Project Report

**Full Unit – Final Report**

Project Title

Student’s first and last name

**BSc in Computer Science**

**Supervisor:** Supervisor Name

Department of Computer Science

July 23, 2020

# 

**Declaration**

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Student Name:

Date of Submission:

Signature:

Table of Contents

[Abstract 3](#_Toc367967467)

[Project Specification 4](#_Toc367967468)

[Chapter 1: Introduction to the approaches 5](#_Toc367967469)

[Chapter 2: Comparison of approaches 6](#_Toc367967471)

[Chapter 3: Critical parts of algorithms 8](#_Toc367967472)

[Chapter 4: Difficulties faced 9](#_Toc367967473)

[Bibliography 13](#_Toc367967481)

Abstract.

Artificial intelligence is now a part of almost every field and is helping them grow. One such area is of health sciences where the proper diagnosis of heart disease is crucial in providing relevant medical attention in less time than manually diagnosing it. Two most common heart related emergencies are Myocardial infarction and Angina, and it is time consuming to differentiate between them manually. Given two features, I used two different AI methodologies, KNN and single layer perceptron, to classify between the patients of M.I. and Angina. The results drawn from both of the model were 100% accurate while the difference is in the resources each model uses.

Key words: Perceptron, Gradient descent, KNN

Project Specification

White blood count (WBC) and Blood pressure (BP) are the two key factors in determining the kind of heart disease a patient is facing. On basis of these two features, the model will classify between the patients that are having myocardial infarction and the ones troubled by angina.

The first approach that I use is Single Layer Perceptron. It is the simplest form of an artificial neural network.

The second selected method is K Nearest Neighbours (KNN). A machine learning technique that labels a given patient based on the neighbouring majority.

# Introduction to the approaches

## Single Layer Perceptron:

The motivation of this technique is taken from the biological neuron. A neuron in our nervous system takes inputs in form of nerve impulses through dendrites, process them, and finally gives output through its axons. A perceptron works similarly a neuron.

A perceptron takes input which are the features from our data set. It then multiplies those features with some constant numbers which we call weights. After having this dot product, the result of this dot product is then passed onto some activation function which ultimately decides the class to which our patient belongs.

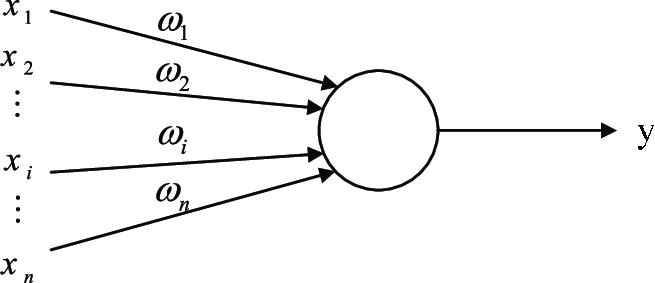


Figure 1: Single layer perceptron

Fig. 1 illustrates the basic architecture of a single layer perceptron. X is a feature vector that contains all the feature of one particular patient i.e. x1, x2, …, xn. In our case this feature vector just contain two features: WBC and BP. The circular region is the layer in which the dot product of our features and pre-selected weights will be taken. Due to this single layer, this model got its name as single layer perceptron.

Hence, Y=W.X or w1x1+w2x2+…wnxn

This Y is called the weighted sum of our features. However, it can not be the final output from our model since it does not give the classification.

Here we introduce an activation function. Activation function gives the probabilities on basis of which our model will be able to properly determine if a feature vector X belongs to a class 0 or 1.

