"Breast Cancer Detection using Neural Network"

by Team Proton, University of Windsor

January 05, 2020

Abstract

The paper examines the root cause of breast cancer through its characteristics like Clump Thickness, Cell Size & Shape, Marginal Adhesion, Class (2 for benign, 4 for malignant) and many more using 699 instances. The data is taken from the UCI Machine Learning Repository. The data is then cleaned as there are 16 missing values. Missing values can be handled by either putting the mean value of the column or putting the most occurred values. Then class values are replaced with 0 and 1 instead of 2 and 4 respectively. The next step is to split the data in which data is divided as training data and testing data. We have used the split ratio of 80-20. Further, feature scaling is performed and that standardizes the range of features of data. The last main step is model generation. For which we are using the Keras library written in Python. With Sequential() function we will initiate the Neural Network and with Dense() function we will add input and hidden layers. We need an activation function as well as an optimizer for our Neural Network, for which we are currently researching. We have used ReLU as an activation function and Adam(Adaptive moment estimation) as an optimizer.

Introduction

According to the World Cancer Research Fund, Breast Cancer was the most common cancer in women all over the world in 2018. The number of Breast Cancer cases diagnosed in 2018 was 2,088,849 and it is 25.4% of the total number of new cases diagnosed. This is our primary motivation behind building a breast cancer classifier that can detect early-stage cancer tumors.

We will take data from UCI Machine Learning repository which has 699 instances, which contains several features/parameters such as Clump Thickness, Cell Size & Shape, Marginal Adhesion, Class (2 for benign, 4 for malignant) and many more. All the features, except class, contain values from 1 to 10. The simple interpretation is: if the value is near 1 that's a benign tumor (safe) and if close to 10, it's malignant (unsafe).

Methodology

Understanding Data

We took data from the UCI Machine Learning repository which has 699 instances, which contains 10 features/parameters as described below.

Feature	**Value**
Sample code number	Id number
Clump Thickness	1–10
Uniformity of Cell Size	1–10
Uniformity of Cell Shape	1–10
Marginal Adhesion	1–10
Single Epithelial Cell Size	1–10
Bare Nuclei	1–10
Bland Chromatin	1–10
Normal Nucleoli	1–10
Mitoses	1–10
Class	2 for benign, 4 for malignant

Table 1 Features and their values

Data Cleaning and Preprocessing

Data cleaning and preprocessing is an important part of every machine learning model present on earth. Without accurate models could not predict accurate results. With the help of robust data sources, for example, UCI Machine Learning Repository, we save a lot of time from cleaning the data. Out of 699 instances of the breast cancer data, only 16 has missing values.

Data Cleaning

There are several ways in which we can handle missing values and clean the data. For example, replacing it with the *most occurred value* in the column or replacing it with the *mean of the column*. In the dataset, 1 and 10 are the most occurring values, therefore we have two values that are most occurring and can not use any one of them. Hence, we replaced those missing values with the mean of the column.

Also, for the ease of understanding, we converted the class values. Here in the data, we have value 2 for class benign(safe) and 4 malignant(unsafe) which we converted to 0 for benign and 1 for malignant.

Data Pre-processing

There are several steps in pre-processing that can be done to ensure our model is accurate and can obtain better results, we will perform these two:

Splitting the data in Machine Learning is a method to divide training and testing data. The data is mainly divided into two parts — Training and Testing (the third split is validation). Training data is for training our algorithm and Testing data is to check how well our algorithm performs. We are keeping the split ration 80-20.

Feature Scaling is a method that will standardize the range of features of data. It helps when we have large numerical values to compute. However, we have values between 1 to 10, therefore it won't make a huge difference. Even so, we are using it because it speeds up gradient descent, which we are using as an optimizer for our network[1].

Model generation and training

We built a neural network using Keras to generate predictions. With the Sequential() function of Keras, we will initiate the Neural Network and with Dense() function we will add input and hidden layers. As there is no rule of thumb for how many hidden layers to choose, we are taking 2 layers with 5 and 3 units respectively.

Activation Function

Activation functions are an important part of any neural network. Activation function decides whether a neuron should be activated or not by calculating a weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron [2]. There are mainly two types of Activation functions:

1) Linear Activation Function

2) Non-linear Activation Functions

Linear or Identity Activation Function

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range. It doesn't help with the complexity of various parameters of usual data that is fed to the neural networks.

