero for the remaining classes. In the handwritten 10-Class classifiers, ten such binary classifiers are required. The prediction is calculated based on which classifier is producing maximum probability (in logistic regression case), the class associated with that classifier is assumed to be the predicted class.

Chapter 4

Results and Discussion

In this chapter, different machine learning algorithms are implemented on four databases. Various logic gates are implemented using PLA, prediction of student for admission into a university is also presented in the form of best fit curve. Then the generation of handwritten dataset is given, what are the features that are used for classification purpose and finally the classification results of handwritten digits are presented.

4.1 Logic gates implementation by PLA

The illustrated plots are the implementations of basic logic gates by PLA. These are the plots of decision boundaries in input plane. For the sake of illustration only two-input logic gates are taken but this concept can be extended to n-input logic gates that is the implementation remains the same as two-input gates. In figure 4.2, the illustrated plots show decision boundaries for every weight modification that occurs whilst training. The final hypothesis or decision boundary is shown in bold black line. The points that are crossed belongs to class one points (for which the desired output is one) and the not-crossed points belong to class zero (desired output is zero). Since these problems are linearly classifiable, only one perceptron is required for the implementation. As observed from the plots all the class one points are on one side of final decision boundary and class zero points on the other, this implies that the model is working correctly and is able to classify with zero errors. Figure 4.2(a) depicts the implementation of OR logic gate, 4.2(b), (c) depicts AND and NAND gates respectively. In figure 4.2 (e) the classification is not possible with single perceptron because the XNOR logic is not linearly separable (minimum two perceptron is required for this problem).

4.2 Datasets

There are four datasets that are utilized in this work; (1) Two Standard datasets that are downloaded from the KAGGLE website [17, 18] (2) manual generation of dataset of handwritten digits of 35 samples (3) MNIST handwritten digits dataset.

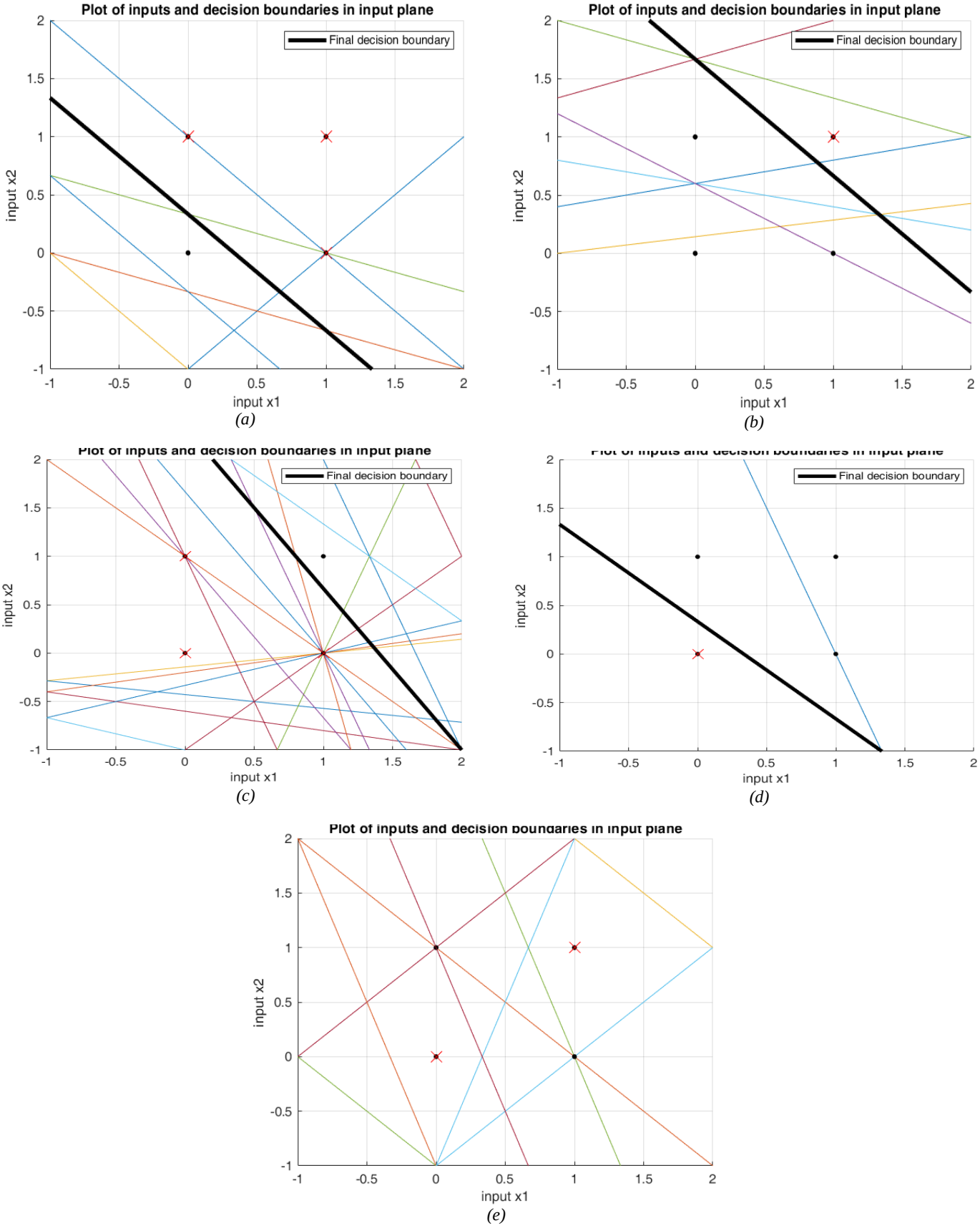


Fig 4.1 PLA implementation of logic gates (a) OR, (b) AND, (c) NAND, (d) NOR logics. (e) XOR implementation showing drawback of single perceptron (not classifiable)

1. The dataset from KRAGGLE is the data of 400 samples giving the chance of admit as output in a university for students. There are seven different input features: GRE score, TOEFL score, university rating out of five, quality of SOP and LOR, CGPA of applicant, is there any research work or not (one for research work done and zero for not any work done).
2. The handwritten digits dataset is generated by hand by asking nine different people to write the digits from zero to nine on a graph paper. The horizontal and vertical densities of each digit is calculated (that is the count of smaller squares in which the ink is present). A matrix of each digit is prepared separately where the rows depicts number of sample and the columns depict the respective densities (first eight columns are for horizontal density and next eight columns for vertical density). Therefore, data of each digit is a matrix of 35 by 16.
3. Two standard datasets that are mainly utilized for benchmarking the machine learning algorithms namely MNIST handwritten digits and a similar dataset for Arabic numerals having same features and size are used. These datasets have dimensions of 28-by-28 pixels, a size of 60,000 training samples and 10,000 testing samples. The raw pixel values are represented in grey scale values ranging from 0-255 and are utilized as features in the ML model. These pixel values are flattened into vectors of size 784 (28*28) for each sample digit. Hence the training dataset now is a matrix of size 784-by-60,000 and testing dataset of 784-by-10,000.

4.3 Linear curve approximation by Pseudo-inverse

Those data samples which can be approximated by linear functions, this algorithm is fast and efficient since it is not an iterative process unlike PLA and gradient descent (discussed in next section) rather it is one-step learning processing. This algorithm is applied on the dataset of chance of admit of students into a university. For the sake of illustration only one parameter is used at a time. Figure 4.4. illustrates the approximation for three important attributes utilized in the criterion for selection of candidate into a particular university. The points depict the target chance of admit on y-axis and corresponding feature used on x-axis and the line depicts the predicted values also known as approximation. The major caveat of using this technique is the condition for the pseudo-inverse to exist (i.e. X^TX must be invertible) which is not always the case for practical datasets. That is the pseudo-inverse

equation is numerically unstable. The mentioned problem occurs if the features used are redundant in nature, have zero variance etc. This is overcome by utilizing feature filtering

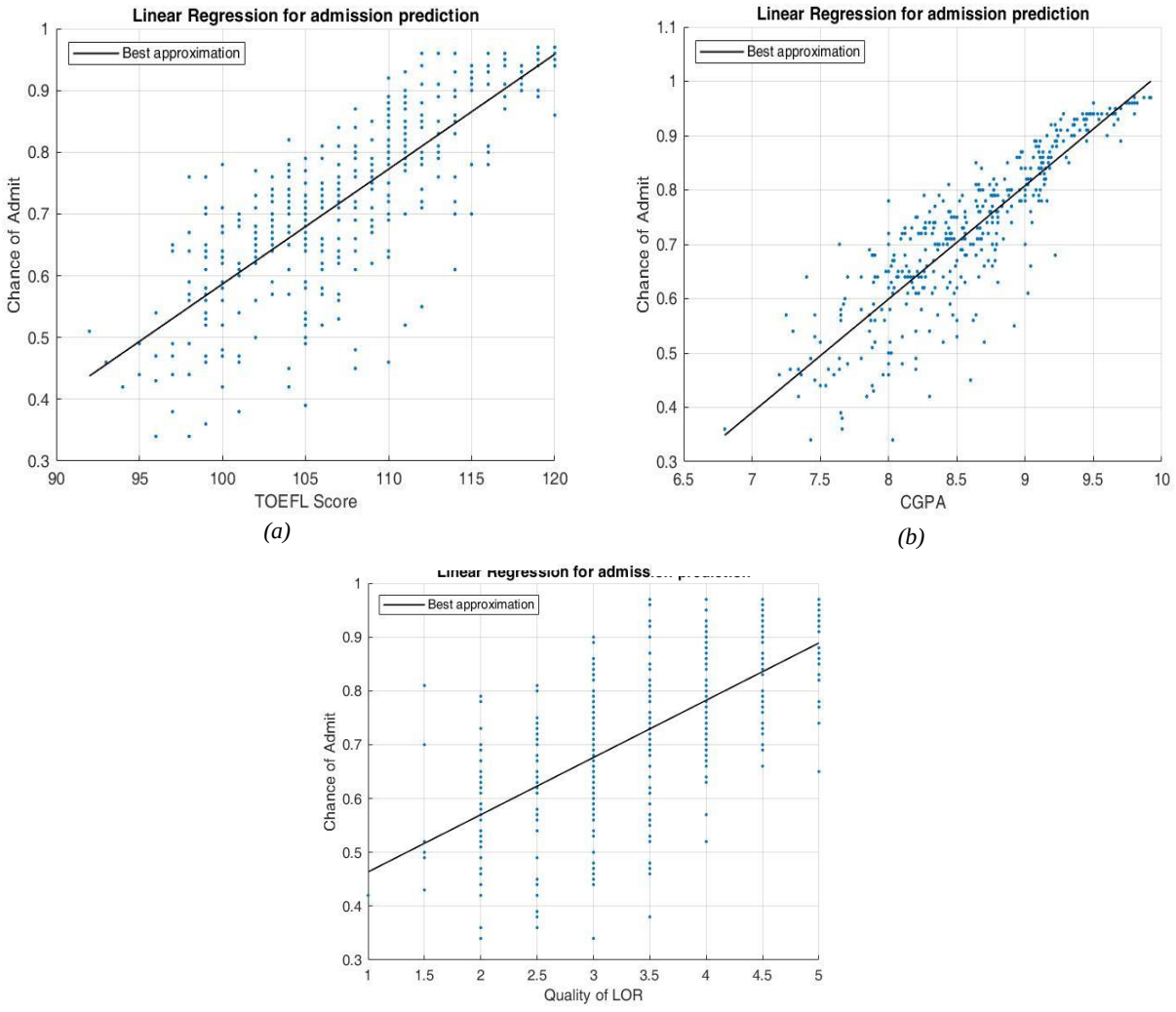


Fig 4.2 Best fit plot by using regression Giving mean absolute error of (a) 0.0610, (b) 0.0835, (c) 0.0653

methods like removal of features containing no or very little information. Another disadvantage of this approach is that the amount of approximation or error cannot be controlled. But this algorithm is easy to implement.

4.4 Linear curve approximation by Gradient Descent

As the number of iterations is increased the quality of approximated curve is improved upto certain extent. Also the generalization error capability of this method is good. The error from applying gradient descent is slightly lower than one-step learning. But this gap is increased as the number of iterations are decrease. For convex optimization problem like for this dataset, this is not the suitable approach for learning. Because this method never reaches global

minimum and there is one extra hyper-parameter to control. There are faster learning algorithms than gradient descent for this type of application area.

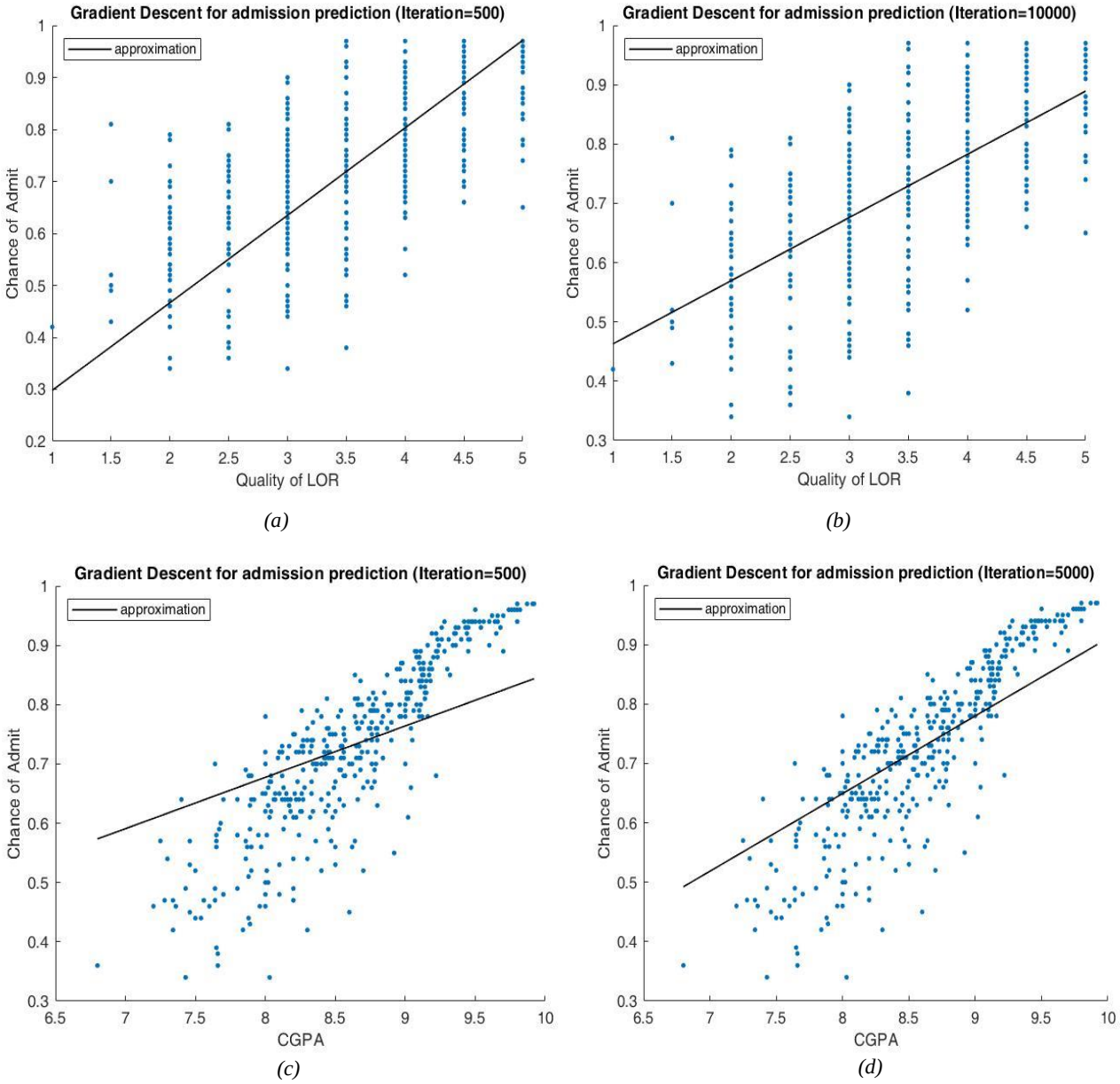
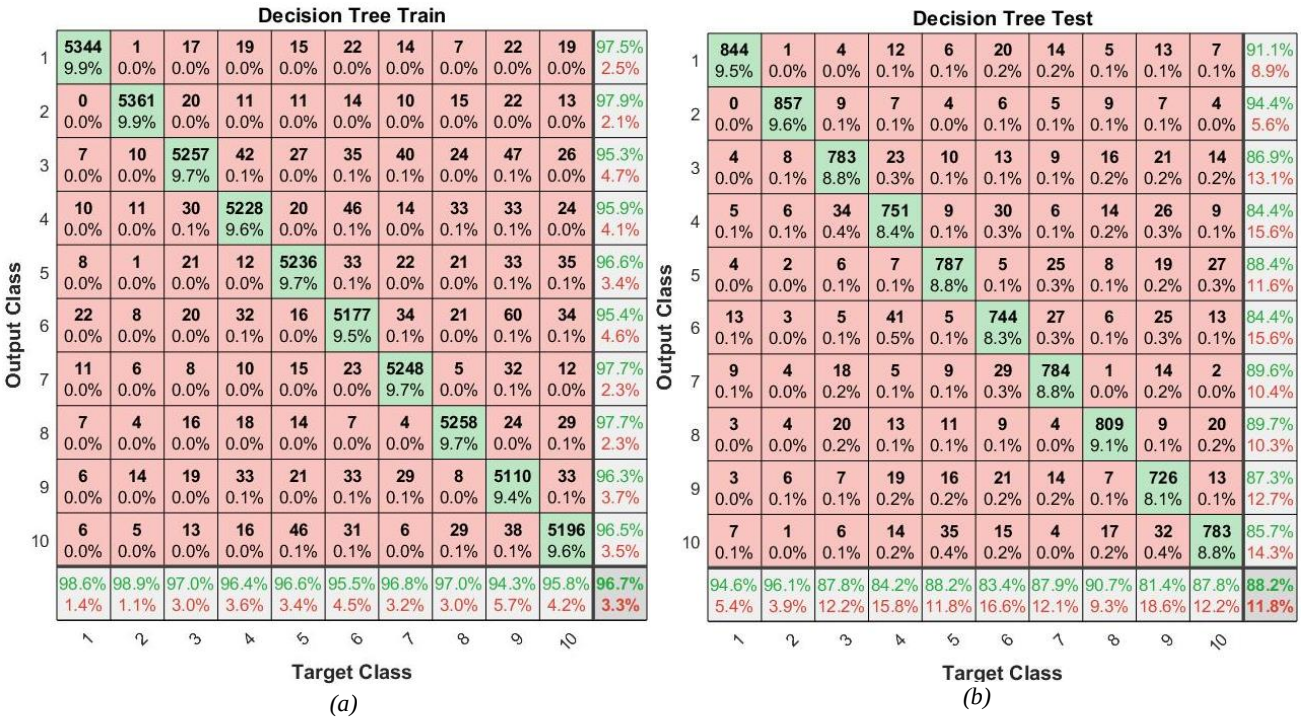


Fig 4.3 Best fit curve plot using gradient descent. Given absolute error of (a) 0.0800, (b)0.0735, (c) 0.0936, (d) 0.0510

4.5 MNIST 10-class classification

Decision Tree:

This method can be utilized for categorical as well as continuous variable classification. For the sake of easier implementation and lesser complexity, only binary decision tree is trained and tested. These trees are deeper and faster than non-binary decision trees.



