ML BOOTCAMP

By Vaibhav Jain - 23JE1057

Introduction

The goal of this project is to create a straightforward and user-friendly machine learning library tailored for educational and prototype development. Our objective is to build a fundamental machine learning library entirely from the ground up, utilizing libraries such as NumPy, Pandas, and Matplotlib. This endeavor aims to simplify the learning and experimentation process in the field of machine learning by offering a clear and easily accessible framework for educational purposes and quick prototyping.

Dataset Overview

Four datasets were given, each intended for a distinct purpose: one for linear regression, another for polynomial regression, one for classification tasks, and a fourth for K-means clustering.

In the dataset focused on linear regression, the model confronted the task of analyzing data with 20 features, aiming to identify inherent patterns and relationships. In contrast, the polynomial regression dataset consisted of three characteristics, requiring the model to capture more intricate nonlinear connections.

The classification component introduced an intriguing aspect by utilizing pixel-by-pixel representations of image data for categorization. These images were characterized by numbers ranging from 0 to 9, and the goal was to train the model to recognize these numbers based on the provided pixel data. This varied set of tasks not only encompassed a range of regression challenges but also delved into the complexities associated with image-based classification.

Linear Regression

Dataset:

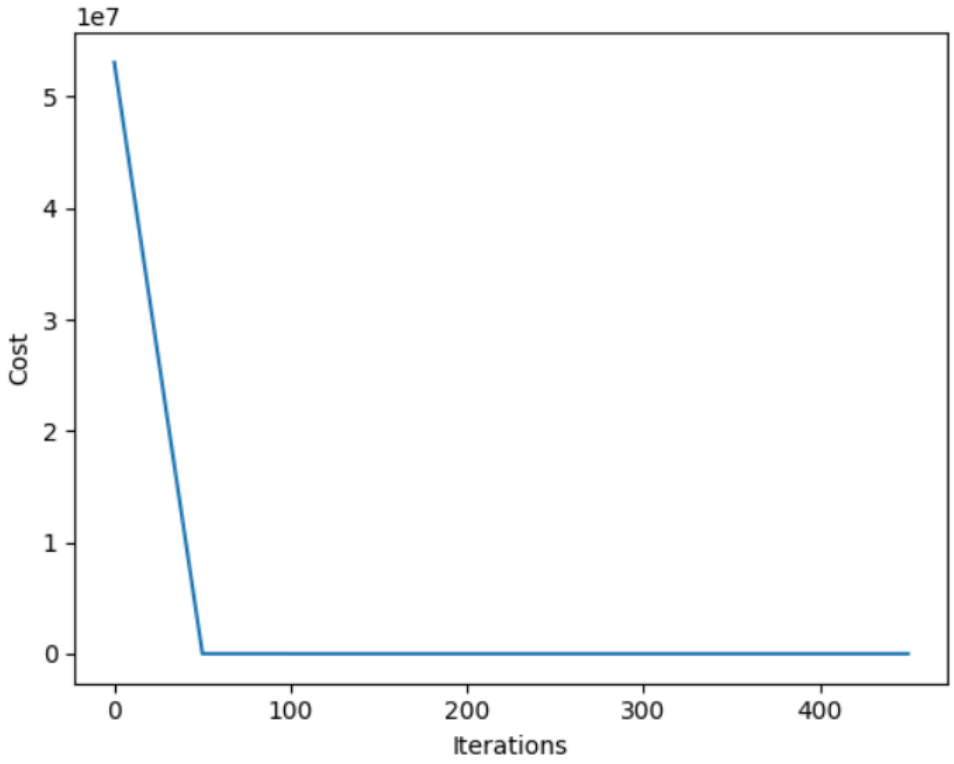
- Training examples = 50000
- Features = 20

Hyperparameters:

- Learning rate = 0.1
- Iterations = 500
- Normalization = Z-score normalization
- Lambda regularization = 0
- Train - CV split = 49000:1000

Results:

- R2 score for training data: 0.9999999999229953
- R2 score for cross validation data: 0.9999999999187164
- Final Cost: 0.0050497333594476685



(Cost plot for every 50th iteration)

After completing the 1st course by Andrew Ng I started this project.