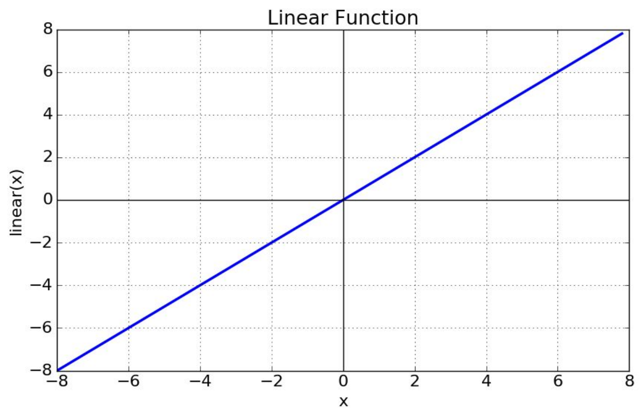


Figure 1 Linear Function

Non-linear Activation Function

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to make the graph look something like this. It makes it easy for the model to generalize or adapt with a variety of data and to differentiate between the output.

Derivative: Change in y-axis w.r.t. change in the x-axis. It is also known as a slope.

Monotonic: A function that is either entirely non-increasing or non-decreasing.

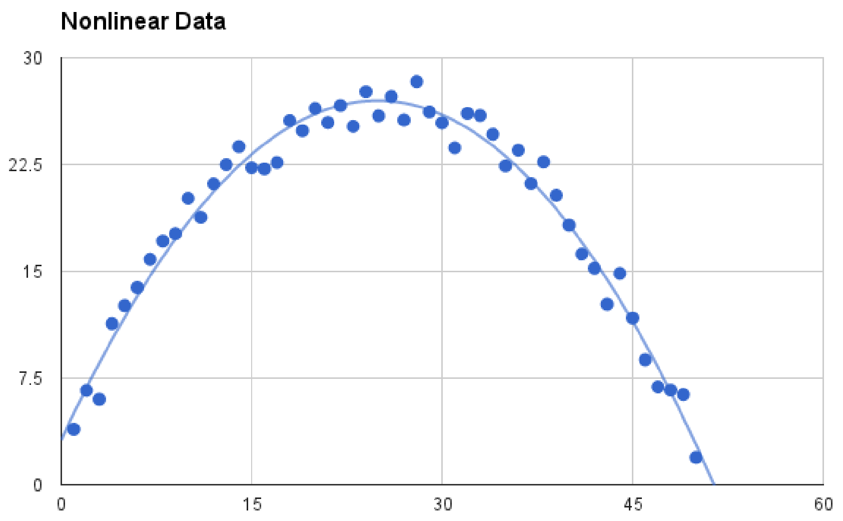


Figure 2 Non-linear function

Sigmoid or Logistic Activation Function

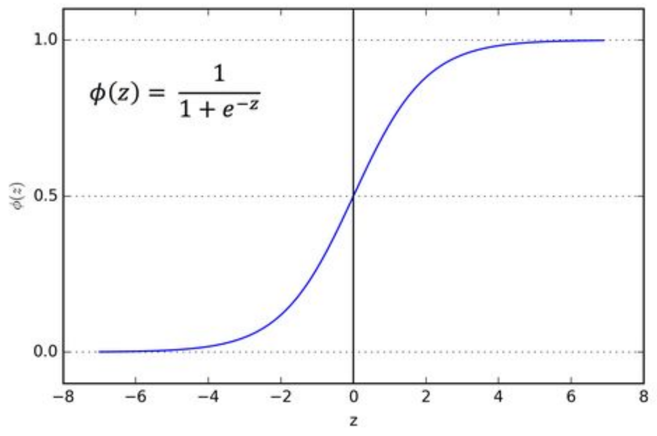


Figure 3 Sigmoid Function

The main feature of the sigmoidal function is that it works between 0 and 1. So where the probability prediction comes, it is very useful as probability always lies between 0 and 1. The function is differentiable which means, we can find the slope of the sigmoid curve at any two points. The function is monotonic but the function's derivative is not.

Tanh or hyperbolic tangent Activation Function

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s-shaped).

