Wireless Healthcare Monitoring Systems

By Gefan Xie

2020

A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Electrical (Computer or Telecommunications) Engineering at The University of Newcastle, Australia.

Abstract

With the development of technology, smaller and more precise electronic components can apply to produce wearable health monitoring devices. Since the1960s, researchers have begun to study the prospects of smart wearable technology. In [1], P.F. Binkley pointed out there are three significant limitations in the diagnosis and treatment of cardiovascular disease field if patients only have a brief period of clinical monitoring in the hospital: 1) they are likely to fail in sampling rare events that may be of profound diagnostic, prognostic or therapeutic importance; 2) they fail to measure a more realistic index of the health status of a patient and the patient response to therapeutic intervention, including the physiological responses during regular periods of activity, rest and sleep. It can eliminate the psychological pressure brought by the hospital environment to the patient, which will affect the measured physiological index. 3) monitoring in a short period cannot capture the circadian variation in a physiological index that seems to mirror the progression of the disease. Therefore, it is necessary to develop wearable and real-time health monitoring devices. Accompanied by improvement in data science, it is relatively simple to establish a monitoring system based on vast amounts of databases and datasets. This project will focus on creating an early detection model based on a supervised machine learning algorithm. It is utilizing the existing datasets to build the model, to make prediction and monitoring by using a measured physiological index as input.

Acknowledgement

I would like to express my great appreciate to Dr. Ngo for his valuable and constructive suggestions during the planning and implementation of this final year project. His willingness to give his time so generously has been very much appreciated.

I would also like to thank my friends Junping Wu, Dong Zhang and Jiahoo Loh. They helped me when I was struggling with the difficulties. They gave me a lot of excellent ideas.

List of Contribution

The key contributions are listed below:

- Literature study on the implemented algorithms
- Implemented Logistic Regression by Matlab
- Implemented Neural Network by Matlab
- All code in the implementations are made by me, no build-in function is used
- Modified MTD function with polynomial term to achieve feature extension
- Proposed a binary features selection called Ratio Feature Selection (RFS)
- Modified optimisation method Gradient Descent by adding two mechanisms
- Simulated and compared the results in Matlab
- Modified Hold-out cross-validation to avoid contingency.

Contents

Chapter 1: Introduction

1.1 Background and Motivation

Heart disease is a catch-all term covering a variety of conditions that affect the heart's structure and function, including coronary heart disease (CHD). In Australia, heart disease is the single leading cause of mortality. Deaths from CHD represents 68,532 years of potential life lost in Australia. Although the rate of death caused by heart disease has decreased by 22 % during the past decade, 76 Australian died from heart disease each day in 2018 [2]. Furthermore, 17.8 million people die from CVDs in 2017, of which more than three quarters were in low-income and middle-income countries [3].

Heart disease is always an essential issue that modern medicine and science need to face. Although there are lots of medical institution and organisation spend much money, and human resource on those kinds of research about heart disease, the mortality rate created by CVDs is

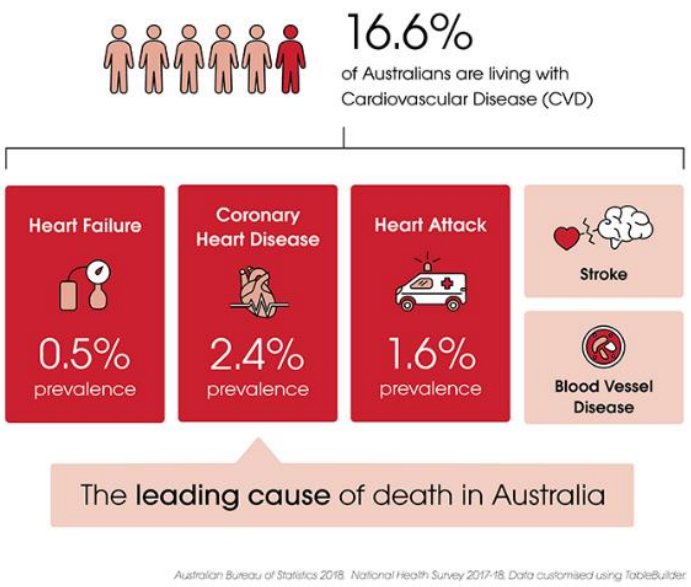


Figure 1.1.1 Heart Disease in Australia [source: [2]]

still high. Therefore, the early detection and prediction of heart disease are significant issues to diagnose and protect patients. Even though people could have physical examination regularly, people may suffer from heart disease or have a heart disease attack in each interval. Furthermore, people cannot have a physical examination every day in the hospital. Therefore, developing a smart wearable device is an appropriate way to achieve real-time monitoring and early detection. In [4], they introduced about four emerging unobtrusive and wearable technologies, which are essential to the realization of pervasive health information acquisition in the electronic field to solve the early prediction and treatment for chronic disease, including the collection of heart data. Figure 1.1.2 shows the timeline of medical devices for ECG measurement.

The problem is then how to utilize the data collected from wearable devices. Nowadays, many databases possess many datasets about the heart. Because in early detection field, not only accurate sensors are needed, but also to predict according to the data acquired from those sensors. It is not difficult to build a prediction model; the issue is how to improve the accuracy of the prediction model. Since there are lots of datasets that could be utilized, this issue could solve by machine learning. In this project, supervised machine learning which is an approach to achieve Artificial Intelligence (A.I.) was applied to establish the early detection and prediction

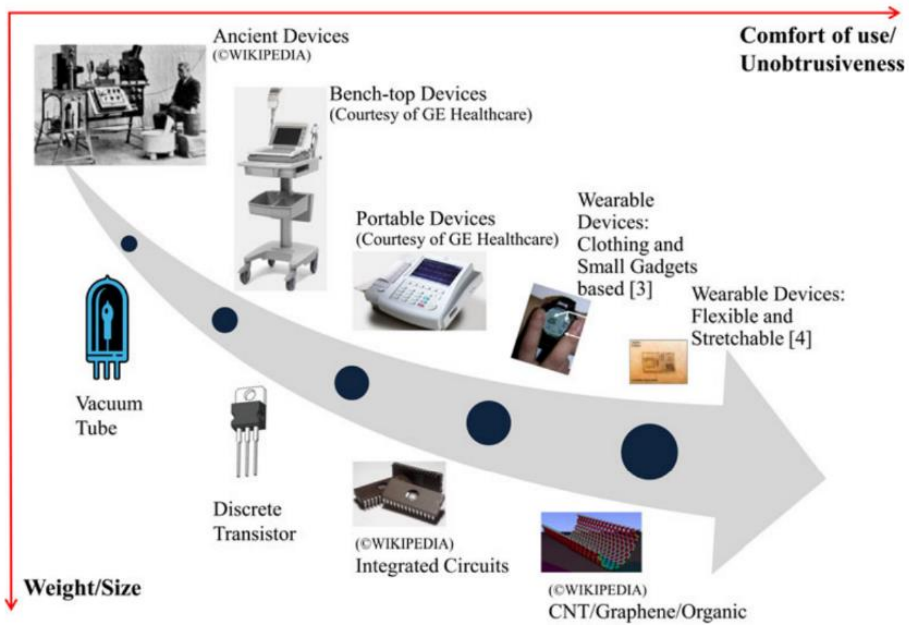


Figure 1.1.2 timeline of medical devices for ECG measurement

model based on the training datasets. Moreover, some adopted, modified, and proposed methods were employed to enhance the accuracy of prediction.

Because of the combination of machine learning and medicine, heart disease could be detected earlier, and heart attack could be predicted earlier. It would significantly reduce the mortality rate. Those collected data could also be useful for research to find a way to heal heart disease.

1.2 Outline of Report

-Chapter1: provides the introduction to this project, including background and motivation.

-Chapter2: provides the materials or concepts from other journals or conference papers which are relevant to this project, also including the theory about the supervised machine learning algorithms used in this project.

-Chapter3: in this chapter, the principle of methodology which used to improve the algorithms will be introduced here, including adopted, modified, and proposed methods.

-Chapert4: this chapter will talk about the implementation of code to achieve the prediction model.

-Chapter5: this chapter will demonstrate the results from the models and comparisons.

-Chapter6: the conclusion and extension.

Chapter 2: Literature Review

This chapter will take about the principle of what algorithm of the method used in this project.

2.1 Analyse Heart Disease

In [3], the WHO CVD Risk Chart Working Group revised models to estimate risk in heart disease. The model includes information on age, smoking status, systolic blood pressure, history of diabetes and total cholesterol. Especially for the information on smoking status and diabetes, Figure 2.1.1 shows predicted 10-year cardiovascular disease risks for different gender in several regions with total cholesterol concentrations of 5 mmol/L and systolic blood pressure of 140 mm Hg. The worst condition in all sub-plots is the sample with diabetes and smoker, no matter what gender the sample is. Therefore, diabetes and smoking status play an essential role in health monitoring for heart disease.

According to the different impact risk factors have on men and women, J. Lennep et al. [5] listed a table which discussed the role of cardiovascular risk factors (shown in Figure 2.1.2). The conclusion states that except for women hormonal status, no risk factor that impacts on one gender but not on the other. As a result, gender is also significant to build a model. The diagnosis of heart disease depends on the ECG signal as well. [6] introduced an approach to predict cardia disease by analysing ECG signal, there are lots of information contained in the ECG signal (Figure 2.1.3). The P wave is the first electrical positive signal in the standard ECG. Q wave is the first negative or downward deflection. R wave is the first positive deflection and S wave is the negative deflection followed by the R wave. The most distinguishable time-domain features of the ECG signal are WRS-complex. RR interval, ST-segment etc.

According to [3] [5] [6], the dataset including age, gender, smoking status, systolic blood pressure, history of diabetes, total cholesterol and ECG signal will be preferred to train the prediction model in this project.

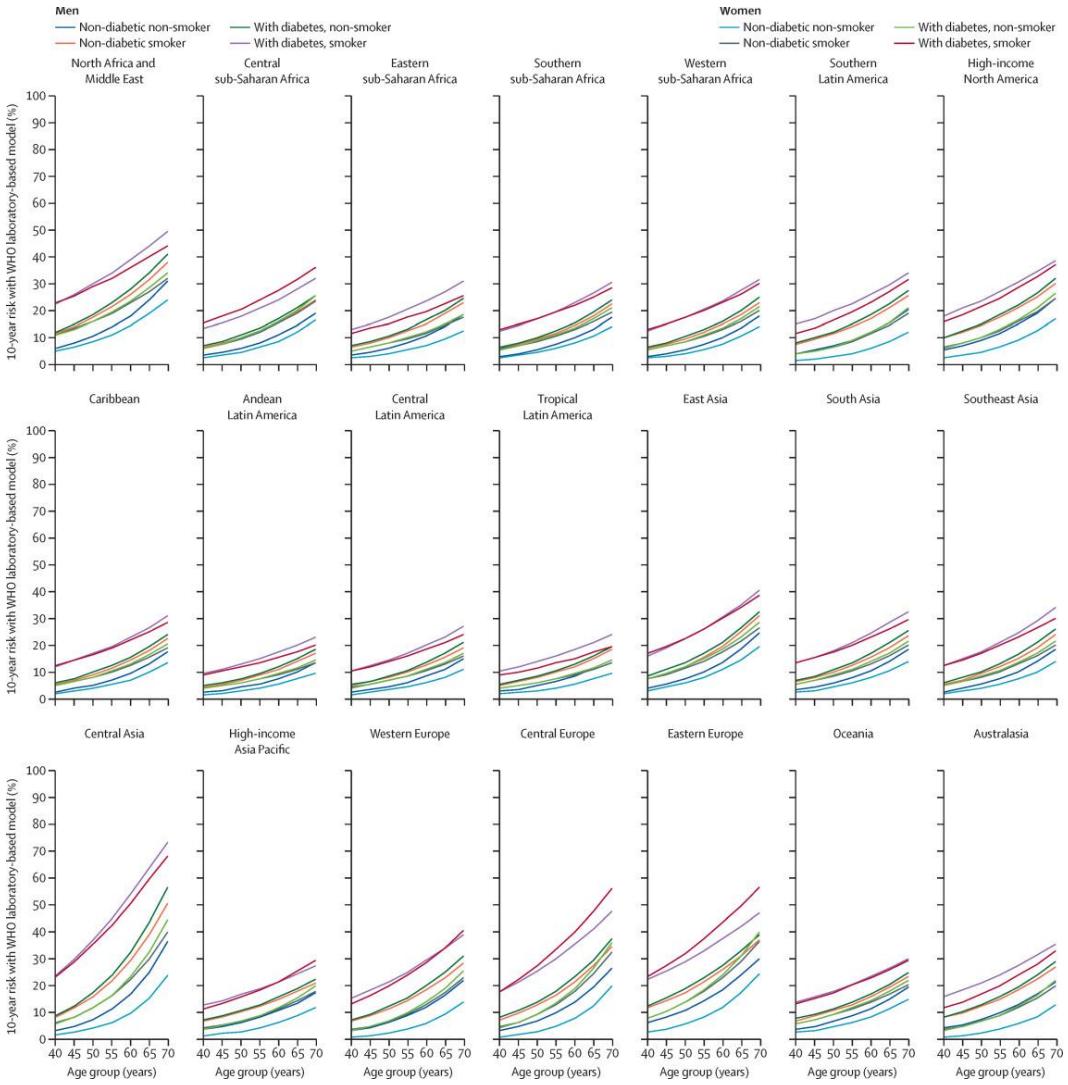


Figure 2.1.1 predicted 10-year cardiovascular disease risks [source: [3]]

Risk factors for men and women

Risk factor	Men	Women
Total cholesterol	+++	+++
LDL	+++	+++
HDL	++	+++
Triglycerides	+	++
Apo A-I	+++	+++
Apo-B	+++	+++
Apo(a)	++	+(+)
Smoking	++	++(+)
Diabetes	++	+++
Obesity		
BMI	++	++
WHR	+++	+++
Hypertension	++	++
Family History	++	++(+)
Hormones		+++
Homocysteine	+	+
Fibrinogen	++	++
Inflammation (CRP)	+	++
Infection (HP, ChP)	–	–
Psychosocial factors	+	+

Apo(a), apolipoprotein (a); Apo A-I, apolipoprotein A-I; Apo B, apolipoprotein B; BMI, body mass index; ChP, *Chlamydia pneumoniae*; CRP, C-reactive protein; HDL, high density lipoprotein cholesterol; HP, *Helicobacter pylori*; LDL, low-density lipoprotein cholesterol; WHR, waist hip ratio.

Figure 2.1.2 risk factors for men and women

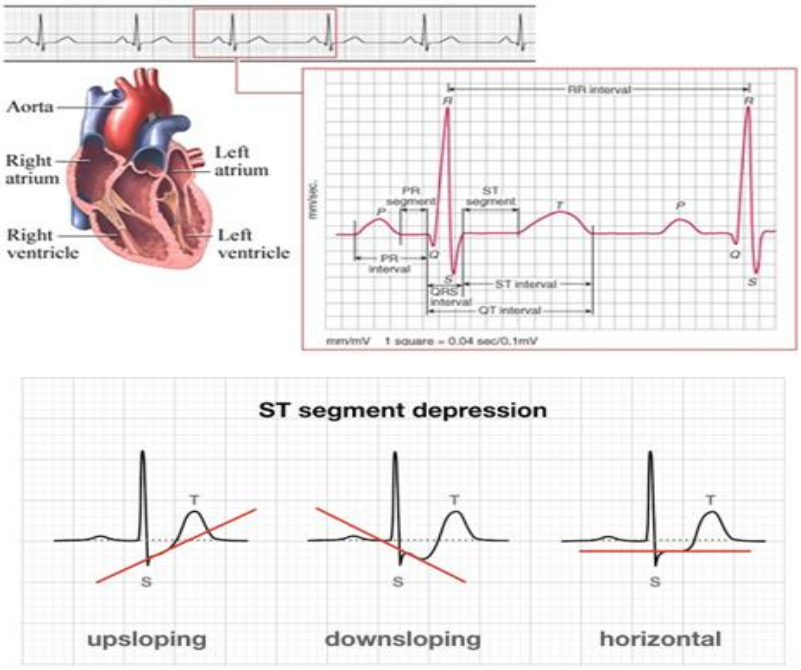


Figure 2.1.3 ECG signal

2.2 Machine Learning

Artificial Intelligence is the branch of science that deals with helping machines to find solutions for various complex problems in a similar fashion like humans [7]. It includes observing the behaviour, intelligence, and the way to think of a human being and apply those observations as an algorithm to build a model. It is like let a computer to have a brain to think and solve problems. AI is not only used in computer science, but also the machine, biology, psychology, and cognition field. The first concept of machine learning is defined by *Arthur L. Samuel* [8]. Machine learning is an application of artificial intelligence (AI). It provides computers with the ability to learn from data automatically without explicitly programming. Give database or examples to train this model. Based on the training, this model can make decision or prediction on the given data which need to be predicted or classified under uncertainty. Further, there are three major types of machine learning.

(a) Supervised machine learning: the mode will be trained with labelled data (desirable output). The data has many features and a target value to indicate classification. Training data can be classified according to how many targets. For example, to diagnose disease, the target value would be True or 1 for patients, false or 0 for healthy people. The measurements from people, also known as features, would be analysed by the model. Eventually, the decision, whether people are patient or healthy, will be made by this model.

(b) Unsupervised machine learning: the model will be trained with unlabelled data. The typical algorithm is k-mean clustering. This kind of algorithms are useful for analysing the data without a label, and human cannot find the pattern or relationship inside it.

(c) Reinforcement learning: this algorithm (called the agent) continuously learns from environment iteratively. The agent learns from its experience of the environment until find full range of possibility [9].

2.3 Application of Supervised Learning in Heart Disease Detection and Prediction

As briefly talked about some popular algorithms of supervised machine learning in section 2.2, several ways that can be used to build a detection and prediction model. Due to the different methodology is applied in each algorithm, the model made by them will perform differently. Therefore, the result or accuracy is different. The whole healthcare system can be represented in the flowchart 2.3.1.