(a) *(b)*

Fig 4.4 Confusion matrix by decision tree for (a) Training dataset, (b) Testing Dataset

The input training subset for decision tree is constructed by taking 5421 samples from every class of MNIST training set and similarly testing subset is constructed by taking 892 samples. The splitting criterion used in this decision tree is Gini-Diversity index (GDI). It is the quantification parameter for degree of classification accuracy of a variable when chosen randomly. Since the dataset used have large sample and feature size, the obtained decision tree is very deep and hence cannot be visually represented in this report. The total number of splitting criterion applied while training is 4353 which results in 4353 total number of nodes in this decision tree. As observed from the figure 4.6, training accuracy is higher than testing accuracy which is 96.7% and 88.2% respectively. The most difficult or the poorest classification is of class 9 (digit 8) in training (94.3%) as well as in testing (81.4%). The easiest or the best classification is of class 2 (digit 1); 98.9% accuracy for training and 96.1% for testing. These results are well aligned with the actual observation and expectations. Since the features used in this model are grey values of pixels for each digit, the digit eight is the most complex of all the digits and have highest correlation with a number of other digits like 5,2,9 etc. Whereas the digit 1 is the simplest of them all and have little correlation with other digits. This algorithm is suitable to problems which have noisy dataset, have less number of attributes (instances could be represented in attribute-value pairs tractably), have discrete output values, have missing attribute values for some samples and the expressions are

disjunctive in nature. The training time of obtained tree comes out to be 0.3099 minutes which is small for this size of dataset.

k-Nearest Neighbour:

This method is the most basic easy to implement of all the other supervised machine learning classification algorithms. It is considered lazy algorithm because there is no training involved after training data is supplied instead it only stores the data. The training dataset used here have 54210 samples; 8920 new data samples are classified. K=6 is used here that is 6 nearest neighbours are considered for classification of incoming data sample. The proximity metric used here is Euclidian Distance which is the most basic and commonly used parameter. The

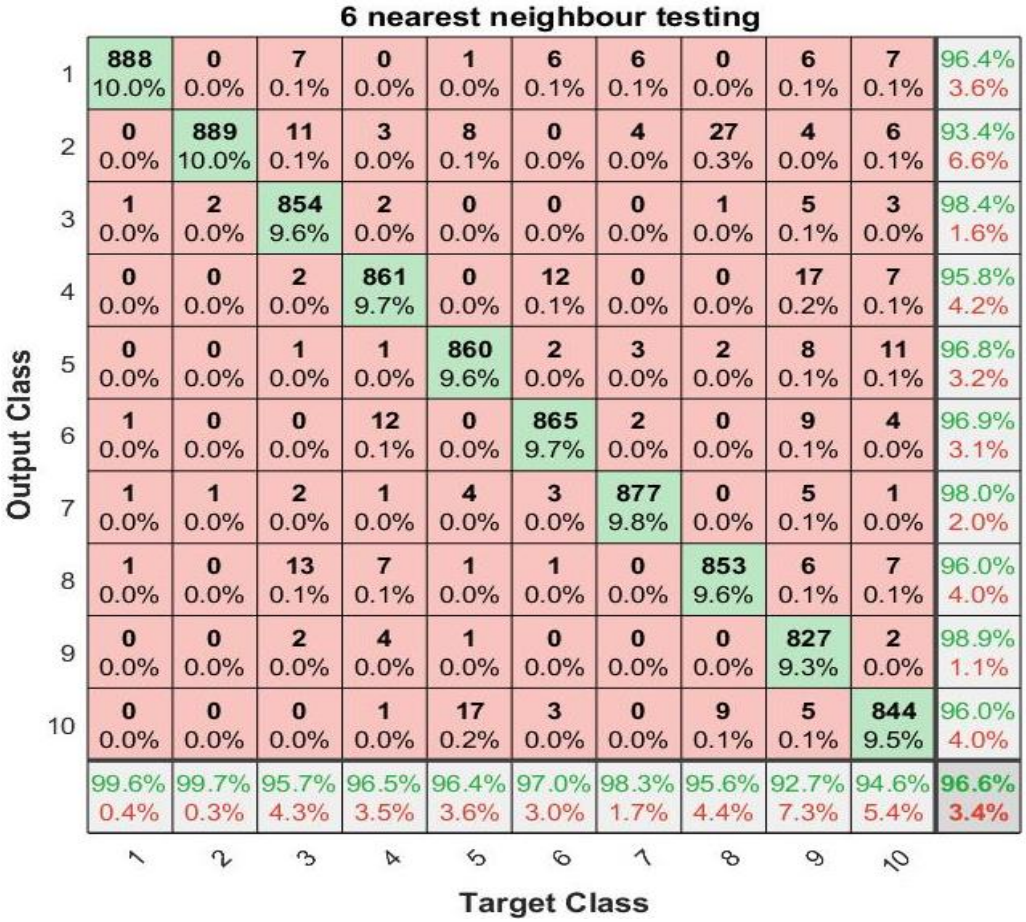


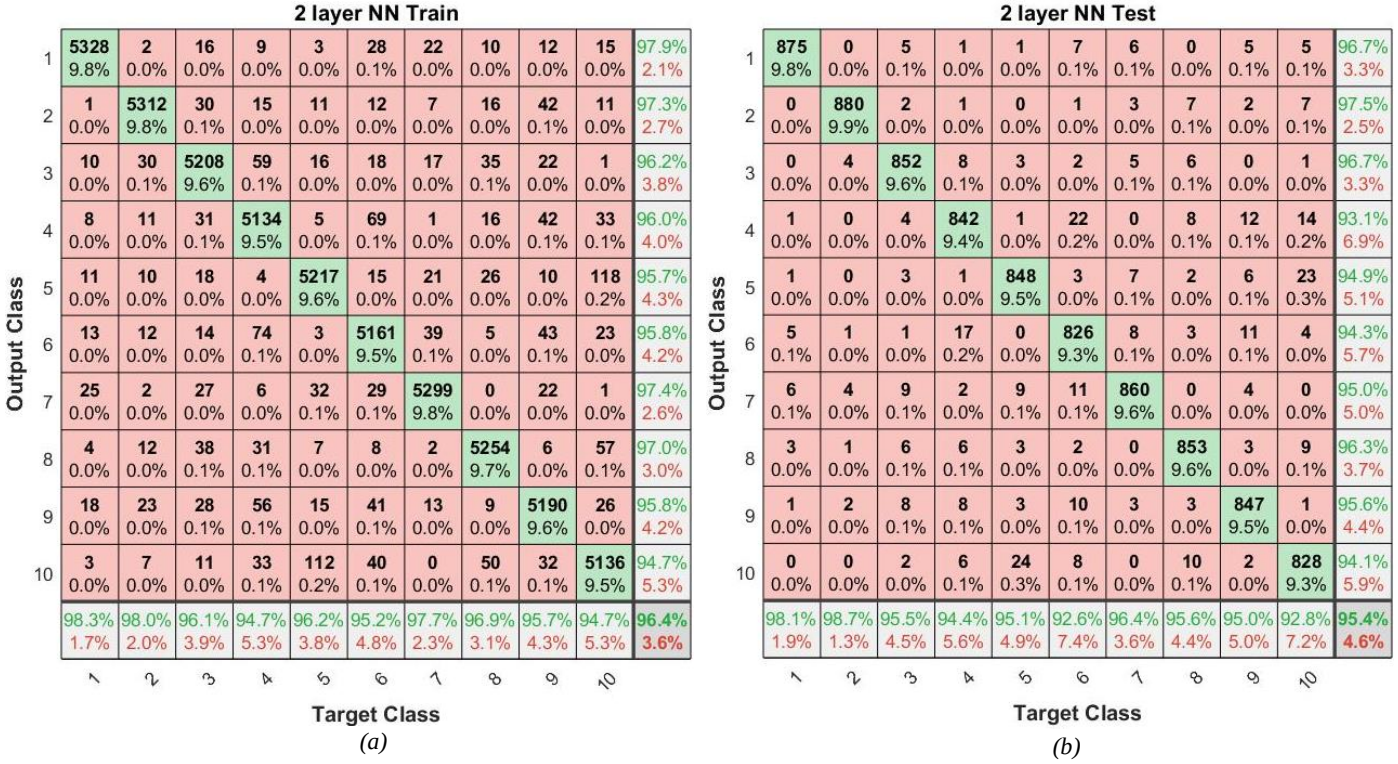
Fig 4.5 Confusion matrix showing 10-Class classification by k-NN method

testing accuracy obtained is substantially better than decision tree but at the expense of increased time and memory complexity. This method is very slow and sensitive to feature scaling and is not suitable for datasets having very high dimensionality. Although the testing accuracy is 96.6% but it takes approximately 30 minute to predict the output. Moreover, for

every new query sample, it calculates distance 54210 times (from every training samples) and this process is repeated 8920 times (for every testing samples); therefore a total of 54210×8920 distance calculations are performed. Here also the minimum efficiency is obtained for digit 8 (92.7%) and maximum efficiency is obtained for digit 1 (99.7%).

2-Layered Artificial Neural Network:

This method is the most prominent and recent supervised machine learning model. This model utilizes backpropagation coupled with gradient descent for learning. The gradients are calculated by using backpropagation method and the parameters of the network is updated by gradient descent. Two type of GD is implemented on the MNIST dataset; normal GD (also known as vanilla GD); Mini-batch GD. When size of dataset is large normal GD results in slower training and is computationally expensive although the performance is good. Therefore, instead of using whole training dataset for calculation of gradient, a small portion of samples are used called mini-batch; it is much quicker and have less computational complexity. The neural network used here has two layers; one hidden layer and one output layer. The hidden layer has 20 neuron units and output layer has 10 neuron units (one for



(a) *(b)*

Fig 4.6 (a) Training results, (b) Testing results by two-layered ANN

each class). The activation function used for hidden layer is Leaky Relu and for output layer Softmax function.

$$LeakyRelu(x) = \begin{cases} \propto x & if\ x < 0 \\ x & if\ x \geq 0 \end{cases} \quad (22)$$

Where $\propto$ is the multiplying factor less than one.

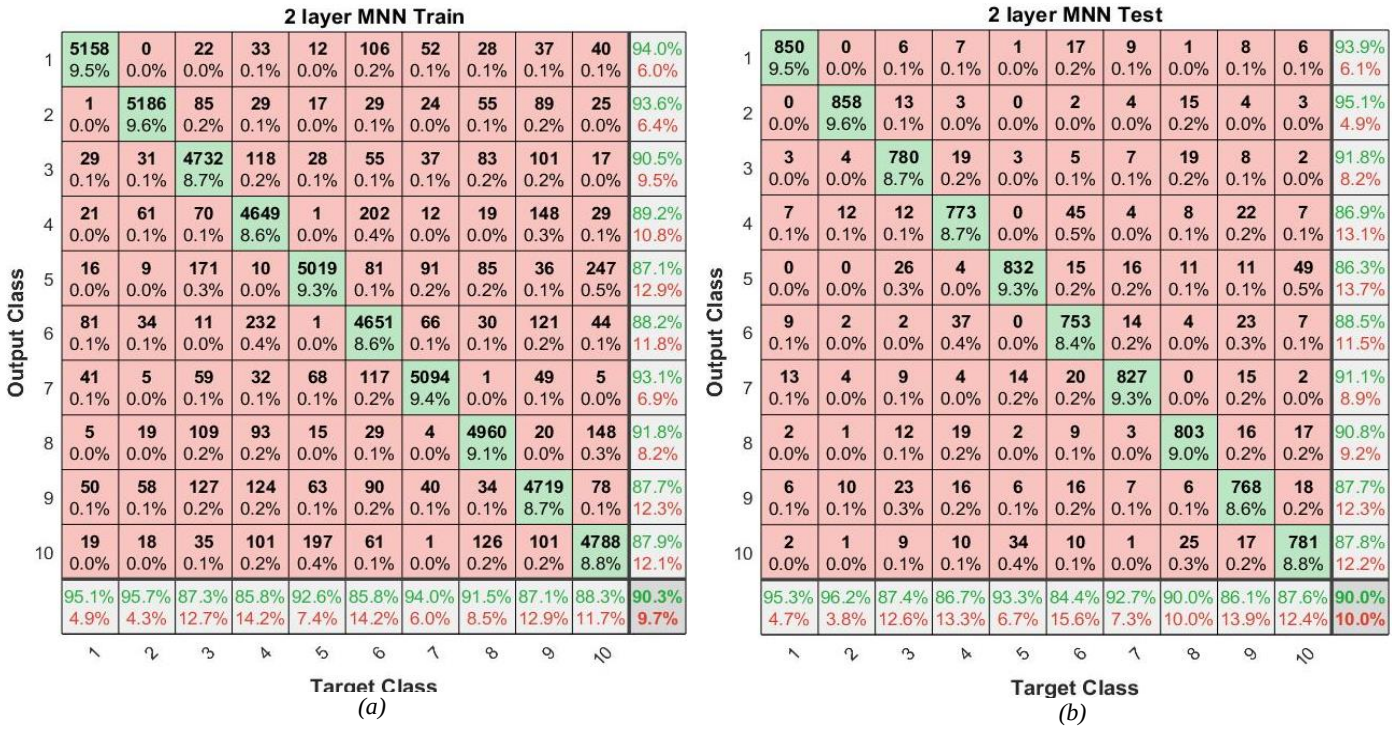
$$Softmax(x)_i = \frac{exp^{x_i}}{\sum_{j=1}^{c} exp^{x_j}} \quad (23)$$

Where numerator is the standard exponential function applied to each element of input vector x; c is the number of output class

The learning rate used here is 0.02 and a total of 10,000 iterations is utilized in training the network. The loss function used in this model is known as categorical Cross Entropy Loss-

$$Loss = -\sum_{i=1}^{c} t_i \log(y_i) \quad (24)$$

Where cis the number of class; t is target value and y is the predicted value for the corresponding class. From figure 4.8, an accuracy of 96.4% and 95.4% on training and testing samples respectively is obtained with a training time of 24.63 minutes. The best and worst classification is of digit 0 and digit 9&3 respectively for training; for testing digit 1 is best and digit 9 is worst classified. Although the accuracy is high and time is lower than k-nn, this method is still slow. In order to resolve this issue a mini-batch of 2500 training samples are used instead of full dataset. This results in substantial decrease in training time (1.0266 minutes) but the accuracy of training as well as testing is decreased to 90.3% and 90% respectively as can be observed from figure 4.9. Here the accuracy is decreased because the direction of movement along the gradient is less accurate than normal GD (less number of training samples are used here). The accuracy can be further optimized by selecting the optimal value of number of samples for mini-batch by hit and trial, although intuitively if the size of mini-batch is increased the training time will increase and the classification accuracy will also increase. The optimum point is that point at which the increase in training time does not provide sufficient amount of increase in classification accuracy that is the training time will increase faster than accuracy.

