**Machine Learning**

**Bootcamp**

Project report

17/01/24

Presented by:**AAditya POrwal**

**Introduction:**

In this project report, we will discuss the implementation and performance evaluation of five machine learning algorithms: Linear Regression, Polynomial Regression, Logistic Regression, K-Nearest Neighbours (K-NN), and Neural Networks. These algorithms will be applied to different datasets and their performance will be compared based on various metrics.

**Tech Stack:**

* Jupyter Notebook
* .csv files
* .npy files

**Libraries used:**

* NumPy
* Pandas
* Matplotlib.pyplot

**Linear Regression**

Linear regression is a regression algorithm used to model the relationship between a dependent variable and one or more independent variables. The goal is to find the linear relationship that best describes the data, which is achieved by fitting a straight line through the data points that minimizes the sum of the squared errors between the observed and predicted values.

In simple linear regression, there is only one independent variable, and the equation of the line can be represented as

                                                                                   𝑦 = 𝑚𝑥 + 𝑐

In multi-variate linear regression, there are multiple independent variables, and the equation of the line is represented as

                                                                     𝑦 = 𝑚0 + 𝑚1𝑥1 + … + 𝑚𝑛𝑥𝑛

where 𝑦 is the dependent variable or hypothesis, 𝑥𝑖 are the independent variables, and 𝑚𝑖are the coefficients.

In matrix form:

                                                                                   𝑌 = 𝑀T. 𝑋

     𝑌 is the matrix of the batch of outputs, 𝑀 is the matrix denoting the weights and biases, and 𝑋 is the batch of all independent variables.

**Standardization:**

Standardization is a data pre-processing technique used to transform data in such a way that it has zero mean and unit variance. The goal of standardization is to make the data comparable.

**Feature Standardization:**

𝑥𝑖: = (𝑥𝑖− 𝑥̅𝑖 )

𝜎

, where 𝑥𝑖 is the 𝑖𝑡ℎ feature, 𝑥̅𝑖 is the mean of all 𝑖𝑡ℎ features and 𝜎 is the standard deviation of all 𝑖𝑡ℎ features.

**Cost Function:**

Cost function used is the Mean Squared Error (MSE). It calculates the average squared difference between the predicted and actual values.

The formula for MSE is:

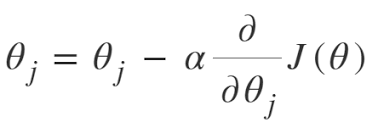
                                                            J = (1/2𝑏) ∗ ∑ (𝑦𝑖− 𝑦̂𝑖 )²

Where J is the loss, 𝑏 is the number of data points in the dataset, 𝑦𝑖 is the actual value, and  𝑦̂𝑖 is the predicted value.

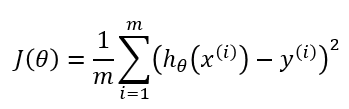
**Batch Gradient Descent Algorithm:**

Batch Gradient Descent (BGD) is an optimization algorithm used to minimize the cost or loss function in machine learning. It is a type of gradient descent that computes the gradient of the cost function over the entire training dataset to update the model's parameters

Weights are updated as:



, where ∝ is the learning rate.



**R2 Score:**

The R-squared score, also known as the coefficient of determination, is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s). In other words, it measures the goodness of fit of a regression model to the data.

Mathematically:

                                                                 𝑅2𝑠𝑐𝑜𝑟𝑒 = 1 - 𝑆𝑆𝑟𝑒𝑠

                                                            𝑆𝑆𝑡𝑜𝑡

                                                                 𝑆𝑆𝑟𝑒𝑠 = ∑𝑏𝑖=1(𝑦𝑖 − 𝑦̂𝑖)²

                                                                 𝑆𝑆𝑡𝑜𝑡 = ∑𝑏𝑖=1(𝑦𝑖 − 𝑦̅)²

, where 𝑦𝑖 is the actual value, 𝑦̅ is the mean of all 𝑦I, and 𝑦̂𝑖 is the predicted value.

