Final Report

MAT399 - Undergraduate Seminar

Adhitya Swaminathan
1910110025

Contents

Abstract

Given a binary classification problem, we want to predict accurately which class a given data point belongs to using a Logistic Regression model and bring out the corresponding maths. We then implement Principal Component Analysis(PCA) and compare the Logistic Regression model with Random Forest Model and Support Vector Machine(SVM) with the criteria being the accuracy of each model. For this, we use the Heart Disease prediction dataset from Kaggle.

Now we extend to a multiclass classification problem and build a neural network to model the given data. Here, we use the FIFA 19 dataset and build a neural network which predicts the position of a player given parameters such as their skill level, accuracy of shot, etc. While doing so, we again observe in detail, the mathematical workings behind the model.

Introduction to Binary Classification

The goal here is to build a model which predicts whether a person has a heart disease or not given a dataset containing various medical parameters.

This is called a binary classification problem since our end result is either 0(no heart disease present) or 1(presence of heart disease).

Dataset

The dataset 12 columns and 1025 rows.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Notation Used

y: Target Variable

X: Predictors

n: Number of features

m: Number of training examples

w: Weights

b: Bias

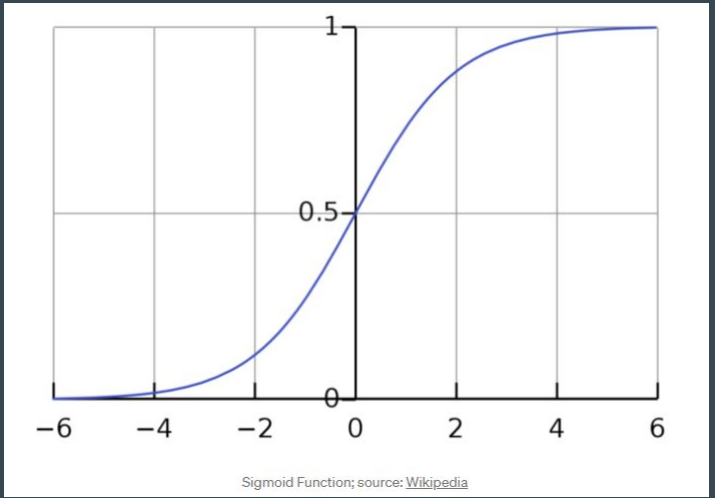
y_hat: Hypothesis(Takes values between 0 and 1)

Sigmoid Function

Sigmoid function maps any input to any value between 0 and 1. The function is as follows:

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

source: Andrew Ng



Sigmoid Function; source: Wikipedia

Log Loss function

Every machine learning algorithm is based on a loss function which needs to be minimized. Here in our case, we need a loss function which adapts to a Binary Classification problem. Hence we use the Log loss function.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[(y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})]$$

Binary Cross-Entropy Loss Function; source: Andrew Ng

Explaining the Algorithm

Here, we are modelling $P(Y=1 \mid X=x)$

In the logistic model : $P(Y=1 \mid X=x) = \frac{e^{\beta_0 + \beta^T x}}{1 + e^{\beta_0 + \beta^T x}}$

$\beta \in (\beta_1, \beta_2, \ldots, \beta_{12}) \in \mathbb{R}^{12}$

$\therefore P(Y=0 \mid X=x) = 1 - P(Y=1 \mid X=x)$

Suppose we have only one predictor variable:

$P(Y=1 \mid X=x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$, $0 \leq RHS \leq 1$

If $P(Y=1 \mid X=x) > P(Y=0 \mid X=x)$, then $X=x$ is assigned to target value 1.

Loss function : $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

$= -\frac{1}{m} \sum_{i=1}^{m} [(y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$

J → overall cost.
L → cost for i^{th} training example

Here we are taking $P(Y=0 \mid X=x) = (1 - \hat{y}\_hat)$
$P(Y=1 \mid X=x) = \hat{y}$

Now we calculate parameters w (weights) & b (bias) through the gradient descent algorithm:

$w = w - lr(\text{learning rate}) \cdot dw$

$dw = \left(\frac{1}{m}\right) \cdot (\hat{y} - y) \cdot x$

$b = b - lr \cdot db.$

$db = \left(\frac{1}{m}\right) \cdot (\hat{y} - y)$

Note: dw does not contain X

Introduction To PCA

PCA(Principal Component Analysis) is a popular dimensionality reduction technique used to pre-process data in Machine Learning.

Using PCA, we can represent the dataset as linear combinations of principal components which are vectors. The first principal component explains the greatest amount of variance in the original features. The second component is orthogonal to the first and it explains the greatest amount of variance left after the first component.

Here, we are using PCA to see how the accuracy of our Logistic Regression model is affected once we transform the dataset.