In [10], A. N. Repaka et al. implemented Sequential Minimal Optimisation (SMO), Bayes Net (BN), Multi Player Perception (MLP) and Navies Bayesian (NB) to predict heart disease. The accuracy of these four algorithms are 84.07%, 81.11%, 77.4% and 89.77, respectively. [11] used seven algorithms and compared the results. M. Raihan et al. [12] developed a prototype design for ischemic heart disease (heart attack) risk prediction based on smartphone. Refer to [13], authors applied five algorithms to build detection model and found the performance of each algorithm. From Figure 2.1.6 the comparison of the performance of each method, SVM has good property among these methods. Another online source [14], the author applied six algorithms by using the same dataset, which is used in this project. The results indicated that SVM and LR has a good performance on this specific dataset.

Logistic regression will be used in this project to build a heart disease prediction model. Additionally, it is a regression model which can have further monitoring function. Another algorithm called neural networks will also be used because neural networks were used the most at this stage in the supervised machine learning field.

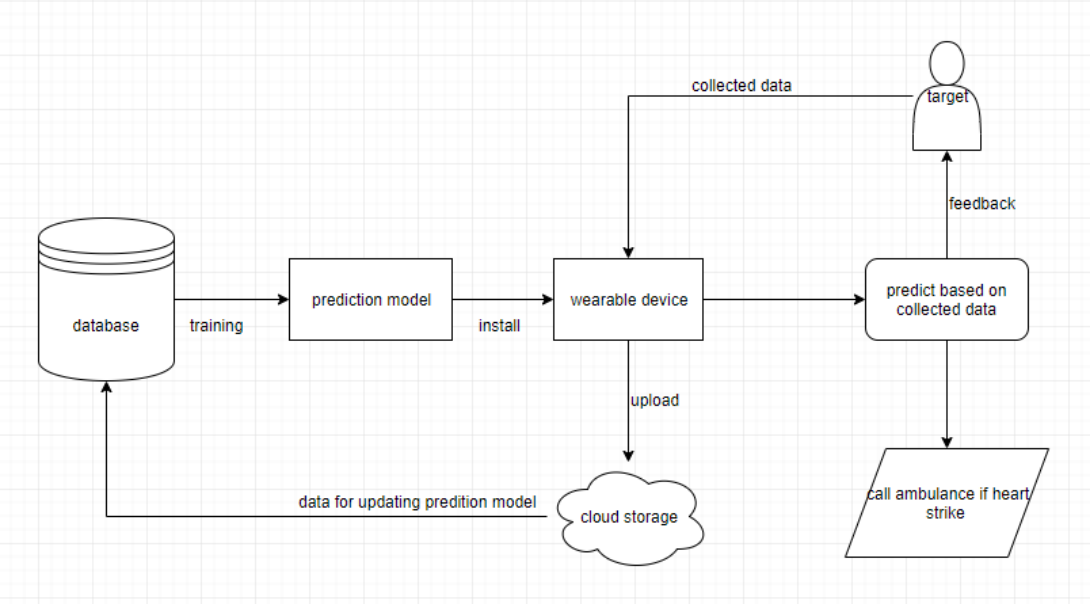


Figure 2.3.1 flowchart of whole system

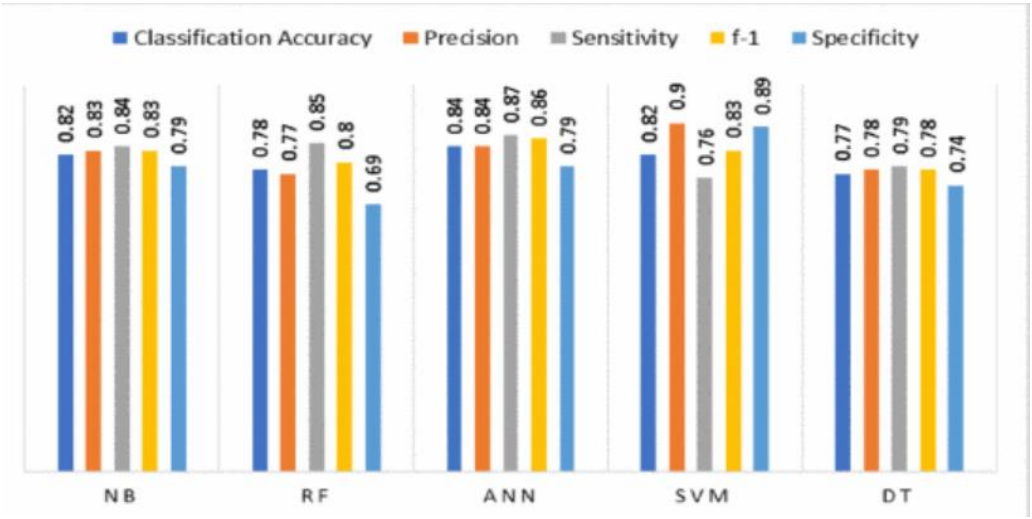


Figure 2.1.6: performance evaluation [13]

2.4 Logistic Regression (L.R.) [15]

As briefly discussed in section 2.1 about the logistic regression, this algorithm can perform in classification and regression field. There is a hypothesis function (equation 2.4.1) in this algorithm to calculate the target value. When LR work in classification model, the target value will be either 0 or 1 through decision rule in discriminant 2.4.3 in this project, 0 means this is a false sample (or negative sample) which is healthy. 1 means this is a true sample (positive samples) which is patient. When LR works in the regression model, the value of the output of the hypothesis function will be a value between 0 and 1. θ is a weight vector, and the size of it depends on how many features in the dataset. $g()$ is the sigmoid function, represented in equation (2.4.2).

$$h_\theta(x) = \mathrm{g}(\theta^T x) \tag{2.4.1}$$

$$\mathrm{g}(z) = \frac{1}{1+e^{-z}} \tag{2.4.2}$$

$$\text{decision rule: } \begin{cases} label\ with\ 1, & h_\theta(x) \geq 0.5 \\ label\ with\ 0, & h_\theta(x) < 0.5 \end{cases} \tag{2.4.3}$$

As shown in Figure 2.4.1, this is the linear regression to do the classification. Circle and cross indicate the positive samples and negative samples, respectively. Hypothesis function of linear regression is just $\mathrm{h}(\theta) = \theta^T \mathrm{x}$. The threshold is y=0.5. When the target value is less than 0.5, the sample will be labelled with negative. When the target value is greater than 0.5, then label this sample with positive. samples. If it just changes around a tiny range, the condition of this sample is stable. The following content will describe the way to obtain θ.

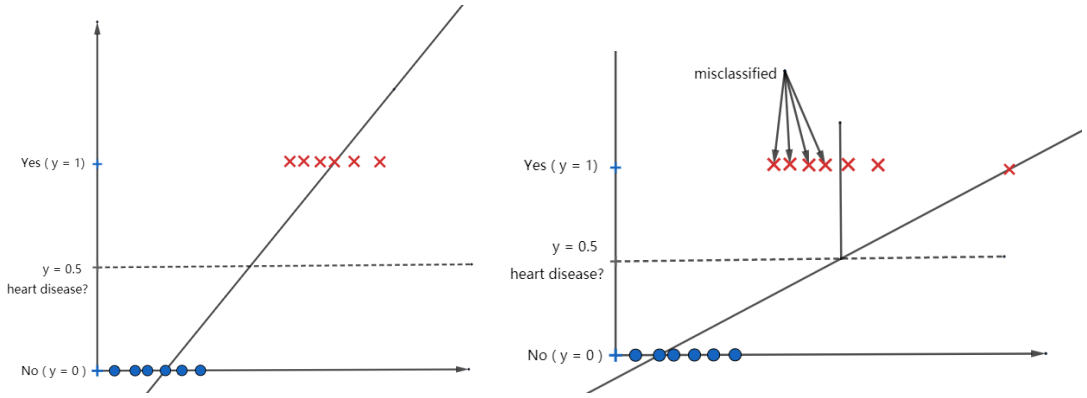


Figure 2.4.1: linear regression

A dataset has m samples, and each sample has n features. Therefore, an $\mathrm{m} \times n$ matrix will be used to represent the dataset. Every sample has a label y, so, each sample can be represented as $(x^{(i)}, y^{(i)})$, the superscripts indicated this is i_{th} sample. There is a function called Cost Function. It is a function that measures the performance of a machine learning model for given data [16]. Equation (2.4.4) is the cost function of LR. Furthermore, equation (2.4.5) is used to calculate for all samples.

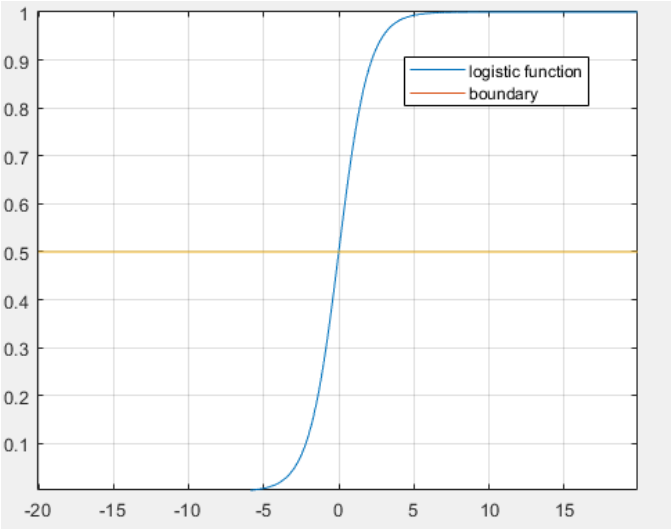


Figure 2.4.2: sigmoid function

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

combination:

$$\min_\theta Cost(h_\theta(x), y) = -\text{y} \times \log\{(h\}_\theta(x)\big) - (1 - y) \times \log\big(1 - h_\theta(x)\big) \tag{2.4.4}$$

$$\min_\theta J(\theta) \ = \ \sum_{i=1}^{m} Cost\big(h_\theta\big(x^{(i)}\big), y^{(i)}\big) \tag{2.4.5}$$

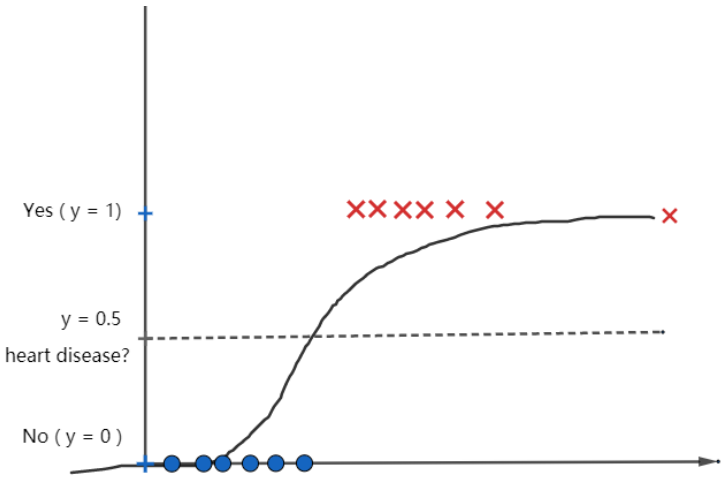


Figure 2.4.3: classification by logistic regression

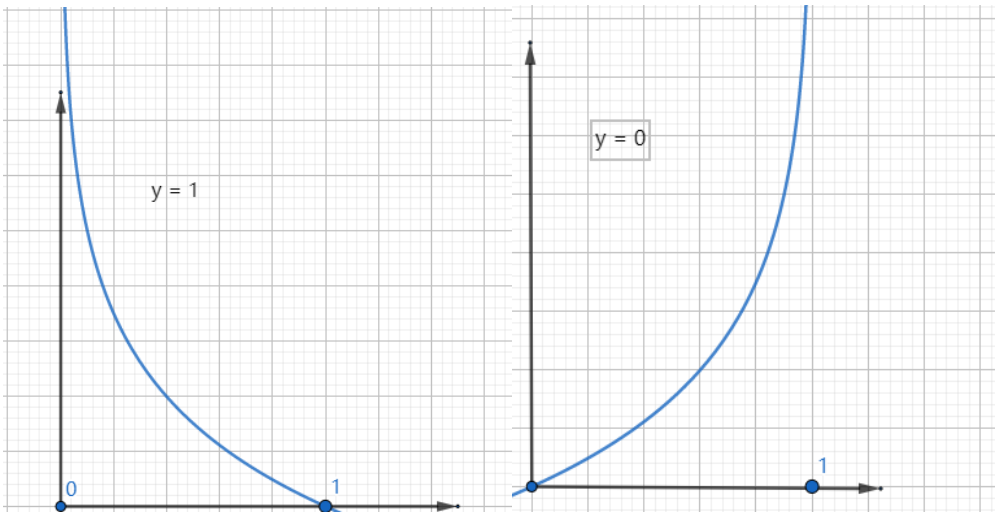


Figure 2.4.3 curve of the cost function

Figure 2.4.3 shows the curve of the Cost function in the light of different label. Captures intuition that if $h_\theta(x) = 0$, (predict $\mathrm{P}(y = 1|x; \theta) = 0$), but actual y = 1, or if $h_\theta(x) = 0$, (predict $\mathrm{P}(y = 0|x; \theta) = 1$), but actual y = 0, there will be a considerable cost to penalize the learning algorithm as a result [15]. Due to that, the cost function must be minimized. At that minimum cost function value, the weight vector will be the best of this prediction model.

2.5 Backpropagation Neural Networks

According to [17], R. Hecht-Nielsen defined a backpropagation neural network. *A neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and can carry out localized information processing operations) interconnected together with unidirectional signal channels called connections each processing element has a single output signal). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local: i.e., it must depend only upon the current values of the input signals arriving at the processing element via impinging connections and upon values stored in the processing element's local memory.*

General speaking, a neural network works like a human brain, and each unit (or called processing element in the definition) performs like a neuron in the brain. The simple structure of a neural network shows in Figure 2.5.1. The first layer in the architecture is called the input layer, and this layer contains attribute $x_i^i, x_2^i, \cdots, x_n^i$ for sample x^i in the training dataset X, the number of units equal the number of attributes plus a bias unit which the value is 1. The last layer is called the output layer. The size of this layer depends on the classification problem. This project is a binary classification problem. Therefore, the size is 1, and the value of the output unit is either 1 or 0. The layers in the middle are called hidden layers. The connection between neurons called weight. The issue here is to determine the number of hidden layers and the neurons in each hidden layer. In [18], H. Mustafidah et al. stated that *using too few neurons in hidden layers will result in something that is underfitting. Likewise, when too many neurons in hidden layers can lead to overfitting.* In the training process, there

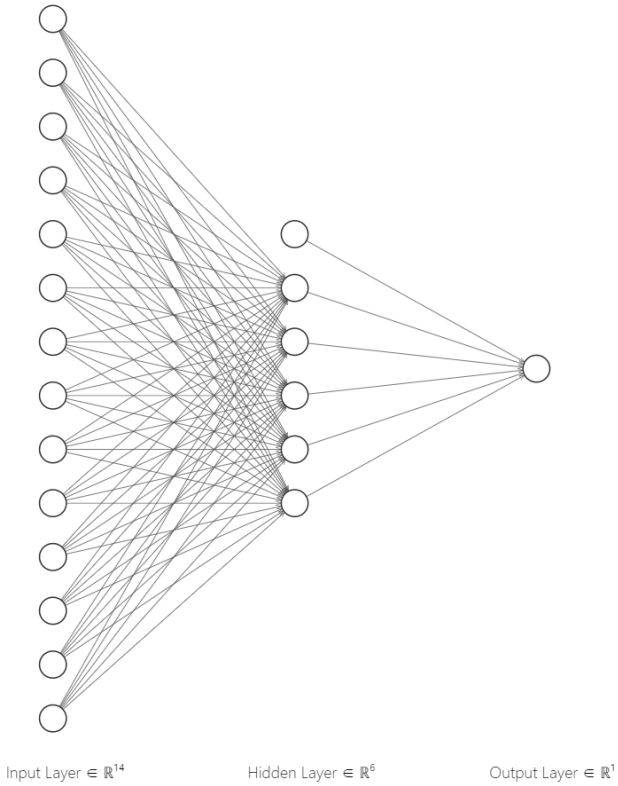


Figure 2.5.1 neural network structure

are two steps contained in the backpropagation neural network. The first step is feedforward. in this step the hypothesis function (equation (2.4.1)) is used to compute the value (or activation) of neurons (or hidden units, processing elements) in each hidden layer. The size of the weight matrix could be determined by equation (2.5.1). The cost function with the regularisation term could be represented by equation (2.5.2), which is like the cost function (equation (2.4.4)) of logistic regression.

$$size\ of\ weight\ matrx\ \Theta^{j} = s_{j+1} \times \left(s_j + 1\right) \qquad (2.5.1)$$

$where,$

$$s_j\ units\ in\ layer\ j, s_{j+1}\ units\ in\ layer\ j+1.$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \times \log\left(h_\theta\left(x^{(i)}\right)_k\right) + \left(1 - y_k^{(i)}\right) \times \log\left(1 - h_\theta\left(x^{(i)}\right)_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2 \quad (2.5.2)$$

$where,$

$$L = total\ No.of\ layers; s_l = No.of\ units\ (exclude\ bias\ unit)\ in\ layer\ j$$

$$m = No.of\ samples;\ K = No.of\ output\ units$$

Gradient descent is used to find the minimum value of cost function. Hence, the sconed step is backpropagation to compute the gradient. There is an intuition: $\delta_j^{(l)} = error\ of\ node\ j\ in\ layer\ l.$

The following procedure shows how to compute the gradient, refer to [15].