**Result:**

After 500 iterations of the code, the R2 score came out to be 84.45% and the loss converged to approximately 4779.97 ,with a learning rate of 0.03.

**Polynomial Regression**

Polynomial regression is an extension of Linear Regression that allows for modelling non-linear relationships between the outcome variable and predictor variables by modelling it as an nth-degree polynomial.

In univariate polynomial regression, there is only one independent variable, and the equation can be represented as

𝑦 = 𝑚0 + 𝑚1𝑥1 + 𝑚2𝑥12 + ⋯ + 𝑚𝑛𝑥1𝑛

In multivariate polynomial regression:

𝑦 = 𝑚0 + 𝑚1𝑥1 + 𝑚2𝑥12 + ⋯ + 𝑚𝑛𝑥1+ 𝑚𝑛+1𝑥2 + 𝑚𝑛+2𝑥22 + ⋯ + 𝑚2𝑛𝑥2𝑛+ . ..

, where 𝑥1,𝑥2 ,. .. are the independent variables and 𝑚𝑖 is the weight or coefficient.

In Matrix form:   𝑌 = 𝑀𝑇. 𝑋

Y is the matrix of the batch of outputs, M is the matrix denoting the weights and bias, and X is the batch of

all independent variables and their different combinations.

**Standardization:**

Standardization is a data pre-processing technique used to transform data in such a way that it has zero mean and unit variance. The goal of standardization is to make the data comparable.

**Feature Standardization:**

𝑥𝑖: = (𝑥𝑖− 𝑥̅𝑖 )

𝜎

, where 𝑥𝑖 is the 𝑖𝑡ℎ feature, 𝑥̅𝑖 is the mean of all 𝑖𝑡ℎ features and 𝜎 is the standard deviation of all 𝑖𝑡ℎ features.

**Label Standardization:**

Reduces the no. of iterations required for convergence.

𝑦 

𝜎

, where 𝑦 is the label, 𝑦̅ is the mean of all labels and 𝜎 is the standard deviation of all labels.

**Cost Function:**

Cost function used is the Mean Squared Error (MSE).

It calculates the average squared difference between the predicted and actual values.

The formula for MSE is:

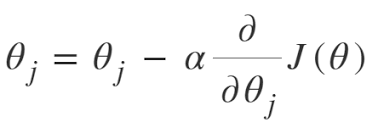
                                                            J = (1/2𝑏) ∗ ∑ (𝑦𝑖− 𝑦̂𝑖 )²

where J is the loss, 𝑏 is the number of data points in the dataset, 𝑦𝑖 is the actual value, and  𝑦̂𝑖 is the predicted value.

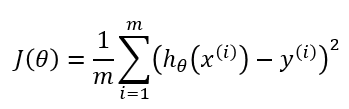
**Batch Gradient Descent Algorithm:**

Batch Gradient Descent (BGD) is an optimization algorithm used to minimize the cost or loss function in machine learning. It is a type of gradient descent that computes the gradient of the cost function over the entire training dataset to update the model's parameters

Parameters are updated as:



,where ∝ is the learning rate.

Cost is calculated as : 

**R2 Score:**

 The R-squared score, also known as the coefficient of determination, is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s). In other words, it measures the goodness of fit of a regression model to the data.

Mathematically:

                                                                 𝑅2𝑠𝑐𝑜𝑟𝑒 = 1 - 𝑆𝑆𝑟𝑒𝑠

                                                            𝑆𝑆𝑡𝑜𝑡

                                                                 𝑆𝑆𝑟𝑒𝑠 = ∑𝑏𝑖=1(𝑦𝑖 − 𝑦̂𝑖)²

                                                                 𝑆𝑆𝑡𝑜𝑡 = ∑𝑏𝑖=1(𝑦𝑖 − 𝑦̅)²

, where 𝑦𝑖 is the actual value, 𝑦̅ is the mean of all 𝑦I, and 𝑦̂𝑖 is the predicted value.

**Result:**

The most optimum degree was found out to be 5.

