Using Machine Learning on a Diverse Class of Problems: from Rainfall to Criminal Actions

Thesis submitted in partial fulfilment of the requirement for the degree of

Bachelor of Science
In
Computer Science

Under the Supervision of

Abu Mohammad Hammad Ali

By

Sirajum Munira (12101038),

Tonmona Tonny Roy (12101039),

Tasmia Rahman (15341031),

Md.Aquib Javed (15341032)

School of Engineering & Computer Science

Department of Computer Science & Engineering

BRAC University

Declaration

This is to certify that the research work titled "Using Machine Learning on a Diverse Class of Problems: from Rainfall to Criminal Actions" is submitted by Sirajum Munira, Tonmona Tonny Roy, Tasmia Rahman and Md.Aquib Javed to the Department of Computer Science & Engineering, BRAC University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science. We hereby declare that this thesis is based on results obtained from our own work. Due acknowledgement has been made in the text to all other material used. This thesis, neither in whole nor in part, has been previously submitted to any other University or Institute for the award of any degree or diploma.

Signature of Supervisor:

Abu Mohammad Hammad Ali

Department of Computer Science and Engineering

BRAC University

Signature of Authors:

Sirajum Munira, 12101038

Tonmona Tonny Roy, 12101039

Tasmia Rahman, 15341031

Md. Aquib Javed, 15341032

Acknowledgements

We would like to express our utmost gratitude to Mr.Abu Mohammad Hammad Ali for giving us the opportunity to work under his supervision and providing us with great advice and support. His guidance, teachings and knowledge have kept us motivated throughout the entire curriculum, especially during this thesis. We are also grateful to Coursera and Dr. Andrew Ng for the free online resources on Machine Learning.

Abstract

We intend to compare and analyze certain machine learning algorithms by taking two different datasets tackling separate real world issues. The first one relates to agriculture. Bangladesh, being an agrarian country, is heavily dependent on rain. Being able to predict the rainfall amount accurately would enable successful and sustainable production. Machine learning algorithms can also help predict the category of crimes in a particular area. This will enable law enforcers in a certain region to predict and categorize recent crimes based on past incidents. Our aim was to approach these problems by labelling and processing the data and then comparing the results of different cost functions.

Contents

List of Figures

Chapter 1

Introduction

Machine learning performs the task of detecting certain patterns from data automatically often from a large data set. Based on its newly learned patterns, we can then predict outputs from future data [1]. For instance, a computer program that plays checkers, can be trained to learn from its own experience and improve. If we denote the experience of playing against itself with E, the performance measure with P and the task with T, we can say that the program can learn from experience E, with respect to the task T and performance P [2]. Machine learning fits in complex problems where a specific algorithm is not the answer and where observation of events is required.

One of the pioneers of machine learning was Arthur Samuel. In his words, machine learning is a “Field of study that gives computers the ability to learn without being explicitly programmed” [3]. He built a checkers playing program and made it play 10,000 games against itself and during this he made the program determine which positions were favourable or not depending on wins or losses.

Enhancing predictive ability is one of the core tenets of machine learning. This can be achieved by creating or sometimes modifying the existing algorithms, thus enabling them to process more data. The current paper presents an attempt at solving two distinct problems using regression, classification and neural networks and compares the results obtained.

1.1 Motivation

Machine learning algorithms have facilitated scientific advancements in numerous ways throughout the world. Bangladesh is yet to be familiarized with it. In this paper, we propose machine learning approaches in predicting rainfall measurements in the agriculture, and classify crimes in a certain region to categorize and predict curved data based on past events. So far, data from radar values have been used to estimate rainfall amounts. In our research, we took polarimetric radar data. Since smaller drops tend to evaporate more, by using polarimetric radar data we will be able to infer raindrop sizes more accurately. Applying learning algorithms will thus make the prediction better than the existing ones. Adding to this, categorizing crimes based on time and place will effectively assist a program meant to predict the type of crimes.

1.2 Goals

Our primary aim is to implement machine learning algorithms and compare the results in order to select which ones perform the best in the two separate situations. Using our rain data, we would like to build an intelligent system that would be able to predict hourly rainfall. The agricultural system in Bangladesh has not been facilitated with sufficient technological advancement. For measuring rainfall, we are highly dependent on rain gauges which often leads to an incorrect result. Our proposal for the first problem is thus to get values from radar data, analyse the results of the algorithms and use the one that performs the best. This will not only be useful for agriculture, it will also help environment specialists gain correct information to tackle natural disasters like drought.

In addition to this, our country has a trade record with criminal activities. Crime records have been recorded manually thus far, and does not have much technological scope for classifying them. Using the best learning algorithms for this problem, our approach would be to classify these crimes based on their time and place in a certain region. This would enable us to predict the category of a crime given a specific time and location.

Chapter 2

Literature Review

2.1 Related Work

Estimation of rainfall has been done in Bangladesh by a few researchers in the past few years. Most of these work encompassed rainfall measurement and characterizing rainfall trends [4]. In order to quantify the precipitation over Bangladesh, the Tropical Rainfall Measuring Mission (TRRM) satellite coupled with the 3B42 algorithm have been applied using remote sensing data [5]. Most of the data have been collected using rain gauges up until a few years ago, when remotely sensed radar- data started emerging as an option. For our research, we have used data provided by NEXRAD and MADIS. The data consists of polarimetric data and using these values will also be convenient because of their two sided orientation, one having horizontal and the other having a vertical component.

On the other hand, Bangladesh is yet to experience sufficient research on criminal records. Some of the algorithms that have been used for analysing crime patterns are

Linear Regression, Additive Regression and Decision Stump algorithms. Upon further research, it has also been found that Linear Regression performed better than the other ones [6].

2.2 Introduction to Machine Learning

Machine learning can be of several categories.

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Recommender systems

2.2.1 Supervised Learning

Supervised learning is the most widely applied form of machine learning. It can be defined as the process when the data is labelled, in other words, when the output variable is known. Also, the predictors and responses are present in the data, the features are known and this information is used during the training of the data. [1]

For example, let's say we have a basket full of marbles with different colours. We have a program that would sort same coloured marbles together. From our previous work, we have already instructed the program about different colours (i.e. RGB values, hue, saturation etc.) and they are acquainted with these colours. This task is performed by response variables, whose task is to instruct that if a marble has a specific colour value, it should be categorized as a specific colour. This is a type of supervised learning[7].

2.2.1.1 Classification

Classification organizes the given data in distinct classes by first modelling the data. This supervised learning technique assigns instances to previously defined classes. If we have inputs x and outputs y and a set of classes C, then the goal of classification is to learn a mapping from x to y, where y G 1,... ,C. Function approximation is used in classification problems. For this, we assume y = f(x), where f is an unknown function. On a given labelled set, our aim is to estimate the function f and then make predictions. Furthermore, classification also tries to predict on a completely new set of inputs, otherwise known as novel inputs.