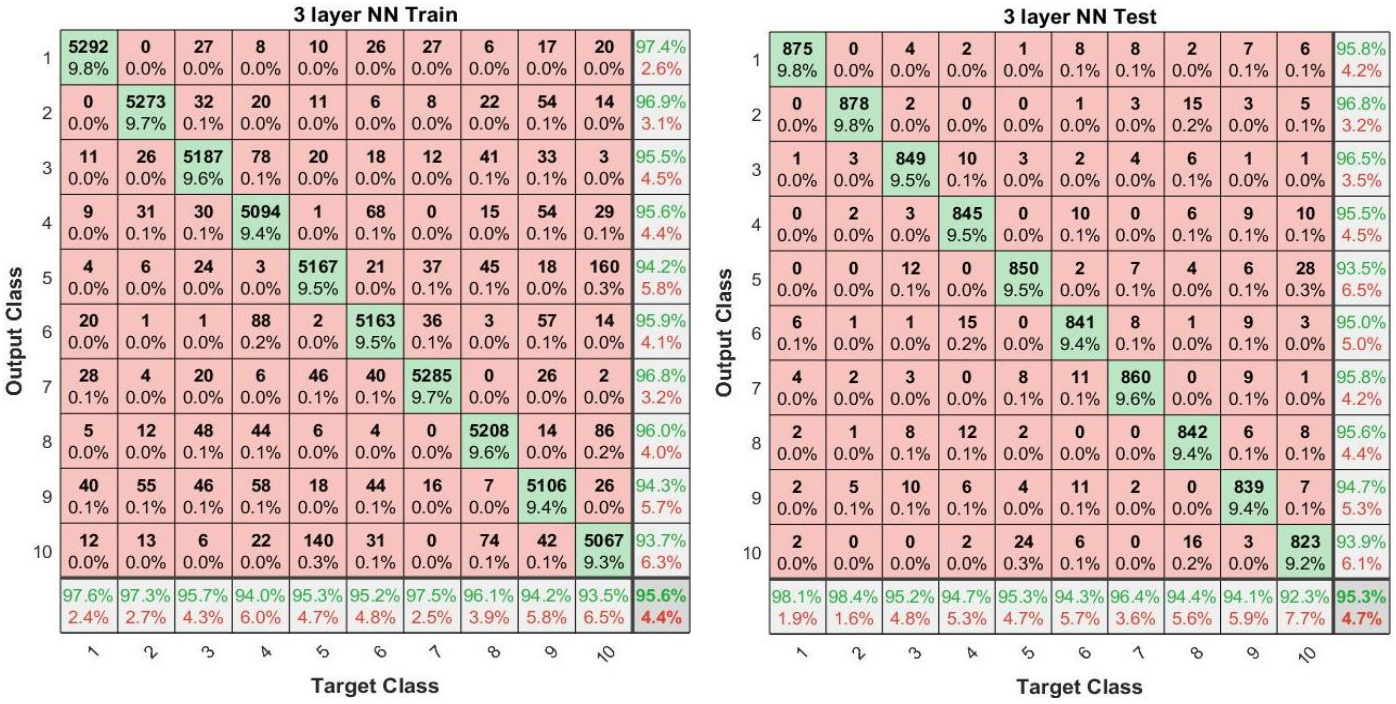


(a) *(b)*

Fig 4.7 (a) Training results, (b) Testing results by using mini-batch of 2500 samples in GD

3-Layered Artificial Neural Network:

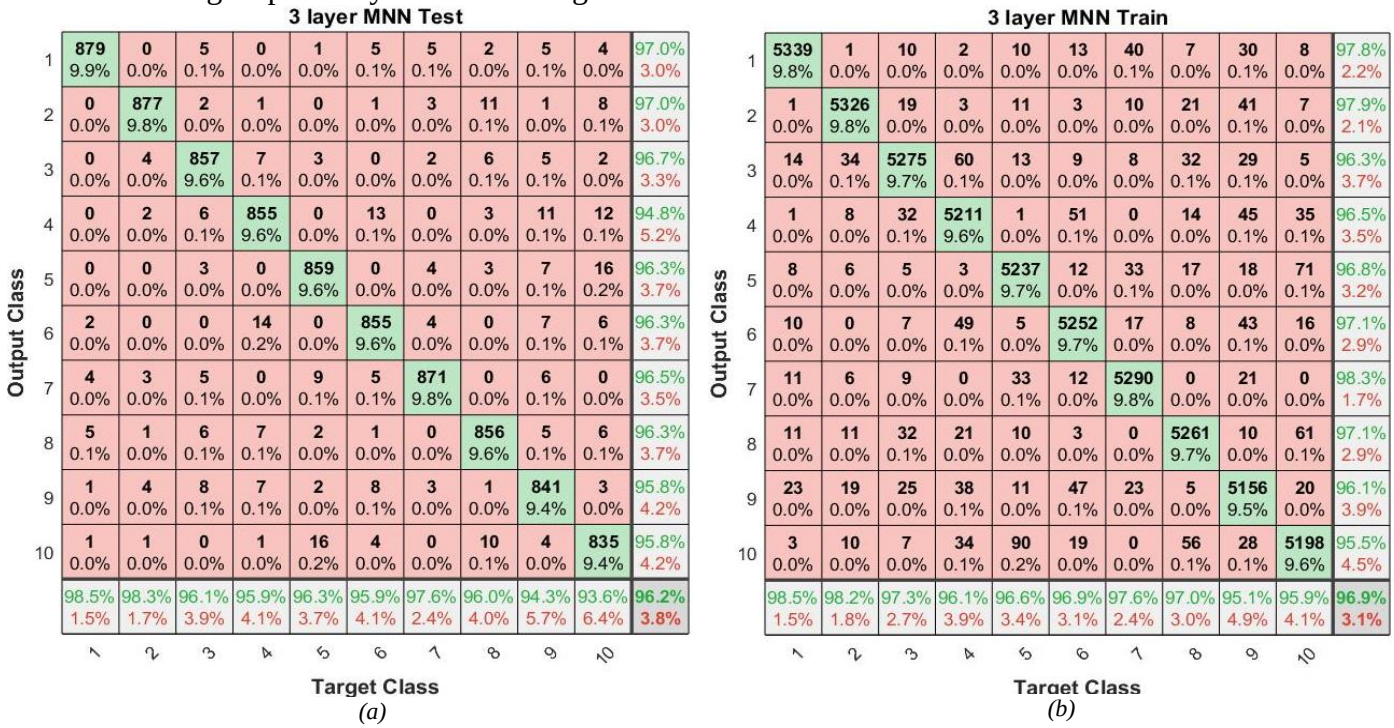
To further effectively increase the efficiency of classification another hidden layer is added into the network. Now the network has two hidden layers and one output layer. The first hidden layer contains 20 neuron units; the second hidden layer consist of 10 hidden units and the output layer again consist of 10 neuron units. The activation functions for each layer remains the same as before Leaky Relu for all hidden layers and Softmax for output layer. The loss function and learning rate are also same. This network is able to provide excellent accuracy with less number of iterations. Here the main difference in performance is observed in mini-batch GD. The normal GD provides an accuracy of 95.6% and 95.3% for training and testing respectively with lesser amount of iterations (5000) as shown in figure 4.10. Although the training time is slightly lesser than 2 layered network (22.21 minutes) but the computational complexity is increased per iteration. The trade-off for increasing computational complexity for decrease in training time does not benefit the classification accuracy as shown in figure 4.10. It can be observed that the difference in training and testing accuracies is decreased in this network and hence it can be inferenced that this model is providing better generalization error than previous network. The overall generalization capacity of neural network in this work is appearing to be better than decision tree.



(a) *(b)*

Fig 4.8 Confusion matrices for 3-layered NN for (a) Training samples, (b) Testing samples by normal GD

The main advantage of increasing the depth of NN appears when the mini-batch GD is applied. Due to the increase in number of layer and neuron units the requirement of number of iterations for training samples is decreased and hence the training time is decreased significantly. The classification efficiency is also increased to 96.9% and 96.2% for training and testing respectively as shown in figure 4.11.



(a) *(b)*

Fig 4.9 Confusion matrices for 3-layered NN for (a) Testing samples, (b) Training samples by mini-batch GD

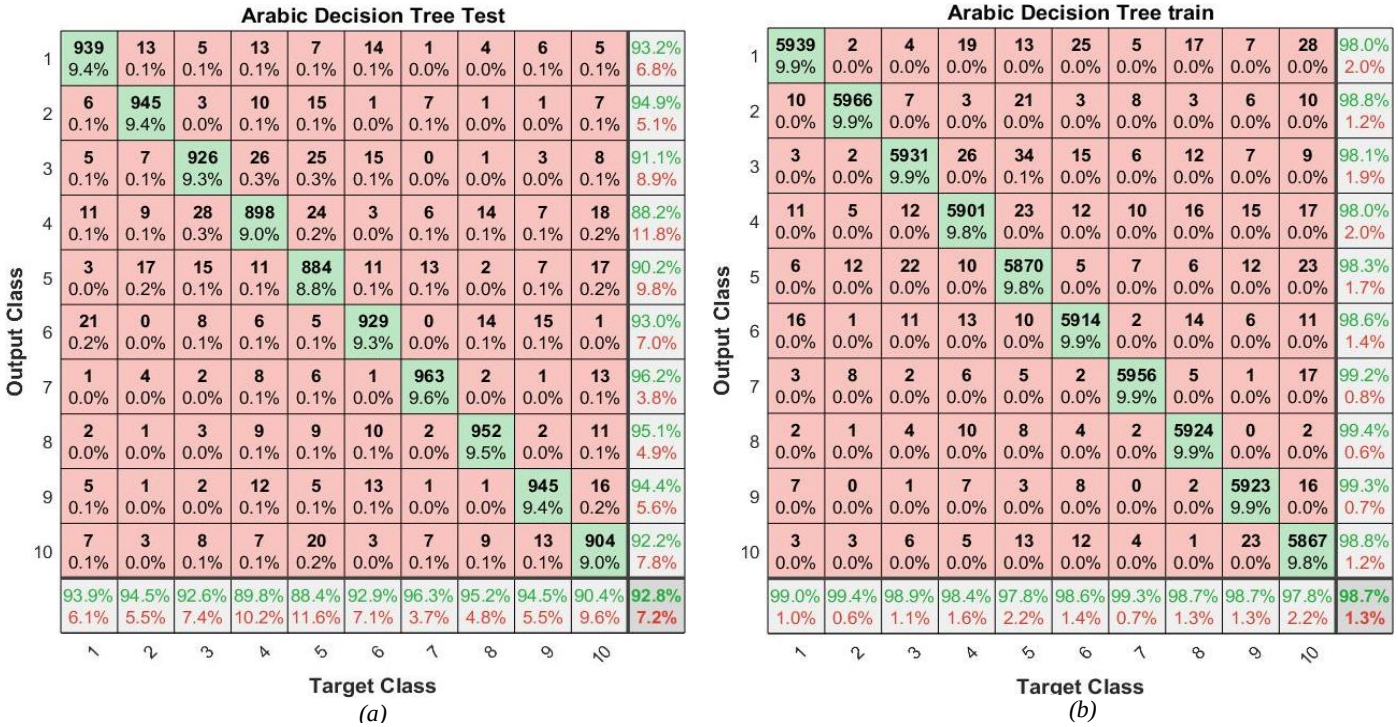
Table 4.1 Summary of implementation of different algorithms on MNIST dataset

Algorithm	Training accuracy (%)	Testing accuracy (%)	Training time (min)
Binary Decision Tree	96.7	88.2	0.3099
k-NN	NA	96.61	30
2-layered NN			
Normal GD	96.4	95.4	24.63
Mini-batch GD	90.3	90	1.0266
3-layered NN			
Normal GD	95.6	95.3	22.21
Mini-batch GD	96.9	96.2	2.198

4.6 Arabic Digits 10-class classification

Binary Decision Tree:

The dataset used here is downloaded from KRAGGLE [18] containing data of Arabic digits 0-9. The training set consist of 60,000 samples and testing set of 10,000 samples; all the remaining characteristics are similar to as of MNIST dataset. There is no normalization applied to the data before supplying to the decision tree algorithm. The performance on this dataset is better than MNIST; this could be due to the normalization technique and the nature of difference in shape of Arabic and English numerals. The splitting criterion used here is



(a) *(b)*

Fig 4.10 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by binary decision tree

Gini's Diversity Index and the total number of nodes in the final tree after training is 2,239. The training and testing accuracy obtained are 98.7% and 92.8% respectively. The worst case classification for training is digit 9 and 5 with 97.4% accuracy and for testing digit 4 with an accuracy of 88.4%.

k-Nearest Neighbour:

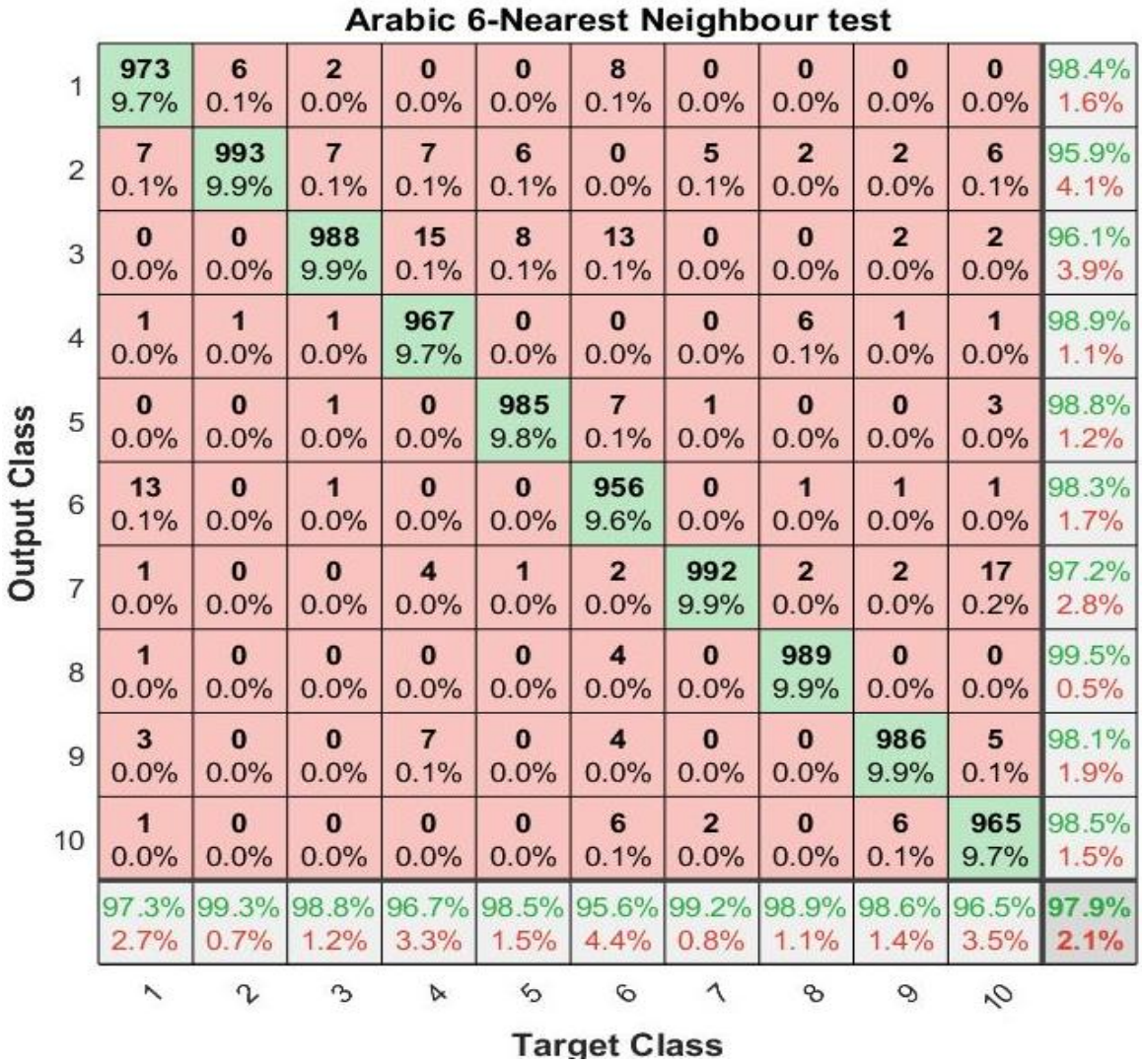
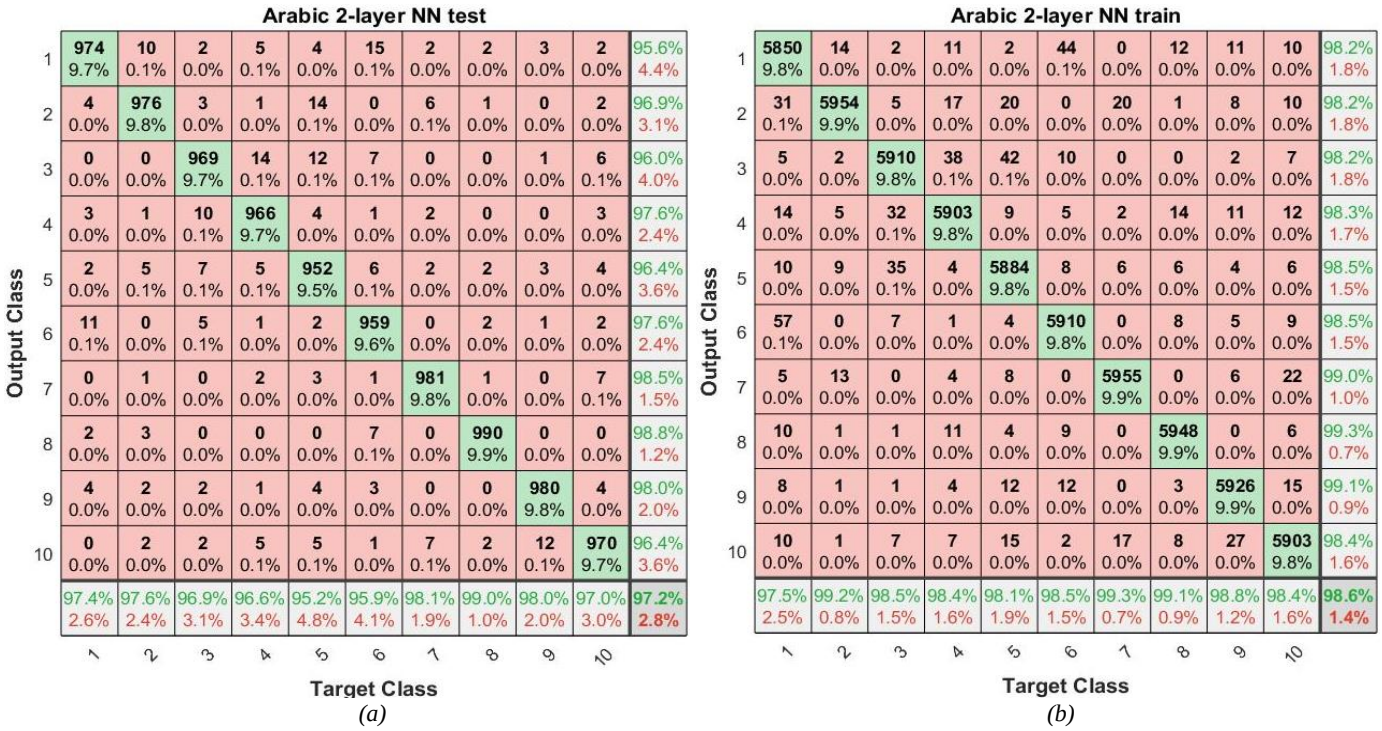