Training set $\left\{\left(x^{(1)}, y^{(1)}\right), \cdots, \left(x^{(m)}, y^{(m)}\right)\right\}$

Set $\Delta_{ij}^{(l)} = 0\ (for\ all\ l, i, j).$

For $i = 1{:}m\ (for\ every\ sample\ in\ the\ dataset)$

Set $a^{(1)} = x^{(i)}$ $\quad (set\ activation\ of\ input\ layer\ to\ features\ for\ i_{th}\ sample)$

Compute $a^{(l)} for\ l = 2,3, \cdots, L$ $\quad (the\ value\ of\ units\ in\ layer\ l)$

Using $y^{(i)}, compute\ \delta^{(L)} = a^{(L)} - y^{(i)}$ $\quad (compute\ the\ error\ in\ output\ layer)$

Compute

$\delta^{(L-1)}, \delta^{(L-2)}, \cdots, \delta^{(2)}\ by\ using\ equation\ (2.5.3)$ $\quad (compute\ the\ error\ in\ the\ rest\ layers)$

$\Delta_{ij}^{(l)} \coloneqq \Delta_{ij}^{(l)} + a_j^{(l)} \times \delta_i^{(l+1)}$ $\quad (variable\ to\ compute\ the\ gradient)$

$D_{ij}^{(l)} \coloneqq \frac{1}{m} \times \Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}$ $\quad , if\ j \neq 0$ $\quad (compute\ the\ gradient)$

$$D_{ij}^{(l)} \coloneqq \frac{1}{m} \times \Delta_{ij}^{(l)} \qquad , if\ j \neq 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

$$\delta^{(l)} = \left(\Theta^{(l+1)}\right)^T \times \delta^{(l+1)} .* g'\left(z^{(l)}\right) \tag{2.5.3}$$

$where,$

$$z^{(l)} = \Theta^{(l-1)} \times a^{(l-1)};\ g'\left(z^{(l)}\right) = a^{(l)} .* \left(1 - a^{(l)}\right);$$

After finding the minimum cost value, the corresponding weight matrix will be the best to make a prediction based on the new input dataset.

2.6 Feature-based Approaches

During the training procedure, the model could experience many issues. An excellent way may avoid some of those issues is do optimize the dataset. There are two methods called feature transformation and feature normalisation. This section will discuss the former, next section will talk about the latter.

Liu and Motoda [19] introduced these three approaches. They could reduce the amount of data, focusing on the relevant data and improve the quality of the data.

2.6.1 Feature (Attribute) Construction

If the model experiences the underfitting, it could be solved by employing several approaches. In the logistic regression model, the overfitting problem will be solved by adding more attributes. Nevertheless, the attributes in the dataset have been settled. Therefore, the method could be used here is called feature construction to extend the attributes in the dataset. The aim of this method is using the existing features in the dataset to construct new

features to train the learning algorithm. There are plenty of ways to do so. The most straightforward approach applied by Richard S. Sutton and Christopher J. Matheus [20] is adding polynomial by using operators with existing attributes. For example, a dataset has two features x_1 and x_2, and suffer the underfitting. The added features could be $x_1^2, x_2^2, x_1 \times x_2, x_1 + x_2$ and so on. This method is straightforward, and there is no data analysis applied. A kind of method will analyze the dataset first, Figure out which feature will play an important role in the algorithm. A. Zien et al. [21] introduced an approach called Feature Importance Ranking Measure (FIRM). By retrospective analysis of arbitrary learning machines, allows to achieve both excellent predictive performance and superior interpretation. One technique called mega-trend diffusion (MTD) which utilized by Li and Liu [22] to generate new attributes to increase the accuracy in their research. In this project, MTD will be applied with modification.

2.6.2 Feature Selection

Feature selection is a process to select a subset containing m_{sub} features from the original dataset containing m features $(m_{sub} < m)$, and the feature space is optimally reduced under a criterion. According to [23], feature selection could reduce the dimensionality of feature space, speed up the process of data training and improve the comprehensibility of the learning result.

In this project, a method called Ration Feature Selection (RFS) is proposed to make the feature selection. This method could improve performance when the dataset includes many binary features.

2.6.3 Feature Extraction

In this approach, the original dataset will be transformed to a new set of features by some functional mapping. For example, there are n features in a dataset, represents as $x^{(i)}\{A_1, A_2, \cdots, A_n\}$, a new set of m $(m < n)$ features $\{B_1, B_2, \cdots, B_m\}$ will come out after

feature extraction. The procedure of this transformation could be indicated as equation (2.6.1).

$$x^{(i)}\{B^{(j)}\} \coloneqq F_j(x^{(i)}\{A_1, A_2, \cdots, A_n\}) \tag{2.6.1}$$

$where,$

$$i = 1,2,\cdots, number\ of\ samples;\ \ j = 1,2,\cdots, m$$

According to different learning model, mappings can be categorized into linear or non-linear, and classification or clustering. The objective of this approach is to create a minimum set of new features and ensure that "the true nature" of the data remains after transformation [19].

2.7 Feature Normalisation

Nnormalisation transforms the features into a joint range, typically between 0 and 1. Therefore, the large numeric feature values cannot dominate the small numeric features value. The main goal is minimizing the bias of the features which have a higher numeric contribution in classification. In [24], the authors compared 16 normalisation methods with 21 datasets. Z.Mustaffa and Y.Yusof [25] provide a comparison of accuracy by applying several normalisation approaches. According to these two papers, there are three normalisation methods adopted in this project.

2.8 Gradient Descent

In the machine learning model, the most crucial step is to optimize the algorithm. The optimization method employed for Logistic Regression in this project is Gradient Descent. The Gradient Descent contains several algorithms which are commonly used in the whole world. S. Ruder [26] introduced some of the gradient descent algorithms, including Vanilla gradient descent (as known as Batch gradient descent), Stochastic gradient descent (SGD) and Mini-batch gradient descent. The Gradient Descent is an approach to minimize the cost function parameterized by the weight vectors Theta (θ) until the cost function reaches the global or local minimum. Simply saying, the process of Gradient Descent is just two steps.

The first step is choosing the elements in weight vector. It would be zeros by default. Then keep updating weight vector in the opposite direction of the gradient of the cost function respect to the weight vector to reduce Cost Function until it ends up at a minimum. If the $J(\theta)$ is a convex function, the minimum will be the global minimum. If $J(\theta)$ is a non-convex function, the minimum could be a local minimum. Once we find the minimum $J(\theta)$, the corresponding weight vector θ is the optimized weight vector that can bring the lowest value from the cost function.

2.9 Overfitting

When a dataset trains a prediction model with a vast number of features, the learned model may perform perfectly on the training set. The cost value during the training process can close to zero. However, it cannot generalise to the new samples [15] [27]. The new samples will be misclassified. A prediction model trained by a dataset with little features cannot make a correct prediction because the insufficient attributes cannot teach the model well, and the underfitting problem comes. Once there is a considerable number of features, overfitting problem may occur, as shown in Figure 2.9.1. The options to solve overfitting include reducing the number of features and regularisation. The latter is preferred. Regularisation can keep all features but reduce the magnitude of weight units. L1 and L2 regularisation are applied often. A novel method called 1/4 regularisation was proposed by [28]

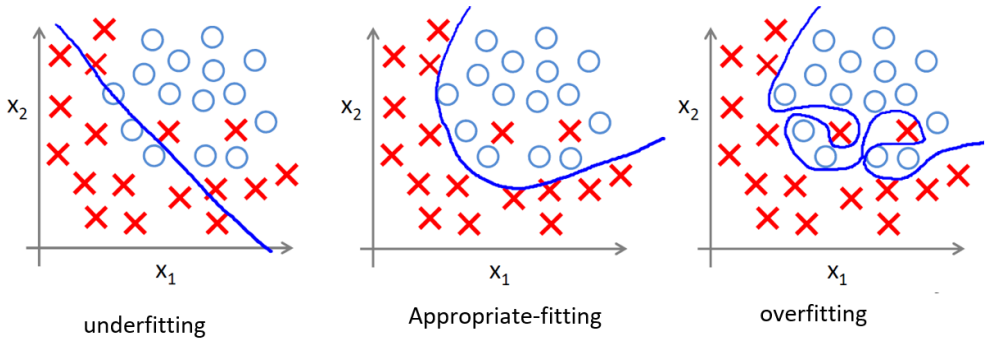


Figure 3.9.1 [15]

Chapter 3: Methodology

The adapted, modified, and proposed methods which were used to improve the performance of algorithms will be discussed deeply in this chapter.

3.1 Modified Gradient Descent

3.1.1 Comparison

The gradient descent algorithm used here is called Batch gradient descent, and the following two paragraphs are the comparisons between Batch and other gradient descent.

In the Stochastic gradient descent (SGD), the formula becomes to equation (3.1.1). The procedure of computation is that there will be two For loops, the index of the inner loop is i, the index of the outer loop is j. in this algorithm, the weight vector will be updated for each sample. In other words, the weight vector will be not be updated simultaneously, the element the weight vector will be updated one by one, which differ from Batch gradient descent.

$$\theta_j \coloneqq \theta_j - \alpha \times \frac{\partial J(\theta, x^i, y^i)}{\partial \theta_j} \tag{3.1.1}$$

The difference between Batch and Mini-batch gradient descent is only one point. The former one used all dataset to compute each element in the weight vector. The latter one just uses the part of the dataset to calculate the weight vector, usually using 50 samples as a batch, but it can vary for a different model. The equation (3.1.2) is the formula. There also will be two For loops. The inner one is for index i, and the outer one is for index j.

$$\Theta_j := \Theta_j - \alpha \times \frac{\partial J\left(\theta, x^{(i:i+n)}, y^{(i:i+n)}\right)}{\partial \theta_j} \tag{3.1.2}$$

In the Batch gradient descent, the formula for updating the weight vector shows in equation (3.1.3).

$$repeat\ until\ convergence \left\{ \theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \ , j = 1, \dots, n \right\} \tag{3.1.3}$$

α is the learning rate. It should be set to an appropriate number when building an L.R. model. If α is too small, Gradient Descent could converge very slowly. If α is too large, Gradient Descent could overshoot the minimum. It may fail to converge, or even diverge [15]. The new weight vector will be assigned to the old weight vector minus the gradient of the cost function with respect to the weight vector times the learning rate. Minus sign there is because when there is a negative gradient, substituting this gradient back into equation (3.1.3), the new weight vector will be larger, by using the new weight vector to compute the cost value, the cost will be smaller. Oppositely, when the gradient is positive, the new weight vector will be

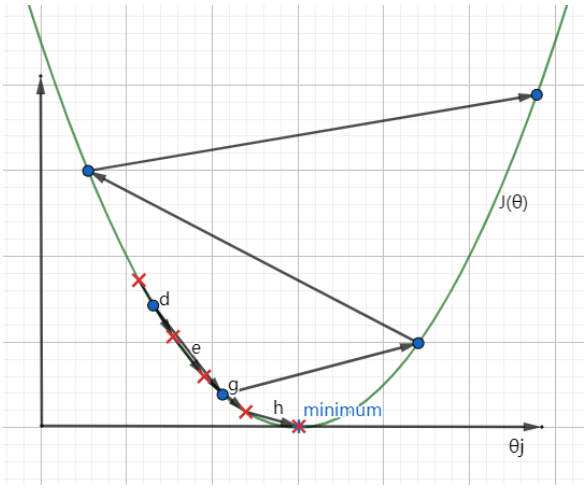


Figure 3.1.1: learning rate

smaller, then the cost value computed by using the new weight vector will be smaller, too. Figure 3.1.1 illustrates the curve due to large α (points in the blue circle) and small α (point in the red cross). In this plot, we can find the change cost value and weight vector according to the sign of gradient. Furthermore, one important thing is that all elements in the weight vector should be updated simultaneously before the next iteration.

There are advantages and disadvantages to each algorithm here. For Batch, although the computation process in this algorithm is slow due to the mechanism of the calculation, it will be the most precise and stable among these three algorithms. So, it is not suitable for the large dataset, but in this project, the dataset is not too huge to use Batch gradient descent. For Stochastic, the process of computation is much quicker than the Batch. However, in this algorithm, the overshooting can occur likely, and it performs with a high variance that causes the cost function undulates heavily, as shown in Figure 3.1.2.

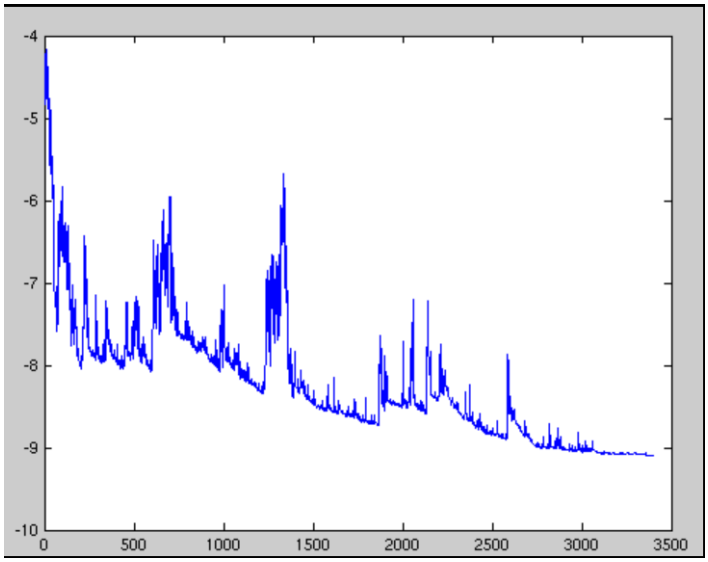


Figure 3.1.2 Cost function of SGD (source: Wikipedia)

3.1.2 Modification

There are two convergence conditions will be considered. The first one is that the cost function decreases continuously. The second condition is that the cost function in a convex function. If the model belongs to the first one, the process is just updating the minimum cost and the number of iterations. Moreover, comparing the previous cost value with the current cost value, if the difference is smaller than a threshold, then the cost value could be considered as stable and almost reaches the minimum point. For a convex model, it will compute the cost value for every iteration, and compare and update the cost value. One crucial step is that the overshoot will be tested in every iteration. The modification here is that when overshoot is detected, the value of iteration will be reduced by 100, as well as the learning rate will be half of the current value. A parameter called $overshootFlag$ which will be used in another mechanism, is set to 1.

Furthermore, a condition will be tested, if the iteration is less than 1, then the new value of iteration will be assigned to 1 to restart the computation. Except for the mechanism for detecting the overshoot, another mechanism is about the learning rate. The curve of the cost function in the logistic regression model in this project is continuously decreasing after running 5000 iterations. As a result, the learning rate could be large at the beginning stage to speed up to converge. In this mechanism, the computation process is divided into 5 stages. The value of iteration equals 0 to 1000, 1000 to 2000, 2000 to 3000, 3000 to 4000 and beyond 4000. Once the learning rate is set to a large number, to avoid overshoot occurring, the learning rate will become smaller than the previous value. The parameter $overshootFlag$ is mentioned before. Once the overshoot occurs, the learning rate value will not be affected by the value of iteration. Otherwise, if the specific learning rate value in a stage is greater than the learning rate value after overshoot happens, there is a risk that the overshoot may occur again. Therefore, this mechanism may avoid this condition. The procedure of detection shows in the chapter of implementation.

3.2 Modified Mega-Trend Diffusion (MTD)

The first step is to build the mega-trend diffusion function which will be the triangular fuzzy membership function for each class in every attribute shows in equation (3.2.1), the value of the function will be between 0 and 1 which, represents the class-probability. The Figure 3.2.1 show the plot of membership function, $M^1(x)$ and $M^2(x)$ is the value of membership function, which indicate the class-possibility for this specific x. The following step is to calculate the overlap area of the fuzzy membership function of each class. There are two classes in the datasets. The next step is to find the overlap area of the membership functions of these two classes.

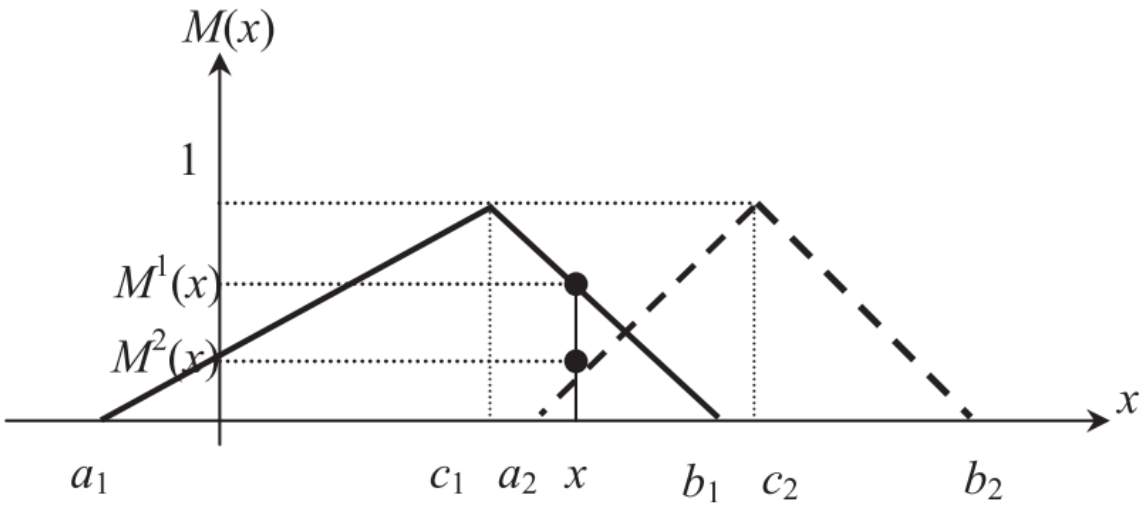


Figure 3.2.1 membership function for a two-class model [22]

$$Given\ the\ data\ set\ X = \{x_1, x_2, \cdots, x_m\},$$

$$M(x) = \begin{cases} \dfrac{x - a}{u_{set} - a}, & a \le x \le u_{set} \\ \dfrac{b - x}{b - u_{set}}, & u_{set} \le x \le b \\ 0, & otherwise \end{cases} \tag{3.2.1}$$

$where$:

$$boundary\ \ a = u_{set} - skew_L \times \sqrt{(-2) \times \frac{S_x^2}{N_L} \times \ln(f(t))}$$

$$boundary\ \ b = u_{set} + skew_U \times \sqrt{(-2) \times \frac{S_X^2}{N_U} \times \ln(f(t))}$$

$$u_{set} = \frac{\min X + \max X}{2},$$

$\min X\ and\ \max X\ are\ the\ minimum\ and\ maximum\ of\ every\ attribute\ respectively,$

$S_X^2\ is\ the\ sample\ variance\ of\ X, f(t)\ is\ real\ number\ which\ is\ 1.2665 \times 10^{-9}$

$N_L\ is\ the\ number\ of\ data\ points\ smaller\ than\ u_{set},$

$N_U\ is\ the\ number\ of\ data\ points\ greater\ than\ u_{set}$

$$skew_L = \frac{N_L}{N_L + N_U},$$

$$and$$

$$skew_U = \frac{N_U}{N_L + N_U}\ , show\ the\ rates\ of\ skewness\ in\ the\ distribution.$$

After the MTD function is derived, the overlap area of the curves of two classes could be computed. Suppose a low overlap of MTD functions for two classes occurs, it denotes that this informative feature plays a vital role because the samples which only has this certain feature can be easily classified in the proper class. In other words, when the overlap area is high, the samples may be misclassified according to this feature as a result of the similar class-possibility. There is a lack of how to determine the overlap area is large or small.