Let's take an example where we have to determine breast cancer to be malignant or benign based on the size of a tumour. In this case, we will have to classify the data into two specific classes, it will be either malignant or benign. For the output, we can have a discrete number of possible values. We can also take one particular attribute, plot data and based on it can differentiate the results, or segment them into groups. Similarly, by taking other attributes and making classes, we can assign different individuals to these classes [8].

2.2.1.2 Regression

Regression is used to predict continuous values, meaning we try to estimate continuous valued output. Contrary to classification, in regression instead of determining the class we want to derive a numeric function.[9]

A real example for regression would be to predict the next day's stock price if the current market conditions along with other supporting information are given. The key difference here is that we are not determining which class the stock price or any other attribute should belong to, determining if something belongs to a class or not. Rather, we are predicting a numeric value. For instance, suppose there is a particular video on YouTube, where classification would determine whether children below the age of 10 can watch it or not, regression would predict the age of the viewer [1]. Since our problem is predicting the amount of rainfall, it is possible to apply regression.

2.2.1.3 Neural Networks

Neural Networks is one of the most powerful machine learning algorithms. They attempt to fit derived parameters given a training set [10], and are found to be significantly efficient in recognizing patterns. Neural Networks can be seen as simplified models of human brains. The brain has neurons and synapses. These synapses perform the task of connecting the neurons. These neurons can be thought of as Booleans while the synapses can be signified as small numbers ranging between -1 to 1.
For our problem, we will be focusing on the classification problems of Neural Networks.

2.2.2 Unsupervised Learning

Unsupervised learning is the opposite of supervised learning, where the outcome variable is unknown. It means the data is not labelled, so the program will have to extract values and information from the given data by itself. Unsupervised learning algorithms can be used for clustering, reduction and visualization, but in general it is not used for prediction. Finding interesting structure in the data is the concern here, which is also sometimes referred to as knowledge discovery [1].

If we take the example of coloured marbles in a basket like we did in the case of supervised learning, this time we would not know anything about the marbles. The program will have to take a specific example, categorize it and repeat for all the marbles. The program, unlike the previous one, does not know anything from past experience. The training data and response variable will also be absent in this case.

Chapter 3

Methodology

3.1 Data Collection

Rain gauges have been used to measure the rain for a certain location, however they have some drawbacks. The primary reason is that they cannot be placed everywhere. Also, the data radars that are used to estimate rainfall do not match the measurements from rain gauges. For this reason, we have used polarimetric data [11]. This will prove to be more effective because polarimetric data consists of horizontal and vertical orientations, so they provide higher quality data. Our data consists of NEXRAD and MADIS data collected between April to August 2014. This was collected from Midwestern corn-growing regions. For our crime classification problem, we collected data from SFPD Crime Incident Reporting system, provided by SF Open Data, with a range of around twelve years. A successful implementation and analysis of the selected algorithms on both these data set has the potentiality to help us in societal and economical ways. Both the data, besides their original sources, have been collected from kaggle.com.

3.2 Data Description

The data for rainfall estimation consists of multiple radar observations over the course of hours. There are multiple rows with the same ID. The columns are as follows:

TimeToEnd: How many minutes before the end of the hour was this radar observation?

DistanceToRadar: Distance between radar and gauge. This value is scaled and rounded to prevent reverse engineering gauge location

Composite: Maximum reflectivity in vertical volume above gauge

HybridScan: Reflectivity in elevation scan closest to ground

HydrometeorType: One of nine categories in NSSL HCA. See presentation for details

Kdp: Differential phase

RR1: Rain rate from HCAbased algorithm

RR2: Rain rate from Zdrbased algorithm

RR3: Rain rate from Kdpbased algorithm

RadarQualityIndex: A value from 0 (bad data) to 1 (good data)

Reflectivity: In dBZ

ReflectivityQC: Qualitycontrolled reflectivity

RhoHV: Correlation coefficient

Velocity: (aliased) Doppler velocity

Zdr: Differential reflectivity in dB

LogWaterVolume: How much of radar pixel is filled with water droplets?

MassWeightedSD: Standard deviation of drop size

Expected: the actual amount of rain reported by the rain gauge for that hour

The dataset for crime classification is derived from San Francisco Police Department Crime Incident Reporting System. The data fields are as follows:

Dates: timestamp of the crime incident

Category: category of the crime incident (only in train.csv). This is the target variable you are going to predict

Descript: detailed description of the crime incident (only in train.csv)

DayOfWeek:the day of the week

PdDistrict : name of the Police Department District

Resolution: how the crime incident was resolved (only in train.csv)

Address: the approximate street address of the crime incident

X: Longitude

Y: Latitude

3.3 Data Processing

- In both of the data sets, in multiple occurrences the data was separated by commas and we have taken comma separated values.

- When faced with multiple data on a single column, we separated them based on spaces. For instance, the first data separated was placed on the first row, the second one on second row and likewise. In this way, we divided a row into multiple rows. This is because we wanted to do vectorised implementation.

- In our rain data, the rightmost column contains the value for rainfall amount. While running the algorithms, that rightmost column was the Y, in other words, the output. Moreover, in the data there were some features with errors and nan values. The reasons that can be attributed to these errors are echo below signal-to-noise threshold, beam blockage and pixel at edge of echo. Since they cannot be processed and there is no way to get those data, we have omitted those columns. The remaining columns that were not omitted are X (features).
- For the crime classification problem, the data was divided into categories. However, they were string values. So we had to convert them into numeric values.
- The data for crime classification has a column where the timeframe is divided in dd-mm-yy and then the hour, minute and second with a space in between. We placed the year, month and date on different columns, meaning we have used them as separate features.
- The two rightmost columns on the crime data are X and Y, which are representing the locations. So, while running the algorithms, these two were given higher priority.

3.3.1 Mean Normalization

To make gradient descent work well some practical tricks are usually followed. One of them is mean normalization. The idea is to make features have zero mean[12]. The steps are as follows:

- Take a feature x_i

- Replace it by $(xi - mean)/max$

$$x_i = (x_i - \mu_i)/s_i$$

μ_i	is the average value of x_i in training set
s_i	range value of the feature $(maximum - minimum)$

3.4 Applying Algorithms

3.4.1 Gradient Descent

Gradient Descent is a numerical optimization algorithm that is reliable for minimizing differentiable functions. Beginning with a primary set of parameter values, gradient descent iteratively moves toward the values that minimize the cost function and other functions mainly in linear regression. Using a calculus approach, the function takes steps in a negative direction [13].