Fig 4.11 Confusion matrices of Arabic numerals by 6-Nearest Neighbour algorithm

Since this dataset has larger number of samples than MNIST dataset, the time taken to predict the output is also greater (31 minutes) but the accuracy is best being 97.9%. Here also no normalization is applied before supplying the data to the model. The worst and best classification is obtained for digit 5(95.6%) and digit 1 (99.3%) respectively.

2-Layered ANN:

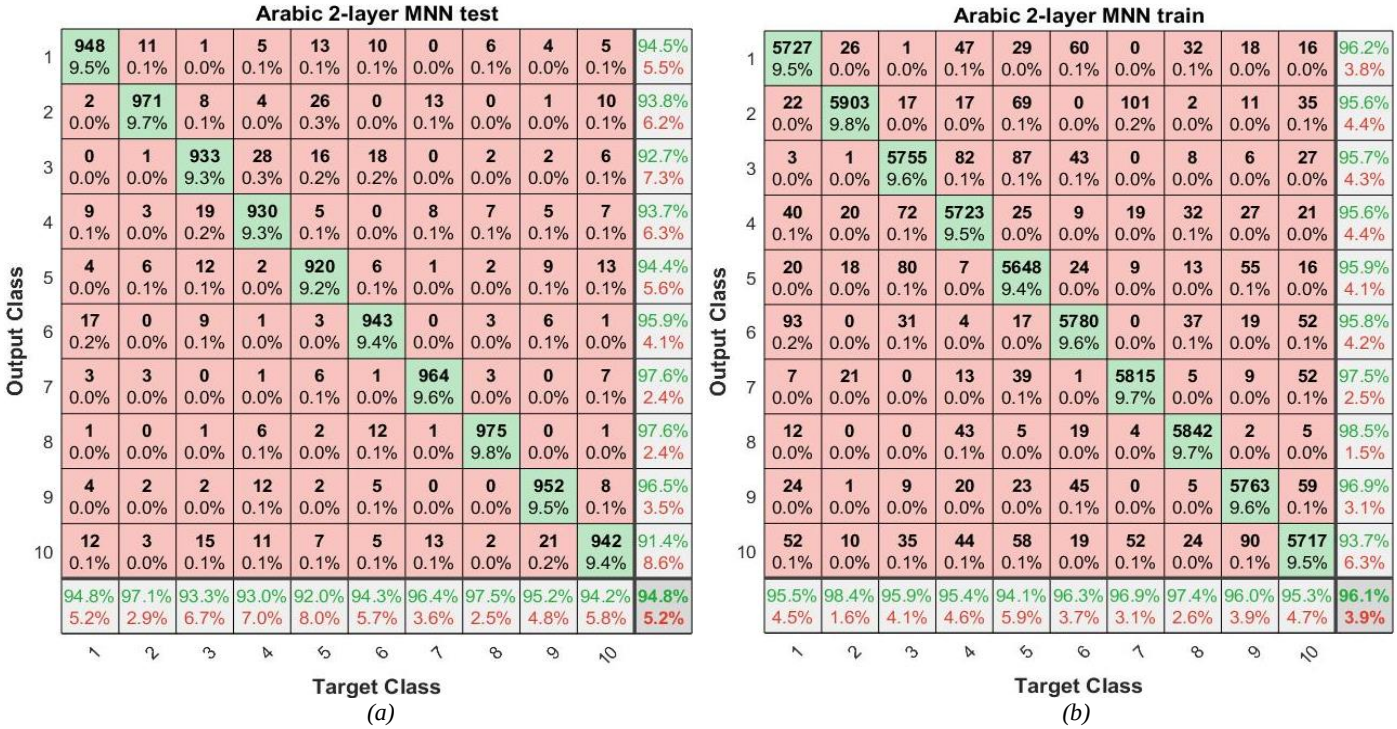


(a) *(b)*

Fig 4.12 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by 2-layered NN using normal GD

This neural network utilizes two layers; one hidden and one output layer. Hidden layer consists of 20 neuron units and output unit consists of 10 neuron units. All hidden units use Leaky Relu and all output units use Softmax as their activation functions. The learning rate used here is 0.02 and the iteration used for training is 10,000. The output in figure 4.14 is obtained from implementing normal GD and that is the reason training of this network on the Arabic dataset is slow (26 minutes) but still faster than k-NN. The testing and training accuracies obtained are 97.2% and 98.6% respectively with worst case classification for digit 5 (95.9%) for testing and digit 0 (97.5%) for training. Figure 4.15 is showing the results obtained by applying the mini-batch GD using only 2500 samples at random instead of whole training dataset. The training time is reduced significantly (0.6512 minutes). The computational complexity is also drastically reduced. The size of parameters for layer 1 is 20×784 and for layer 2 is 10×20 that is 15,880 parameters excluding the bias parameters. In one iteration a total of 15,880×60,000 number of gradients are calculated for normal GD whereas for mini-batch GD only 15,880×2500 gradient calculations are performed. Moreover, the number of iteration is also lower (3,386), Overall the mini-batch computes 25

times less gradient calculations than normal GD in one iteration. The accuracy drops to 96.1% for training and 94.8% for testing.

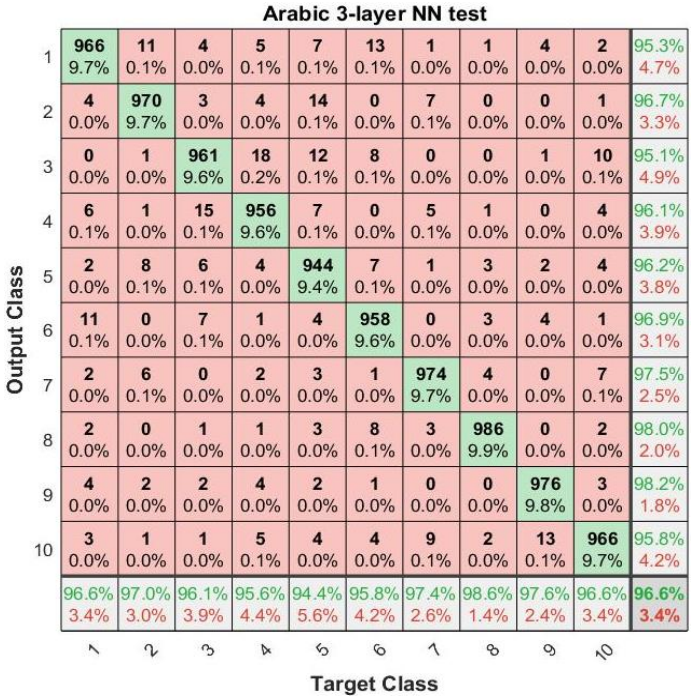


(a) *(b)*

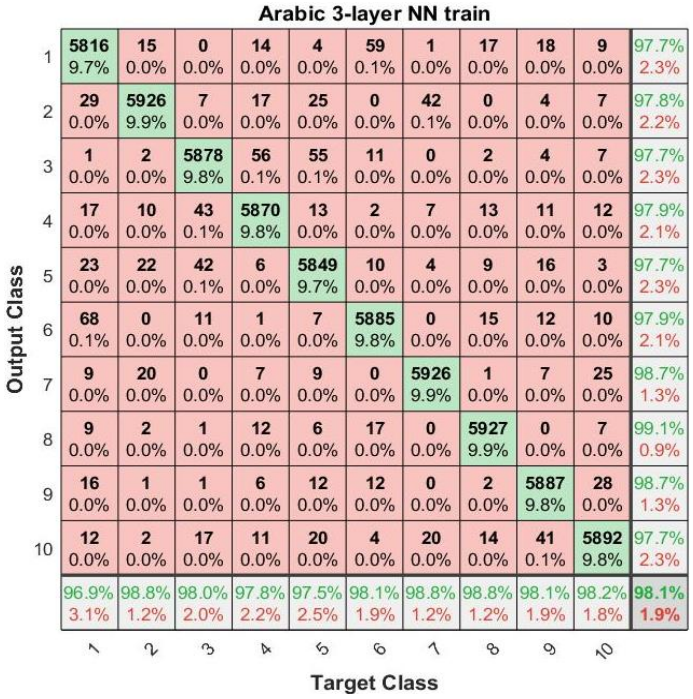
Fig 4.13 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by 2-layered NN using mini-batch GD

3-Layered ANN:

One additional hidden layer of 10 neuron units is added to the previous network. Adding a layer generally results in more detailed characteristic extraction and increase in the level of abstractness of data but it could also lead to overfitting due to the increase in number of neurons. Here the training accuracy remains almost similar to 2-layered network (98.1 %) but the testing accuracy decreases to 96.6%. But the training time is decreased slightly to 24.12 minutes which is still slow. The number of weights are now increased to 15,980 and hence resulting in the increase of computational complexity per iteration. But the number of iterations used is 5000 which results in decrease of overall number of calculations. Figure 4.16 shows the results for 3-layered network by normal GD. Here also a mini-batch variant of GD with batch size of 2000 samples is implemented. Since this method is much faster and less computationally expensive the number of iterations are increased to 10,000 to achieve higher accuracy of 97.1% for testing and 98.7% for training. Figure 4.17 shows these results.

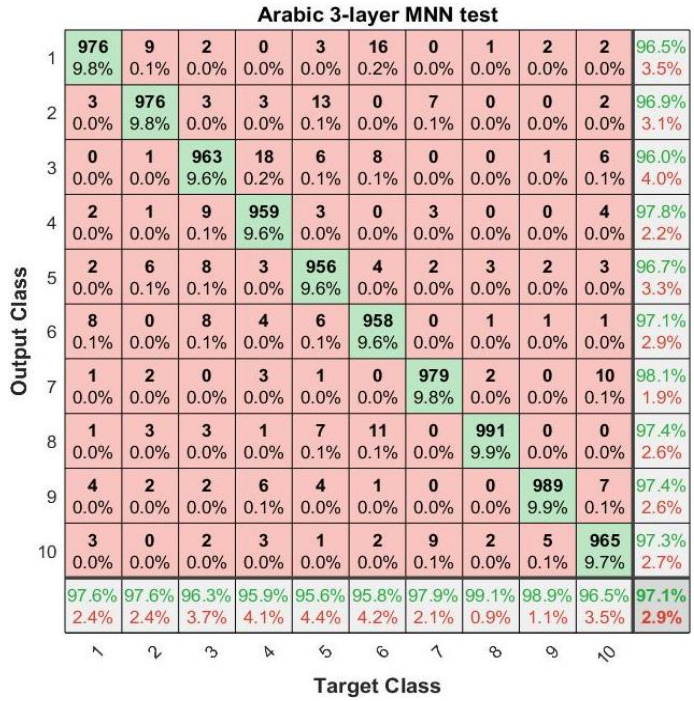


(a)

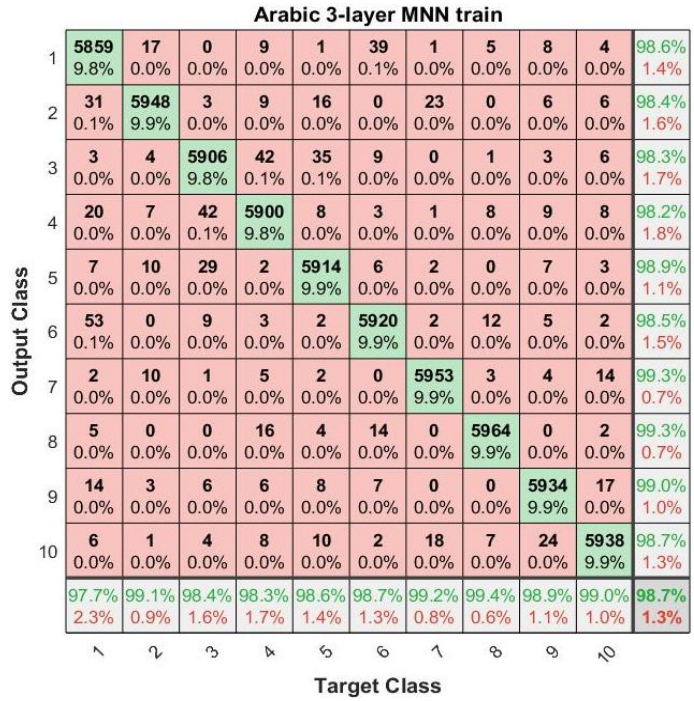


(b)

Fig 4.14 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by 3-layered NN using normal GD



(a)



(b)

Fig 4.15 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by 3-layered NN using mini-batch GD

Custom modified approach:

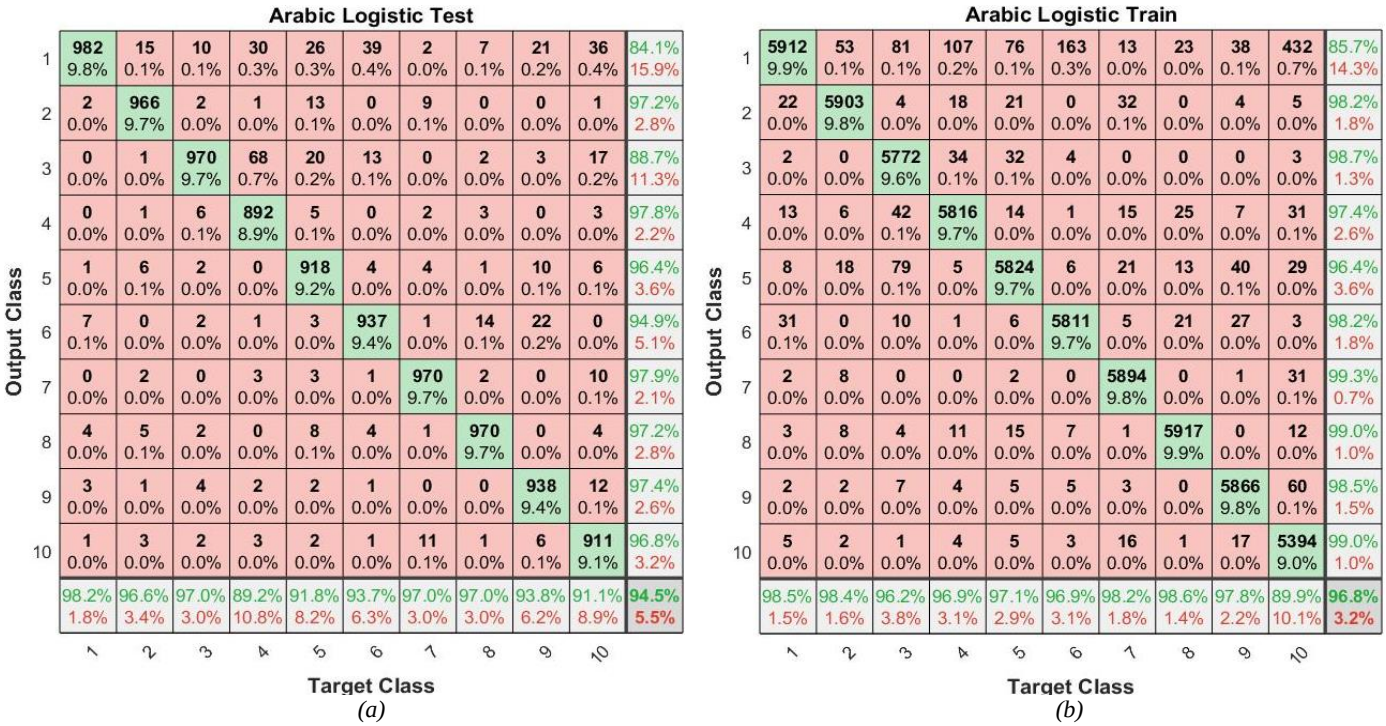
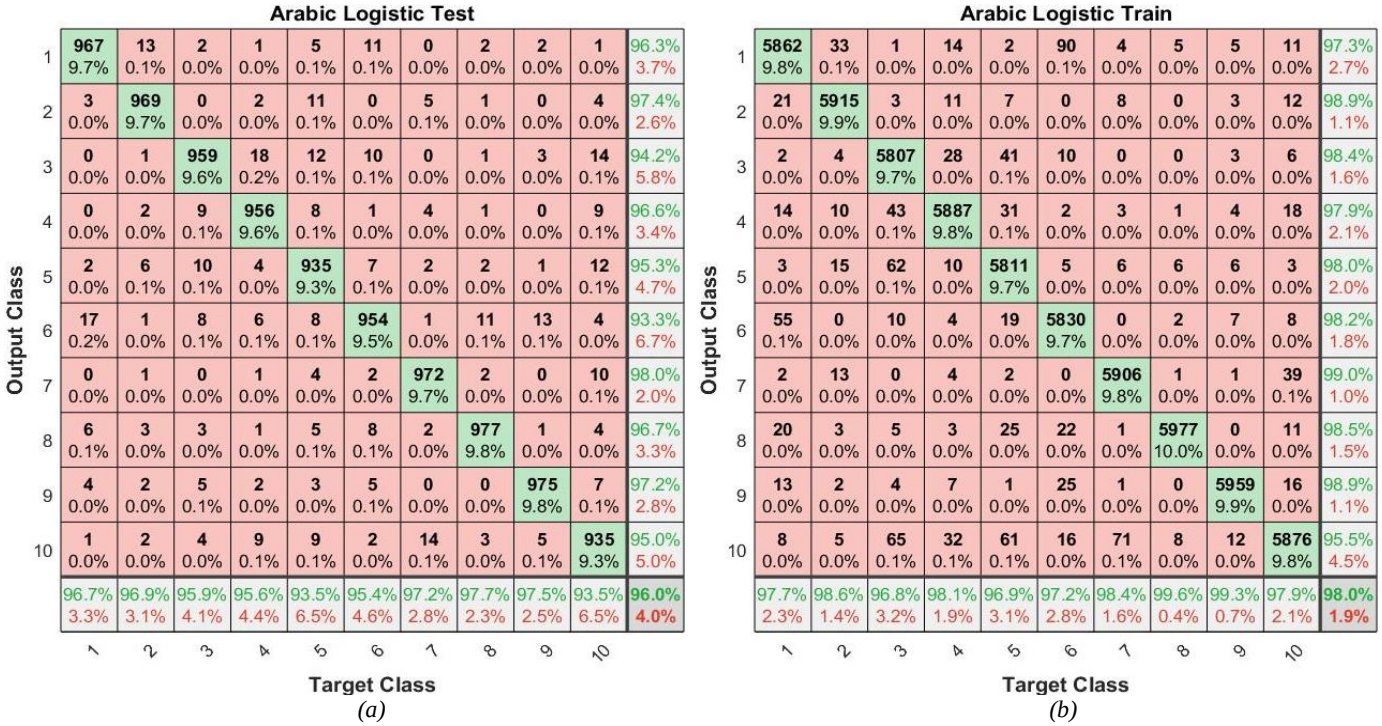


Fig 4.16 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by One Vs Rest binary LR

Figure 4.16 shows the results on training and testing by applying ten One Vs Rest binary classifiers units based on Logistic Regression. Each classier has a target=1 for only one class and target=0 for rest of the classes. The accuracy by this model is slightly smaller as compared to 3-layered NN but the difference in number of neuron units compensates. Here only 10 units are used whereas 3-layered network uses 30 neuron units thereby increase the calculation complexity and training time.

To further improve the accuracy, additional binary classifiers are used. By analysing the classification results of One Vs Rest model, 10 additional binary units are constructed taking only pairs of classes that are closest to each other that is there classification is poorest. The design of layer 1 is complete. To combine the effects of layer 1, another layer is designed which is output layer of the network. This output layer has 10 units corresponding to each class (digit). The output of these units are again obtained by applying One Vs Rest approach. Therefore, the first layer has 20 units and output layer has 10 units. The output of first layer is used as input of output layer. All these classifier units in the network use Logistic Regression.



(a) *(b)*

Fig 4.17 Confusion matrices of Arabic numerals. (a) Testing, (b) Training by custom approach

In table 4.2 all the algorithms applied to the Arabic dataset are summarized for better comparison and comprehension purpose.

Table 4.2 Summary of implementation of different algorithms on Arabic dataset

Algorithm	Training accuracy (%)	Testing accuracy (%)	Training time (min)
Binary Decision Tree	98.7	92.8	0.3668
k-NN	NA	97.94	31
2-layered NN			
Normal GD	98.6	97.2	26.3
Mini-batch GD	96.1	94.8	0.6512
3-layered NN			
Normal GD	98.1	96.6	24.12
Mini-batch GD	98.7	97.1	2.3563
One Vs Rest	96.8	94.5	15
Custom Approach	98	96	20

4.7 Self-made Handwritten Digits 10-class classification

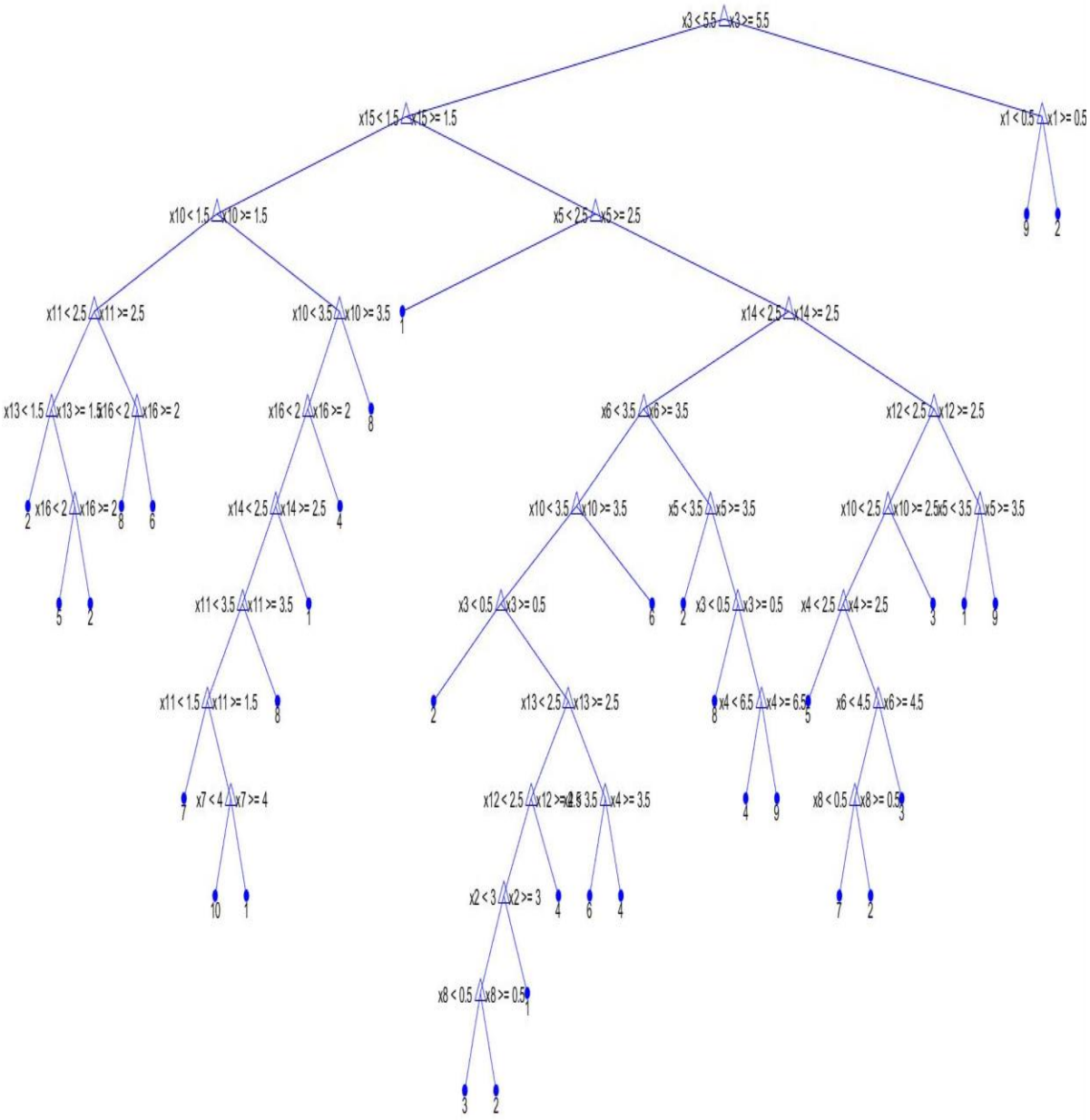
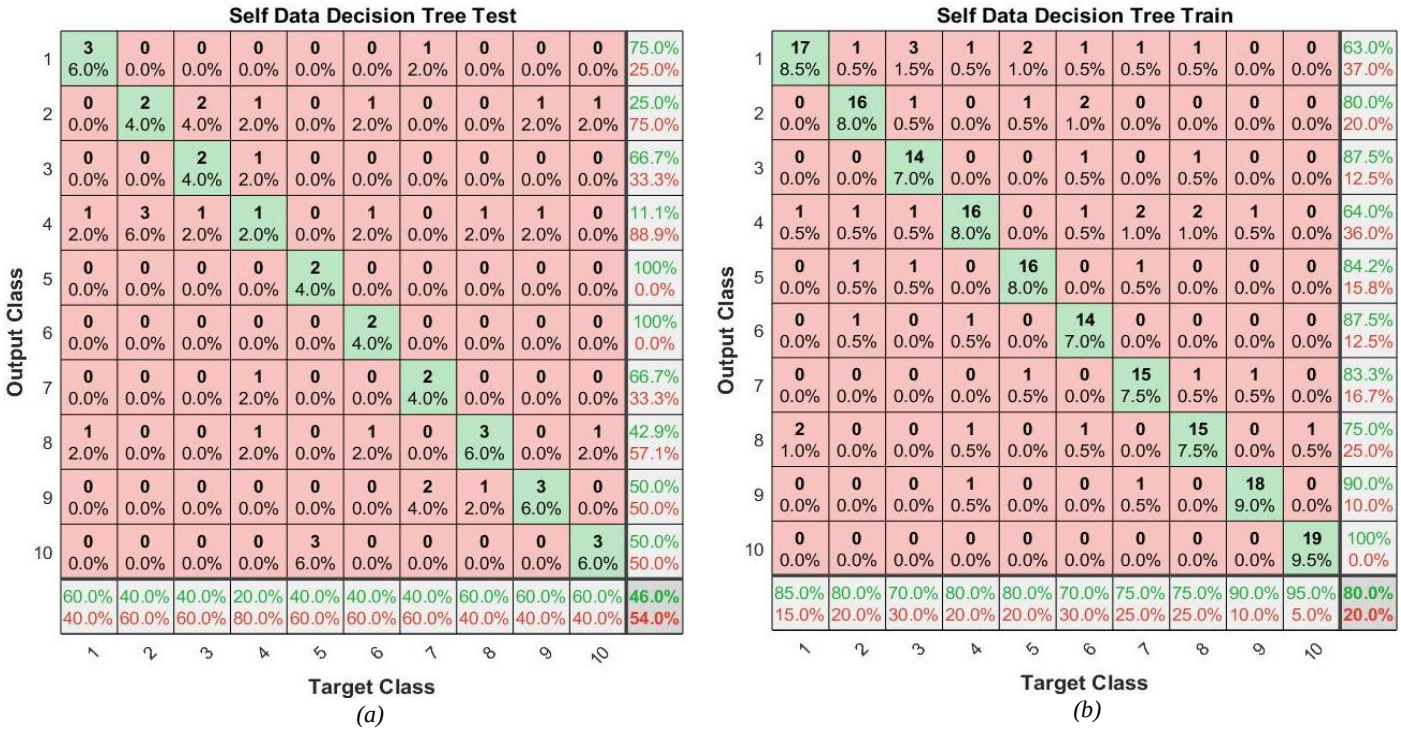


Fig 4.18 Decision Tree constructed by training on self-made handwritten digits data

The dataset used here is small containing only 20 samples for training and 5 samples for testing. This dataset is generated for experimental purposes that is to develop a new algorithm for classification. Figure 4.18 illustrates the binary decision tree obtained, it contains 67 nodes that is the number of times the Gini's splitting criterion performed is 67. Since the feature size or attributes of generated dataset is small (16 features), the training (80%) and testing (46%) accuracies are not good as shown in figure 4.19.



(a) *(b)*

Fig 4.19 Confusion matrices of Self-made numerals. (a) Testing, (b) Training by binary decision tree

k-Nearest Neighbour:

As discussed earlier, this algorithm is best suitable for small datasets thereby the classification results obtained are better but still worse than standard MNIST and Arabic datasets. The training time (0.3381 seconds) is very small as compared to other methods, this the direct consequence of training set size being small. The accuracy obtained through this method is 74% as can be observed from figure 4.20 and is the highest for the generated dataset. Since the dataset is small and scattered, the highest accuracy is achieved for the value of k=1 that is the classification is done by consideration of only one nearest neighbour. The worst classification is obtained for digit 4 (40%).