The R2 score of the model exceeded 99.995% after 10000 iterations and reached almost 100% after 1000000 iterations with a learning rate of 1e-5.

The time taken to train model for 10000 iterations was 37 seconds.

**Logistic Regression**

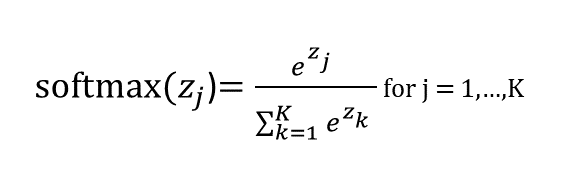
Logistic regression is an algorithm used for classification problems.

**SoftMax Regression:**

SoftMax Regression is an algorithm used for multiclass classification problems by using the softmax function to generate probability distributions of each class or label.

            𝐹𝑜𝑟 𝑦̂𝑖 = 𝑚𝑇.𝑥𝑖

, where z𝑖 denotes the predicted output of 𝑖𝑡ℎ input, 𝑚 is the matrix of weights for each label and 𝑥𝑖 is the matrix of input features.

  
The softmax function exponentiates the predictions, making any negative values positive, and then divides it by the sum of all exponentiated labels, thus generating a probability distribution over all labels.

**Normalization:**

Normalization is a process of scaling the data so that it falls within a specified range, typically between 0 and 1.Upon analysing the labelled data, the noticeable things were that the 784 features were pixel values ranging from 0 to 2. Uponplotting the 784 pixels into a grid of 28 × 28 pixels by using the pixel values as grayscale values, different wearable items such as clothes, shoes, bags, etc.

To normalize the data, we can divide each feature by the maximum of all values i.e., 255

X = X/255.0

**Cost Function:**

The cost function used here is the Cross-Entropy Loss.

Cross entropy loss is a measure of the distance between the predicted probability distribution and the true probability distribution. The loss is calculated as the negative log likelihood of the true class labels, given the predicted class probabilities.

                                               𝐶𝑟𝑜𝑠𝑠 𝐸𝑛𝑡𝑟𝑜𝑝𝑦 𝐿𝑜𝑠𝑠 = 𝐽 = − ∑ 𝑦𝑖 . log (𝑦̂𝑖  ) + (1 − 𝑦𝑖 ). log(1 − 𝑦̂𝑖)

, where 𝑦𝑖  is the One-Hot Encoded matrix form of the true label of the 𝑖𝑡ℎ sample and 𝑦̂𝑖 is the matrix of probability distributions for each label of the 𝑖𝑡ℎ sample.

**Mini-Batch Gradient Descent Algorithm:**

Mini-Batch Gradient Descent is a variant of the Gradient Descent Algorithm in which instead of computing the gradient on the entire dataset, the dataset is divided into smaller subsets called mini-batches. The gradient is computed on each mini-batch separately, and the model parameters are updated based on the each gradient across all the batches. This process is repeated for a number of iterations, where each iteration consists of one pass through the entire dataset.

**Accuracy:**

The accuracy is measured by direct comparison of predicted labels of the labelled test features and test labels.

𝐴𝑐𝑐𝑢𝑟𝑎𝑐𝑦 = ( 𝑁𝑜. 𝑜𝑓 𝑚𝑎𝑡𝑐ℎ𝑒𝑑 𝑐𝑎𝑠𝑒𝑠× 100 ) %

𝑇𝑜𝑡𝑎𝑙 𝑐𝑎𝑠𝑒𝑠

**Result:**

The Accuracy of the model converged at around 82.98% after 1000 iterations with a learning rate of 1 and 100 minibatches.

Time taken by the model to run up to 1000 iterations was around 5 mins.

**K-Nearest Neighbours**

K-nearest neighbours (K-NN) is a supervised learning algorithm used in machine learning for classification and regression tasks.

The K-NN algorithm works by finding the K closest data points (i.e., nearest neighbours) to the new data point that needs to be classified or predicted. The value of K is a hyperparameter. Once the K nearest neighbours are identified, the algorithm takes a majority vote for classification tasks or a mean for regression tasks to predict the label or value of the new data point.