Therefore, instead of using overlap arear as standard, a parameter called Overlap Degree (OD) which can be computed through equation (3.2.2) and discriminant (3.2.3) to indicate high or low. In [22] will utilize the value of class-possibility to generate new attributes and other further works. However, the steps referred in this project ended at this point in the Logistic Regression model, the follow-up procedures are completely different. After the overlap degree of each feature obtained, creating some polynomials as new features by utilizing those features with low overlap degree is the final step which followed [20]. In other words, the modification is that two features construction method are combined.

$$OD^i = \sqrt{\frac{\beta_O^i}{\beta_A^i} \times \frac{\beta_O^i}{\beta_B^i}} \tag{3.2.2}$$

$where,$

$$\beta_A^i \; is \; the \; area \; of \; the \; MTD \; function \; of \; attribute \; i \; of \; class \; A$$

$$\beta_B^i \; is \; the \; area \; of \; the \; MTD \; function \; of \; attribute \; i \; of \; class \; B$$

$$\begin{cases} OD^i < mean(OD), & low \; overlap \\ OD^i \geq mean(OD), & high \; overlap \end{cases}, i = 1,2,\cdots,N \tag{3.2.3}$$

3.3 Proposed Ratio Feature Selection (RFS)

According to the method name, the selection procedure will depend on the ratio of the features. However, the method is proposed for the binary features in two-class classification problems, meaning that this method only applies to features whose value is either 1 or 0. Another requirement for employing this method is that the ratio between the number of positive samples and the number of negative samples is around 0.5. Specifically, there are two ratios required to compute. One is the ratio between the number of samples with value

equals to 0 and a total number of samples in feature x, if the ratio is greater than 0.95 or less than 0.05, thus, this feature will not be considered as a vital role to train the model, we could remove it. Because almost all the samples are or are not affected by this attribute, this attribute will not influence the classification result. Suppose the first condition is satisfied in discriminant (3.3.1). The second discriminant is to validate whether the ratio between the number of positive samples with value equals to 0 and the number of positive sample in feature x is similar to the ratio between the number of negative samples with value equals to 0 and the number of negative samples in feature x or not, which means that this attribute has the same effect in the positive and negative samples. The classification will not depend on this attribute once the second condition satisfied in discriminant (3.3.2), feature x will be used to build the training model. If those features in the dataset are not filtered, it may impact the classification result. Here, the reference is the value equals to zero in feature x, this method will work in the same way if the reference changes to the value equal to one in feature x.

$for\ attribute\ x$:

$$satisfied\ condition\ 1\ ?\ \begin{cases} 0, & \frac{n}{T} \geq 0.95\ or\ \frac{n}{T} \leq 0.05 \\ 1, & otherwise \end{cases} \tag{3.3.1}$$

$where,$

$n\ is\ the\ number\ of\ samples\ whose\ value\ of\ attribute\ x\ equals\ to\ 1, T\ is\ the\ total\ number\ of\ samples$

$$satisfied\ condition\ 2\ ?\ \begin{cases} 0, & \left|\frac{a}{P} - \frac{b}{N}\right| \leq 0.05 \\ 1, & otherwise \end{cases} \tag{3.3.2}$$

$where\ (in\ attribute\ x),$

$P\ is\ the\ number\ of\ positive\ samples.\ a\ is\ the\ number\ of\ positive\ samples\ whose\ value\ is\ 1.$

$N\ is\ the\ number\ of\ negative\ samples, b\ is\ the\ number\ of\ negative\ samples\ whose\ value\ is\ 0.$

3.4 Normalisation Approaches

Min-max Normalisation

Min-max normalisation is an approach by utilizing the minimum and maximum value of each feature. The procedure follows the equation (3.4.1) for any feature X.

$$X_{normalized} = \frac{X_{original} - X_{min}}{X_{max} - X_{min}} \tag{3.4.1}$$

$where,$

$$X_{normalized} = the\ dataset\ after\ normalization,$$

$$X_{original} = \ the\ dataset\ need\ to\ be\ normalized,$$

$$X_{min}, X_{max} = minimum\ and\ maximum\ value\ of\ feature\ X, respectively$$

Z-Score Normalisation

Z-Score normalisation approach utilizes the mean value and standard deviation of any feature X. The procedure follows the equation (3.4.2).

$$X_{normalized} = \frac{X_{original} - X_{mean}}{\sigma_X} \tag{3.4.2}$$

$where,$

$$X_{mean} = average\ avlue\ of\ feature\ X$$

$$\sigma_X = standard\ deviation\ of\ feature\ X$$

Decimal-Point Normalisation

This approach was used in [25] to compare the result with different normalisation methods. Furthermore, this approach had the best performance. The dataset is normalized by moving

the decimal point of each attribute. Thus, this approach is employed in this project to make a comparison. The procedure of this approach displayed in equation (3.4.3).

$$X_{normalized} = \frac{X_{original}}{10^{j}} \tag{3.4.3}$$

$where,$

$$j = the\ smallest\ integer\ such\ that \max(X_{normalized}) < 1.$$

3.5 Cross-validation

Sometimes, during the process of training a machine learning model, there is not exactly two datasets with the same attributes in the databases. Therefore, one way estimate the accuracy by applying accuracy estimation methods to avoid generalization problem. One of these methods is cross-validation which is applied in this project. In this approach, the dataset will be divided into some subsets randomly, as training sets and test sets. Three ways are included in this approach, called Hold-out, K-Fold and Leave-one-out.

Hold-out and Modification

In this way, the dataset will be broken up into two subsets as the training set and test set. The value of parameter Holdout will be a number from 0 to 1, which means how many percentages of the dataset will be used to test the algorithm. The rest of the dataset will be used to train the model. In most of the research papers, the Holdout parameter equals 0.3, meaning that 70% of the dataset for training, 30% of the dataset for testing.

The modification in the project is using Holdout method to train the learning algorithm several times and take the average as a result. It is similar to the K-Fold, but the difference is

the partition occurs randomly every time. Although there could be some fluctuations of accuracy, it is still an excellent way to compare the result when different optimization methods applied in the algorithm when there is plenty of training time. The advantage is that it can be implemented simply without long codes to divide the dataset into several parts, and the results are near to the result from K-fold. Nevertheless, sometimes results will fluctuate within 2% or 3%. Furthermore, it can avoid contingency of a single evaluation to make the result reliable.

K-Fold

The dataset will be split into k subsets (as known as k folds) in this way. One if these k folds are used as a validation set, the remaining $k - 1$ folds are used to training the model. Every fold will be used as the validation set. As a result, k times will take to train the model. The final result of the accuracy is the average result in every single time. The brief procedure shows in Figure 3.5.1. In this method, the validation method will be stable. Therefore, when the improvement is small, K-fold will be used instead of modified Holdout.

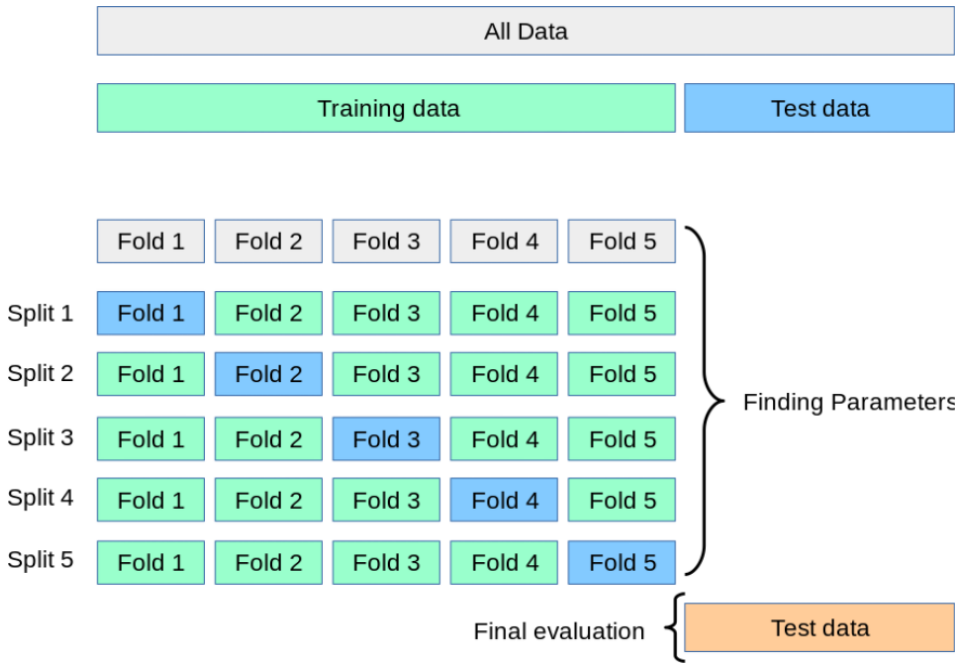


Figure 3.5.1 K-Fold procedure [source: scikit-learn]

Leave-one-out

The procedure of this method is the same as the procedure of K-Fold. The only difference is that the k value will be fixed here, and it equals to the number of the samples in the dataset. In other words, the training time will be a number according to the number of samples in the dataset. The result is the average of the result in the single training.

3.8 Number of Hidden Layers and Thump Rule

There is an issue in the procedure of building a backpropagation neural network model. Figure 2.5.1 shows the architecture of a neural network. The size of the input layer and the output layer can be determined because they depend on dataset. However, the number of hidden layers and units in the hidden layers should be determined by some methods. In [29], K. Gnana Sheela and S.N Deepa researched different methods to fix the number of hidden neurons in a neural network. They stated that most predicting research fields had been heuristic in nature. There is no generally accepted theory to determine how many hidden neurons are needed to approximate any given function in a single hidden layer.

For the number of hidden layers, most of the research in the neural network only used one hidden layer to construct the model. Moreover, in the process of building a neural network model, characteristics of the dataset, cost function, optimization function, optimization methods, etc. will be considered. There are lots of parameters that need to be optimized. Heaton [30] said a single hidden layer neural networks are capable of [universal approximation]. In this project, the number of hidden layers of backpropagation neural network is one.

For the number of neurons N_{hidden} in the hidden layer, there is thump rule, N_{hidden} is between the size of the input layer and output layer. Additionally, in [31], they used a $\frac{2}{3}$ size of the input and output layer shown in equation (3.8.1). However, half of the size of the input layer and the output layer has the same outcomes.

$$N_{hidden} = \frac{2}{3}(N_{in} + N_{out}) \tag{3.8.1}$$

$where,$

$$N_{in}\ and\ N_{out}\ are\ the\ size\ of\ input\ and\ output\ layer, respectively.$$

3.9 Symmetric Breaking [15]

The weights in BPNN require initialization, but it cannot be the same way as the process in logistic regression which is zeros. If the values in weight matrix are the same, then the activation or the value of units in the hidden layer will be the same after applying the sigmoid function (equation (2.4.2)) to compute the values. It is a pointless step.

A method called symmetric breaking could be employed to randomly initialize the weight matrix. Initializing each weight to a random value in $[-\epsilon, \epsilon]$ according to equation (3.9.1).

$$\theta_i^{(l)} = rand \times 2 \times \epsilon - \epsilon \tag{3.9.1}$$

$where,$

$$l = No.of\ hidden\ layer, i = No.of\ unit\ in\ this\ hidden\ layer$$

$$rand = a\ number\ in\ [0,1]\ selected\ randomly$$

$$\epsilon = \frac{\sqrt{6}}{\sqrt{N_{in} + N_{out}}},$$

$$N_{in}\ and\ N_{out}\ are\ the\ sizes\ of\ input\ and\ ouput\ layer\ respect\ to\ current\ hidden\ layer.$$

3.10 L2 Regularisation

According to Occam's Razor, to reduce the complexity of the prediction model can avoid overfitting. Therefore, the objective is not only to minimise the cost value, but also minimise the complexity of the prediction model. Through weakening the magnitude of weights as the penalty to reduce the complexity. Then weight matrix Θ becomes smaller and the hypothesis becomes simpler. Less prone to overfitting.

Adding regularisation term $\lambda \times \sum_{j=1}^{n} \theta_j^2$ into cost function. The new cost function shows in equation (3.10.1).

$$J(\theta) = original\ cost\ function + \lambda \times \frac{1}{2m} \times \sum_{j=1}^{n} \theta_j^2 \tag{3.10.1}$$

Because the cost function has been varied, the computation in the gradient descent algorithm will vary as well. The update process of weight units is shown in equation (3.10.2)

$$\theta_o \coloneqq \theta_o - \alpha \times \frac{1}{m} \times \frac{\partial}{\partial \theta_o} J(\theta)$$

$$\theta_j \coloneqq \theta_j - \alpha \times \frac{1}{m} \times \frac{\partial}{\partial \theta_j} J(\theta) - \frac{\lambda}{m} \times \theta_j \tag{3.10.2}$$

Once the weight is large, it will be weakened by $-\frac{\lambda}{m} \times \theta_j$ as a penalty. Except the weight unit of bias unit.

Chapter 4: Implementation Aspect of Algorithms

All the codes in this chapter were implemented in Matlab. All functions were written by me, no build-in function except the basic functions.

4.1 Datasets

Three datasets have been used in this project. The first one comes from UCI Machine Learning Repository which was created by A. Janosi [32] et al has 303 samples, although this database contains 76 attributes, only a subset of 14 of them were used in all published experiments. Table 4.1 shows the features included in this database.

FEATURE NAME	INTERPRETATION	RANGE
AGE	Age in year	29 - 77
SEX	1 = male; 0 = female	0, 1
CP	Chest pain type, 1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic	1, 2, 3, 4
TRESTBPS	Resting blood pressure (in mm Hg on admission to the hospital)	94 - 200
CHOL	Serum cholesterol in mg/dl	126 - 564
FBS	Fasting blood sugar > 120 mg/dl, 1 = true; 0 = false	0, 1

RESTECG	Resting electrocardiographic result, 0 = normal; 1 = having ST-T wave abnormality; 2 = showing probable or definite deft ventricular hypertrophy by Estes' criteria	0, 1, 2
THALACH	Maximum heart rate achieved	71 - 202
EXANG	Exercise induced angina, 1 = yes; 0 = no	0, 1
OLDPEAK	ST depression induced by exercise relative to rest	0 – 6.2
SLOPE	The slope of the peak exercise ST segment, 1 = upsloping; 2 = flat; 3 = downsloping	1, 2, 3
CA	Number of major vessels (0-3) colored by flourosopy	0, 1, 2, 3
THAL	3 = normal; 6 = fixed defect; 7 = reversable defect	3, 6, 7
TARGET	Diagnosis of heart disease, 1 = true; 0 = false	0, 1

Table 4.1 features in the first dataset

The second dataset has 1190 samples with 11 features which extracted from the scientific literature from 1992 and 2018 [33]. This dataset and the first one have 11 same features. As a result, they can be combined as one training set and one testing set respectively once two redundant features called CA and THAL were deleted. However, in this dataset there is some samples lack of important feature, there are 1018 samples left after those samples were removed.

The third dataset named Z-Alizadeh Sani Data Set [34]with 303 samples and 56 attributes was donated in 2017.

Some of the features after optimized shows in table 4.2. (feature name (interpretation): range)

Age: 30 - 86	Weight: 48 - 120	Length: 140 - 188	Sex: 1 = male 0 = female	BMI: (body mass index) 18.1 – 40.9	DM: (Diabetes mellitus) 0,1
HTN: (hypertension) 0, 1	Current smoker: 0, 1	PR: (period in electrocardiography) 50 - 110	FH: (familial hypercholesterolemia) 0, 1	Obesity: 0, 1	CP: (typical chest pain) 0, 1
DLP: (Dyslipidemia) 0,1,2,3	BP: (blood pressure) 90 - 190	Dyspnea: 0, 1	LVH: (Left ventricular hypertrophy) 0, 1	FBS: (fasting blood sugar) 62 - 400	HDL: (High-density lipoprotein) 15.9 - 111

Table 4.2 features in the third dataset

The procedure for how to utilize the datasets will be demonstrated in the implementation of each algorithm.