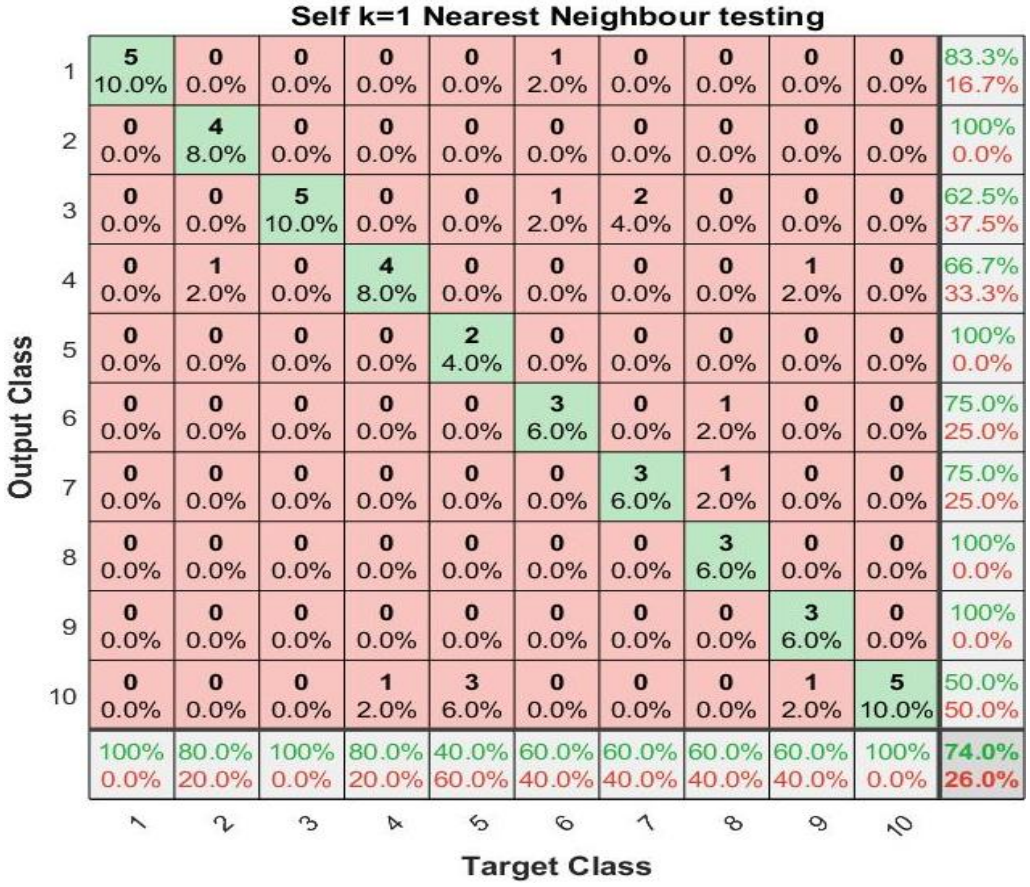
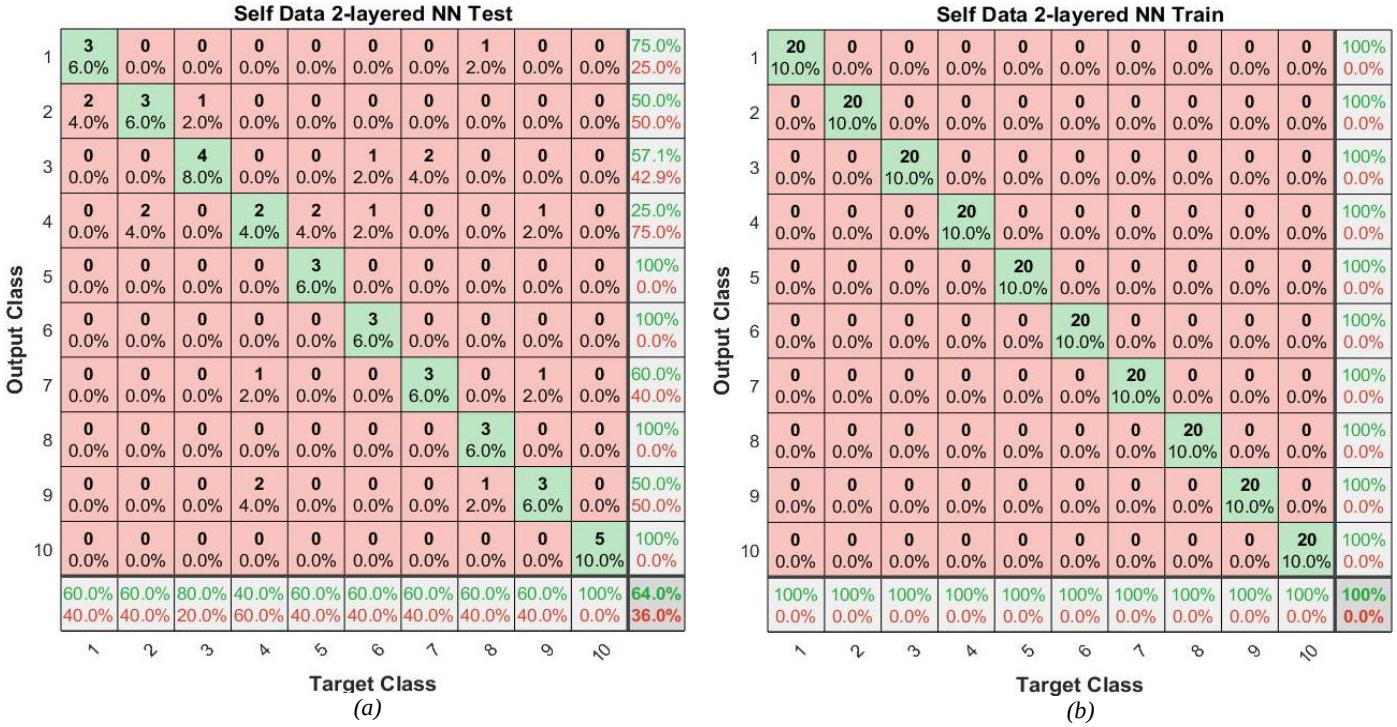


Fig 4.20 Confusion matrix for self-made dataset by k-NN at k=1

2-layered ANN:

The network consists of two layers; one hidden and one output layer. The hidden layer contains 20 units and output layer contains 10 units. Iterations used for training are 10,000, this whole process takes 8.8769 seconds. Although the network is able to excellently learn the training dataset as shown in figure 4.21 with 100% accuracy but the testing accuracy (64%) is still lower than k-NN. But here the accuracy of digit 4 (60%) is greater than k-NN. Since the sample size is very small the implementation of mini-batch GD is irrelevant and unnecessary. Also, the increase in depth of neural network does not produce any good results.



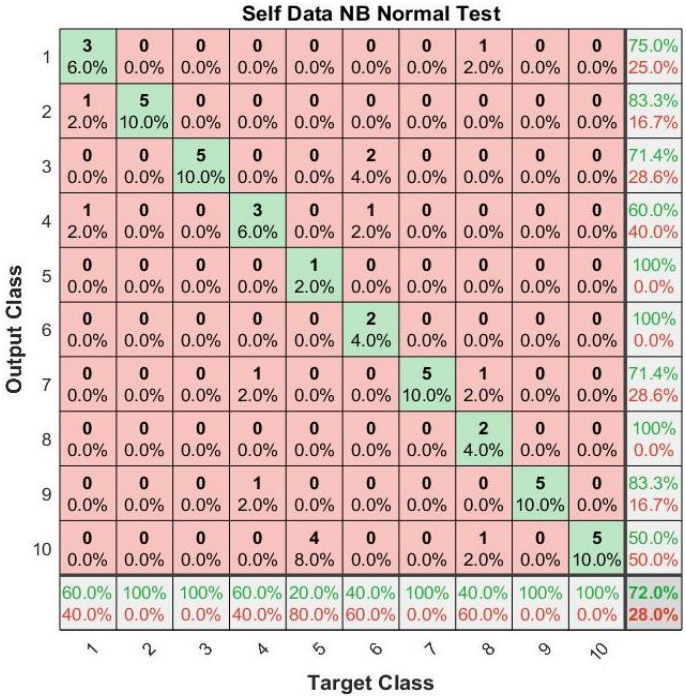
(a) *(b)*

Fig 4.21 Confusion matrices of Self-made numerals. (a) Testing, (b) Training by 2-layered ANN

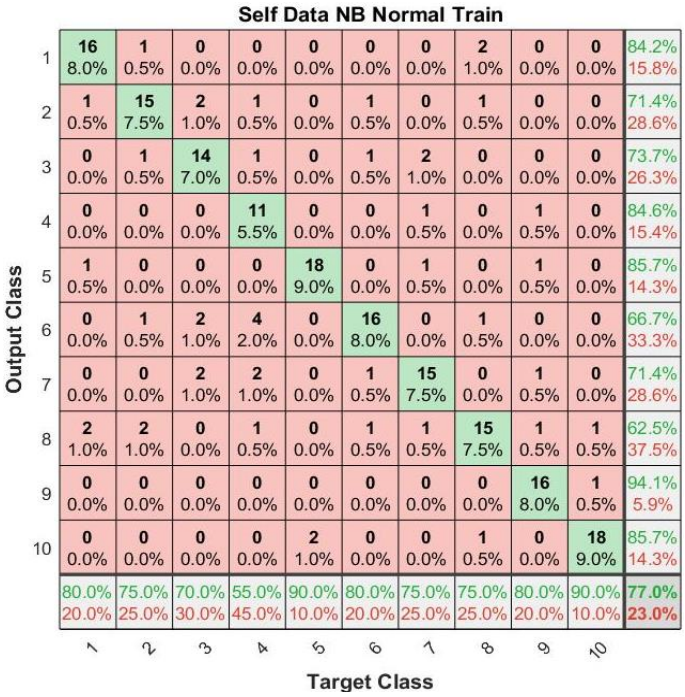
Naïve Bayes Classifier:

This method is very sensitive to statistical parameters like mean and variance. It calculates mean and standard deviation of every class for all attributes separately. If in any class, even one attribute has zero mean or variance that is the value is either zero or constant throughout all samples of corresponding class, this method could not be able to train. These type of attributes contribute negligible amount of information; the simplest way to tackle this is to remove that feature. To implement this method third and sixth feature values are removed for every class thereby the input matrix is now 14×200. Likelihood of data is calculated by utilizing normal probability distribution function. Although the training accuracy (77%) is lower than ANN but the testing accuracy (72%) is 8% greater this implies that this model is generalizing better.

This method works best with discrete categorical dataset. Although it can be applied to continuous datasets but the assumptions are very strong and often the real world datasets did not follow these assumptions. The dataset is assumed to follow a standard probability distribution, the most popular distribution used in normal or Gaussian. The distribution is selected which best describes the data.



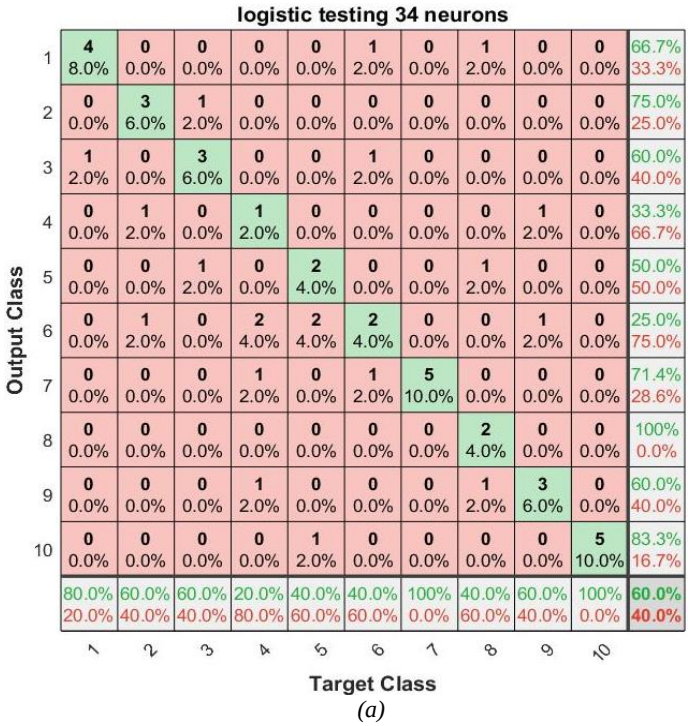
(a)



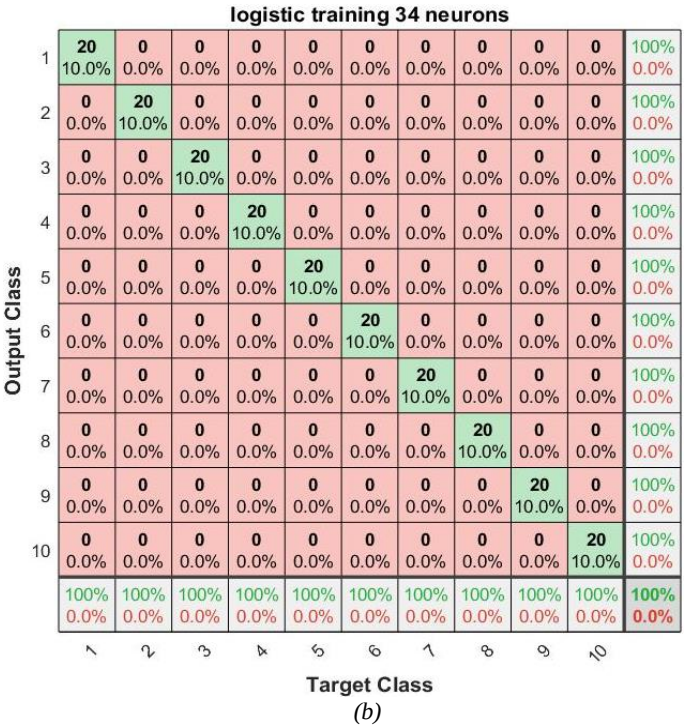
(b)

Fig 4.22 Confusion matrices of Self-made numerals. (a) Testing, (b) Training by Naïve Bayes Classifier

Custom modified approach:



(a)



(b)

Fig 4.23 Confusion matrices of Self-made numerals. (a) Testing, (b) Training by custom approach

The technique of designing of network layers as well as each of binary classifiers are same as used in Arabic dataset. The only difference is the number of neuron units (24) in the first layer. As can be observed from the figure 4.23 the poorest classification is of digit 3 with recall (20%) and precision (33%) while testing.

Table 4.3 summarizes the results obtained by implementation of different algorithms on self-generated handwritten digits dataset. K-NN is giving the best result as observed from the table.

Table 4.3 Summary of implementation of different algorithms on Self-made dataset

Algorithm	Training accuracy (%)	Testing accuracy (%)	Training time (secs)
Binary Decision Tree	80	46	0.3381
k-NN	NA	74	0.0137
2-layered NN	100	64	8.87
Naïve Bayes	77	72	0.0875
Custom Approach	100	60	4.4

Chapter 5

Conclusion

After implementing the logic gates by PLA, it is evident that the one perceptron can e used to implement any linear function. The XOR logic is not linearly separable and hence cannot be implemented by single perceptron, although just by increasing one more perceptron and connecting it in proper way can implement XOR. The one-step pseudo inverse method for regression is more fast and efficient but less stable than the iterative gradient descent having linear hypothesis for the dataset of chance of admit. As the dimension of input data is increased gradient descent performs better.

In handwritten digit classification, the dataset generated is binary classifiable if only two classes are taken at a time; thus PLA, logistic regression, linear regression can be used to obtain the results although they are not shown in this report. For 10-class classification four different supervised machine learning algorithms are implemented on three different datasets but of same category. The best result is obtained on Arabic dataset by k-nearest neighbour algorithm using k=6 with an accuracy of 97.94% but the training time is longest (31 minutes). By observing the results, it can be concluded that the mini-batch gradient descent outperforms other implemented algorithms substantially in terms of training time as well as accuracy for large datasets (MNIST and Arabic datasets) but k-NN performs better for smaller datasets as in the case of self-generated handwritten digits dataset.

Results obtained from the Arabic dataset are better than MNIST dataset, this could be due to the difference in nature of structure of digits. The digits in Arabic language are less complex and similar in comparison to English digits. All the obtained results can be further improved by using different techniques like method of normalizing input data, validation techniques, regularization etc.

References

[1] D. O. Hebb. (1949). *The Organization of Behaviour. A Neuropsychological Theory* (1st ed.). John Wiley & Sons. Inc.

[2] Frank Rosenblatt. (1958) *The perceptron: a theory of statistical separability in cognitive systems.* Cornell Aeronautical laboratory

[3] Tom Mitchell (1997). *Machine Learning.* McGraw Hill

[4] Y.Lecuu, B.Boser, J.S Denker, D.Herderson, R.E Howard, W.Hubbard and L.D Jackel, "Handwritten Digit Recognition with a Back-Propagation Network", AT&T Bell Laboratories, Holmdel,N.J.077331990

[5] Daniel Cruces Alvarez, Fernando Martin Rodriguez and Xulio Fernandez Hermida, "Printed and Handwritten Digits Recognition Using Neural Networks", *The Ninth International Conference on Signal Processing Applications & Technology,* Sep 1998, Vol. 1

[6] Dewi Nasien, Siti S. Yuhaniz and Habibollah Haron, "Recognition of Isolated Handwritten Characters using one continuous route of freeman chain code representation and feed forward neural network classifier", *International Journal of Computer and Information Engineering,* 2010, Vol. 4, No. 7

[7] P. Pandi Sevi and Dr. T. Meyyappan, "Recognizing Handwritten Numerals Using Multilayer Feedforward Back-propagation Neural Network", *International Journal of Computer Technology and Applications,* Vol3(6) 1939-1944,2012.

[8] S. Iamsa-at and P. Horata, "Handwritten Character Recognition Using Histograms of Oriented Gradient Features in Deep Learning of Artificial Neural Network," 2013 International Conference on IT Convergence and Security (ICITCS), 2013, pp. 1-5, doi: 10.1109/ICITCS.2013.6717840.