I wrote the code in vectorized form from start and implemented Z-score normalization as suggested in the course. Great r2 scores were observed from the start because maybe the dataset was already linear. After trying different values for learning rates and iterations I used the above mentioned values for best result. Mean squared error was used for cost function

Polynomial Regression

Dataset:

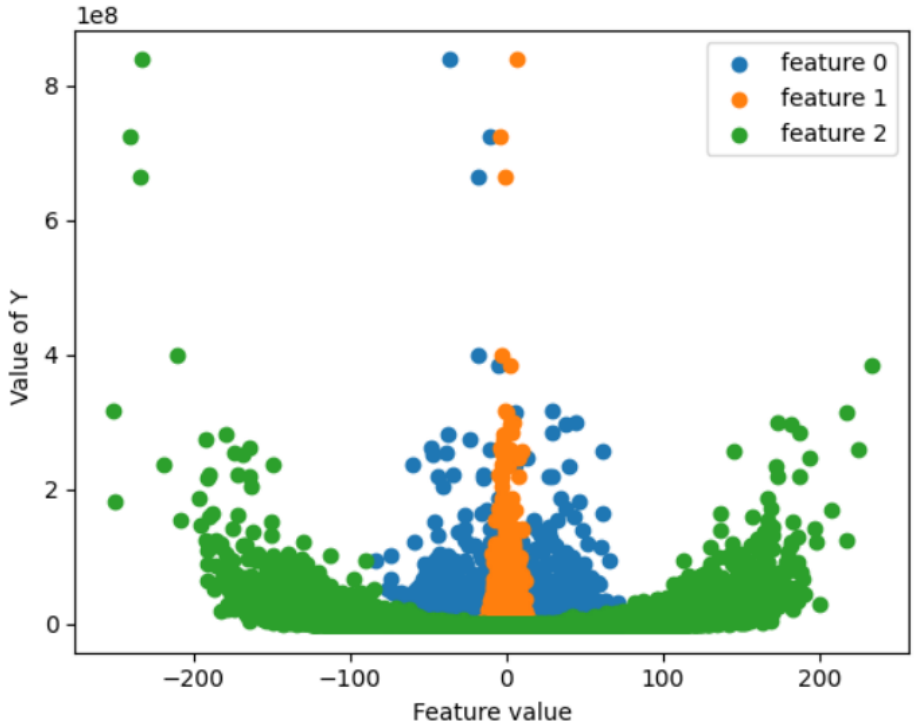
- Training examples: 50000
- Features: 3

Hyperparameters:

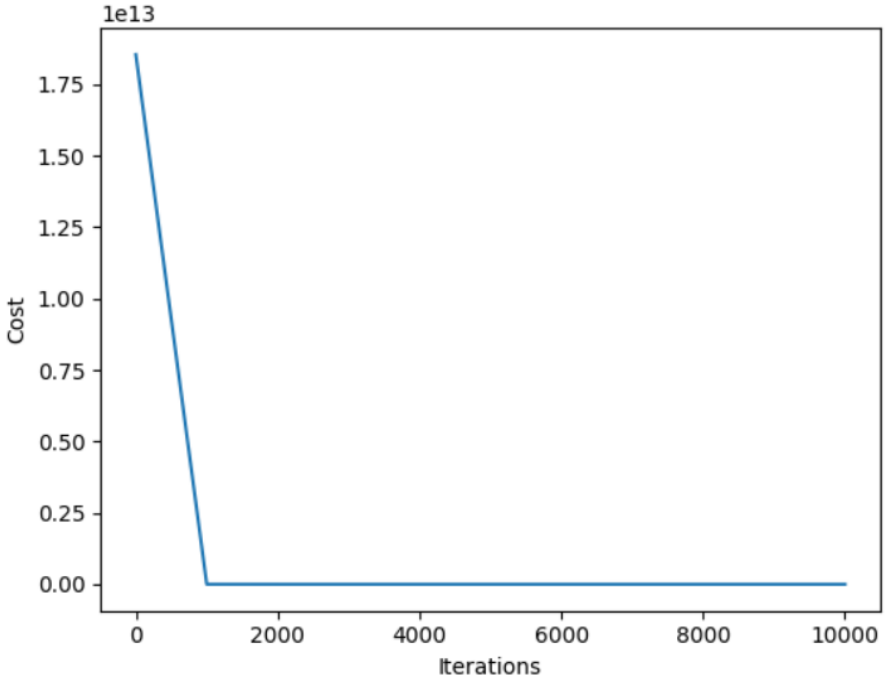
- Learning rate: 0.2
- Iterations: 10000
- Lambda : 0
- Degree : 6
- Normalization = Z-score normalization
- Train - CV split = 46000:4000

Results:

- R2 score on training: 1.0
- R2 score on cross validation: 1.0
- Final cost = 5.421293306320952e-07



(Scatter plot for 3 features)



(Cost plot for every 1000th iteration)

After coding feature engineering function for this was same as linear regression. I used cross features also like for degree 2 X1*X2 was also used.