4.2 Implementation of L.R.

The first step is the load the dataset and extracts the attributes and label in two separate matrices. There are three datasets, a parameter called $readFlag$ is used to distinguish which

dataset should be used to train the logistic regression model. An IF-ELSE function makes a selection.

```
if readFlag == 0;
    heartData = readtable('dataset.csv','PreserveVariableNames',true);
    heartFeatures = heartData{:,1:end-1};
    actualResults = heartData.target;
elseif readFlag == 1;
    trainingData = readtable('heartTrain.csv','PreserveVariableNames',true);
    trainingFeatures = trainingData{:,1:end-1};
    trainingResults = trainingData{:,end};
    testData = readtable('heartTest.csv','PreserveVariableNames',true);
    testFeatures = testData{:,1:end-1};
    testResults = testData{:,end};
elseif readFlag == 2;
    heartData = readtable('heartTrain.csv','PreserveVariableNames',true);
     actualResults = heartData.target;
elseif readFlag == 3;
    heartData = readtable('heartTest.csv','PreserveVariableNames',true);
    heartFeatures = heartData{:,1:end-1};
    actualResults = heartData.target;

end
```

According to the training results (will discuss the details in chapter 5), as a step of pre-processing the attribute selection needs to be applied in the dataset with 303 samples and 56 attributes. There are 30 binary attributes contained in the attributes. Thence the Ratio Feature Selection (RFS) proposed in chapter 3.3 can have good performance in this dataset. When $readFlag = 0$, one more step is required:

```
heartFeatures = RFS(heartFeatures,actualResults);
```

Ratio Feature Selection function shows in the following,

```
function x_selected = RFS(x,y)
    [m, n] = size(x);
    index = 1;
    posIndex = y == 1;
    negIndex = y == 0;
    for i = 1:n
        maxVal = max(x(:,i)) == 1;
        minVal = min(x(:,i)) == 0;
        if minVal + maxVal == 2;
            condition1_1 = sum(x(:,i)) > m * 0.95;
            condition1_2 = sum(x(:,i)) < m * 0.05;
            if condition1_1 + condition1_2 == 0;
                posSample = x(posIndex,i);
                negSample = x(negIndex,i);
                ratio_p = (m - sum(posSample))/length(posSample);
                ratio_n = (m - sum(negSample))/length(negSample);
                if abs(ratio_p - ratio_n) > 0.05;
                   x_selected(:,index) = x(:,i);
                   index = index + 1;
                else
                end
            else
            end
        elseif sum(x(:,i)) == 0 || sum(x(:,i)) == m;
            continue;
        else
           x_selected(:,index) = x(:,i);
           index = index + 1;
        end
    end
end
```

Because dataset with 11 attributes has a risk to experience underfitting, two methods are provided to extend the attributes in the dataset. Mega-trend diffusion and Polynomial Addition,$extendFlag$ indicates which method will be used.

```
if extendFlag == 1; %MTD
        extenedDataset = ExtendFeature(heartData);
        heartFeatures = extenedDataset;
elseif extendFlag ==2;  %polynomial
    heartFeatures = heartData{:,1:end-1};
    extenedDataset = extend_feature(heartFeatures,2);
    heartFeatures = extenedDataset;
else %no
    heartFeatures = heartData{:,1:end-1};
end
```

Another pre-processing step that could be used is normalisation, there is a probability that the results may be better without normalisation. Hence the parameter $scalarMethod$ and a $Switch$ function indicate whether the normalisation methods should be employed and which method should be employed.

```
switch scalarMethod
        case 1
            trainingFeatures = [ones(samplesNum,1) min_max(trainingFeatures)];  % Normalize the
data by function min_max().
            testFeatures = [ones(testNum,1) min_max(testFeatures)];
        case 2
            trainingFeatures = [ones(samplesNum,1) z_score(trainingFeatures)];  % Normalize the
data by function z_score().
            testFeatures = [ones(testFeatures,1) z_score(testFeatures)];
        case 3
            trainingFeatures = [ones(samplesNum,1) decimal_point(trainingFeatures)];    %
Normalize the data by function decimal_point().
            testFeatures = [ones(testNum,1) decimal_point(testFeatures)];
        otherwise
            trainingFeatures = [ones(samplesNum,1) trainingFeatures];   % No normalization will
take place
            testFeatures = [ones(testNum,1) testFeatures];
```

No matter it is the training features or test features, both of them are the original dataset, a column of 1s is required to add in front of the original dataset as a bias unit. As a result, the size of dataset become $m \times (n+1)$, m is the number of samples, n is the number of attributes.

Three normalisation methods:

```
function normalized_data = min_max(dataIn)
    %minimum-maximum normalization method.
    [smapleNum, featureNum] = size(dataIn);
    [featureMin, ~] = min(dataIn);  %obtain the minimum
    [featureMax, ~] = max(dataIn);  %obtain the maximum
    normalized_data = (dataIn - featureMin)./(featureMax - featureMin);
end

function normalized_data = z_score(dataIn)
    %Z-score standardization method
    [sampleNum, featureNum] = size(dataIn);
    featureMean = mean(dataIn); %obatin the mean
    featureStd = sqrt(sum((dataIn - featureMean).^2)/(sampleNum-1)); %obtain std
    normalized_data = (dataIn - featureMean)./featureStd;
end

function [x_nor, powerIndex] = decimal_point(X)
    [m, n] = size(X);
    powerIndex = zeros(1,n);
    x_nor = 10 * ones(m,n);
    for i = 1:n    %for every feature
        j = 0;     %devide by 10^j
        while max(x_nor(:,i)) >= 1
            x_nor(:,i) = X(:,i)/(10^j);
            j = j + 1;
        end
        powerIndex(i) = j - 1;
    end
end
```

The dataset will be selected according to the value of $readFlag$. There is only one option when $readFlag = 1$ that contains the training set and test set separately. In other options, only one dataset will be included. Therefore, it supposes to use cross-validation, and modified Holdout method is used here. The value of Holdout can be adjusted from 0 to 1.

```
cvpt = cvpartition(heartData.target,'HoldOut',0.3); % use holdout, and adjust the value
trainingFeatures = normalizedFeatures(training(cvpt),:);    % according to the random index
trainingResults = actualResults(training(cvpt));            % to split dataset into training set
testFeatures = normalizedFeatures(test(cvpt),:);            % and test set.
testResults = actualResults(test(cvpt));
```

After the pre-processing is accomplished, step into the core part of the algorithm, which is the optimization method. The method employed here is Batch gradient descent. Substitute equation (2.4.1) into equation (2.4.4) to obtain equation (4.2.1). For computation convenience, we add factor $\frac{1}{m}$. It does not affect the result.

$$\mathrm{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)} \log\left(\left\{\frac{1}{1+e^{-\theta^{(i)}x^{(i)}}}\right) - \left(1-y^{(i)}\right)\log\left(1-\frac{1}{1+e^{-\theta^{(i)}x^{(i)}}}\right)\right\} \qquad (4.2.1)$$

The variables in the equation are in scalar form, represent these variables in matrix or vector form. It would be convenient when using Matlab to implement the code. Furthermore, matrix or vector form will avoid lots of For loops or While loops.

Equation (4.2.1) in matrix form:

$$\mathrm{J}(\theta) = \frac{1}{m}\left[-y\log\left(\frac{1}{1+e^{-x\theta^{T}}}\right) - (1-y)\log\left(1-\frac{1}{1+e^{-x\theta^{T}}}\right)\right]$$

The size of each matrix: ($m\ samples\ and\ n\ attributes\ in\ matrix\ x$)

$$x = m \times (n+1), \qquad y = m \times 1, \qquad \theta = (n+1) \times 1$$

Sometimes there will be an overfitting issue. Regularisation could be used in the LR model. T this method can keep all features but reduce the magnitude of parameters θ_j. The equation (4.2.2) is the regularised Cost Function. λ is the regularisation parameter selected manually, but if λ is too large, algorithm result will be in underfitting, and Batch gradient descent will

fail to converge. For the regularisation terms, only elements from 1 to m-1 need regularisation. Because the element θ_0 is the bias unit in weight vector does not need to this step.

$$\mathrm{J}(\theta) = \frac{1}{m}\left[-y\log\left(\frac{1}{1+e^{-x\theta^T}}\right) - (1-y)\log\left(1-\frac{1}{1+e^{-x\theta^T}}\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{m-1}\theta_j^2 \qquad (4.2.2)$$

At the same time, when we compute the cost, we could calculate gradient as well.

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\times\sum_{i=1}^{m}\left[\left(\frac{1}{1+e^{-x^i\theta^T}}\right) - y^i\right]\times x_j^i,\ \ i = 1,2,\cdots,m; j = 1,2,\cdots,n \qquad (4.2.3)$$

$$in\ vector\ form:\ \ \frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}\times\left[\left(\frac{1}{1+e^{-x\theta^T}}\right) - y\right]\times x$$

$$regularization\ term = \frac{\lambda}{m}\times\theta, except\ element\ for\ bias\ unit$$

```
function [J_Regularisation, gradient_regularisation] = CostFunction(X,y,theta,lambda)
    %COST_FUNCTION_REGULARISATION Summary of this function goes here
    %   here is the calculation for Cost function woith regularisation
    NumerOfSamples = length(y); % number of training examples
    %initialization
    J_Regularisation = 0;
    gradient_regularisation = zeros(size(theta));
    n = size(X,2);
    h = hypothesis_function(theta , X);
    J1 = ((-y')*log(h)-(1-y')*log(1-h))/NumerOfSamples ; %theta(1) is just the coeff of
X0 which is 1s.
    J2 = (lambda * sum(theta(2:end,:).^2)) / (2*NumerOfSamples);    %regularization term
    J_Regularisation = J1 +J2;
    gradient_regularisation(1,1) = ((h-y)')* X(:,1)/NumerOfSamples; %compute gradient
    gradient_regularisation(2:n,1) = ((h-y)')* X(:,2:n)/NumerOfSamples +
lambda*theta(2:n,1)/NumerOfSamples;
end
```

In LR model, Batch gradient descent is used to do optimisation. Continue the equation (3.1.3), equation (4.2.4) can be derived.

$$\theta_0^{new} \coloneqq \theta_0^{old} - \frac{a}{m}\times\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)\times x_o^i$$

$$\theta_j^{\text{new}} \coloneqq \theta_j^{(old)} - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta\left(x^{(i)}\right) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j \qquad (4.2.4)$$

$$in\ vector\ form:\ \theta^{new} \coloneqq \theta^{old} - \frac{\alpha}{m}\times(h_\theta(x) - y)\times x + \frac{\lambda}{m}\times\theta$$

Using new θ to calculate Cost function, repeat this process until Cost function converges.

The parameter $maxIteration$ I set to 5000, the maximum repeat time is 5000. The minimum value of the difference between the cost value of this iteration and the cost value of the last iteration is the parameter $threshold$. This parameter is set to $10e-9$, when the difference is less than this threshold which means this point is around the local minimum and we can take this value as a local minimum, iteration will be stopped and the θ will be the optimized weight vector. For the learning rate α, to avoid overshooting around the local minimum, the algorithm will adjust this parameter when iteration beyond 1000, 2000, 3000 and 4000 times. Therefore, in the beginning, we could have a large α to converge quickly, but no worries about overshooting issue.

```
if iteration > 1000 && overshootFlag(Time)==0;
                alpha = 1;
            elseif iteration > 2000 && overshootFlag(Time)==0;
                alpha = 0.5;
            elseif iteration > 3000 && overshootFlag(Time)==0;
                alpha = 0.25;
            elseif  iteration > 4000 && overshootFlag(Time)==0;
                alpha = 0.1;
 end
```

Just in case, there is still a mechanism to deal with overshooting. Once the previous cost value is between the current cost value and cost value before the previous cost value, it indicates that an overshoot occurs. α becomes the half to avoid overshooting occurs again, and the parameter overshootFlag becomes 1, no matter what iteration value is, α will not change. Furthermore, the value of iteration will be reduced by 100 to recompute cost value.

```
elseif iteration > 2 && cost(iteration)>cost(iteration-1) && cost(iteration-2)>cost(iteration-1);
    iteration = iteration - 100;
    overshootFlag(Time) = 1;
    alpha = alpha/2;
    if iteration < 0;
        iteration = 1;
    end
end
```

The cost value will be store when the current value is less than the previous one. Otherwise, it will maintain.

```
elseif cost(iteration) < cost(iteration-1);
    if cost(iteration-1) - cost(iteration) < threshold;
        minCost(Time) = cost(iteration);
        minIndex(Time) = iteration;
        iteration = maxIter + 1;
    else
        minCost(Time) = cost(iteration);
        minIndex(Time) = iteration;
    end
```

After the optimized weight vector found, the logistic regression model is built up. Validate this model by calculating two types of error. One is called resub-loss, as the name states, resubstitute the training set to test the model. Another one is called test error, substitute test set into the model to compute the accuracy. There is a function called predictor to make predictor based on the input dataset. One more error type called critical error still can be an assistance to validate. It only accounts for the actual patient who is misclassified into health. Three formulas are shown in the equation (4.2.5).

$$test\ error = \frac{sample\ number\ of\ misclassification\ in\ the\ test\ set}{total\ sample\ number\ in\ the\ test\ set}$$

$$resub - loss = \frac{sample\ number\ of\ misclassification\ in\ the\ training\ set}{total\ sample\ number\ in\ the\ training\ set} \quad (4.2.5)$$

$$critical\ error = \frac{sample\ number\ of\ patient\ misclassified\ into\ health\ in\ the\ test\ set}{total\ sample\ number\ in\ the\ test\ set}$$

When there are a high resub-loss and a low test error, this condition will be considered as overfitting. This model does not generalize to other datasets. Feature selection and extraction, high regularisation parameter and more training samples could be the methods to fix it. It will be considered as underfitting with low resub-loss. Additional attributes, decrease regularisation parameter could fix it.

```
prediction = predictor(testFeatures,optimisedTheta);    %make prediction
[cm ,grp] = confusionmat(testResults,prediction);   %compare the actual result and prediction
h = heatmap(grp,grp,cm);
accuracy = (cm(1,1)+cm(2,2))/size(testResults,1)    %test error
criticalError = cm(2,1)/length(testResults)
resubLoss = 1 - mean(predictor(trainingFeatures, optimisedTheta) == trainingResults)    %resub-
loss
```

```
function ypredicted = predictor(X,theta)
%PREDICTOR Summary of this function goes here
%   use logistic regression as a claasifier
    %initialisation
    NumberOfSamples = size(X,1);
    ypredicted = zeros(NumberOfSamples,1);
    % calculate the hypothesis function result
    h = hypothesis_function(theta,X);
    for i = 1:NumberOfSamples
        if h(i,1)>=0.5
            ypredicted(i,1) = 1;
        else
            ypredicted(i,1) = 0;
        end
    end
end
```

Another validation method is K-Fold which is used to compare the results of modified Holdout. Parameter $validationMethod$ is employed to indicate which method, 0 means Holdout, 1 means K-fold. K equals to 10.

```
for foldIndex = 1:foldNum
        if foldIndex == foldNum;
            testK = heartFeaturesK((1 + (foldIndex - 1) * 30):end,:);
            testResultsK = actualResultsK((1 + (foldIndex - 1) * 30):end,:);
            temp = heartFeaturesK;
            temp((1 + (foldIndex - 1) * 30):end,:) = [];
            trainingK = temp;
            temp = actualResultsK;
            temp((1 + (foldIndex - 1) * 30):end,:) = [];
            trainingResultsK = temp;
        else
            testK = heartFeaturesK((1 + (foldIndex - 1) * 30):(30 * foldIndex),:);
            testResultsK = actualResultsK((1 + (foldIndex - 1) * 30):(30 * foldIndex),:);
            temp = heartFeaturesK;
            temp((1 + (foldIndex - 1) * 30):(30 * foldIndex),:) = [];
            trainingK = temp;
            temp = actualResultsK;
            temp((1 + (foldIndex - 1) * 30):(30 * foldIndex),:) = [];
            trainingResultsK = temp;
        end
```

After the optimized weights were captured, when data is collected from a wearable device, then the algorithm inside this device can analyse the data to make a prediction. If the

prediction is positive, a message will notice this observation object to having a further medical examination in hospital.

4.2 Implementation of Backpropagation Neural Network

According to the performance in the logistic regression prediction model, the dataset with 303 samples and 55 features has the best outcomes. So, it will be used in the neural network as a training set and testing set by applying cross-validation. Because in these datasets mentioned in section 4.1, although there is dataset contains 1018 samples, this dataset lacks features, only 11 features. Another dataset has 303 samples. However, it just owns 13 features which is much less than 55. Therefore, the dataset with 303 samples and 55 features was chosen as a compromise. If there are no apparent comparisons in results or other issues, other datasets will be used instead.