3.4.1.1 Linear Hypothesis

Gradient Descent is a way to automatically improve the hypothesis function. The Linear Hypothesis for Linear Regression is

$$h_\theta(x) = 1/(1 + e^{-\theta^T X})$$

3.4.1.2 Cost Function

Cost function measures the accuracy of hypothesis function.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

The variable used are:

m	The number of training examples
$x^(i)$	The input vector of the ith training example
$y^(i)$	The output vector of the ith training example
θ	The chose parameter values or weights
$h_\theta(x^{(i)})$	The algorithm's prediction for the i th training example using the parameters θ

Cost function lets us figure out how to fit the best straight line to our data. If the model yields a poor result, the cost becomes higher. Hence, the objective is to find the parameters θ which give the minimum possible cost J. In order to solve a minimization problem we need to minimize $(h_\theta(x) - y)^2$ for each sample and sum this over the training set [10].

3.4.1.3 Gradient Descent Algorithm

The cost minimization is done by solving the values of theta which sets the derivative to zero. This can be done analytically with calculus and algebra. However, in case of complex functions it is done by gradient descent. The principal gradient descent is an iterative algorithm that uses theta and the first derivative. The main algorithm is as follow:

```
repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
        (for $j = 1$ and $j = 0$)
}
```

where alpha is the learning rate that decides how big or small the step size would be towards the convergence. If alpha is too small the algorithm takes baby steps and will process very slowly. However, if it is too large the algorithm might not converge and might even diverge. Hence, it is very critical to select the right learning rate [14].

The following two figures illustrate why this might occur. In the first figure, α is sufficiently small so each iteration in the algorithm results in a step towards the minimum, resulting in convergence of the algorithm.

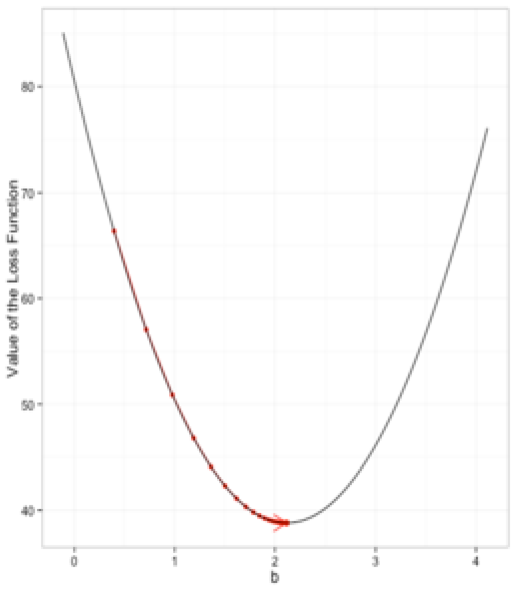


Figure 3.1: when α is too small

In the second figure, α is too large and each subsequent iterate increasingly overshoots the minimum, resulting in divergence of the algorithm [15].

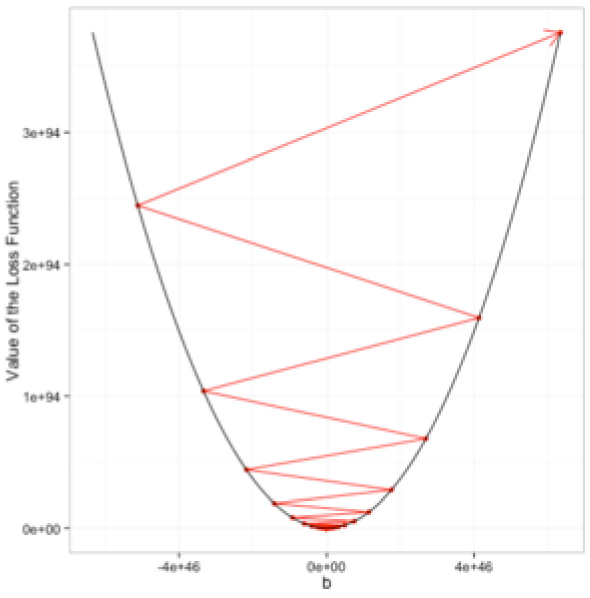


Figure 3.2: when α is too large

The above algorithm is processed for the value of θ_0 and θ_1 and simultaneously update both of them. By simultaneously, it is meant that we have to compute the right hand side for both θ_0 and θ_1. Then we can update θ_0 and θ_1 at the same time.

$$
\begin{aligned}
&\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
&\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
&\theta_0 := \text{temp0} \\
&\theta_1 := \text{temp1}
\end{aligned}
$$

The derivative term gets smaller as the value approaches the global minimum even with alpha is fixed. So there is no need to change alpha over time [10].

3.4.1.4 Gradient descent for multi variable

The function for gradient descent is

$$J(\theta_1, \theta_2) = {\theta_1}^2 + {\theta_2}^2$$

Our goal is to minimize the cost function to

$$\min_{\theta_1, \theta_2} J(\theta_1, \theta_2)$$

This function uses two variables. However, in case of multiple features we need to have more parameters. Considering this, for the hypothesis can be written as

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Instead of treating J only as a function, $J()$ is now written as a parameter function $J(\theta)$. The algorithm now is done through a simultaneous update of every θ_j value

```
Repeat {
    $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
        (simultaneously update $\theta_j$ for
        $j = 0, \ldots, n$) }
```

[14]

3.4.2 Classification

In binary classification, the data falls into two categories, the positive (1) and the negative (0). Logistic regression is then used to set the threshold. However, logistic regression generates a value which is always either 0 or 1 for Y. But hypothesis can give values greater than 0 or less values than Therefore, the algorithm needs to be developed so that it can give values between 0 and 1 [10].

3.4.2.1 Hypothesis

In case of linear regression the hypothesis was,

$$h_\theta(x) = (\theta^T x)$$

For classification hypothesis representation,

$$h_\theta(x) = g((\theta^T x))$$

where

$$g(z) = 1/(1+e^{-z})$$

This is the sigmoid or the logistic function. If we combine these equations we can write out the hypothesis as

$$h_\theta(x) = 1/(1+e^{-\theta^T x})$$

3.4.2.2 Interpreting hypothesis output

When the hypothesis $(h_\theta(x))$ generates a number, that value is treated as the estimated probability that $y = 1$, given x and parameterized by theta.

X is the feature vector and $h_\theta(x)$ gives probability

$$h_\theta(x) = P(y = 1|x;\theta)$$

Since this is a binary classification task we know $y = 0$ or 1
So the following must be true

- $P(y = 1|x;\theta) + P(y = 0|x;\theta) = 1$
- $P(y = 0|x;\theta) = 1 - P(y = 1|x;\theta)$[1]

3.4.2.3 One vs All

In binary classification using logistic regression, the positive and negative classes are separated by a straight line. By using ‘one vs all’, sometimes called ‘one vs rest’ classification the same idea can be applied for multiclass classification.

