Using Perceptron to Predict Diabetes

Kamalnain Taank
University of Adeliade
North Terrace Adelaide SA 5000
a1805326@student.adelaide.edu.au/

Abstract

A human can learn how to perform a task but computers must be programmed to do so. But programming is a very time consuming and error prone task. A perceptron, invented in late 1950, can help machine to learn to perform tasks using some examples. It is the simplest learning machine and a basic building block of Neural Network that can perform complex tasks if fed enough examples. In this paper we are going to predict whether a person is diabetic or not based on certain diagnostic measurements provided in the dataset named Pima Indians Diabetes Dataset(PIDD) sourced from UCI repository and is originally from the National Institute of Digestive and Kidney Diseases.It is a chronic and hereditary disease considered as a serious health problem which can lead to death. There are several constraints considered whiles selecting data points in this dataset. All the samples are from females aged minimum 21 years old. This paper will cover most of the basic elements and mathematics behind a perceptron and also implementations of those methods in python. We will be comparing some competing approaches to the problem as well.

1. Introduction

Diabetes is a disease in which a person suffers from high blood sugar levels. Insulin is a hormone produced by pancreas which regulates blood sugar level and helps glucose obtained from food to be carried into body cells to be used for production of energy. There There are basically 2 main reasons for that: 1 - there is insufficient production of insulin, also called type1[1]. Some treatments and a healthy lifestyle can help people to manage their condition. 2- body cells refuse to respond to the insulin produced in the body. Therefore lack of insulin or insensitivity of cells towards insulin is the key factor leading to diabetes. In diabetes a person suffers from prolonged thirst and hunger, frequent urination, blurry vision and poor wound healing. If the disease remains untreated, it can lead to kidney damage, eye damage and increased risk of heart strokes. Many researches are developing machine learning technique or classification algorithms to diagnose diabetes. Some of the techniques employed for this task are SVM [2], Naive Bayes[3] and Decision Tree. Machine Learning can handle large datasets and do complex mathematical computations. It is gaining popularity and various organizations investing ample amount of resources in it. In this project we are going to predict whether a person is diabetic or not with the help of data points present in the dataset using a single layer perceptron with Hyperbolic tangent function as our output function.

1.1. Perceptron

In machine Learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.[4]. In simple terms, perceptron is an algorithm for learning a binary classifier called a threshold function that maps input **x** (a real-valued vector) to an output $f(x)$. Here we are using Hyperbolic Tangent Function as our activation function to because the outcomes given in the PIDD sourced from UCI repository are 1 and -1. The equation for $f(x)$ is as follows:

$$\begin{aligned} iff(x) &= tanh(w.x + b) \geq 0 \\ thenf(x) &= 1 \\ elsef(x) &= -1 \end{aligned}$$

where $\underline{w}$ is a vector of real-values weight, **w.x** is a dot product $\sum_{i=1} w_i.x_i$, where *m* is the number of inputs to the perceptron and *b* is the bias. The value of $f(x)$ will be used to identify whether a person is diabetic or not.

1.2. Dataset

The Pima Indian Diabetes Dataset from UCI is a collecction of 768 records of female patients aged at least 21 years old of Pima Indian Heritage, it has 9 columns and **0** being the outcome column with values 1 for positive results and -1 for negative results. 500 of the results are positive and 268

being negative. The data is scaled and all the values varies from **[-1,1]**. See:

	0	1	2	3	4	5	6	7	8
0	-1	-0.294118	0.487437	0.180328	-0.292929	-1.000000	0.001490	-0.531170	-0.033333
1	1	-0.882353	-0.145729	0.081967	-0.414141	-1.000000	-0.207153	-0.766866	-0.666667
2	-1	-0.058824	0.839196	0.049180	-1.000000	-1.000000	-0.305514	-0.492741	-0.633333
3	1	-0.882353	-0.105528	0.081967	-0.535354	-0.777778	-0.162444	-0.923997	-1.000000
4	-1	-1.000000	0.376884	-0.344262	-0.292929	-0.602837	0.284650	0.887276	-0.600000