Steps to load the dataset and extract the features and the actual targets of each sample and the RFS function was applied to filter the dataset.

```
heartData = readtable('dataset.csv','PreserveVariableNames',true);  % load dataset to a table
heartFeatures = heartData{:,1:end-1};                               % read the features from table
actualResults = heartData.target;                                   % read the targets from table
heartFeatures = RFS(heartFeatures,actualResults);                   % use RFS function to select
dataset
```

The following codes are setting and initializing some parameters and followed by interpretation.

```
[sampleNum, featureNum] = size(heartFeatures);      % get number of samples and features
lambda = 2;                                         % set regularization parameter
alpha = 0.5;                                        % set learining rate
maxIter = 5000;                                     % set max iteration to optimize weight
threshold = 10e-6;                                  % threshold in weight optimization
kfold = 10;                                         % fold number in k-fold cross-validation
foldsize = round(sampleNum/kfold);                  % sample number in each fold in k-fold cv
trainingTime = 1;                                   % training time for modified hold-out cv
accuracy = zeros(1,trainingTime);                   % initialize accuracy
resub_loss = zeros(1,trainingTime);                 % initialize resub-loss
overshootFlag = zeros(1,trainingTime);              % initialize overshoot flag
cost = zeros(1,maxIter);                            % initialize cost value
minCost = zeros(1,trainingTime);                    % initialize minCost in each training time
minIndex = zeros(1,trainingTime);                   % initialize the index of minCost
labelNum = size(actualResults,2);                   % number of classification, here just a binary
classification
layer1size = featureNum;                            % size of input layer
layer3size = labelNum;                              % size of output layer
```

In light of section 3.8, there is only one hidden layer to construct the backpropagation neural network. Therefore, equation (3.8.1) was used to calculate the number of neurons in this hidden layer, the result is 28 units, and this output of this model is the prediction of heart disease. Hence, this is a binary classification, $k = 1$. There is a parameter called $hiddenflag$, which indicates that the method will be used to calculate the number of neurons. So far, there is only one method was used.

```
hiddenFlag = 1;     % 1: equals 2/3 of size of input layer and output layer;

if hiddenFlag == 1;                                 % when hidden flag == 1, 2/3
    layer2size = neurons(layer1size, layer3size);   % size of hidden layer
end
```

The codes of neuron function:

```
function number = neurons(inputlayersize,outputlayersize)
    % this function is used to calculate the number of neurons when
    % there is only one hidden layer.

    % arguement 1 = the size of input layer.
    % arguement 2 = the size of output layer.

    number = round(2 / 3 * (inputlayersize + outputlayersize));
end
```

In equation (2.5.1), the size of the weight matrix should be $s_{j+1} \times (s_j + 1)$, a function called $initWeight$ was implemented to create and initialize the weight matrix. There are three layers in the BPNN, and two weight matrices are required to link these three layers. The size of the first matrix Θ_1 is 28×42, the size of second weight matrix Θ_2 is 1×29.

```
if initFlag == 1;                                % if == 1, use symmetric breaking method to
initialize weight
        initialWeight1 = initWeight(layer1size,layer2size);     % weight matrix 1
        initialWeight2 = initWeight(layer2size,layer3size);     % weight matrix 2
    end
```

```
function weights = initWeight(sizeIn, sizeOut)
    % function of initializing the weight maxtrix.
    % output is the initialized weight matrix
    % arg 1 =  input layer size,
    % arg 2 = output layer size.

    weights = zeros(sizeOut, sizeIn + 1);   % create weight matrix
    initialEpsilon = sqrt(6) / sqrt(sizeIn + sizeOut);  % calculate the epsilon

    weights = rand(sizeOut, sizeIn + 1) * 2 * initialEpsilon - initialEpsilon;  %
initialized weight matrix
end
```

The next step is the same as the step in the previous model, which is to choose the normalisation method. The comparisons in the next chapter will process according to the different normalisation method.

```
switch scalarFlag                                % select different scalar method
    case 1
        % Normalize the data by function min_max().
        normalizedFeatures = [ones(sampleNum,1) min_max(heartFeatures)];
    case 2
        % Normalize the data by function z_score().
        normalizedFeatures = [ones(sampleNum,1) z_score(heartFeatures)];
    case 3
        % Normalize the data by function decimal_point().
        normalizedFeatures = [ones(sampleNum,1) decimal_point(heartFeatures)];
    otherwise
        % No normalization will take place
        normalizedFeatures = [ones(sampleNum, 1) heartFeatures];
end
```

The modified hold-out cross-validation and k-fold cross-validation could be used to evaluate the model. Parameter $cvflag$ represents which method will be used. However, k-fold cross-validation is preferred because the improvement after optimization is not obvious.

The user determines the parameter $kfold$. The dataset will be partitioned to $kfold$ parts. A for loop was used to training the model $kfold$ times. Due to the number of samples in the dataset, the samples cannot be divided into groups neatly. Therefore, the last fold will contain 33 samples, while other folds contain 30 samples. The codes are parameterized, thence, it can adapt the different size of the dataset. A parameter named $foldsize$ represents how many samples in each fold calculated by equation (4.2.1).

$$foldsize = round\left(\frac{Number\ of\ samples}{kfold}\right) \tag{4.2.1}$$

```
foldsize = round(sampleNum/kfold);                  % sample number in each fold in k-fold cv
```

```
if cvflag == 1;                                                    % steps for modified hold-out
    for Time = 1:trainingTime                                      % train trainingTime times,
and take the average
        cvpt = cvpartition(heartData.target,'HoldOut',0.3);        % use hold-out to do the par-
tition.
        trainingFeatures = normalizedFeatures(training(cvpt),:);   % according to the random in-
dex
        trainingResults = actualResults(training(cvpt));           % to split dataset into train-
ing set
        testFeatures = normalizedFeatures(test(cvpt),:);           % and test set.
        testResults = actualResults(test(cvpt));
        % codes for optimize weights and calculate the accuracy in each training time.
    end
    % then take average as final results
elseif cvflag == 2;                                % k-fold cross-validation when cvflag == 2.
    for foldIndex = 1:kfold                        % train the model k times
        if foldIndex == kfold;                     % last fold contains 33 or 118 samples
            testk = normalizedFeatures((1 + (foldIndex - 1) * foldsize):end, : );   % treat last
fold as testing set
            testResultsk = actualResults((1 + (foldIndex - 1) * foldsize):end, : );
            temp = normalizedFeatures;
            temp((1 + (foldIndex - 1) * foldsize):end, : ) = [];                    % delete the
testing set from dataset,
            trainingk = temp;                                                       % the rest
will be training set.
            temp = actualResults;
            temp((1 + (foldIndex - 1) * foldsize):end, : ) = [];
            trainingResultsk = temp;
        else                                       % other folds contains 30 or 100 samples
            testk = normalizedFeatures((1 + (foldIndex - 1) * foldsize):(foldsize * fold-
Index), : );    % treat selected part as testing set.
            testResultsk = actualResults((1 + (foldIndex - 1) * foldsize):(foldsize * fold-
Index), : );
            temp = normalizedFeatures;
            temp((1 + (foldIndex - 1) * foldsize):(foldsize * foldIndex), : ) =
[];                    % rest of the dataset is training set.
            trainingk = temp;
            temp = actualResults;
            temp((1 + (foldIndex - 1) * foldsize):(foldsize * foldIndex), : ) = [];
            trainingResultsk = temp;
        end
        % codes for optimize weights and calculate the accuracy in each training time.
    end
    % then take average as final results
end
```

The cost function of the backpropagation neural network in this project is like the cost function of logistic regression. In equation (2.4.4), all parameters are represented in scalar form, but in the implementation, all parameters are in vector form. In the earlier steps, the

architecture of this model has been determined and shown in Figure 4.2.1. To compute the error through cost function and gradient following the procedure in section 2.5. Firstly, calculating the activation of neurons in the hidden layer and the units in the output layer.

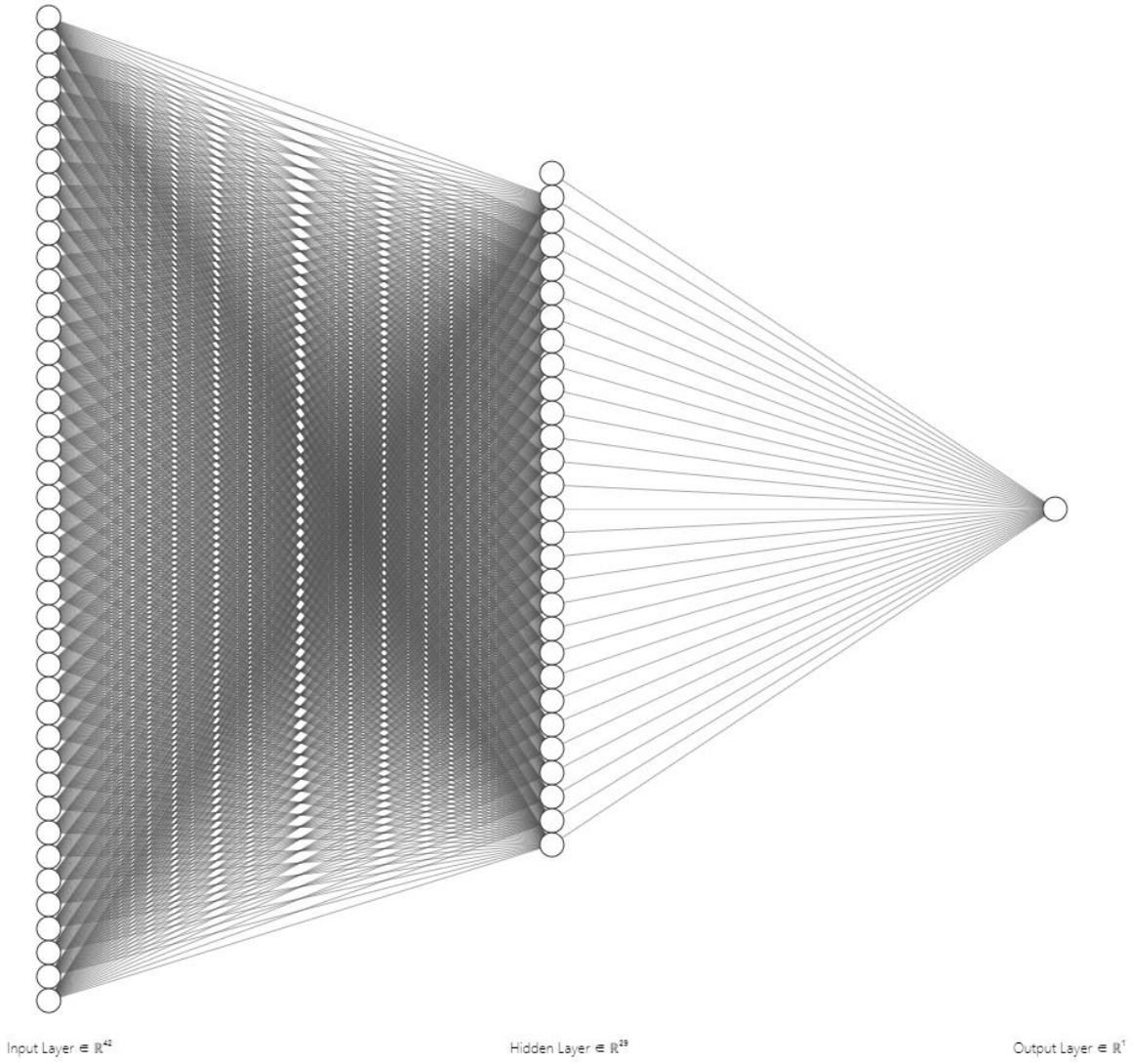


Figure 4.2.1 Architecture of BPNN

Cost function in vector form:

$$J = -sum\left(\frac{\left(y.* \log\left(sigmoid(\theta \times X)\right) + (1-y).* \log\left(1 - sigmoid(\theta \times X)\right)\right)}{(number\ of\ sample)}\right)$$

After obtained the gradients, the θ^{new} could be updated according to equation (4.2.4).

Codes for computation:

```
% compute the cost and gradients by nnCostFun
            [cost(iteration),grad(:,iteration)] =
nnCostFunc(initPara(:,iteration),layer1size,layer2size,layer3size,trainingk,trainingResultsk,lambd
a);
            % store the updated thetas
            initPara(:,iteration + 1) = initPara(:,iteration) - alpha .* grad(:,iteration);
```

Cost function:

```
function [J gradient] = nnCostFunc(Thetas, layer1size, layer2size, layer3size ...
                                   ,X, y, lambda)
    % output 1: cost value; output 2: gradient; arg 1: unrolled theta; arg
    % 2 - 4: size of layers; arg 5: features of training samples; arg 6:
    % targets of training samples; arg 7: regularization parameter
    X = X';
    y = y';
    numOfSamples = size(X,2);   % obtain sample number
    Theta1 = reshape(Thetas(1:(layer2size * (layer1size + 1))), layer2size, layer1size +
1);        % obtain weight 1
    Theta2 = reshape(Thetas((layer2size * (layer1size + 1) + 1):end),layer3size,
layer2size + 1);   % obtain weight 2 from input
    theta1Grad = ones(size(Theta1));    % parameter stores the gradient of theta 1
    theta2Grad = ones(size(Theta2));    % parameter stores the gradient of theta 2
    a1 = X ;                    % activation in firts layer is features of sample
    a2 = [ones(1, numOfSamples) ; sigmoid(Theta1 * X)]; % activation of units in the
hidden layer including bias unit.
    hypothesis = sigmoid(Theta2 * a2);  % activation of last output layer, used to
calculate the error
    delta3 = hypothesis - y;    % error in the output layer
    delta2 = Theta2' * delta3 .* a2 .* (1 - a2);    % error in the hidden layer
    delta2 = delta2(2:end,:);   % error excludes bias unit
    Delta2 = delta3 * a2';      % variable to compute gradient
    Delta1 = delta2 * a1';
    costWithoutRegularization =( - sum(y .* log(hypothesis) + (1 - y) .* log(1-
hypothesis),"all"))/numOfSamples;    % cost value without regularization
    regularizationTerm = (sum((Theta1(:,2:end)).^2,"all") +
sum((Theta2(:,2:end)).^2,"all"))*(lambda/(2 * numOfSamples));
    J = costWithoutRegularization + regularizationTerm;     % total cost
    theta1Grad = Delta1/numOfSamples;                       % compute gradient of each
unit in matrix 1
    theta1Grad(:,2:end) = theta1Grad(:,2:end) + lambda .* Theta1(:,2:end) / numOfSamples;
    theta2Grad = Delta2/numOfSamples;                       % compute gradient of each
unit in matrix 2
    theta2Grad(:,2:end) = theta2Grad(:,2:end) + lambda .* Theta2(:,2:end) / numOfSamples;
    gradient = [theta1Grad(:) ; theta2Grad(:)];             % unroll gradients
end
```

In the BPNN model, the optimization method used gradient descent as well. The steps are the same as the steps in logistic regression before.

After the weight matrices were optimized, a prediction could be made based on the testing dataset.

```
function prediction = predictor(x,theta1,theta2)
    % function to make prediction based on the inputs
    % output: preidciton
    % arg 1: testing features; arg2: weight matrix 1; arg2: weight matrix 2
    [m, n] = size(x);   % size of input dataset.
    a1 = x;             % activation in input layer.
    a2 = [ones(m,1), sigmoid(a1 * theta1')];    % activation of hidden layer
    h = sigmoid(a2 * theta2');  % activation of output layer
    prediction = zeros(m,1);    % initialize prediction value

    for index = 1:m             % make prediction for each sample
        if h(index) >= 0.5;     % threshold is 0.5
           prediction(index,1) = 1;
        else
           prediction(index,1) = 0;
        end
    end
end
```

After this prediction model was trained, it can be installed into a wearable device to make an early prediction of heart disease. Besides, the collected data can be store into Cloud storage. The researcher can utilize those data to update the algorithm by mixing old dataset and new dataset through the above steps.

Chapter 5: Results and Outcomes

At the first time, the model could only be trained by one dataset with 303 samples and 13 attributes.

After building the model with pre-processing methods, the resub-loss is 85.15%, not high. The next step is to test the model. So, one more dataset to test is required. After some days, one dataset with 1190 samples was found which has 11 features also included in the first dataset. However, there was a consideration of lacks feature. Then another dataset with 303 samples and 55 features was found following.

Dataset1: 303 samples, 55 attributes

Dataset2: 1019 samples (reduced from 1190, some are lack of attributes), 11 attributes

Dataset3: 303 samples, 13 attributes

The comparison will be the results from different aspects by using the following 4 sets.

Set1: $readFlag = 0$; training, test: dataset1; cross-validation

Set2: $readFlag = 1$; training: dataset2, test: dataset3

Set3: $readFlag = 2$; training, test: dataset2; cross-validation

Set4: $readFlag = 3$; training, test: dataset3; cross-validation

5.1 Logistic Regression Results

The results of logistic regression in the tables are the average of results from plenty of training times. The result shown in the Figure is the most accurate.

Aspect1: without any pre-processing or optimisation method

Figure 5.1.1 shows the prediction made by the model trained by 4 sets without any pre-processing or optimization method. It can be found that there is a common problem. The prediction misclassified all test samples into a true target for every dataset. It is a highly inaccurate prediction model.

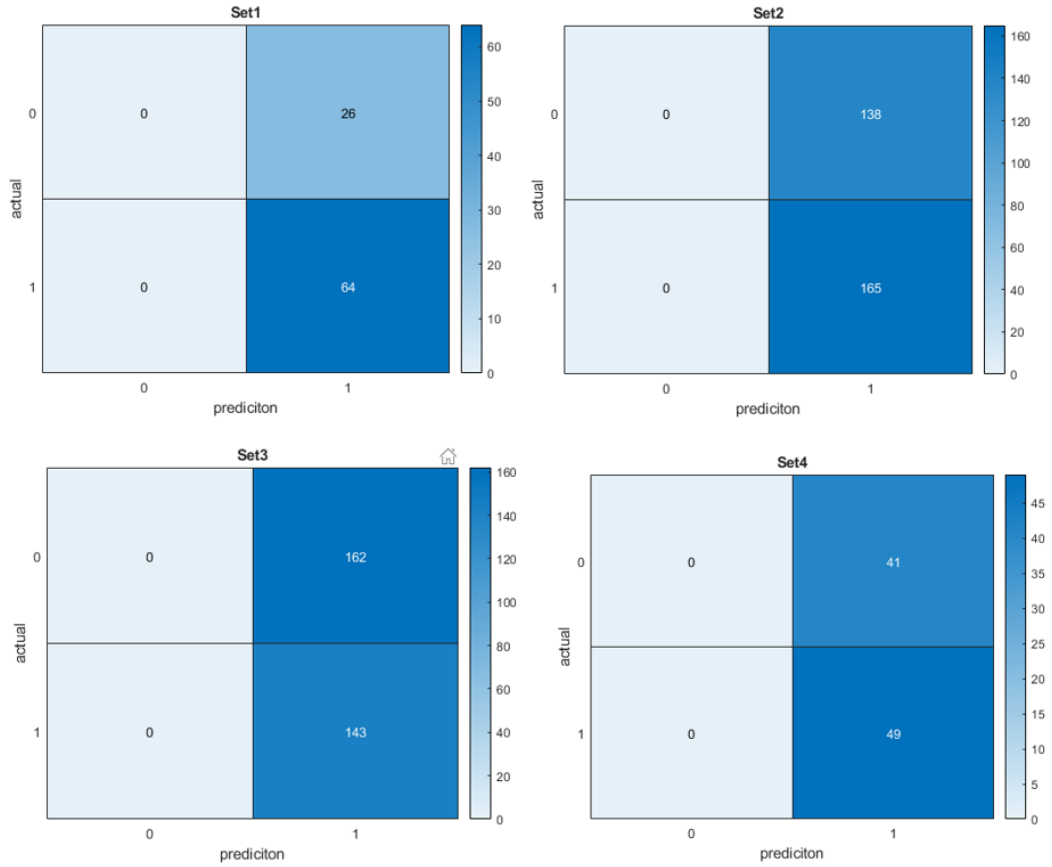


Figure 5.1.1 prediction of 4 sets without methods

Aspect2: with normalisation method

RFS was used in the set1 to make features selection. Because after the analysis of the dataset, several features that do not play a role in the building model, only a few people have those symptoms, no more than 15 samples. It can avoid many misclassifications as in the aspect 1. Min-max normalisation method was used for all sets. Results are shown in table 5.1 and Figure 5.1.2.