For example if there arc three classes, it can be used as three separate classification problem and can be defined in numerical order. Thus a fake training set is created where one set is put in a positive class and the rest in the negative classes. The positive class is assigned the value 1 and the negative class is assigned value 0. Now a standard logistic regression crossfire is trained which will give a positive boundary.

For each training set a classifier $h_\theta^{(i)}(x)$ is fit

The superscript i here represents the class which is currently considered as the positive class and the rest of the class will be considered as negative at that time. Thus a logistic regression classifier $h_\theta^{(i)}(x)$ is trained for each class i in order to determine what is the probability that y is equal to class i given x and prioritize by theta. The other classes are processed in the same way.

$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1,2,3,...)$$

Finally, when a new input x is given to make a prediction, all i number of classifiers is run on the input x and then the class i can be chosen that maximizes all of them[16]. So, whichever value of i gives the highest probability, y can be predicted to be that value.

$$\max_{(i)} h_\theta^{(i)}(x)$$

finally to make a prediction when we're given a new input$x$, we just run all i number of classifiers on the input x and we then pick the class i that maximizes the three.[10].

3.4.2.4 Cost Function

As previously stated, linear regression uses the following function to measure theta.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Instead of writing the squared error term, we can write,

$$\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

We can redefine the cost function as,

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

Though this cost function worked well for linear regression, this particular function yields a non-convex function of the parameter's data for logistic regression since our hypothesis function has a non-linearity (sigmoid function of $h_\theta(x)$). Hence, if gradient descent is run on this sort of function, it is not guaranteed to converge to the global minimum.

To solve this, a new logistic regression cost function is formed:

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Here, when the prediction is right, the cost function is 0. Else it slowly increases cost function as it becomes more inaccurate

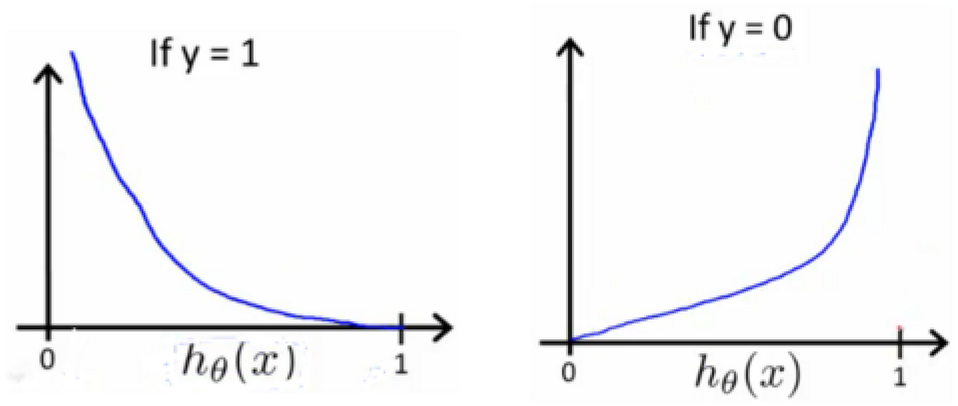


Figure 3.3: Cost function representation when $y = 1$ and $y = 0$

Here X axis is what we predict and Y axis is the cost associated with the prediction. If $y = 1$ and $h_\theta(x) = 1$ and if hypothesis predicts exactly 1 then the cost is equal to zero. As $h_\theta(x)$ goes toward 0, cost goes to infinity.
If $y = 0$, then cost is evaluated as $-log(1 - h_\theta(x))$ and yields inverse of the other function.
Since y is always either 0 or 1, the cost function can be re-written in a simpler way

$$cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)log(1 - h_\theta(x))$$

When $y = 1$, the equation simplifies to $log(h - \theta(x))$ which is stated above when $y = 1$. Now, if $y = 0$, the equation simplifies to $-log(1 - h_\theta(x))$ which also turns out to be the same as above when $y = 0$.

Finally, the cost function for θ parameters can be defined as

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

3.4.3 Neural Network

The motivation behind the idea of ”neural network” is identifying the computational difference between a human brain and digital computers. The specialty of the brain is not limited to being able to perform highly complex non-linear computations, but also the ability to perform parallel information processing. For example, human brain performs complex computations such as pattern recognition, motor control etc. As a matter of fact, the ability to build up its own rules through experiences is the reason behind the human brain’s ability to do complex computations. That human brain learns from experiences is the central idea behind neural networks[17]. Following is a formal definition of neural netwok:

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. According to Aleksander and Morton (1990), it resembles the brain in two respects:

1. Knowledge is acquired by the network from its environment through a learning process.

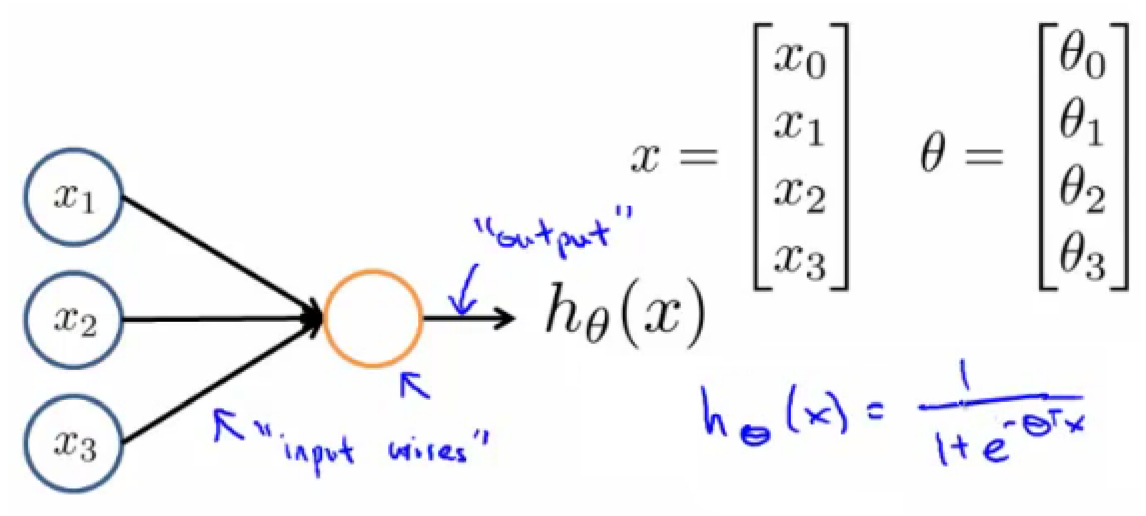
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Artificial neural networks are known for having potential for very complicated behavior as they consist of many interconnected simple nonlinear systems, which are typically modeled by sigmoid functions. As said before, the primary motivation for neural networks was the human brain where simple neurons are massively interconnected [18]. An artificial neural network learn from examples(experiences) by operating in parallel, composed of a huge number of interconnected plain neurons(processors) .In case of artificial neural networks, exploitation is the easiest in case of these known biological neural systems' characteristics [19]. The strength of interconnections of simple elements of a neural network, defined as neurons, are known as weights . Adjustment is done to these weights to boost performance [18].