Best results were obtained by using above parameters.

The cost did not converge at 10000 iterations but they were enough as r2 score for both training and cv set was 1 also doing further regression could have led to overfitting

Logistic Regression

Dataset:

- Training examples: 30000
- Features: 784

Hyperparameters:

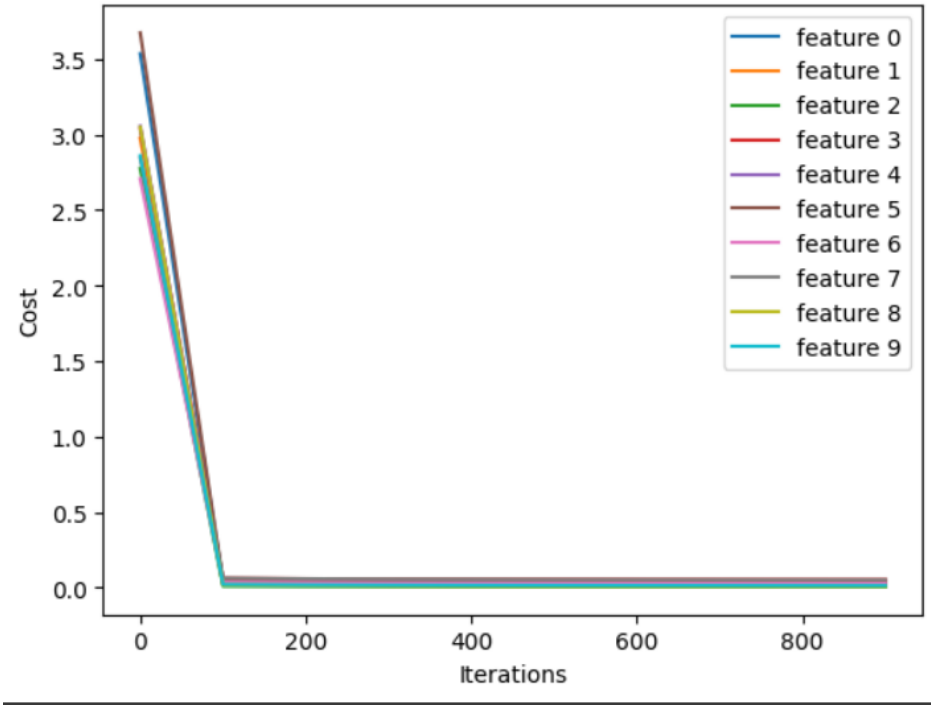
- Learning rate: 0.0001
- Iterations: 900
- Lambda : 0.1
- Normalization = Not used
- Train - CV split = 29000:1000

Results:

- Accuracy on cross validation: 0.973

- Final cost (array of 10 numbers which are cost for binary logistic regression for each category)

```
Iteration  900: Cost [0.03025954 0.01136483 0.00543206 0.05280724 0.0227937  0.02900285
 0.02785292 0.05221645 0.01059525 0.01389091]
```



I used one vs rest method in this and applied binary classification for each category. Sigmoid function was used in this method. Normalization was not needed in this as it didn't affect the result much.

N-layer Neural Network

Dataset:

- Training examples: 30000
- Features: 784

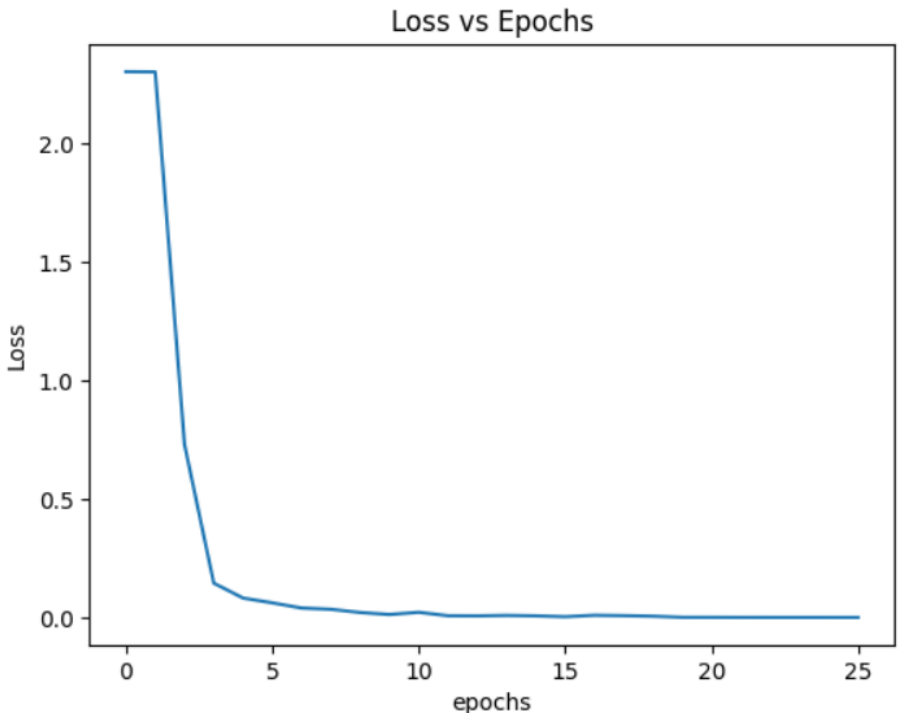
Hyperparameters:

- Learning rate: 0.1
- Layers: 512, 256, 128
- Epochs: 25
- Lambda : 0.1
- Normalization = Scale to 0 to 1
- Train - CV split = 25000:5000

- batch size =32

Results:

- Accuracy on training: 1.0
- Accuracy on cross validation: 0.9856
- Final cost: 0.0001758823993 (Sparse Categorical Cross Entropy)
- Training time: 299.69



I used relu for hidden layers and softmax for output layer. Architecture of neural network was 3 hidden layers with 512,256,128 neurons respectively and 784 ,10 neurons for Input and Output layer respectively. Best outputs were obtained on above mentioned parameters.

KNN

Dataset:

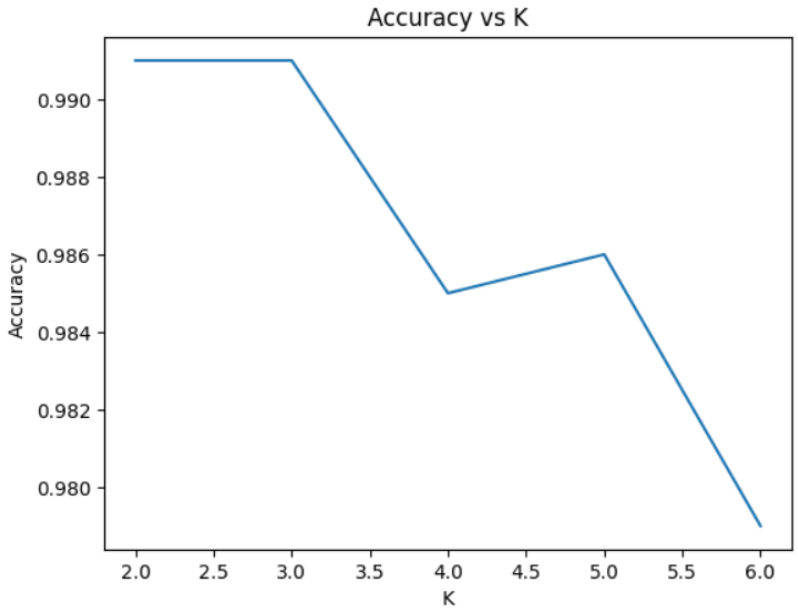
- Training examples: 30000
- Features: 784

Hyperparameters:

- Train - CV split = 25000:5000
- Max K = 6
- Optimum K = 3

Results:

- Accuracy on cross validation: 0.9772
- Time taken: 538.91



To choose optimum K I shuffled the data and used 5000 datapoints as training set and 1000 points as test set. Then I calculated the accuracy for K going from 2 to 6(max K)

According to the plot 3 is the best K and was chosen for KNN algorithm, 2 was avoided so that there is no problem when there is a tie i.e 2 points from different categories in closest points