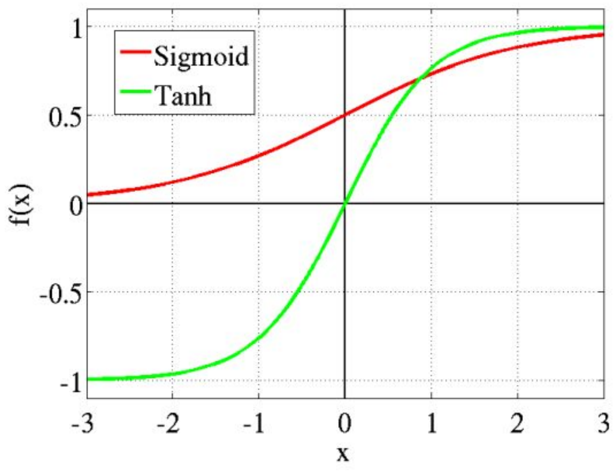


Figure 4 Tanh function

The function is differentiable and monotonic while its derivative is not monotonic. The tanh function is mainly used for classification between two classes.

3). ReLU (Rectified Linear Unit) Activation Function:

The ReLU function is a commonly used activation function. ReLU function is half rectified. The function value is zero when z is less than zero and function is equal to z when z is above or equal to zero. The function and its derivative both are monotonic. Be that as it may, the issue is that all the negative qualities become zero promptly which diminishes the capacity of the model to fit or prepare from the information appropriately[2].

Optimizer

Each time a neural system wraps up a group through the system and creating forecast results, it must choose how to utilize the contrast between the outcomes it got and the qualities it knows to be consistent with modifying the loads on the hubs so the system ventures towards an answer. The calculation that discovers that progression is known as the optimization algorithms.

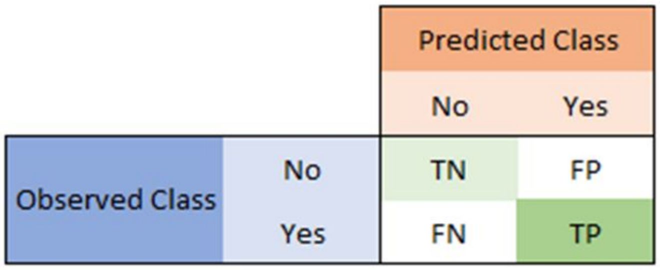
Adam: Adaptive Moment Estimation

Adam is a first-order gradient-based optimization algorithm it gives better optimization results at present than other similar algorithms. It is based on RMSProp and AdaGrad. It is popular because of its straightforward implementation and its computational efficiency, it works best for problems where there is a large dataset or more parameters.

We have used ADAM because it requires relatively low memory requirements (though higher than gradient descent and gradient descent with momentum) and it usually works well even with the little tuning of hyperparameters.

Results

To check the accuracy of our model, we have used a confusion matrix. A confusion matrix is nothing but a 2x2 matrix indicating all correct and incorrect values. Here, TN is True Negative, which means the actual output was no and our model also predicted no. TP is True positive, I think I don't have to explain that now. These both are our correct answers and the other two are false. To find the accuracy, we will add the number to correct observations and divide it by the total number of observations.



		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

Figure 5 Confusion Matrix [3]

		Predicted Class	
		No	Yes
Observed Class	No	**82**	**3**
	Yes	**1**	**54**

Here, in our test data, we have 140 entries. Out of which our confusion matrix says we have 82+54 = 136 correct observations and 3+1 = 4 incorrect. Therefore the accuracy of our model on new training data would be 136/140 = 0.9714 which is 97.14% accuracy.

Conclusion

Breast Cancer is the most common cancer in women. Building a classifier to detect cancer in the early stage can save lives around the globe. We tried creating a neural network model that predicts whether the cancer cell is benign or malignant using data from the UCI machine learning repository. Things we learned include how neural network works, what an activation function is, how optimizers work, how to measure the accuracy of the generated model. In the end, we achieved 97.14% in our first attempt. However, there is still room for growth and we can still improve our model beyond this point.

References

[1] Retrieved December 26, 2019, from https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e

[2] Retrieved December 27, 2019, from https://www.geeksforgeeks.org/activation-functions-neural-networks/

[3] Retrieved December 29, 2019, from http://scaryscientist.blogspot.com/2016/03/confusion-matrix.html