3.4.3.1 Artificial Neural Network - representation of a neuron

A neuron is a logistic unit in an artificial neural network where

- input is fed via input wires
- Logistic unit does computation which is like calculation of hypothesis $h_\theta(x)$ for logistic regression
  Here
  $$h_\theta(x) = 1/e^{-\theta^T x}$$
- output is sent down via output wires



x_1
x_2
x_3
$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$
$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$
"output"
$h_\theta(x)$
"input wires"
$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Figure 3.4: Neuron's computation model

It is often good to include the bias unit x_0 which is equal to 1

3.4.3.2 Feed forward neural network

The feed forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

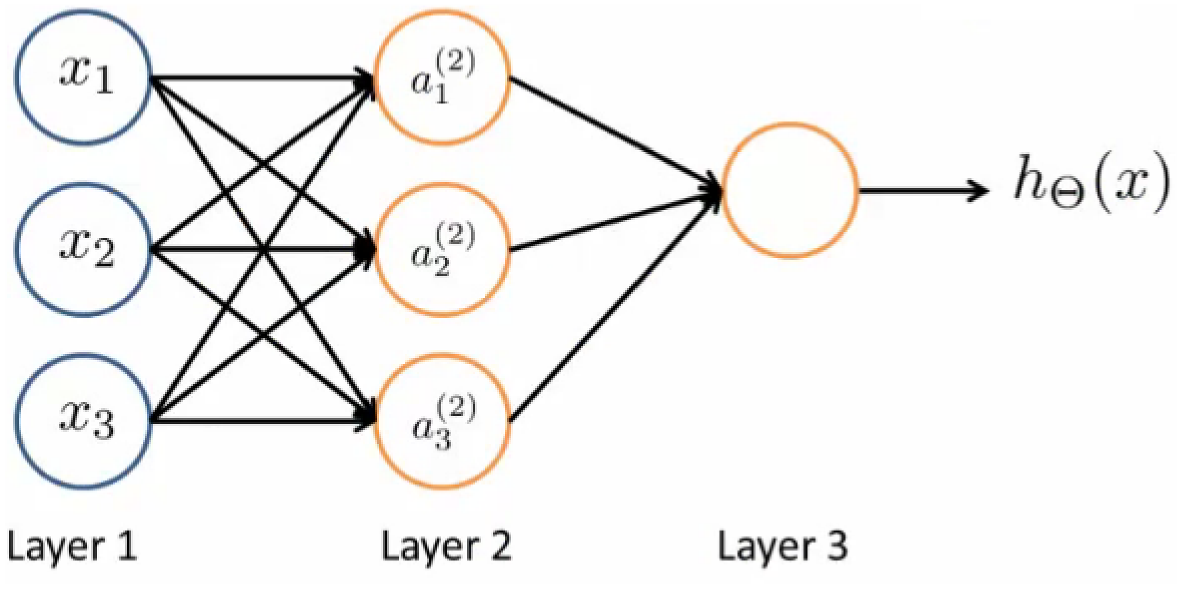


Figure 3.5: Feed-forward Neural Network block diagram

- Here, input is x1, x2 and x3
- We could also call input activation on the first layer - i.e. (a11, a21 and a31). Activation actually stands for the value computed as well as outputted by the node. For each node, we have to calculate activation.
- Three neurons in layer 2 (a12, a22 and a32)
- Final fourth neuron which produces the output
  - Which we call $a13$
- input layer is the first layer
- output layer is the final layer which produces value computed by a hypothesis
- Middle layer(s) are called the hidden layers [10]

3.4.3.3 The Perceptron

The term perceptron has no established definition, but is used to describe a feed forward network which contains a neuron that has predetermined weighted connection with the input layer and used only for acquisition of data.

3.4.3.3.1 Single-Layer Perceptrons The simplest form of an artificial neural network is known as perceptron. It is used to classification of linearly separable patterns. It contains a single neuron and the weight and bias of this neuron is adjustable. Additionally, a single layer perceptron (SLP) has only one layer of output neurons.

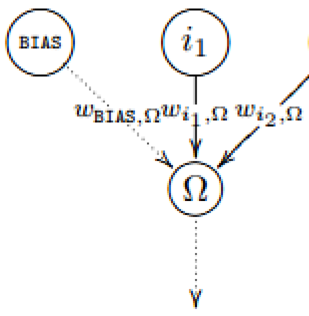


Figure 3.6: A single layer perceptron with two input neurons and one output neuron

3.4.3.3.2 Multi-layer perceptron Feed forward nets with single or multiple layers of neurons between the input and output nodes are known as multi-layer perceptron. Besides there are hidden units in these additional layers. Moreover, these hidden units are neither directly connected to input or to output nodes[20]. Back-propagation is the most popular algorithm among the wide range of learning techniques that Multi-layer networks use. Here comparison between the correct answer and output values are used while computing some predefined error function value. Through the network, this error value is fed back via several methods. This information is then used by the algorithm for weight adjustment of each connection. This helps to reduce the error function value by small amount. Convergence of the network is determined when calculated error is small. This convergence usually takes repetition of the procedure for a huge number of training cycles [21].

3.4.3.4 Back-propagation

Though the back-propagation algorithm was at first introduced in the 1970s, but its significance wasn't fully valued until a famous 1986 paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams. That paper describes several neural networks where back-propagation works far faster than earlier approaches to learning, making it possible to use neural nets to solve problems which had previously been insoluble. Today, the back-propagation algorithm is the workhorse of learning in neural networks.

3.4.3.4.1 The Algorithm The error signal at the output of neuron j at iteration n (i.e., presentation of the nth training example) is defined by

$$e_j(n) = d_j(n) - y_j(n)$$

neuron j is an output node.

Correspondingly, the instantaneous value $E(n)$ of the total error energy is

$$\mathscr{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

where the set C includes all the neurons in the output layer of the network. Let N denote the total number of patterns (examples) contained in the training set. The average squared error energy is obtained by

$$\mathcal{E}_{av} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}(n)$$

For a given training set, E_{av} represents the cost function as a measure of learning performance. The objective of the learning process is to adjust the free parameters of the network to minimize E_{av}

3.4.3.4.2 The Two Passes of Computation Two distinct passes of computation are distinguished in the application of the back-propagation algorithm. The first pass is referred to as the forward pass, and the second is referred to as the backward pass. The synaptic weights remain unaltered in the forward pass throughout the network, and the function signals of the network are computed on a neuron-by-neuron basis. The backward pass, on the other hand, starts at the output layer by passing the error signals leftward through the network, layer by layer, and recursively computing the 0 (i.e., the local gradient) for each neuron[22].