**Normalization:**

Normalization is a process of scaling the data so that it falls within a specified range, typically between 0 and 1.

Upon analysing the labelled data, the noticeable things were that the 784 features were pixel values ranging from 0 to 255. Upon plotting the 784 pixels into a grid of 28 × 28 pixels by using the pixel values as grayscale values, different wearable items such as clothes, shoes, bags, etc.

To normalize the data, we can divide each feature by the maximum of all values i.e., 255

X = X/255.0

**Distance:**

The distance used for the algorithm is the square of the Euclidean distance to reduce computational cost.

                                                            𝐷𝑖𝑠𝑡𝑎𝑛𝑐𝑒 = 𝑑 = (𝑥 − 𝑎)2

, where x and a are values of the same feature.

For all features:

                                                                                    𝑑 = ∑(𝑥𝑖 − 𝑥̂𝑖)2

, where 𝑥𝑖 and 𝑥̂𝑖 are values of the same feature.

**Loss:**

Cost function used is the Mean Squared Error (MSE).

It calculates the average squared difference between the predicted and actual values.

The formula for MSE is:

                                                             J = (1/2𝑏) ∗ ∑𝑏𝑖=1(𝑦𝑖 − 𝑦̂𝑖)²

Where J is the loss, 𝑏 is the number of data points in the dataset, 𝑦𝑖 is the actual value, and  𝑦̂𝑖 is the predicted value.

**Result:**

The optimum value of K was found to be 4 by plotting Loss vs K over 1000 examples from the training and testing labelled datasets.

**Neural Networks**

Neural Networks are a family of algorithms that are inspired by the structure and function of the human brain. They consist of layers of interconnected neurons that can learn to recognize patterns in data. They can be used for Regression and Classification.

**Neurons:**

A Neuron in a neural network functions as:  It takes an input, multiplies it with weights, adds a bias to it and then applies an activation function on the output. The neuron sends this activated output forward to each neuron of the next layer.

                                                 𝑧 = 𝑥. 𝑤 + 𝑏

     𝑎 = 𝐴𝑐𝑡𝑖𝑣𝑎𝑡𝑖𝑜𝑛(𝑧)

, where 𝑥 is the input to the neuron, 𝑤 is the weights, 𝑏 is the bias,𝑧 is the output and 𝑎 is the activated output.

**Activation Function:**

Activation function as a mathematical operation applied to the outputs of a neuron in order to remove its linearity and increase the accuracy of the model.

The Activation function used for Linear and Polynomial Regression is the Sigmoid Activation:

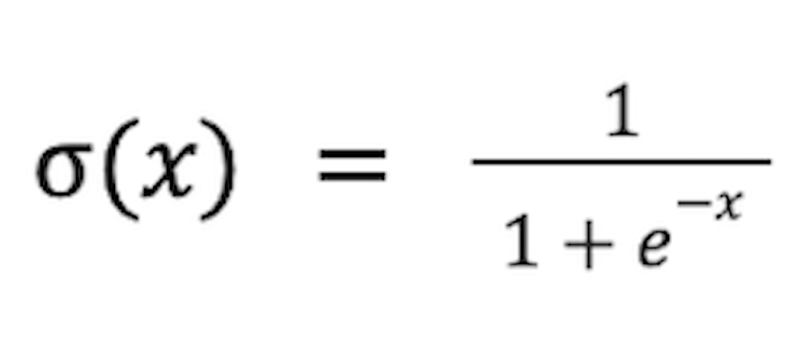
The Activation functions used for classification are Sigmoid Activation for each hidden layer neuron and SoftMax Activation for the output layer neurons.

**Sigmoid Activation:**

The sigmoid function maps all real numbers into a range from 0 to 1.

**SoftMax Activation:**

Converts Outputs into a probability distribution, which provides the probabilities for each class or label.



**Input Layer:**

The input layer of the network is the layer in which the input features of the data are introduced into the neural network.