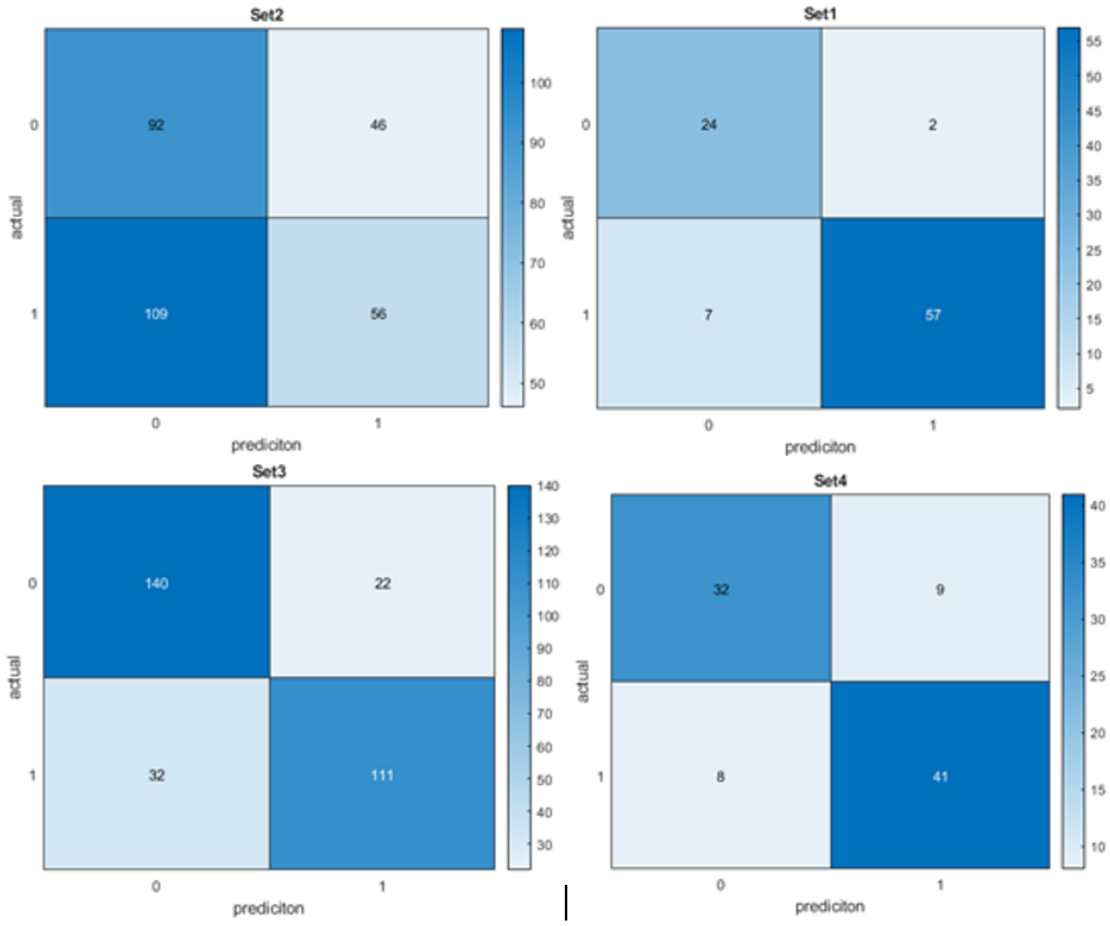


Figure 5.1.2: Cost function

Min-max	*Set1*	*Set2*	*Set3*	*Set4*
Average accuracy	86.33	48.84	82.33	80.44
Average resub-loss	4.46	16.9	16.27	18.26
Average critical error	6.67	35.97	9.02	9.33
Maximum accuracy	90		86.23	84.44

Table 5.1 min-max normalisation

If the RFS method was not applied in the set1, but the dataset was still normalized by the min-max method, the results show in Figure 5.1.3. All the test samples were classified as negative samples. Values of four measurements in table 5.1 are 28.89, 71.36, 71.11 and 28.89, respectively. **Comparing these values with the values of set2 in table 5.1, the performance of the prediction has a significant improvement after RFS applied to optimize the dataset**.

Comparing the results of set3 and set4 with set1, they have lower accuracy, higher resub-loss and critical error. It may be due to the number of features in the dataset. They only have 13 and 11 features. Therefore, the comparison will be between feature construction and without it in aspect3. Furthermore, from the comparison between set3 and set4, the model performs better when the training set has more samples.

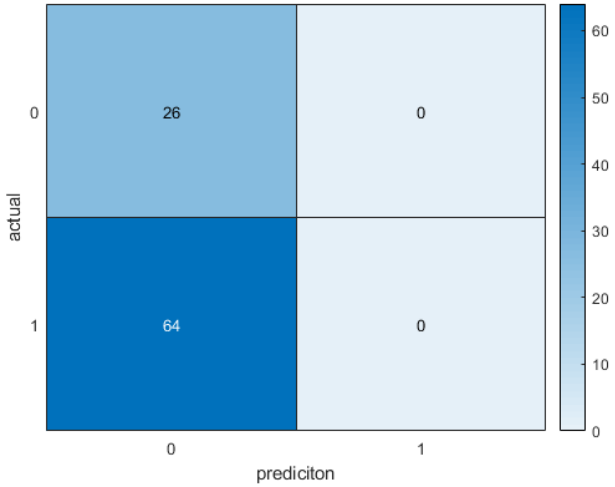


Figure 5.1.3

Z-score	*Set1*	*Set2*	*Set3*	*Set4*
Average accuracy	84.33	44.22	83.7	79.78
Average resub-loss	4.04	16.9	16.73	17.56
Average critical error	6.78	32.67	8.52	8.44
Maximum accuracy	93.33		87.54	84.44

Decimal Point	*Set1*	*Set2*	*Set3*	*Set4*
Average accuracy	87.22	44.22	82.85	80.67
Average resub-loss	8.76	16.6	17.25	18.4
Average critical error	5.67	53.47	8.59	9.33
Maximum accuracy	93.33		86.89	87.78

Table 5.2 Z-score and decimal point

By comparing the results from distinct normalisation methods in table 5.2, the decimal point method could perform better with a dataset has many attributes except the resub-loss. Also, the model trained by dataset1 has the best capability to make a prediction. The measurements are similar in Z-score and Min-max. They could have a good performance in the other machine learning fields.

No matter what normalisation method was applied, the set2 performed the worst, and the resub-loss is much lower than the critical error. It could be due to the standard of measurement, although they have the same features. After analysed two datasets in set2, there is some difference, as the value of the chest pain type is labelled in different standard. Furthermore. The training set was collected between 1992 and 2018, the test set was uploaded in 1987. These two datasets may be not compatible with each other. Consequently, in the rest of the project, this set is removed, and these two datasets are utilized separately.

Obviously, the results are better for the algorithm with normalisation. Because by applying normalisation, the range of the value of attributes will reduce to between 0 and 1. It is much smaller than the original range. All the attributes are consistency.

Aspect3: with feature construction

From aspect2, the result of set3 and set4 have a high resub-loss, underfitting is considered here. Therefore, feature construction was used. Because dataset2 is better to train the learning algorithm, therefore, only dataset2 with z-score normalisation method was used in this section.

After applying modified MTD in dataset2, 5 attributes could be used to construct new attributes, age, resting blood pressure, serum cholesterol, oldpeak and ST slope. Figure 5.1.4 shows the MTD function of those 5 attributes. Table 5.3 shows the results after feature construction. This table also includes the results of the polynomial feature construction method. Because the modified Holdout cross-validation will fluctuate in validation results, a stable validation method is required to compare the little difference, thence, K-Fold method was applied here.

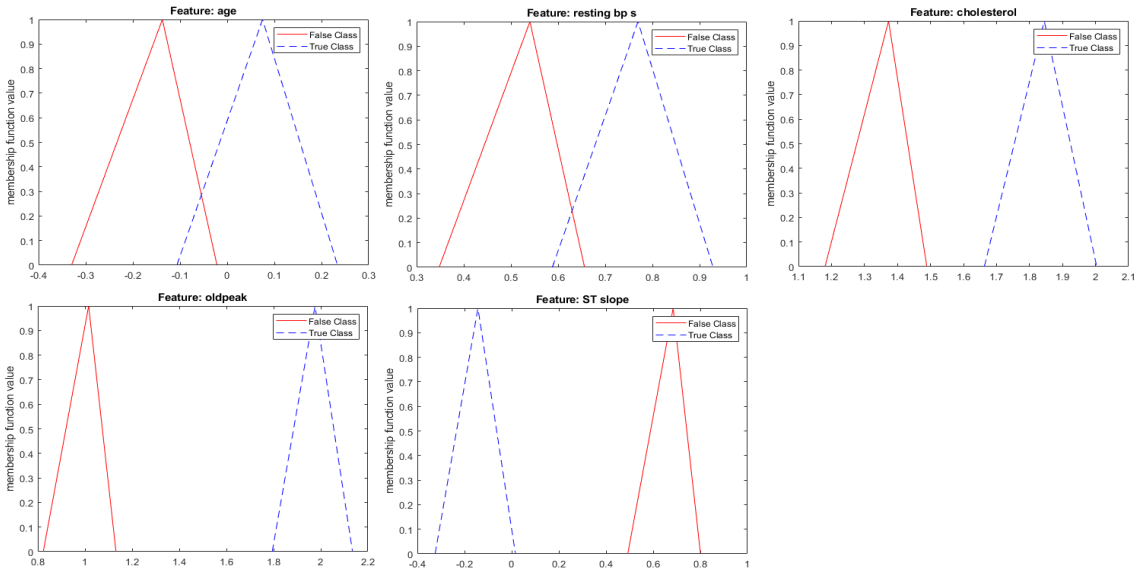


Figure 5.1.4 MTD function

	Without feature construction	Modified MTD features construction	Polynomial feature construction
Accuracy	83.18%	84.8%	84.55%
Resub-loss	16.32%	12.83%	15.3%
Critical error	8.53%	7.76%	7.57%

Table 5.3 comparison between different feature construction by 10-fold

According to the results in table 5.3, the modified MTD feature construction has the best performance in the comparison. It could improve the accuracy and the resub-loss to fix the underfitting problem. If implementing polynomial directly, there will be a high probability of occurring overfitting due to many attributes in the dataset.

Aspect4: comparison with other people's work.

In [35] **used the logistic regression with L2 regularisation, which the cost function is the same as the cost function in this prediction model. The result validated by 10-fold cross-validation is 85.4% which 1.82% lower than 87.22% in this project by using modified hold-out. And the result in this logistic regression model is higher than the result in** [36] **which is 87%.**

The dataset3 is used by many researchers to build a prediction model, in [14] also employed logistic regression with the gradient descent algorithm. **The author validated the model by Holdout method, and the maximum accuracy he obtained is 86.89%, the maximum accuracy in this project of logistic regression is 87.78% which is better. In** [37]**, the accuracy author received is 85.25% which used 5-fold validation method, 2.53% low than the result here.**

For dataset1, there is no research found that used this dataset. Hence, the comparison will be taken between other datasets used in this project and the different pre-processing method or optimization method. **The performance of this dataset is the best throughout all**

pre-processing methods. It has a better prediction result after using normalisation method and MTD feature construction method.

5.2 Backpropagation Neural Network Results

According to the results in section 5.1, the results will be better with the scalar method. Therefore, the comparison in this section will follow a different normalisation method. K-fold cross-validation was applied to evaluate the performance of this prediction model, in case the difference is too tiny to observe. And $k = 10$.

Aspect 1: optimization in the min-max method

When parameter $scalarFlag == 1$, it indicates the min-max normalisation method was used to optimize the prediction model. The threshold in the gradient descent was set to $10e - 6$. Parameter $lambda = 0$. The first comparison will observe how modified gradient descent improves this prediction model. Table 5.2.1 shows the performance of different learning rate (alpha). And the curves of cost value versus iteration are illustrated in Appendix A.

Type	Learning rate	overshoot	Number of iterations reaches minimum	Accuracy	Resub-loss
Keep larger alpha constant	Alpha = 1.4	10 times	Max: 3234 Min: 2894 Average: 3056	84.18%	0%

Keep low alpha constant	Alpha = 0.8	0 times	Max: 4047 Min: 3485 Average: 3820	84.21%	0%
Alpha varies at different stage	Alpha = 1.4, 1.1, 1, 0.95, 0.9 at each stage	2 times	Max: 3200 Min: 2663 Average: 2940	84.24%	0%

Table 5.2.1

From the table, it is easy to find that although if the learning rate is large in the modified gradient descent will overshoot in every training process, the average number of iterations to reach minimum is less than the number in small learning rate test. It means the mechanism to solve overshoot can speed up the computation process at the beginning stage. **The best performance is when the learning rate varies at a different stage, it can reduce overshoot times.** Because every time overshoot occurs, the learning rate will be half of the original value. If less overshoot, quicker computation will be. Furthermore, if there is an overshoot, the iteration value will decrease 100 to recompute the cost value by using new learning rate. It will also add computation time. Therefore, the modified gradient descent with changing learning rate can speed up computation the most in this prediction model.

There is an issue that all types in the table 5.2.1 have 0 resub-loss. It indicates they experience a problem of overfitting with regularisation parameter $lambda = 0$. Because regularisation can improve the generalization ability. Therefore, the next step is to set some values of $lambda$ to settle overfitting and to generalize to other datasets. The results are list in the table 5.2.2.

Performance of different lambda with changing learning rate

Lambda value	Accuracy	Resub-loss	Resub-accuracy
0	84.24%	0%	100%
0.1	85.24%	3.52%	96.48%
0.2	85.21%	4.36%	95.64%

0.5	87.85%	7.74%	92.26%
0.7	87.85%	8.54%	91.46%
1	87.58%	8.8%	91.2%

Table 5.2.2

After transferring the results into curves in Figure 5.2.1. **obviously, with the increase regularisation parameter, the resub-loss decreased, and accuracy raised. When $lambda = 0.5$, this prediction model has the best performance with the highest accuracy and moderate resub-loss.** It implies that regularisation can predict to be generalized to other datasets or samples; not only the training set can have good performance. Otherwise, it cannot make the right prediction when a new sample comes in and delay the treatment of heart disease.

Another advantage is that cost function converges faster. The average iteration decreased from2940 to 637 as lambda increased. In this aspect, BPNN with parameter $lambda = 0.5$, $changing\ learning\ rate$ has the best capability to make a proper prediction for heart disease.

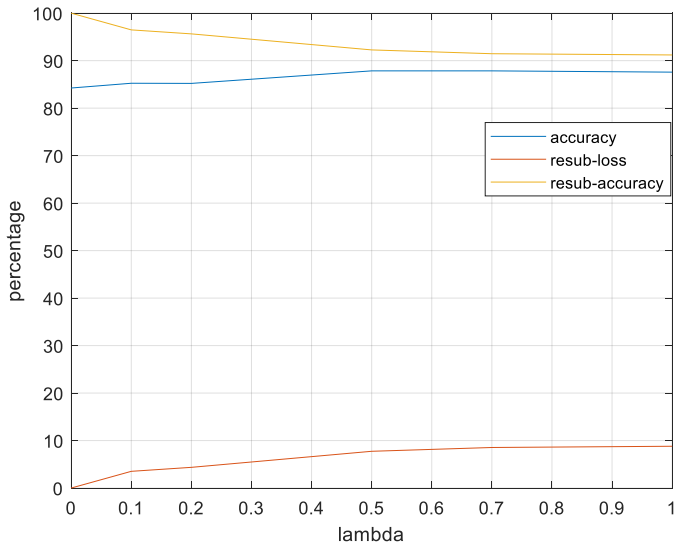


Figure 5.2.1

Aspect 2: z-score normalisation

In the last aspect, changing learning rate during training process can produce the best results. Therefore, in this aspect, no comparison will take between different types of learning rate. When $scalarFlag = 2, lambda = 0$, overshoot occurred only 1 time. The average iteration is 1036. This normalisation method has better performance in computation than the min-max method. The accuracy and resub-loss are 83.85% and 0%, respectively. Suffer from overfitting as well. L2 regularisation was applied to solve this issue. Table 5.2.3 and Figure 5.2.2 illustrate the results.

Performance of different lambda when the training dataset was normalised by z-score method,

Lambda value	Accuracy	Resub-loss	Resub-accuracy
0	83.85%	0%	100%
0.1	84.52%	0%	100%
0.2	84.55%	0%	100%
0.5	84.85%	0.37%	99.63%
1	85.85%	2.86%	97.24%
1.2	85.88%	4.14%	95.86%
1.5	85.55%	5.72%	94.28%
3	86.21%	7.59%	92.41%
5	87.18%	8.73%	91.27%
6	87.18%	9.35%	90.65%
7	86.55%	9.5%	90.5%

Table 5.2.3

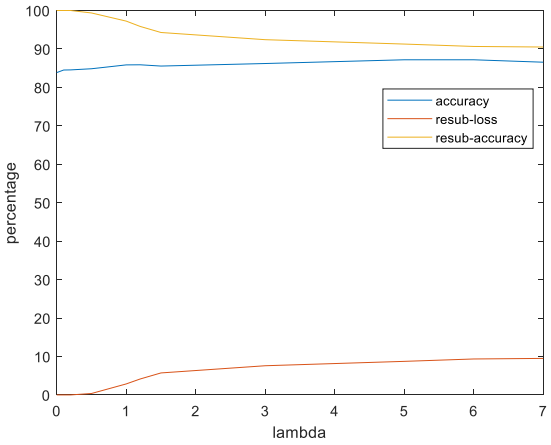


Figure 5.2.2

the prediction model experience overfitting heavily, **by using L2 regularisation, it can overcome this issue, and improve the accuracy from 83.85% to 87.18% while lambda is 5**. Comparing the best results between this section and last section, **training dataset normalized by the min-max method with L2 regularisation can have a better capability to classify the collected data correctly.**

Aspect 3: decimal-point normalisation

The same steps were taken as in the last aspect. When $scalarFlag = 3, lambda = 0$, the accuracy of this prediction model is 87.52%, resub-loss is 8.62%, which has a similar performance comparing to the best results from the previous two aspects. No overfitting was considered.