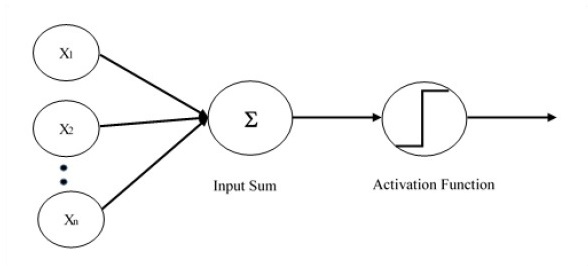


Figure 2 SLP with Activation function

Fig. 2 shows how our model looks like after the introduction of activation function. The weighted sum (y) will then be considered as an input to the activation function. The deliverable of activation function will be compared against a threshold value, and only then it would be possible to classify our feature as class 1 or class 0.

There is a variety of activation functions to choose from, and it all depends upon the nature of problem we are dealing with. Some commonly used activation functions are linear, sigmoid, htan etc.

**Learning rule (Training):**

Trial-and-error is the approach uses by neurons to learn. So is the case with artificial neurons(perceptron). The weights we take are mere guesses and it is very unlikely that the same wights results in accuracy. This is where the concept of training comes from. We train our perceptron to get higher accuracy by changing the weights using a learning rule.

Suppose that a patient has myocardial infarction, but our activation function labels it as Angina patient—this is called an *error*—then our perceptron will stop the further labelling of other patients and will update the weights using a certain learning rule.

The learning will happen as long as most or all of our patients are labelled correctly. Once this is achieved our training will be stopped. It would not be incorrect to say that all of the learning by a perceptron is just to get appropriate weight vector: W.

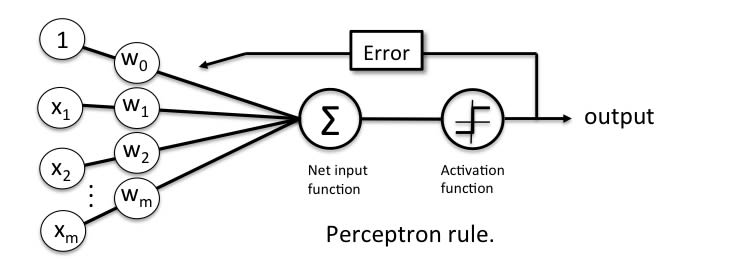
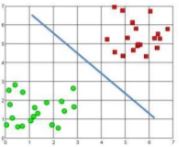


Figure 3 Learning rule

**Decision Boundary:**

The main goal of a perceptron for a classification problem is to obtain a decision surface. This decision surface can be a line, plane, cube. Its nature depends upon the number of dimensions our feature vector has.

Let’s suppose after plotting we get our features vector as the following graph:



All the green circles are the patients with MI while the red ones are Angina patients. It is this blue line that we aim to achieve through perceptron rule. This blue line is called decision boundary.

**Testing:**

This is the valuation phase of our model. Suppose that we started our perceptron training with w1 and w2 as 0.1 and 0.2, and we ended up having these weights with values 1.5 and -0.5. Therefore, the equation of our decision boundary becomes:

Y=1.5x1-0.5x2

Now we will take those points (patients’ features) that were never introduced to the model before. By following the conditions below we will test our model’s accuracy.

If y>0, Label: Angina

Else if y<0, Label: MI

Then we will compare our model’s findings with the pre-obtained values from data set and will check the accuracy of our model.

## K nearest neighbours:

A simple machine learning technique that classifies a feature vector based on the similarity it has with its neighbouring features. It works on the assumptions that all similar things are near to each other.

**Algorithm:**

Load our data set.

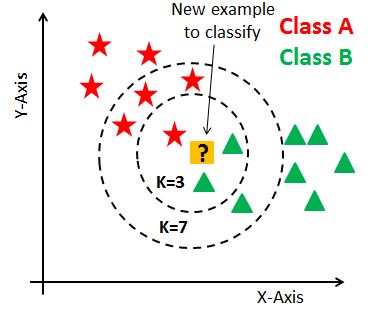
Select the value of K (no of neighbours).

Now calculate the distance of our test feature with all of the loaded features.

Store these distances and sort them in ascending order.

Take first K features (ones with shortest distances i.e. nearer)

Now take the mode of the classes these K features belong to, and assign this label to our test feature.