[9] Rajiv Kumar and Kiran KumarRavulakollu, "Handwritten Devnagari Digit Recognition Benchmarking on New Dataset", *Journal of Theoretical and Applied Information Technology,* Vol. 60, No. 3, Feb 2014

[10] Saeed Al-Mansoori, “Intelligent Handwritten Digit Recognition using Artificial Neural Network” *International Journal of Engineering Research and Application (IJERA),* Vol.5, pp.46-51. May 2015

[11] Haidar Al-Wzwazy, “Handwritten Digit Recognition using Convolutional Neural Networks”, *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 4, pp 1101-1106, Feb 2016

[12] Saqib Ali; Zeeshan Shaukat, Muhammad Azeem, Zareen Sakhawat; Tariq Mahmood, Khalil ur Rehman, “An efficient and improved scheme for handwritten digit recognition based on convolutional neural network”. *SN Applied Sciences,* Vol. 1,pp 1:1125, 2019

[13] Yawei Hou, Huailin Zhao, “Handwritten digit recognition based on depth neural network”, *IEEE International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS),* pp. 35–38, 2017

[14] S. M. Shamim, Md Badrul Alam Miah, Angona Sarker, Masud Rana, Abdullah Al Jobair, “Handwritten Digit Recognition Using Machine Learning Algorithms”, *Indonesian Journal of Science and Technology,* vol. 3, No. 1, 2018

[15] Yuji Roh, Geon Heo, Steven Euijong Whang, “A Survey on Data Collection for Machine Learning”, *A Big Data - AI Integration Perspective*, 12 Aug 2019

[16] Kotsiantis, S.B., “Supervised Machine Learning: A Review of Classification Techniques”, *Informatica 31*, pp. 249-268, July 16 2007

[17] “Graduate Admission 2” https://www.kaggle.com/mohansacharya/graduate-admissions

[18] "Arabic Handwritten Digits Dataset” https://www.kaggle.com/mloey1/ahdd1/version/3?select=Train+%2B+Test+Matlab.mat

[19] Szilard Vajda and BarnaSzocs, “A Neural Network Based Distance Function for the K-Nearest Neighbor Classifier” *International Conference on Frontiers in Handwriting Recognition*, 2014 IEEE

[20] Tsehay Admassu Assegie, Pramod Sekharan Nair, “Handwritten digits recognition with decision tree classification: a machine learning approach”, *International Journal of Electrical and Computer Engineering (IJECE),* Vol. 9, No. 5, pp. 4446-4451, October 2019

[21] Ionut B. Brandusoiu Gavril I. Toderean (2020). *How to fine-tune Neural Networks for Classification.* GAER Publishing House

[22] Szilard Vajda and Barna Szocs, “A Neural Network Based Distance Function for the K-Nearest Neighbor Classifier” *International Conference on Frontiers in Handwriting Recognition,* Vol. 1, pp. 429-433 , 2014 IEEE

[23] Danial Berrar(2018), “Bayes’ Theorem and Naïve Bayes Classifier”, DOI: 10.1016/B978-0-12-809633-8.20473-1

[24] Sonia Singh, Priyanka Gupta, “Comparative Study ID3, CART, C4.5 Decision Tree Algorithm: A Survey”, *International Journal of Advanced Information Science and Technology (IJAIST),* Vol. 27, No. 27, July 2014

[25] Yongli Zhang, “Support Vector Machine Classification Algorithm and its Application”, *International Conference on Information Computing and Applications,* pp. 179-186, 2012

[26] Anshuman Sharma, “Handwritten digit Recognition using Support Vector Machine”, *Neural and Evolutionary Computing,* March 2012, arXiv:1203.3847

[27] Chychkarov, Serhiienko, Syrmamiikh, Kargin, “Handwritten Digits Recognition using SVM, KNN, RF and Deep Learning Neural Networks”, *The Fourth International Workshop on Computer Modelling and Intelligent Systems,* Vol. 2864, April 27 2021

[28] Hafiz Ahmad, IshraqAlam, Md. Manirul Islam, “Handwritten Digit Recognition System based on LRM and SVM Algorithm”, *International Conference on Engineering Research and Education School of Applied Sciences and Technology,* January 2019

[29] Chi Dianwei, “Handwritten Digit Recognition Application based on Improved Naïve Bayes Method”, *IEEE Conference Proceedings,* Vol. 2020, pp. 622-624, 2020

Appendix

A.1 MATLAB Codes

1. Perceptron Learning Algorithm:

```
% ----------Function for training of perceptron----------


function w=pla_1(w,x,t)
[~,c]=size(x); done=0; itr=1; n=1; error=0;
while ~done%iteration
    for i=1:c
        y=perceptron_out(w,x(:,i));%calculating output of perceptron
        w(1)=w(1)+n*(t(i)-y); %updating threshold
        w(2:end)=w(2:end)+n*(t(i)-y)*x(:,i);%updating weight
        decison_plot(x,w,t)% Plotting decision boundaries in input plane
        error=error+0.5*(t(i)-y)^2;%calculating mean square error
    end
    if (error==0)
        done=1;
        k=linspace(min(x(:))-1,max(x(:))+1);
        h=plot(k,-(w(2)*k+w(1))/w(3),'LineWidth',3,'color','black');% Final
decision boundary
        legend((h),{'Final decision boundary'})
    else
        error=0;
    end
     tcount(itr)=itr
    itr=itr+1;
end
end
```

2. Perceptron Pocket Learning Algorithm:

```
% ----------Function for training of perceptron----------

function [w_pocket,tcount,net_sum]=pocket_2(w,x,t,epoch)
tic;
set=size(x); itr=1; n=1; j=1; tcount=[];
while itr<epoch
    miscount=0;
    for i=1:set(2)
        [y,net]=perceptron_out(w,x(:,i));% Calculating output of perceptron
        if y~=t(i)
            miscount=miscount+1;% Number of missclassification
            w(1)=w(1)+n*(t(i)-y); %updating threshold
            w(2:end)=w(2:end)+n*(t(i)-y)*x(:,i);%updating weights
        end
        net_sum(itr,i)=net;
    end
    tcount(itr)=miscount;% storing missclassification for each set
    if miscount<=min(tcount)
        w_pocket=w;%storing best weight having minimum misclassification
```

```
    end
    itr=itr+1
    if miscount==0% loop terminating condition
        break
    end
    end
time2=toc;
end
```

3. Program for calculating output from perceptron:

```
% ----------Function for calculting output of perceptron----------

function [y,net]= perceptron_out(w,x)
dim=size(x);
x_in=[ones(1,dim(2));x];%Adding x0 value to input matrix
% oldparam = sympref('HeavisideAtOrigin',0);
net=w'*x_in;%calculating weighted sum
y=heaviside(net);
end
```

4. Program for linear regression:

```
% ----------Linear Regression using single step learning----------

function w=reg(x,t)
x_psuedo=(x'*x)\x';%Calculation of pseudo inverse of input
w=x_psuedo*t;
end
```

5. Program for linear Gradient descent:

```
% ----------Gradient Descent for linear hypothesis----------

function [theta,mis_count,y_prid,time]=grad_des(x,y,theta,alpha,itr)
tic;
m=length(y);
x=[ones(m,1) x];
for i=1:itr
h=x*theta;%calculating the hypothesis
theta=theta-alpha*(x'*(h-y)); %Updating the weight vector
y_prid=x*theta;% predicted values
end
time=toc;
end
```

6. Program for Training Binary Decision Tree:

```
% -------------Training of Binary Decision Tree-------------

function tree=BinTree(input,target)
    tree=fitctree(input,target); % Training of tree
end
```

7. Program for Prediction by Binary Decision Tree:

```
% -------------Prediction by Binary Decision Tree-------------

function labels=test(tree,testing_input)
    labels=predict(tree,testing_input); % Prediction on testing data
end
```

8. Program for finding the nearest classes by k-NN:

```
% -------------Nearest Classes decider by k-NN-----------------

function out=knn1(x_train,x_valid,t,k)
    i=1;
    while i<=size(x_valid,2)
       dist=sum((x_valid(:,i)-x_train).^2).^0.5;
        [~,ind]=sort(dist,2); % Sorting distances & indices in ascending
order of each respective sample point
        out(i,:)=t(1,ind(1,1:k)); % Finding the k smallest labels of
indices sorted
        i=i+1
    end
end
```

9. Program for finding the classified labels by k-NN:

```
% -----------Function for majority voting/selection of label-------------

function out_predict=pred(cls)
    edges = unique(cls);%Finding which labels are present once only
    for i=1:size(cls,1)
        counts(i,:) = histc(cls(i,:), edges);% Counting the number of times
each label is present respective to each query point
end
    [~,out_predict]=max(counts,[],2);% Classifying into label which is
occuring maximum times
end
```

10. Testbench for k-NN Algorithm:

```
tic
dataset=x;
query=x_t;
labels=t;
 k=1; % Defining number of nearest neighbour for consideration
indices=knn1(dataset,query,labels,k); % Calculating nearest classes
classification=pred(indices); % Performing classification based on majority
voting
log=classification==t_t'; % Comparing predictions with actual targets
prcnt(k)=sum(log(:))/size(query,2)*100; % Finding classification accuracy
time=toc;
```

11. Program for predicting output by Logistic Regression:

```
%-----------Calculating Output------------

function [y,E,s]=output(x,w,t,samples)
    s=w*x; % Calculating net sum
    y=sigmoid(s); % Applying activation function
    E=(-1/samples).*sum(t.*log(y)+(1-t).*log(1-y));%   Calculating   binary
cross-entropy loss
end
```

12. Program for training by GD on Logistic Regression:

```
% ----------Gradient Descent-----------

function [W,y,s,E]=GD(x,t,w,itr,alpha)
    dim=size(x);
    x=[ones(1,dim(2));x];
    for i=1:itr
        [y,E(1,i),s]=output(x,w,t,dim(2)); % Calculating output and error
        fe=round(E,4);
        if i>=30
        log=fe(1,i-29:i)==fe(1,i);   %   Loop   Termination   condition   for
negligible/slow error change
        if sum(log)==30
            break
        end
        end
        dw=derivative(t,y,x); % Calculating partial derivative of weights
w.r.t loss
        W=w-alpha*dw; % Updating weights
        w=W;
        i
    end
end
```

13. One-Vs-Rest classifiers using Logistic Regression:

```
% -----------One vs Rest Classifier for 10-Classes-------------

t_temp=[1 0 0 0 0 0 0 0 0 0];
for i=1:10
    if i~=1
    t_temp=circshift(t_temp,[0,1]); % Defining target for each classifier
    end
    t=repelem(t_temp,c);
    [w(i,:),y(i,:),s(i,:),E]=GD(x,t,[0  rand(1,784)],5000,0.02);  %  Calling
logistic unit for each classifier
end
```

14. Program for Training Naïve Bayes Classifier:

```
% -------------Training of Naïve Bayes Classifier-------------

function nb=NB(input,target)
    nb=fitcnb(input,target); % Training of classifier
end
```

15. Program for Prediction by NB classifier:

```
% -------------Prediction by NB-------------

function labels=test(nb,testing_input)
    labels=predict(nb,testing_input); % Prediction on testing data
end
```

16. Program for Calculating forward output of Artificial Neural Network:

```
% --------------------Function for forward flow of ANN--------------------

function [y,h1,h2,s1,s2,E]=fwrd3(x,w1,w2,w3,b1,b2,b3,t,samples,e)
s1=w1*x+b1;
h1=max(0.02*s1,s1); % Hidden layer-1 output
s2=w2*h1+b2;
h2=max(0.02*s2,s2); % Hidden layer-2 output
y=softmax(w3*h2+b3); % Final output
e=-sum(t.*log(y)); % Calculating categorical cross-entropy loss
E=sum(e)/samples;
end
```

17. Program for Backpropagation of Artificial Neural Network:

```
% -------------Calculating Gradients of weights (Backward flow)------------

function
[delw1,delw2,delw3,delb1,delb2,delb3]=bkwrd3(x,h1,h2,w2,w3,y,t,s1,s2,sample
s)
del_s1=max((0.02*s1)./abs(s1),s1./abs(s1));  % Derivative of activation
function of hidden layers
del_s2=max((0.02*s2)./abs(s2),s2./abs(s2));
del_s3=y-t; % Derivative of activation function of output layer+ Loss
k1=(w3'*del_s3).*del_s2;
k2=(w2'*k1).*del_s1;
One=ones(samples,1);
delw3=(del_s3*h2')/samples; % Gradient of output weight vector
delb3=(del_s3*One)/samples; % Gradient of outputlayer bias
delw2=(k1*h1')/samples; % Gradient of hiddenlayer weight vector
delb2=(k1*One)/samples; %Gradient of hiddenlayer bias vector
delw1=(k2*x')/samples;
delb1=(k2*One)/samples;
end
```

18. Program for Gradient Descent of ANN:

```
% -------------Gradient Descent Algorithm for ANN-------------

function
[w1_new,w2_new,w3_new,b1_new,b2_new,b3_new,E,y]=SGD(x,w1,w2,w3,b1,b2,b3,t,a
lpha,itr,samples)
e=zeros(1,samples);
for i=1:itr
    idx = randperm(size(x,2),samples); % Selecting mini-batch from input
randomly
    B = x(:,idx);
    T=t(:,idx);
    [y,h1,h2,s1,s2,E(1,i)]=fwrd3(B,w1,w2,w3,b1,b2,b3,T,samples,e);%
Calculating network output
```

```
[delw1,delw2,delw3,delb1,delb2,delb3]=bkwrd3(B,h1,h2,w2,w3,y,T,s1,s2,sample
s); % calculating gradients
    w1_new=w1-alpha*delw1;
    w2_new=w2-alpha*delw2;  %   Parameters update
    b1_new=b1-alpha*delb1;
    b2_new=b2-alpha*delb2;
    w3_new=w3-alpha*delw3;
    b3_new=b3-alpha*delb3;
    w1=w1_new;  w2=w2_new;  b1=b1_new;  b2=b2_new;  w3=w3_new;  b3=b3_new;
    fe=round(E,4);
    if i>=30
    log=fe(1,i-29:i)==fe(1,i); % Loop stoping criterion
    if sum(log)==30
        break
    end
    end
    i
end
end
```

19. Program for defining parameters of ANN:

```
% --------------Testbench for ANN----------------

tic;
dim=size(x);
alpha=0.02; itr=10000; % Learning rate & no. of iterations
samples=dim(2);
i_nodes=dim(1); % Defining number of input nodes
h2_nodes=10; % Defining number og hidden nodes
h1_nodes=20;
o_nodes=10; % Defining number of output nodes
avg=1e-5;
dev1=(1/(i_nodes+h1_nodes)).^0.5; % Calculating deviation for parameters
dev2=(1/(h1_nodes+h2_nodes)).^0.5;
dev3=(1/(o_nodes+h2_nodes)).^0.5;

w1=dev1.*randn(h1_nodes,i_nodes)+avg;  % Initializing starting weights and
biases
w2=dev2.*randn(h2_nodes,h1_nodes)+avg;
w3=dev3.*randn(o_nodes,h2_nodes)+avg;
b1=avg;zeros(h1_nodes,1);
b2=zeros(h2_nodes,1);
b3=zeros(o_nodes,1);
[w1_new,w2_new,w3_new,b1_new,b2_new,b3_new,E,y]=SGD(x,w1,w2,w3,b1,b2,b3,tra
in,alpha,itr,samples);
```

20. Miscellaneous:

```
% ------------One-hot Encoding--------------

o=labels;
prid=zeros(10,length(o));
for i=1:length(o)
k=o(1,i);
prid(k,i)=1;
end
```

```
% ------------Function for Sigmoid-------------

function y=sigmoid(x)
    y=1./(1+exp(-x));
end

% ------------Function for Leaky Relu--------------

function [y,dely]=leakyRelu(x)
y=max(0.02*x,x);
dely=max((0.02*x)./abs(x),x./abs(x));
end

%------------Program for plotting normal distribution plot-------------

for i=1:10
yc(i,:)=s(1,c*(i-1)+1:i*c);
end
dat=[min(yc,[],2)  max(yc,[],2)  mean(yc,2)  std(yc,[],2)];  % Calculating
statistical parameters of data
for i=1:10 % Looping for the each and every classes
    pd = makedist('Normal','mu',dat(i,3),'sigma',dat(i,4)); % Constructing
the probability distribution based on the mean and deviation for each
respective class
    g=sort(yc(i,:)); % Sorting into ascending order the netsum values for
each class
%     g=[-20:24];
    p=pdf(pd,g); % Calculating the probability density function for each
netsum point
    hold on
    if rem(i,2)==0
        plot(g,p,'-
o','LineWidth',1,'MarkerSize',6,'DisplayName','Class'+string(i));
    else
        plot(g,p,'-
*','LineWidth',1,'MarkerSize',6,'DisplayName','Class'+string(i));
    end
end
title('10 vs 5')
 legend('Location','best')
 ylabel('Normal Density')
 xlabel('Net Sum')
 hold off
```