3.4.3.4.3 Stopping Criteria It cannot be shown that back-propagation algorithm has hit convergence. However, the back-propagation algorithm is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small[17].

Chapter 4

Experiment and Results

4.1 Training and Testing

4.1.1 Training in Gradient Descent

While we were running the training set in order to train the program, it was critical to choose alpha. The value of alpha was determined by trial and error. The program became properly trained when the cost function yielded minimum error and and converged to zero or near zero[23].

4.1.2 Training for Classification

In case of one vs all algorithm, for training the data of rainfall prediction, we split the training set into 70 separate binary classification problems since our objective was to predict the percentage of amount of rainfall for each mm between 0 to 69.

Similarly, for training the data for crime problem, we calculated the number of categories of crime mentioned in the data. We found that there were 24 categories of crimes. Hence, the training set was divided into 24 separate binary classification problems.

4.1.3 Training a neural network

4.1.3.1 Randomly initializing weights

The weights carried by multilayered high order perceptrons plays a huge role in deciding the learning speed [24]. There are several suggestions for initializing these weights. Among them random weight initialization is the simplest. Whether the training algorithm will result in a satisfactory local minimum or in an unacceptable local minimum, depends on weight initialization. The successful convergence probability is dependent on weight initialization scheme [25].

4.1.3.2 Implementing forward propagation

Forward propagation has two important steps.

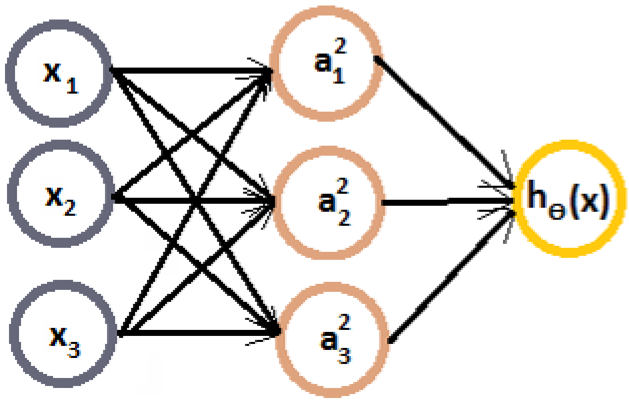
- Activation of each input unit is the very first step.

  Activation also depends on

  - Input(s) to the node
  - Parameters which are associated with that very node.

- Then calculation of activation of each layer sequentially by forward propagation to calculate final hypothesis.

Following figure is an example of a network which includes associated calculations.



$$\begin{aligned}
a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3) \\
a_2^{(2)} &= g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3) \\
a_3^{(2)} &= g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3) \\
h_\Theta(x) = a_1^{(3)} &= g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})
\end{aligned}$$

4.1.3.3 Computing Cost Function

The cost function of neural network is as follows:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

The output of this cost function is a k dimensional vector. Here $h_\theta(x)$ i refers to the ith value in the $h_\theta(x)$ vector.

4.1.3.4 Implement Back propagation

Back propagation, an abbreviation for ”backward propagation of errors”, is used to calculate partial derivatives. Hence it minimizes the cost function. The basic idea, used to compute the partial derivatives $\frac{\partial E}{\partial w_{ij}}$ for each weight in the network, is to repeatedly apply the chain rule:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{w_{ij}}$$

where

$$\frac{\partial s_i}{\partial w_{ij}} = \frac{\partial s_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}} = f'_{log}(net_i)\, s_j$$

To compute $\frac{\partial E}{\partial s_i}$ or the influence of the output s_i of unit i on the global error E, the following two cases are distinguished:

- If i is an output unit, then

$$\frac{\partial E}{\partial s_i} = \frac{1}{2} \frac{\partial (t_i - s_i)^2}{\partial s_i} = -(t_i - s_i)$$

- If i is not an output unit, then the computation of is a little more complicated. Again, the chain rule is applied:

$$\begin{aligned} \frac{\partial E}{\partial s_i} &= \sum_{k \in succ(i)} \frac{\partial E}{\partial s_k} \frac{\partial s_k}{\partial s_i} \\ &= \sum_{k \in succ(i)} \frac{\partial E}{\partial s_k} \frac{\partial s_k}{\partial net_k} \frac{\partial net_k}{\partial s_i} \\ &= \sum_{k \in succ(i)} \frac{\partial E}{\partial s_k} f'_{log}(net_k)\, w_{ki} \end{aligned} \quad (1)$$

where $succ(i)$ denotes the set of all units k in successive layers (successive means closer to the output layer) to which unit i has a non-zero weighted connection w_{ki}.

Equation (1) assumes knowledge of the values $\frac{\partial E}{\partial w_{ij}}$ for the units in successive layers to which unit i is connected. This can be provided by starting the computation at the output layer and then successively computing the derivatives for the units in preceding layers, applying (1). In other words, the gradient information is sequentially moved from the output-layer back towards the input-layer. Therefore it is called 'back propagation algorithm. [7]

4.2 Experiment Result

4.2.1 Gradient Descent

After training the system with the training set, our algorithm yielded the graph shown below:

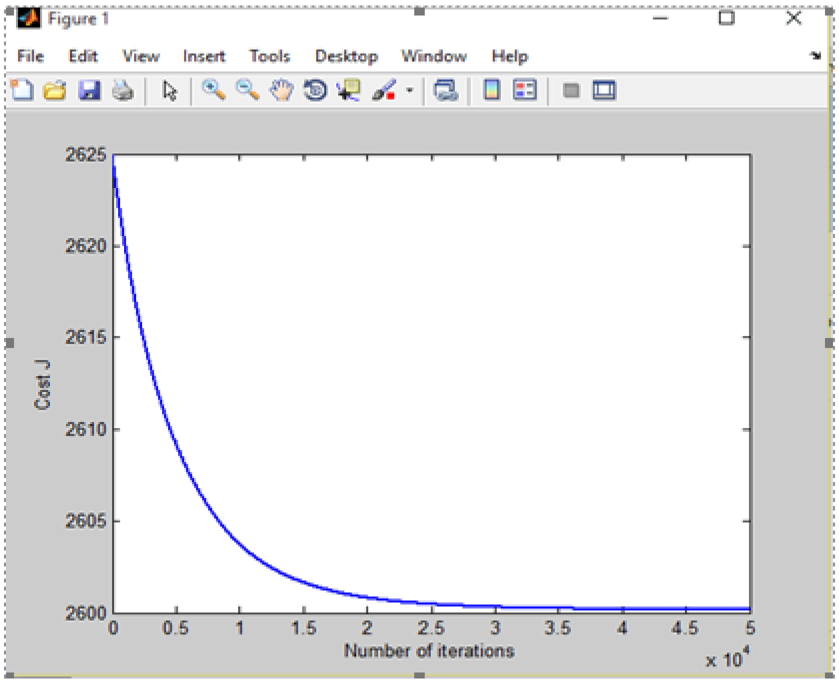


Figure 4.1: Cost function representation for data set

Here X axis represents the number of iterations and Y axis represents the Cost function J. From the graph, we see that as the number of iteration increases, the cost function error moves towards to zero. Hence, it shows that the system has converged and is ready to run the test data. So, using the value of theta that we got, we ran the test data. Thus we got the values of theta from the train data. We took those theta values and the values of the features of test data as input for the hypothesis function of linear regression and thus we got the output which is the amount of rainfall for the corresponding features. For measuring the accuracy, we compared the yielded outputs of rainfall amount and the actual value of rainfall amount. We observed that the error was high. Even though we tried different values of alpha to improve the accuracy, the error could not be minimized in a significant amount. Since, the train data was huge on size, the algorithm could not be trained properly. As a result, the predicted values were not accurate as expected.

4.2.2 Classification

After training, applying the test data for crime, we can see the following histrogram

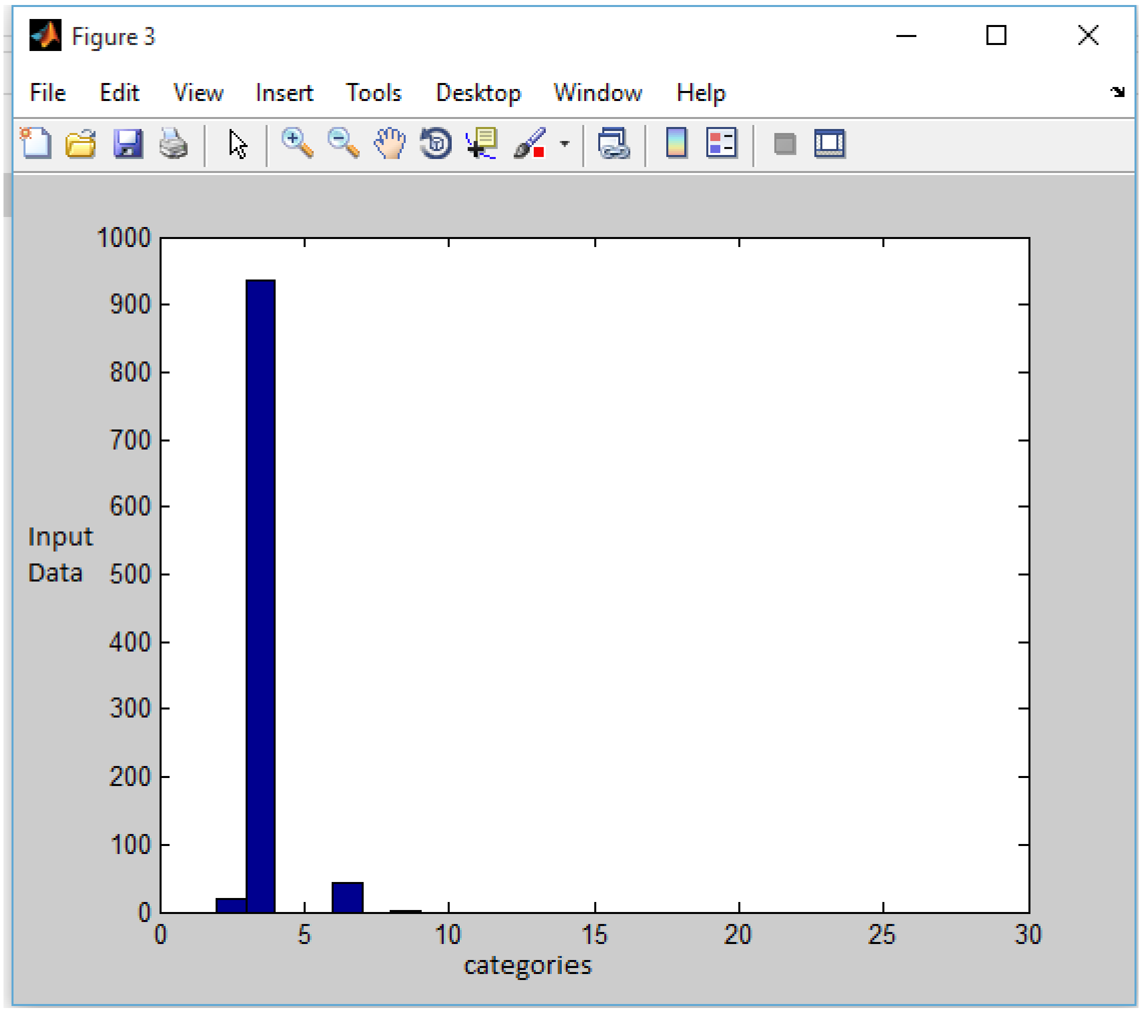


Figure 4.2: Histogram for Crime data set

Here in the X axis the crime data has been categorized into 24 classes. and from the Y axis we can see how much crime is on each category. Here class 3 has the highest occurence crime.