2. Background

2.1. Related work

There are various studies being conducted on this dataset by various researchers and various types of classification algorithms have been tried on this dataset but no-one performs too well. Results of an experiment conducted by Deepti Sisodia and Dilip Singh Sisodia 2018[5] using different classification problems are as follows:

Classification Algorithms	Precision	Recall	F-Measure	Accuracy %	ROC
Naive Bayes	0.759	0.763	0.760	76.30	0.819
SVM	0.424	0.651	0.513	65.10	0.500
Decision Tree	0.735	0.738	0.736	73.82	0.751

It can be be clearly observed that among SVM and Decision tree and Naive Bayes, the accuracy percentage of the Naive Bayes is the highest which is 76.3. And we want to identify diabetes as effectively as possible so Naive Bayes Classification algorithm here shows more promise for the real world test.

2.2. Advantages And Disadvantages of Naive Bayes Classification

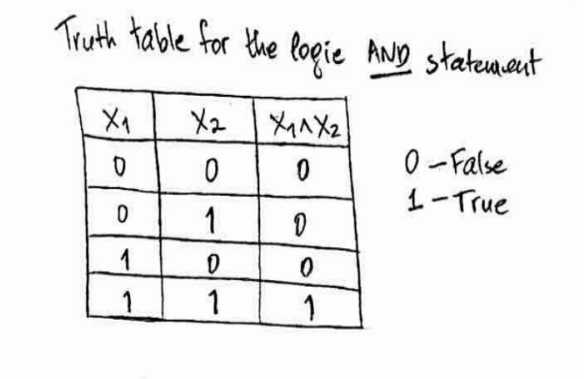
When assumption of independent predictors holds true, a Naive Bayes classifier performs better as compared to other models. It requires a small amount of training data to estimate the test data. So, the training period is less. And Naive Bayes is also easy to implement. But main problem with the Naive Bayes is the assumption of independent predictors. Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it is almost impossible that we get a set of predictors which are completely independent. And If categorical variable has a category in test data set, which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as Zero Frequency.[6]

2.3. Advantages And Disadvantages of Perceptron

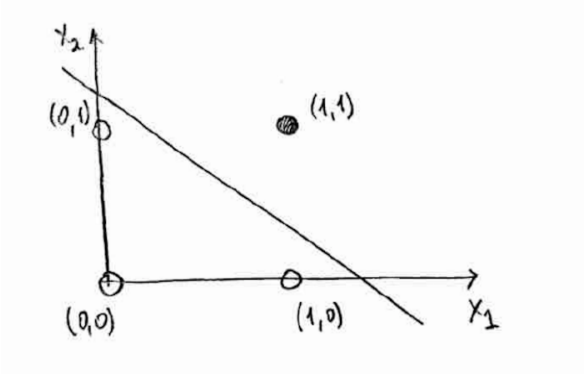
Single Layer Perceptron is quite easy to set up and train.It outputs a function which is can be Sigmoid or Hyperbolic Tangent and that activation function can easily be linked to posterior probabilities.And We can interpret and input the output as well since the outputs are the weighted sum of inputs. It always finds hyperplane which can divide linerarly separable data points in two parts.

3. Methodology

Logically, every statement has a truth value i.e. every statement is true or false[7]. Complex statements also have truth values but to find that, we first breakdown complex statement into simple statement then we can answer these simple statement as true or false and using logical operators like AND or OR we can find the truth value of the complex statement. Also for that we need to know with which rule logical operator should be used to find truth value of complex statement. Suppose we use AND operator to find the truth value of complex statement. Example is given below:



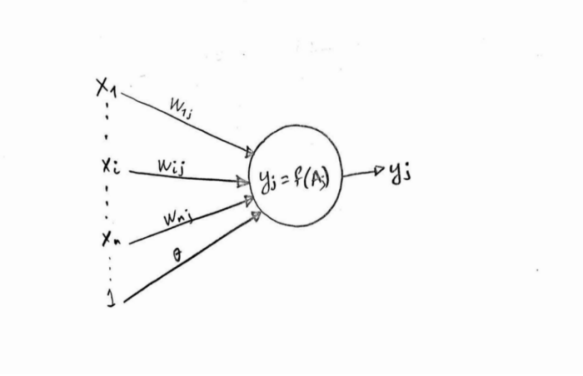
We can observe that for final complex statement to be true, all the inputs are needed to be true for AND operator. If we want to model this truth table we will first plot this table of AND statement:



It can be seen that there is a line which can separate our result for truth values of AND statement. And this line which can separate these data points can be represented as a function called activity function. And that is our true goal while training a Perceptron.