Pros And Cons of PCA

Pros of PCA:

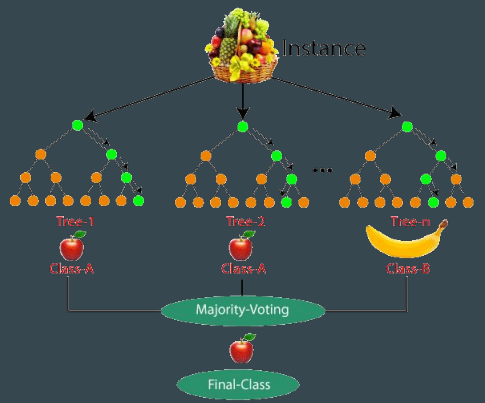
- Removes correlated features
- Faster training time
- Reduces overfitting
- Easy to visualize

Cons of PCA:

- Less interpretable
- Information Loss

Random Forest Model

The Random Forest Model works mainly by the concept of decision trees. A tree consists of different decision levels and branches, which are used to classify data.The Decision Tree algorithm tries to divide the training data into different classes so that the objects within a class are as similar as possible and the objects of different classes are as different as possible.It builds decision trees on different samples and takes their majority vote for classification



Pros and Cons of Random Forest

Pros:

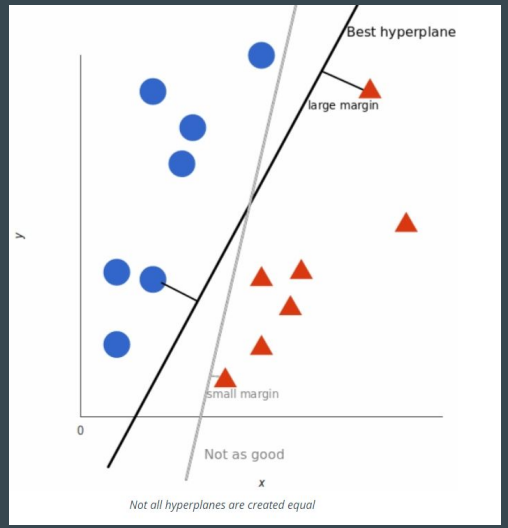
- Handles both continuous and categorical data.
- Not sensitive to outliers
- Works well on huge datasets.
- Works well on non-linear data.

Cons:

- Slow training
- Not suitable for linear datasets with less features.

SVM(Support Vector Machines)

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. Given a point (X,Y), we need a classifier which tells us which class this point belongs to. A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as *class 1*, and anything that falls to the other as *class 2*



Not all hyperplanes are created equal

Pros and Cons of SVM

Pros:

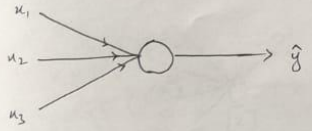
- Works well in high dimensions
- Very effective when number of dimensions > number of training examples
- Memory efficient

Cons:

- Prone to noise in dataset i.e., one set of classes are more than the other.
- Training time is higher in large datasets.

Intro to Neural Networks

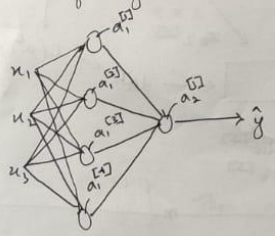
Consider a neuron.



Its function is to take in input x and output $\hat{y}$.

A neural network consists of many layers each consisting of nodes. An input x is passed and calculation takes place through each layer and finally $\hat{y}$ is outputted.

Consider the following neural network:



This is a neural network consisting of input layer x, two hidden layers each consisting of 4 and 1 neuron respectively and output layer $\hat{y}$.

Notation: $a_i^{[l]}$ ⟶ activation function corresponding to the i^{th} node in l^{th} layer.

Detailed Computation of Neural Networks

Consider the previously shown neural network.

We have feature vector a_0 or $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

1) $z_1^{[1]} = w_1^{[1]T} a_0 + b_1^{[1]}$ where w's are weights and b is the bias term as seen in the simple logistic regression model.

$a_1^{[1]} = \sigma(z_1^{[1]})$ ← activation function.

$z_2^{[1]} = w_2^{[1]} a_0 + b_2^{[1]}$

$a_2^{[1]} = \sigma(z_2^{[1]})$

⋮

$z_4^{[1]} = w_4^{[1]} a_0 + b_4^{[1]}$

$a_4^{[1]} = \sigma(z_4^{[1]})$

Here, $a_1 = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$

2) $z_1^{[2]} = w_1^{[2]} a_1 + b_1^{[2]}$

$a_1^{[2]} = \sigma(z_1^{[2]}) = \hat{y}$

3) $\mathcal{L}(\hat{y}, y)$ is calculated.

Steps 1–3 are called forward Propagation.

Here, we still have to keep updating the weights and bias term to minimize our cost.

∴ from $\mathcal{L}(\hat{y}, y)$, we can get $d\mathcal{L}/d\hat{y}$ & $d\mathcal{L}/da^{[1]}$.

From this the following calculations are made:

$dz^{[l]} = da^{[l]} \cdot g^{[l]}(z^{[l]})$

$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$

$db^{[l]} = dz^{[l]}$

$da^{[l-1]} = w^{[l]T} dz^{[l]}$

This is known as back propagation

Neural Network Algorithm Flowchart