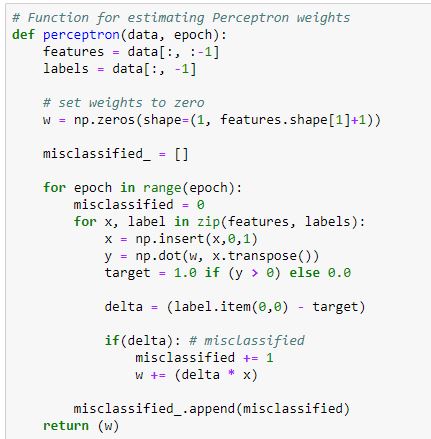
Suppose that the red stars are MI patients and green ones are Angina patients. The yellow feature is the one that we need to classify. It is the value of K that will play pivotal role in determining its class. If I take value of k=3, KNN will classify it as Angina. Whereas a value of k=7 will label it as MI.

# Comparison of approaches

* k-NN is exceptionally straightforward and requires tuning only one hyperparameter (the estimation of k), while SLP training includes numerous hyperparameters controlling the size and structure of the system and the improvement methodology.
* Once a SLP is trained, the training information isn't any longer required to provide new predictions. this is often clearly not the case with k-NN.
* Once a SLP is trained on one task, its parameters are used as an honest initializer for an additional (similar) task. this can be a kind of transfer learning that can't be achieved with k-NN.
* SLP have achieved the state of the art in additional domains than k-NN. (This doesn’t essentially mean neural networks can work higher on your specific problem, however through empirical observation neural networks are effective in several settings.)
* KNN relies upon the neighbours as it were. It doesn't require a training stage. Whereas SLP utilizes weights of every neuron as for the information. So, they need a bigger training stage to get more precision

# Critical Parts of algorithms

## Single Layer Perceptron



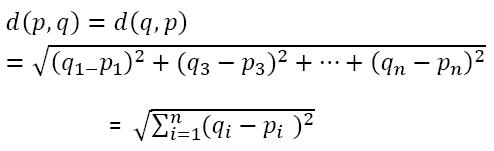
This snippet is taken from my implementation of perceptron rule. All the learning of perceptron and weight updating is encapsulated inside this perceptron() function. It takes our training data and no of epochs (i.e. iterations). The weights initially are set as zeros. Variable y is the weighted sum of our features. We then compare our predicted value with the target(actual) value, and if our results are not correct, we update our weights. *delta* stores the difference between actual and predicted classes and if it’s anything but zero, w gets updated by getting product of delta with feature and then adding it to the current value of w. This function also records the number of misclassifications in a certain epoch and finally returns the weights that correctly classifies our all feature vectors.

## KNN:

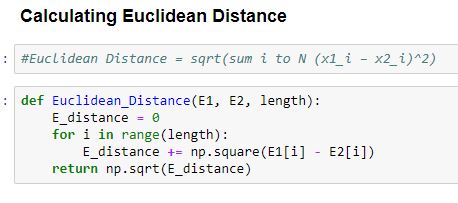
Euclidean distance:

Calculating the distance between two vectors(features) is the key to getting K neighbours of our test feature. There are many ways to find distance between the two points on a plane. The Euclidean distance, however, is the most commonly used and efficient option for the task.

From basic co-ordinate geometry, we know that the formula to calculate distance between two points is:



Where n is the no of dimensions our point has, which, in our case, is 2.



The function Euclidean\_Distance() takes two points E1 and E2, and then takes the number of dimension as *length.* It then uses numpy’s functions np.square() and np.sqrt() to achieve the calculation necessary for Euclidean distance.

# Difficulties Faced

* No built-in ML libraries were used for the implementation of algorithms.
* To implement algorithms from their mathematical formulas was also one of the challenges faced.
* The selection of dataset was a tedious task since our data set was not taken from any open source database. It was taken from the book on image processing while the test data was generated using random generators.