Then, applying the feature extension to find whether the accuracy of this prediction model can be improved. There is no difference when using modified MTD function to extend the features. Adding quadratic term based on all features can improve the accuracy slightly, accuracy increased to 87.82%, resub-loss decreased to 8.1%.

Therefore, utilizing regularisation can fix the overfitting problem. Afterwards, apply features extension still can improve accuracy and resub-loss. However, when looking into the results

for each $foldIndex$, the highest accuracy could reach 100%, which means no misclassification was made. The lowest is 80%. It could be due to the noise in the dataset.

Comparing the results between the logistic regression model and backpropagation model, the dataset with 303 samples and 55 features and decimal-point normalisation method can always generate the best results in each prediction model. **Overall outcomes in BPNN are better than logistic regression. The best and the most stable results among these training is accuracy=87.82% when applying decimal-point normalisation, regularisation, and feature extension.** As a consequent, if more accurate early prediction is required, the vast feature size and more precise dataset are needed.

Chapter 6: Conclusion and Extension

Conclusion

Heart disease has caused 1/3 death around the world. With the development of the electronic field, various sensors are produced more precise and sensitive. Furthermore, the size of the sensors become tiny. Previously, the collect of ECG signal needs to go to a hospital and use equipment with large volume. Figure 1.1.2 shows the timeline of the medical device for ECG measurement. Nowadays, people can use portable to collect at home.

Because it is convenient to gather the heart data and the growth of data science, it can train an early prediction and detection model easily to monitor the status of heart disease by using a vast number of datasets. Machine learning is an approach to achieve this prediction model. In this project, there are two machine learning algorithms were applied, logistic regression and neural network. And three datasets from internet and UCI Machine Learning Repository.

In the most basic version, the performance of the two algorithms is not good. Extremely low accuracy generated lots of misclassification. In order to raise the prediction accuracy, several methods were applied.

Cross-validation was used to evaluate the capability of the prediction model. K-fold can be employed to observe the tiny improvement because the result is stable. The implementation of modified Hold-out is simple to achieve. It can avoid contingency of a single evaluation to make the result reliable.

When the prediction model experiences an underfitting problem, feature construction approach can create new features based on the existed features to fix this problem. The modified MTD function with adding polynomial terms is able to develop new features according to the importance of each feature. It has a better performance comparing to add polynomial directly. The accuracy increased by 1.8%.

When there is an overfitting issue, it can be solved by applying feature selection and regularisation. The proposed method RFS can select the binary features which play a vital role during the training process. It avoided many misclassifications. Regularisation has good performance in training the neural network, and the accuracy was raised around 3.5%.

The modified gradient descent algorithm was used as the optimisation method for weight units in both learning algorithms. The revision has two mechanisms. The first one is detecting the overshoot. No matter what learning rate is used, it can still converge to a minimum. Otherwise, once the overshoot occurs, the cost function will fail to converge to find the optimal weight units. Another mechanism is changing the learning rate at the different training stage. Therefore, the learning algorithm can accept high learning rate at the beginning and avoid overshoot. According to the result, two mechanisms achieved the goal. Moreover, the modified gradient descent speeds up the training process by up to 23%.

All applied optimisation method improved these two prediction models effectively, the prediction results raised after each technique was used.

Extension

The logistic regression can have further functions. According to the different actions human take, the real value from the logistic regression should be within a proper range if the status is stable. Otherwise, it will give a warning message.

By using more novel optimisation method to improve the capability of early prediction. The higher accuracy, more precise prediction will be. Then, the prediction model can be installed on wearable device to achieve early prediction and detection of heart disease to reduce the mortality caused by heart disease.

References List

[1] P. Binkley, "Predicting the potential of wearable technology," *IEEE Engineering in Medicine and Biology Magazine,* vol. 22, no. 3, pp. 23-27, 2003.

[2] "Key Statistics: Heart Disease in Australia," Heart Fundation, 2018. [Online]. Available: https://www.heartfoundation.org.au/About-us/Australia-Heart-Disease-Statistics.

[3] T. W. C. R. C. W. Group, "World Health Organization cardiovascular disease risk charts: revised models to estimate risk in 21 global regions," *The Lancet Global Health,* vol. 7, no. 10, pp. 1332-1345, 2019.

[4] Y. Zheng, X. Ding, C. Poon, B. Lo, H. Zhang, X. Zhou, G. Yang, N. Zhao and Y. Zhang, "Unobtrusive Sensing and Wearable Devices," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING,* vol. 61, no. 5, pp. 1538 - 1554, 2014.

[5] J. E. R. v. Lennep, H. T. Westerveld, D. W. Erkelens and E. E. v. d. Waa, "Risk factors for coronary heart disease: implications of gender," *Cardiovascular Research ,* vol. 53, no. 3, p. 538–549, 2002.

[6] T. Tabassum and M. Islam, "An approach of cardiac disease prediction by analyzing ECG signal," in *International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, Dhaka, 2016.

[7] N.Kumar, N.Kharkwal, R.Kohli and S.Choudhary, "Ethical aspects and future of artificial intelligence," in *International Conference on Innovation and Challenges in Cyber Security*, Noida, India, 2016.

[8] L.Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development,* p. 601, 1959.

[9] D.Fumo, "Types of Machine Learning Algorithms You Should Know," 15 6 2017. [Online]. Available: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.

[10] A. N. Repaka, S. D. Ravikanti and R. G. Franklin, "Design And Implementing Heart Disease Prediction Using Naives Bayesian," in *3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2019.

[11] A. Dewan and M. Sharma, “Prediction of heart disease using a hybrid technique in data mining classification,” in *2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2015.

[12] M. Raihan, “Smartphone based ischemic heart disease (heart attack) risk prediction using clinical data and data mining approaches, a prototype design,” in *9th International Conference on Computer and Information Technology (ICCIT)*, Dhaka, 2016.

[13] M..Ahmed, S.Mahmud, M.Hossin and S.Noori, “A Cloud Based Four-Tier Architecture for Early Detection of Heart Disease with Machine Learning Algorithms,” in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, Chengdu, China, China, 2018.

[14] C.Dabakoglu, “Heart Disease - Classifications (Machine Learning),” 2019. [Online]. Available: https://www.kaggle.com/cdabakoglu/heart-disease-classifications-machine-learning.

[15] Andrew.Ng, “Machine learning,” Coursera, [Online]. Available: https://www.coursera.org/learn/machine-learning/home/info.

[16] K.Krzyk, “Coding Deep Learning for Beginners — Linear Regression (Part 2): Cost Function,” 2018. [Online]. Available: https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f.

[17] H. Nielsen, “Theory of the backpropagation neural network,” in *nternational 1989 Joint Conference on Neural Networks*, Washington, DC, USA, 1989.

[18] H. Mustafidah, Suwarsito and S. N. C. Permatasari, “Accuracy of the Neurons Number in the Hidden Layer of the Levenberg-Marquardt Algorithm,” *International Journal of Recent Technology and Engineering (IJRTE),* vol. 8, no. 4, pp. 2349-2353, 2019.

[19] H. .Liu and H. Motoda, “Feature Extraction, Construction And Selection: Data Mining Perspective,” *Journal of the American Statistical Association,* vol. 94, p. 1390, 1998.

[20] R. S.Sutton and C. J.Matheus, “Learning Polynomial Functions by Feature Construction,” in *Proceedings of the Eighth International Conference, Pages 208 - 212*, Evanston, Illinois, 1991.

[21] A.Zien, N.Krämer, S.Sonnenburg and G.Rätsch, “The Feature Importance Ranking Measure,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Bled, Slovenia, 2009.

[22] D. Li and C. Liu, “Extending Attribute Information for Small Data Set Classification,” *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 24, no. 3, pp. 452 - 464, 2012.

[23] H. Liu and H.Motoda, in *Feature Selection for Knowledge Discovery and Data Mining*, Boston, Kluwer Academic Publisher , 1998.

[24] D. Signh and S. B, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing Journal,* p. In Progress, 2019.

[25] Z.Mustaffa and Y.Yusof, "A Comparison of Normalization Techiques in Predicting Dengue Outbreak," in *2010 International Conference on Business and Economics* , Kuala Lumpur, Malaysia, 2010.

[26] S.Ruber, "An overview of gradient descent optimization algorithms," 15 Jun 2017. [Online]. Available: https://arxiv.org/pdf/1609.04747.pdf.

[27] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks," in *Eighth International Conference on Intelligent Computing and Information Systems*, Cairo, Egypt, 2017.

[28] J. Kolluri, V. K. Kotte, M. S. B. Phridviraj and S. Razia, "Reducing Overfitting Problem in Machine Learning Using Novel L1/4 Regularization Method," in *4th International Conference on Trends in Electronics and Informatics*, Tirunelveli, India, 2020.

[29] K. G. Sheela and S. N. Deepa, "Review on Methods to Fix Number of Hidden Neurons in Neural Networks," *Mathematical Problems in Engineering,* p. 10.1155/2013/425740, 2013.

[30] J. Heaton, "The number of hidden layers," Heaton Research Inc, 2017. [Online]. Available: https://www.researchgate.net/publication/272508666_The_number_of_hidden_layers.

[31] S. Xu and L. Chen, "A Novel Approach for Determining the Optimal Number of Hidden Layer Neurons for FNN's and Its Application in Data Mining," in *5th International Conference on Information Technology and Applications* , Cairns, Queensland, AUSTRALIA, 2008.

[32] A.Janosi, W.Steinbrunn, M.Pfisterer and R.Detrano, "Heart Disease Data Set," UCI Machine Learning Repository, 01 07 1988. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+disease.

[33] R. Alizadehsani, M. Roshanzamir, M. Abdar, A. Beykikhoshk, A. Khosravi, M. Panahiazar, A. Koohestani, F. Khozeimeh, S. Nahavandi and N. Sarrafzadegan, "A database for using machine learning and data mining techniques for coronary artery disease diagnosis," *Scientific Data,* vol. 6, no. 1, p. 227, 2019.

[34] Z. Sani, R.Alizadehsani and M.Roshanzamir, "Z-Alizadeh Sani Data Set," UCI Machine Learning Repository, 17 11 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Z-Alizadeh+Sani.

[35] M. Siddhartha, “Stacked Ensemble for Heart Disease Classification,” Kaggle, 4 2020. [Online]. Available: https://www.kaggle.com/sid321axn/stacked-ensemble-for-heart-disease-classification.

[36] A. Shivapooja, G. Mounika, P. Sahoo and K. Eswaran, “Efficient System for Heart Disease Prediction by applying Logistic Regression 1,” *International Journal of Computer Science and Technology,* vol. 10, no. 1, 2019.

[37] S. Agarwal, “Heart-disease-classifier-RF-LogRegg-Recall=0.87,” Kaggle, 4 11 2020. [Online]. Available: https://www.kaggle.com/suryanshagarwal/heart-disease-classifier-rf-logregg-recall-0-87.

[38] h. fundation, “heart disease fact sheet,” 2017. [Online]. Available: https://www.heartfoundation.org.au/about-us/what-we-do/heart-disease-in-australia/heart-disease-fact-sheet.

[39] UCI, “heart disease data set,” [Online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+Disease.

[40] D.T.Larose, “Discovering Knowledge in Data An Introduction to Data Mining,” *Wiley Interscience,* pp. 90-106.

[41] Okfalisa, I.Gazalba, Mustakim and N. I. Reza, “Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification,” in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 2017.

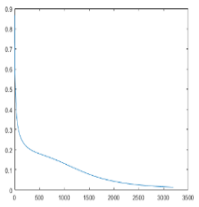
[42] MATLAB, “MATLAB and Simulink Training,” MATLAB, [Online]. Available: https://au.mathworks.com/services/training.html.

[43] SK-LEARN, “Naive Bayes,” SK-LEARN, [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html.

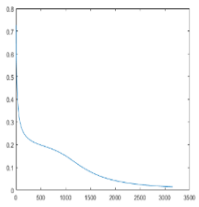
[44] Y.P.Zhang, J.H.Liu, Z.H.Zhang and J.H.Huang, “Prediction of Daily Smoking Behavior Based on Decision Tree Machine Learning Algorithm,” in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Beijing, China, China, 2019.

[45] R.Berwick, “An Idiot’s guide to Support vector,” MIT, Cambridge.

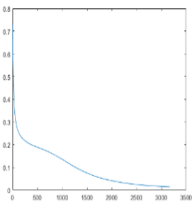
Appendix A



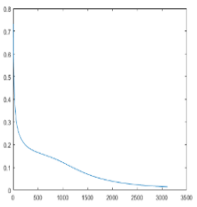
foldIndex = 1



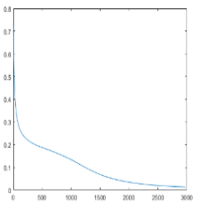
foldIndex = 2



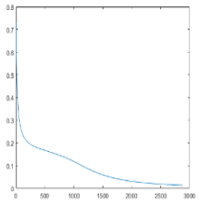
foldIndex = 3



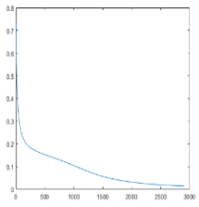
foldIndex = 4



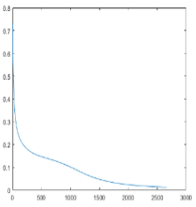
foldIndex = 5



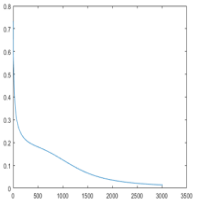
foldIndex = 6



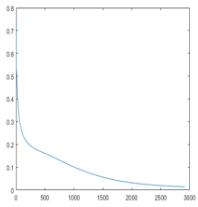
foldIndex = 7



foldIndex = 8

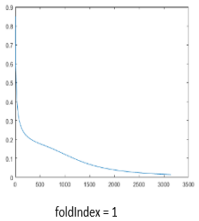


foldIndex = 9

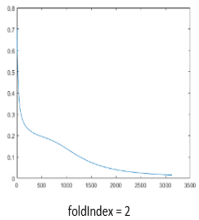


foldIndex = 10

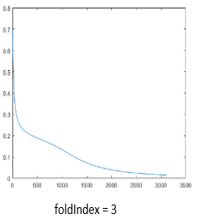
Learning rate varies at each stage



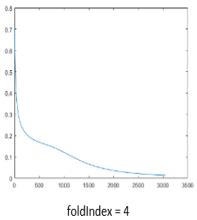
foldIndex = 1



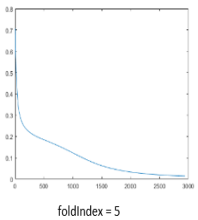
foldIndex = 2



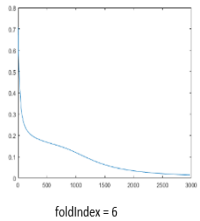
foldIndex = 3



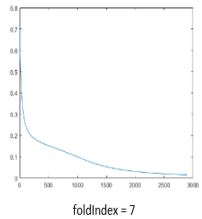
foldIndex = 4



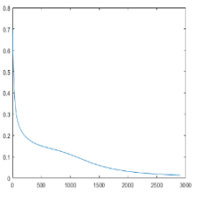
foldIndex = 5



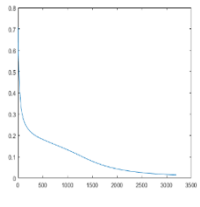
foldIndex = 6



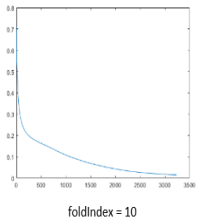
foldIndex = 7



foldIndex = 8

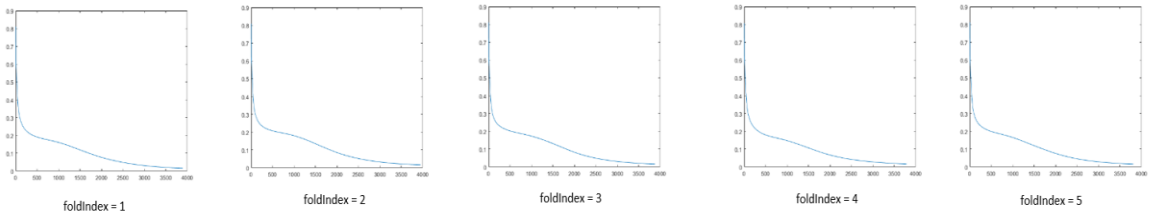


foldIndex = 9

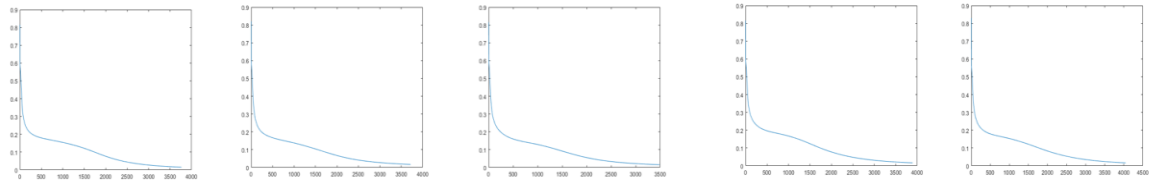


foldIndex = 10

Large learning rate keeps constant



foldIndex = 1 foldIndex = 2 foldIndex = 3 foldIndex = 4 foldIndex = 5



foldIndex = 6 foldIndex = 7 foldIndex = 8 foldIndex = 9 foldIndex = 10

Low learning rate keeps constant