A.2 Training Decision Trees data

1. Part of decision tree on MNIST:

```
1350  if x185<2.78498 then node 1960 elseif x185>=2.78498 then node 1961 else 2
1351  class = 9
1352  if x294<2.18216 then node 1962 elseif x294>=2.18216 then node 1963 else 3
1353  if x268<2.51212 then node 1964 elseif x268>=2.51212 then node 1965 else 8
1354  class = 9
1355  if x128<2.71518 then node 1966 elseif x128>=2.71518 then node 1967 else 7
1356  if x376<1.44608 then node 1968 elseif x376>=1.44608 then node 1969 else 8
1357  class = 4
1358  if x377<1.12247 then node 1970 elseif x377>=1.12247 then node 1971 else 9
1359  if x523<-0.356027 then node 1972 elseif x523>=-0.356027 then node 1973 else 4
1360  if x383<0.265828 then node 1974 elseif x383>=0.265828 then node 1975 else 3
1361  class = 9
1362  class = 3
1363  class = 4
1364  if x231<-0.387755 then node 1976 elseif x231>=-0.387755 then node 1977 else 9
```

```
1365  class = 6
1366  class = 8
1367  if x410<1.09708 then node 1978 elseif x410>=1.09708 then node 1979 else 9
1368  class = 9
1369  class = 5
1370  class = 4
1371  class = 2
1372  class = 10
1373  class = 4
1374  if x464<2.17581 then node 1980 elseif x464>=2.17581 then node 1981 else 5
1375  if x408<0.671937 then node 1982 elseif x408>=0.671937 then node 1983 else 10
1376  if x490<0.183337 then node 1984 elseif x490>=0.183337 then node 1985 else 9
1377  class = 8
1378  if x96<1.52857 then node 1986 elseif x96>=1.52857 then node 1987 else 9
1379  if x220<0.259482 then node 1988 elseif x220>=0.259482 then node 1989 else 9
1380  class = 7
1381  class = 5
1382  class = 6
1383  if x665<2.75959 then node 1990 elseif x665>=2.75959 then node 1991 else 9
1384  class = 9
1385  if x153<-0.108554 then node 1992 elseif x153>=-0.108554 then node 1993 else 2
1386  if x349<1.86488 then node 1994 elseif x349>=1.86488 then node 1995 else 9
1387  class = 3
1388  if x93<-0.298918 then node 1996 elseif x93>=-0.298918 then node 1997 else 5
1389  if x271<-0.152973 then node 1998 elseif x271>=-0.152973 then node 1999 else 6
1390  if x380<-0.165664 then node 2000 elseif x380>=-0.165664 then node 2001 else 6
1391  if x266<0.475228 then node 2002 elseif x266>=0.475228 then node 2003 else 8
1392  class = 7
1393  class = 3
1394  if x351<0.43081 then node 2004 elseif x351>=0.43081 then node 2005 else 6
1395  if x293<-0.102209 then node 2006 elseif x293>=-0.102209 then node 2007 else 10
1396  class = 9
1397  class = 4
1398  if x214<0.399083 then node 2008 elseif x214>=0.399083 then node 2009 else 5
1399  class = 6
1400  if x572<-0.292573 then node 2010 elseif x572>=-0.292573 then node 2011 else 7
1401  class = 9
1402  if x105<2.13139 then node 2012 elseif x105>=2.13139 then node 2013 else 7
1403  class = 3
1404  class = 3
1405  class = 4
1406  class = 3
1407  class = 4
1408  class = 1
1409  if x181<0.0881549 then node 2014 elseif x181>=0.0881549 then node 2015 else 9
1410  if x107<0.475228 then node 2016 elseif x107>=0.475228 then node 2017 else 6
1411  class = 8
1412  if x429<2.79132 then node 2018 elseif x429>=2.79132 then node 2019 else 5
1413  class = 3
1414  class = 6
1415  class = 8
1416  if x212<-0.121245 then node 2020 elseif x212>=-0.121245 then node 2021 else 8
1417  class = 3
1418  if x602<0.583101 then node 2022 elseif x602>=0.583101 then node 2023 else 9
1419  class = 8
1420  if x177<-0.108554 then node 2024 elseif x177>=-0.108554 then node 2025 else 5
```

The above data is showing part of training of huge decision tree implemented on MNIST dataset. The whole decision tree contains 4,353 number of nodes of which only 70 nodes are shown. The conditions are applied on predictors/features of MNIST in each row.

2. Part of decision tree on Arabic Dataset:

```
782  if x259<254 then node 1058 elseif x259>=254 then node 1059 else 6
783  class = 6
784  if x549<34 then node 1060 elseif x549>=34 then node 1061 else 3
785  if x238<1.5 then node 1062 elseif x238>=1.5 then node 1063 else 5
786  class = 9
787  if x242<3 then node 1064 elseif x242>=3 then node 1065 else 1
788  class = 8
789  if x233<189 then node 1066 elseif x233>=189 then node 1067 else 10
790  class = 5
791  class = 1
792  if x688<1.5 then node 1068 elseif x688>=1.5 then node 1069 else 9
793  if x178<2.5 then node 1070 elseif x178>=2.5 then node 1071 else 8
794  if x654<17 then node 1072 elseif x654>=17 then node 1073 else 6
795  class = 3
796  if x324<250.5 then node 1074 elseif x324>=250.5 then node 1075 else 6
797  class = 9
798  if x244<61.5 then node 1076 elseif x244>=61.5 then node 1077 else 8
799  class = 3
800  class = 9
801  class = 1
802  class = 3
803  class = 6
804  class = 4
805  class = 6
806  class = 6
807  class = 1
808  class = 6
809  class = 1
810  class = 6
811  class = 1
812  if x326<120 then node 1078 elseif x326>=120 then node 1079 else 2
813  if x322<11.5 then node 1080 elseif x322>=11.5 then node 1081 else 1
814  class = 5
815  if x288<28 then node 1082 elseif x288>=28 then node 1083 else 2
816  if x123<4 then node 1084 elseif x123>=4 then node 1085 else 4
817  class = 1
818  class = 2
819  class = 9
820  class = 5
821  class = 10
822  if x236<36 then node 1086 elseif x236>=36 then node 1087 else 7
823  class = 5
824  class = 6
825  class = 1
826  if x345<108.5 then node 1088 elseif x345>=108.5 then node 1089 else 10
827  if x522<4.5 then node 1090 elseif x522>=4.5 then node 1091 else 4
828  if x353<108.5 then node 1092 elseif x353>=108.5 then node 1093 else 2
829  if x372<0.5 then node 1094 elseif x372>=0.5 then node 1095 else 5
830  class = 2
831  if x462<157.5 then node 1096 elseif x462>=157.5 then node 1097 else 1
832  if x436<105 then node 1098 elseif x436>=105 then node 1099 else 5
833  if x372<26 then node 1100 elseif x372>=26 then node 1101 else 4
834  class = 2
835  class = 5
836  class = 2
837  class = 1
838  class = 4
839  class = 10
840  class = 10
841  class = 1
842  if x445<178 then node 1102 elseif x445>=178 then node 1103 else 1
843  class = 6
844  if x551<147 then node 1104 elseif x551>=147 then node 1105 else 4
845  class = 5
846  if x570<81 then node 1106 elseif x570>=81 then node 1107 else 1
847  class = 9
848  class = 4
849  class = 1
850  if x464<69.5 then node 1108 elseif x464>=69.5 then node 1109 else 1
851  if x377<243.5 then node 1110 elseif x377>=243.5 then node 1111 else 1
852  if x156<13 then node 1112 elseif x156>=13 then node 1113 else 7
```

3. Decision Tree on Self-made dataset:

```
 1  if x3<5.5 then node 2 elseif x3>=5.5 then node 3 else 1
 2  if x15<1.5 then node 4 elseif x15>=1.5 then node 5 else 1
 3  if x1<0.5 then node 6 elseif x1>=0.5 then node 7 else 9
 4  if x10<1.5 then node 8 elseif x10>=1.5 then node 9 else 10
 5  if x5<2.5 then node 10 elseif x5>=2.5 then node 11 else 3
 6  class = 9
 7  class = 2
 8  if x11<2.5 then node 12 elseif x11>=2.5 then node 13 else 5
 9  if x10<3.5 then node 14 elseif x10>=3.5 then node 15 else 10
10  class = 1
11  if x14<2.5 then node 16 elseif x14>=2.5 then node 17 else 7
12  if x13<1.5 then node 18 elseif x13>=1.5 then node 19 else 5
13  if x16<2 then node 20 elseif x16>=2 then node 21 else 8
14  if x16<2 then node 22 elseif x16>=2 then node 23 else 10
15  class = 8
16  if x6<3.5 then node 24 elseif x6>=3.5 then node 25 else 4
17  if x12<2.5 then node 26 elseif x12>=2.5 then node 27 else 7
18  class = 2
19  if x16<2 then node 28 elseif x16>=2 then node 29 else 5
20  class = 8
21  class = 6
22  if x14<2.5 then node 30 elseif x14>=2.5 then node 31 else 10
23  class = 4
24  if x10<3.5 then node 32 elseif x10>=3.5 then node 33 else 6
25  if x5<3.5 then node 34 elseif x5>=3.5 then node 35 else 4
26  if x10<2.5 then node 36 elseif x10>=2.5 then node 37 else 7
27  if x5<3.5 then node 38 elseif x5>=3.5 then node 39 else 9
28  class = 5
29  class = 2
30  if x11<3.5 then node 40 elseif x11>=3.5 then node 41 else 10
31  class = 1
32  if x3<0.5 then node 42 elseif x3>=0.5 then node 43 else 3
33  class = 6
34  class = 2
35  if x3<0.5 then node 44 elseif x3>=0.5 then node 45 else 4
36  if x4<2.5 then node 46 elseif x4>=2.5 then node 47 else 7
37  class = 3
38  class = 1
39  class = 9
40  if x11<1.5 then node 48 elseif x11>=1.5 then node 49 else 10
41  class = 8
42  class = 2
43  if x13<2.5 then node 50 elseif x13>=2.5 then node 51 else 3
44  class = 8
45  if x4<6.5 then node 52 elseif x4>=6.5 then node 53 else 4
46  class = 5
47  if x6<4.5 then node 54 elseif x6>=4.5 then node 55 else 7
48  class = 7
49  if x7<4 then node 56 elseif x7>=4 then node 57 else 10
50  if x12<2.5 then node 58 elseif x12>=2.5 then node 59 else 3
51  if x4<3.5 then node 60 elseif x4>=3.5 then node 61 else 6
52  class = 4
53  class = 9
54  if x8<0.5 then node 62 elseif x8>=0.5 then node 63 else 7
55  class = 3
56  class = 10
57  class = 1
58  if x2<3 then node 64 elseif x2>=3 then node 65 else 3
59  class = 4
60  class = 6
61  class = 4
62  class = 7
63  class = 2
64  if x8<0.5 then node 66 elseif x8>=0.5 then node 67 else 3
65  class = 1
66  class = 3
67  class = 2
```

A.3 Raw Self-Made Dataset:

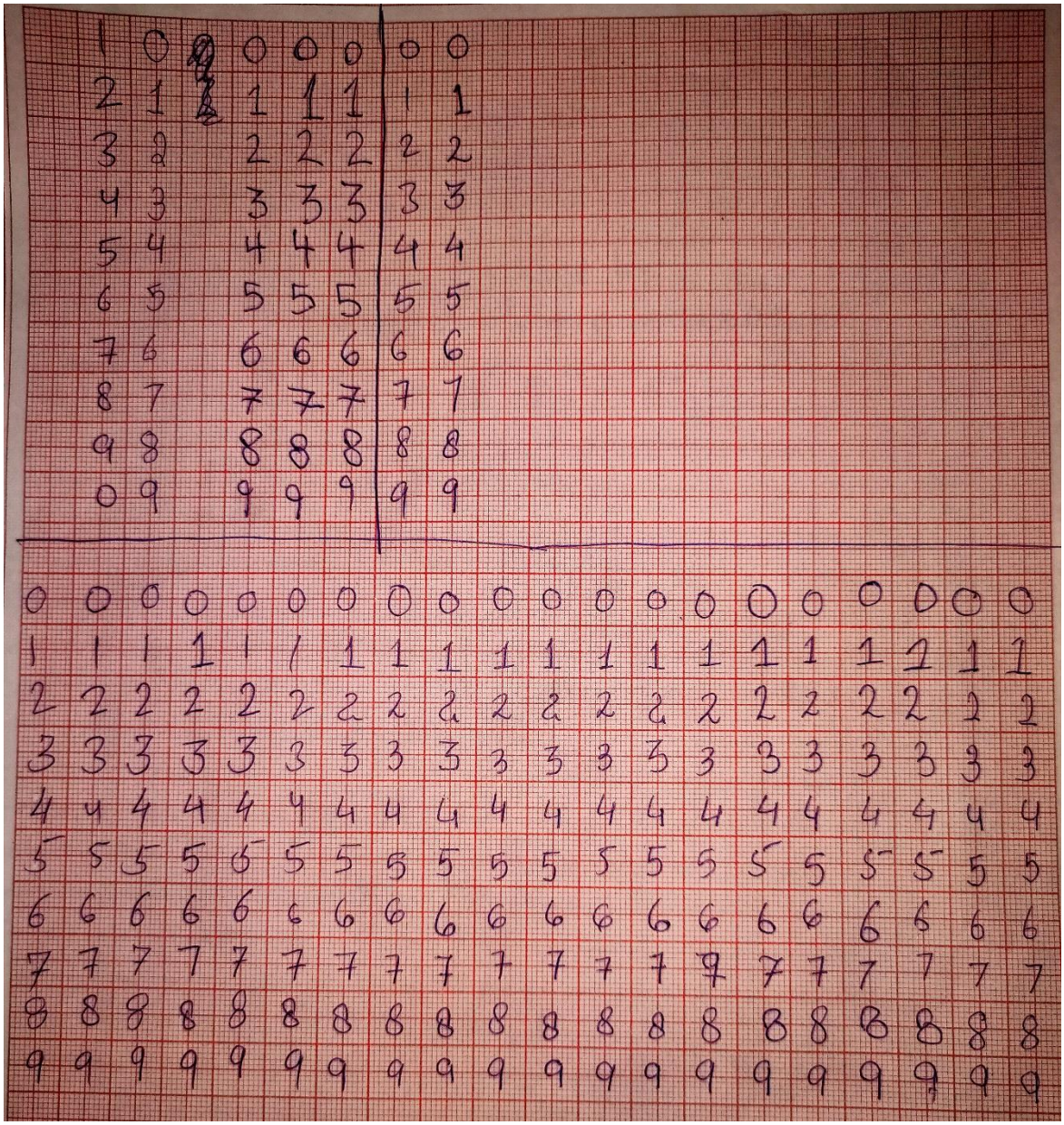


Fig A.1 Snapshot of raw data generated by hand of six different samples

Each digit is written in a box of graph paper having 10×10 grid lines. Each box contains 100 smaller squares. The dataset is generated by taking the horizontal and vertical densities of presence of ink. Horizontal densities are calculated by counting the number of index of smallest boxes from left to right thereby extracting a vector of eight elements. Vertical densities are extracted in a similar fashion except now the counting is done from top to bottom. Now a sample of digit is constructed by appending horizontal and vertical densities respectively; now each sample is a vector of size 16×1.

17 Dissertation Mohd Aqram Khan GF1116 Electronics Engg AMU

ORIGINALITY REPORT

8%	6%	3%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.tudelft.nl Internet Source	2%
2	dzone.com Internet Source	1%
3	www.kdnuggets.com Internet Source	<1%
4	machinelearningmastery.com Internet Source	<1%
5	dokumen.pub Internet Source	<1%
6	Zhou, Y.. "Fault detection and classification in chemical processes based on neural networks with feature extraction", ISA Transactions, 200310 Publication	<1%
7	hdl.handle.net Internet Source	<1%
8	docplayer.net Internet Source	

		<1%
9	Iamsa-at, Suthasinee, and Punyaphol Horata. "Handwritten Character Recognition Using Histograms of Oriented Gradient Features in Deep Learning of Artificial Neural Network", 2013 International Conference on IT Convergence and Security (ICITCS), 2013. Publication	<1%
10	Submitted to Universidad Politécnica de Madrid Student Paper	<1%
11	etd.aau.edu.et Internet Source	<1%
12	Gabriela Souza de Melo. "Winograd schemas in portuguese.", Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2020 Publication	<1%
13	serokell.io Internet Source	<1%
14	www.sas.com Internet Source	<1%
15	towardsdatascience.com Internet Source	<1%
16	Submitted to Cranfield University	

	Student Paper	<1%
17	Submitted to Higher Education Commission Pakistan Student Paper	<1%
18	juniperpublishers.com Internet Source	<1%
19	www.exhibit.xavier.edu Internet Source	<1%
20	Submitted to Arab Open University Student Paper	<1%
21	Submitted to Liverpool John Moores University Student Paper	<1%
22	citeseerx.ist.psu.edu Internet Source	<1%
23	medium.com Internet Source	<1%
24	Submitted to University of Oxford Student Paper	<1%
25	DalSpace.library.dal.ca Internet Source	<1%
26	Kyungchan Son, Jaegak Lee, Haejin Hwang, Wonseok Jeon, Hyunseok Yang, Il Sohn, Younghwan Kim, Hyungsic Um. "Slag Foaming	<1%

Estimation in the Electric Arc Furnace using Machine Learning Based Long Short-Term Memory Networks", Journal of Materials Research and Technology, 2021
Publication

27	Muhammad Awais Shafique, Eiji Hato. "Use of acceleration data for transportation mode prediction", Transportation, 2014 Publication	<1 %
28	sylvainchevalier.free.fr Internet Source	<1 %

Exclude quotes	Off	Exclude matches	< 14 words
Exclude bibliography	Off		