4. Algorithm

A simple illustration of perceptron is given below:



Inputs are multiplied by weights and and then in the perceptron we will apply activation function to it to get the desired output. There are 3 things which we want to find-1: Right weights w_ij. , -2:Right Activation Function $f(A)$ and -3: Bias θ. We will use Hyperbolic Tangent Function as our activation function because it ranges from -1 to 1 and values of our datapoints are aligned with that. Also there will be a threshold value 0 for predicting output. See:

$$iff(x) = tanh(w.x + b) \geq 0$$
$$thenf(x) = 1$$
$$elsef(x) = -1$$

And here w is a vector of real-values weight, **w.x** is a dot product $\sum_{i=1} w_i.x_i$, where *m* is the number of inputs to the perceptron and *b* is the bias. The value of $f(x)$ will be used to identify whether a person is diabetic or not. We will Initialize weights using **Xavier's Initialization**:

$$\sqrt{\frac{1}{size^{[l-1]}}}$$

$$W^{[l]} = np.random.randn(size\_l, size\_l\text{-}1) * np.sqrt(1/size\_l\text{-}1)$$

We use this to avoid vanishing or exploding gradient. It will set weight near 1 so the value of **w.x +b** will not get too high and it will also help in avoiding slow convergence and prevent missing the global minimum. We will be generating a random value for bias. To calculate the error we have devised a error function

$$e_j = d_j - y_j$$

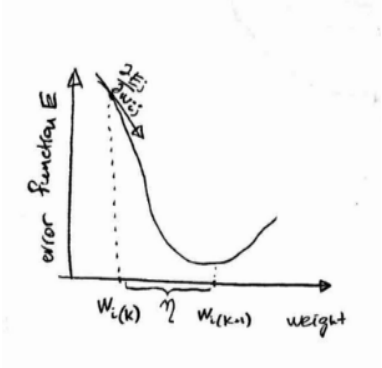
. e_j is error, d_j is desired output and y_j is actual output. We can see that the error function related to desired output and actual output. Here desired output is our predicted output. But there is a chance that the error function might give negative values so we will square the error function and divide it by 1/2 just to make the calculations easier.

$$E_j = 1/2e_j^2 = 1/2(d_j - y_j)^2$$

Now we will adjust the weights after getting results to further reduce the error while back propagating . The expression for updated weight based on the current current value of weight and error function E_j looks like this:

$$W_{ij(}k+1) = -\eta\frac{\partial E_j}{\partial W_ij}$$

$W_{ij(}k+1)$ is updated weight, $W_{ij(}k)$ is current weight, η is the step size or we can say the learning rate and $\frac{\partial E_j}{\partial W_ij}$ is derivative of error function with respect to weight W_j. It is also called **Gradient Descent** which helps in finding the minimum of the function.



Now we will find partial derivative of E_j with respect to W_{ij} using chain rule.

$$\frac{\partial E_j}{\partial W_ij} = \frac{\partial E_j}{\partial e_j} * \frac{\partial e_j}{\partial y_j} * \frac{\partial y_j}{\partial A_j} * \frac{\partial A_j}{\partial W_{ij}}$$

where

$$y_j = (e^x - e^{-x})/(e^x + e^{-x})$$

and

$$A_j = \sum_{i=1}^{n} w_{ij} * X_i + \theta$$

we get

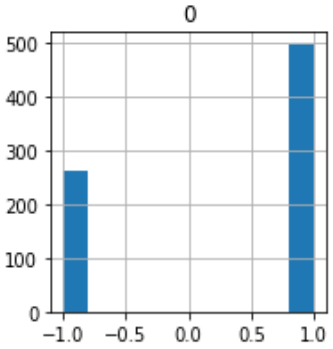
$$\frac{\partial E_j}{\partial W_ij} = -e_j * [1 - (\frac{e^x - e^{-x}}{e^x + e^{-x}})^2] * X_i$$

And that is the equation which we will use to update our weights or so to say train our weights.