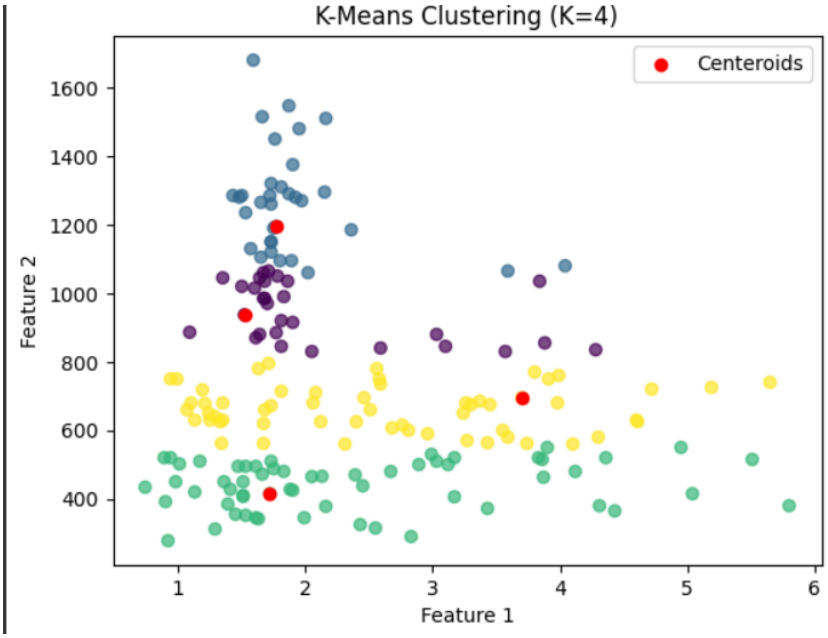
K means Clustering

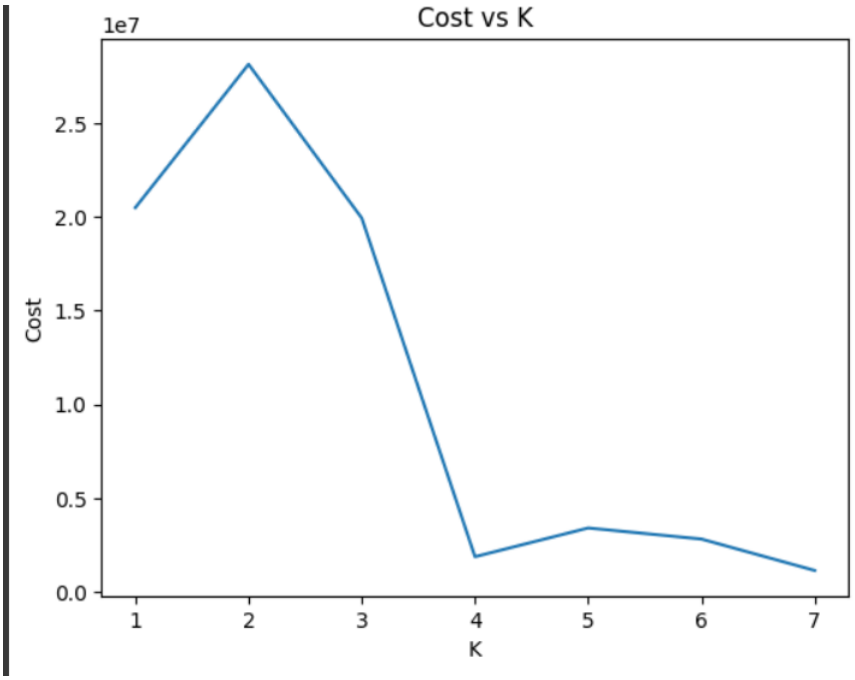
Dataset:

- Training examples: 178
- Features: 13

Hyperparameters:

- Max K = 8
- Optimum K = 4
- max iterations: 300
- max times: 200





I implemented K means clustering, in which we had to group the data points in K groups.

I used for loop to take values for K from 1 to Max K (8) and plot the data after plotting scatter plots with different colored clusters for different groups.

Firstly I used to initialize the centeroids by taking random values between the max and min value of their respective features but it didn't yield good results. So I switched to using randomly selected datapoints for initial centeroids.

Max iters is the maximum iterations allowed to converge the centeroids, it is used to avoid infinite loops.

Max times is the number of times random centeroids are initialized for each K, Best results based on cost are kept as final results

I used SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid for cost function.

After plotting Cost vs K I realized that K=4 is a optimum value for the dataset