Again from the rain dataset we get

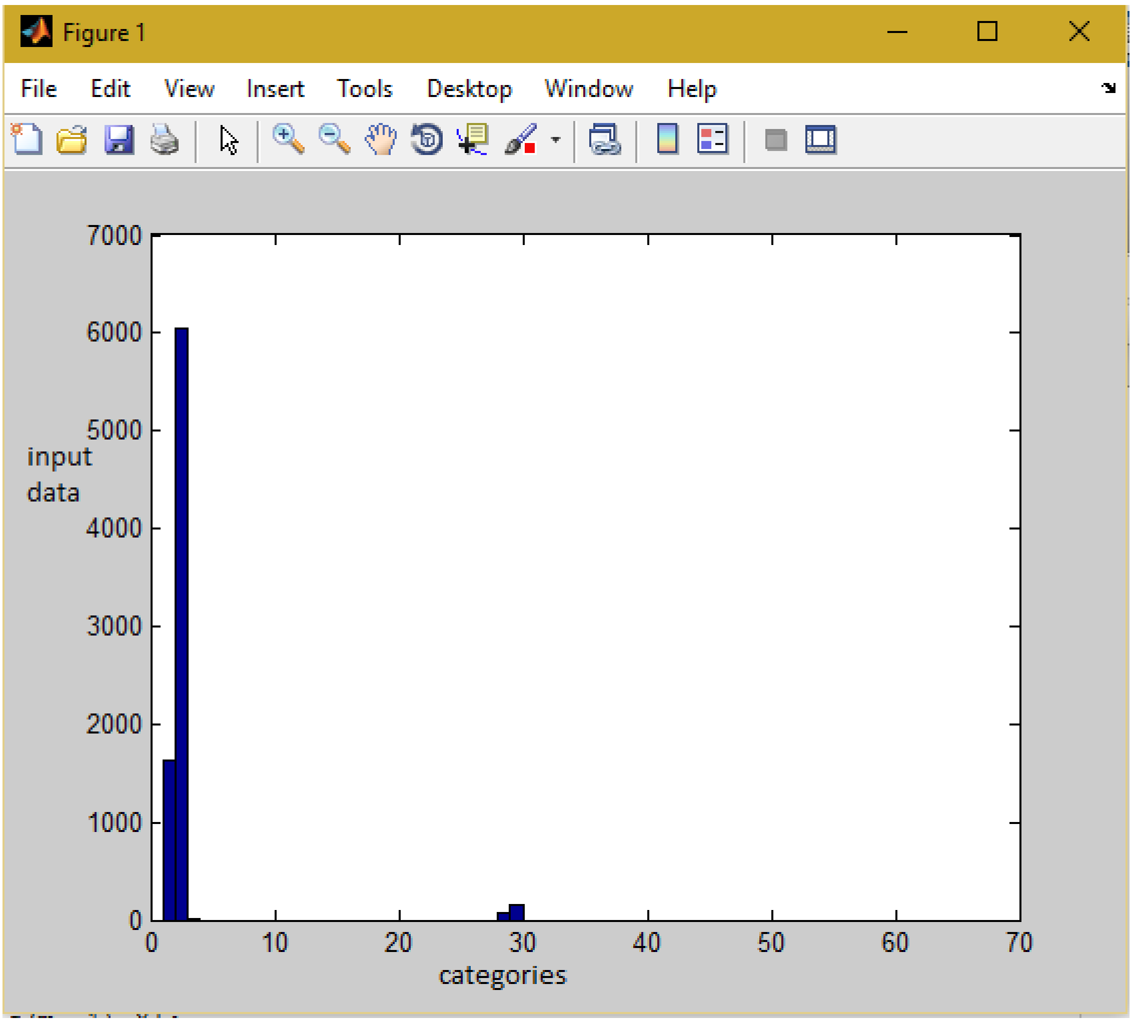


Figure 4.3: Histogram for Rain data set

Here in the X axis showa that the rain data has been categorized into 70 classes. and from the Y axis we can see the probability of each rain in mm. Since our data was short and consisted most of 0 values, the maximum probability is on 0.

Accuracy from running classification on crime dataset is around 40%. Since classification result depends on the value of λ, choosing λ was really crucial here. Choosing the value of λ, we had to go through 'trial-and-error' process. Hence we can conclude a better λ would result into better accuracy.

4.2.3 Neural Network

Running neural network gave us around 50% accuracy. Here we have used two hidden layers. Again choosing λ was crucial here. The neural network algorithm should have worked better than classification. However, since our data is complex and we used backpropagation, getting our result was very slow.

Chapter 5

Future Work

Since our processed data set for rainfall prediction was very large, gradient descent turned out to be very slow to run the train set and test set. Therefore, in future, we plan to use stochastic gradient descent since it provides a drastic simplification. The algorithm randomly picks example and from this, we plan to estimate the gradient in each iteration instead of computing the gradient of $E_n(f_w)$ exactly.

$$w_{t+1} = w_t - \gamma_t \delta_w Q(z_t, w_t)$$

At each iteration, some examples are randomly picked and the stochastic process $wt, t = 1, ...$ is dependent on it. Thus the system is expected to converge much faster[26].

Moreover, since our rainfall data was very complex, therefore neural networks did not work well in these case. Backpropagation being a slow algorithm plays a role behind this slow convergence. Therefore we plan implement deep learning and SVM for predicting rains.

Furthermore,in future, we intend to apply more complex algorithms, for instance, deep learning which should give more accurate results than neural network as it has more hidden layers. We would like to examine how the data performs on these algorithms. This will enable us to highlight more on the accuracy as well as compare them.

References

[1] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[2] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[3] John McCarthy and Edward A Feigenbaum. In memoriam: Arthur samuel: Pioneer in machine learning. *AI Magazine*, 11(3):10, 1990.

[4] Rahman Md. M Farhana S. *Characterizing rainfall trend in Bangladesh by temporal statistics analysis*. PhD thesis, Dhaka, Bangladesh, 2011.

[5] Islam M. Nazrul Islam, A.K.M. S. *Rainfall Estimation Over Bangladesh Using Remote Sensing Data*. PhD thesis, Dhaka, Bangladesh, June, 2006.

[6] Meghanathan N. McClendon, L. *USING MACHINE LEARNING ALGORITHMS TO ANALYZE CRIME DATA*. PhD thesis, 1400 Lynch St, Jackson, MS, USA, March, 2015.

[7] Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(5):1060–1089, 2013.

[8] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[9] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[10] Andrew Ng. Stanford machine learning course. Available at http://www.holehouse.org/mlclass/index.html.

[11] A. Kleeman J. Boshard R. Minkowsky A. Pasch Lakshmanan, V. The ams-ai 2015-2016 contest: Probabilistic estimate of hourly rainfall from radar. *13th Conference on Artificial Intelligence, American Meteorological Society*, 2015.

[12] M. Welling. *A First Encounter with Machine Learning*. Donald Bren School of Information and Computer Science University of California Irvine, 2010.

[13] Eric C. Chi Jocelyn T. Chi. Getting to the bottom of regression with gradient descent. Available at http://www.statisticsviews.com/details/feature/5722691/Getting-to-the-Bottom-of-Regression-with-Gradient-Descent.html.

[14] Matt Boggard. Regression via gradient descent in r. Available at http://www.r-bloggers.com/regression-via-gradient-descent-in-r/.

[15] Eric C.Chi Jocelyn T.Chi. Getting to the bottom of regression with gradient descent. Available at http://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/.

[16] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[17] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[18] Panos J Antsaklis. Neural networks for control systems. *Neural Networks, IEEE Transactions on*, 1(2):242–244, 1990.

[19] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Netw.*, 2(3):183–192, May 1989.

[20] Richard P Lippmann. An introduction to computing with neural nets. *ASSP Magazine, IEEE*, 4(2):4–22, 1987.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. chapter Learning Internal Representations by Error Propagation, pages 673–695. MIT Press, Cambridge, MA, USA, 1988.

[22] Martin Riedmiller. Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278, 1994.

[23] Alex Smola and SVN Vishwanathan. Introduction to machine learning. *Cambridge University, UK*, pages 32–34, 2008.

[24] Georg Thimm and Emile Fiesler. Neural network initialization. In *From Natural to Artificial Neural Computation*, pages 535–542. Springer, 1995.

[25] *ESANN 2001, 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 25-27, 2001, Proceedings*, 2001.

[26] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.