**Hidden Layers:**

In a neural network, a hidden layer is a layer of neurons that sits between the input layer and the output layer. The hidden layer is where the network performs most of its computation and learns to represent the input data in a way that is useful for making predictions.

The activation function applied to the neurons in the hidden layer is what introduces non-linearity into the output of the network.

**Output Layer:**

It is the last layer of the Neural Network which provides the final output for the Network. No activation is applied to this output in the case of Regression and a SoftMax activation is applied in this output in the case of Classification.

In Regression, there is only one neuron in the output layer and in Classification, there are as many neurons as there are classes in the output layer.

**Forward Propagation:**

In this process, the input features are passed into the network from the input layer, then they are propagated through all the hidden layers, updating the outputs of each neuron and then finally returning the final output.

𝑧[1] = 𝑥. 𝑤[1] + 𝑏[1]

𝑎[1] = 𝑆𝑖𝑔𝑚𝑜𝑖𝑑(𝑧[1])

, 𝑥 where is the input from the input layer, 𝑧[1] is the output, 𝑤[1] is the weights, 𝑏[1] is the bias and 𝑎[1] is the activated output for the first layer.

𝑧[𝑖] = 𝑎[𝑖−1]. 𝑤[𝑖] + 𝑏[𝑖]

𝑎[𝑖] = 𝑆𝑖𝑔𝑚𝑜𝑖𝑑(𝑧[𝑖])

, where 𝑎[𝑖−1] is the activated output from the (𝑖 − 1)𝑡ℎ layer, 𝑧[𝑖] is the output, 𝑤[𝑖] is the weights, 𝑏[𝑖] is the bias and 𝑎[𝑖] is the activated output for the (𝑖)𝑡ℎ  layer.

**Backward Propagation:**

Backpropagation is an algorithm used to train neural networks by adjusting the weights and biases at each layer in the network. It works by propagating the error from the output layer back through the network to adjust the weights and biases at each layer through mini-batch gradient descent. It uses the chain rule of differentiation for updating errors at each layer.

|  |  |  |
| --- | --- | --- |
| Backpropagation Neural Network |  |  |

**Loss for Linear and Polynomial:**

Cost function used is the Mean Squared Error (MSE).

It calculates the average squared difference between the predicted and actual values.

The formula for MSE is:

J = (1/2𝑏) ∗ ∑𝑏𝑖=1(𝑦𝑖 − 𝑦̂𝑖)²

Where J is the loss, 𝑏 is the number of data points in the dataset, 𝑦𝑖 is the actual value, and  𝑦̂𝑖 is the predicted value.

**Cost Function for Classification:**

The cost function used here is the Cross-Entropy Loss.

Cross entropy loss is a measure of the distance between the predicted probability distribution and the true probability distribution. The loss is calculated as the negative log likelihood of the true class labels, given the predicted class probabilities.

                                      𝐶𝑟𝑜𝑠𝑠 𝐸𝑛𝑡𝑟𝑜𝑝𝑦 𝐿𝑜𝑠𝑠 = 𝐽 = − ∑ 𝑦𝑖 . log (𝑦̂𝑖  ) + (1 − 𝑦𝑖 ). log(1 − 𝑦̂𝑖)

, where 𝑦𝑖  is the One-Hot Encoded matrix form of the true label of the 𝑖𝑡ℎ sample and 𝑦̂𝑖 is the matrix of probability distributions for each label of the 𝑖𝑡ℎ sample

**Result:**

Linear Regression showed an R2 score of 84.37%, within 1000 iterations and learning rate of 0.01.

Polynomial Regression showed an R2 score of 99.95% within 10000 iterations and with a learning rate of 0.001

Multi-Class Classification showed an Accuracy of 80.46%, within 1000 iterations and learning rate of 1.

**Conclusion**

In conclusion, Implementation and evaluation of five machine learning algorithms: Linear Regression, Polynomial Regression, Logistic Regression, K-NN, and Neural Networks has been achieved. We have used three different datasets to test the performance of these algorithms. The hyperparameters of each algorithm were optimized to achieve the best possible performance.