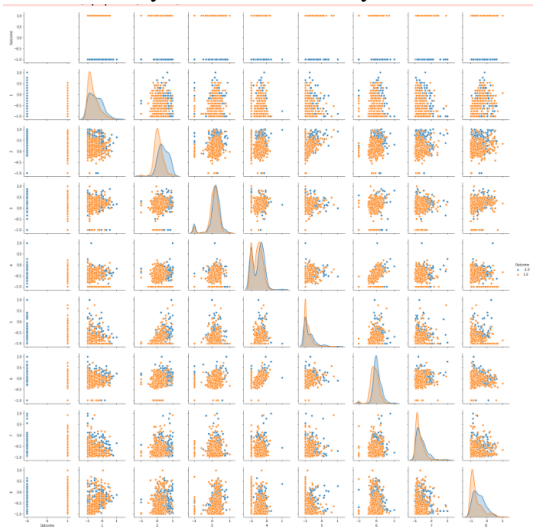
5. Experimental Analysis

First, the dataset is checked for missing values. Second, using histogram we checked whether our data is balanced or not i.e. ratio of both the classes because if the classes are

unbalanced then our model will get biased towards single class.



Then plotted pairplot clearly indicated that no pair of features can clearly draw a boundary between two classes.



The training set contains 80% of the data points as the dataset is a To iterate over the rows and columns of dataframe and to avoid potential errors during runtime, we converted the dataframe into matrix using funtion to numpy(). We first ran the model for 1 epoch then for 10 and up to 200 and observed that after increasing the accuracy has not effected model accuracy by significant amount. Started from 65% now reached 70.4% after 150 epochs and does not change much after 150 so we left it there.

6. Code

The code for this experiment is given on this Link. Please inform if you cannot access the link.

7. Conclusion

After increasing the epochs while training weights, there was no significant change in the accuracy, meaning our model does not converge a lot from it's starting point. Current accuracy is 70.39% and comparing this to the accuracy of classification algorithms made by Deepti Sisodiaa Dilip Singh Sisodiab 2018, perceptron algorithm stands after SVM Classifier which has the accuracy of 76.30%, which means that Perceptron algorithm performed quite well. Overall the algorithm can be further improved if we have more data points and if we work with hidden layers i.e. Multi Layer Perceptron (MLP) having different activation function like Sigmoid function.

7.1. References

[1]. American Diabetes Association Reports of the Experts Committee on the Diagnosis and Classification of Diabetes Mellitus. Diabetes Care. 2001;23:S4–19

[2]. Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., Chouvarda, I., 2017. Machine Learning and Data Mining Methods in Diabetes Research. Computational and Structural Biotechnology Journal 15, 104–116. doi:10.1016/j.csbj.2016.12.005.

[3]. Dhomse Kanchan B., M.K.M., 2016. Study of Machine Learning Algorithms for Special Disease Prediction using Principal of Component Analysis, in: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication, IEEE. pp. 5–10.

[4]. Freund, Y.; Schapire, R. E. (1999). "Large margin classification using the perceptron algorithm" (PDF). Machine Learning. 37 (3): 277–296. doi:10.1023/A:1007662407062.

[5]. Deepti Sisodiaa Dilip Singh Sisodiab, National Institute of Technology, G.E Road, Raipur and 492001, India. Prediction of Diabetes using Classification Algorithms Analysis, in: International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

[6]. Niresh Kumar, Advantages and Disadvantages of Naive Bayes in Machine Learning Accessed from: http://theprofessionalspoint.blogspot.com/2019/03/

[7]. Ani Karenovna K May 14, 2019, The Perceptron — A Building Block of Neural Networks Accessed From: https://blog.usejournal.com/the-perceptron-the-building-block-of-neural-networks-5a428d3f451d