Introduction to Neural Networks Assignment 01

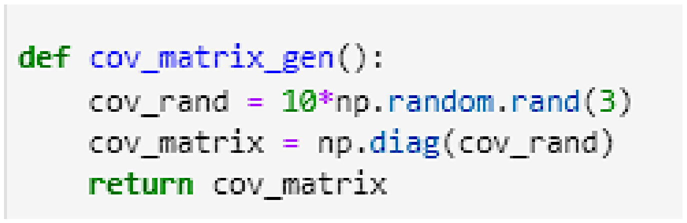
Mangesh Sakordekar

The aim of this project was to create a random dataset that met thefollowing criterion:

- The dataset should have 4 labels tied to it.
- The dataset should be a 3 features corresponding to the 3 axes in 3D space.
- The points for each feature should be picked from randomly from a multivariate normal distribution generated from a random covariance matrix and a random mean.
- The mean is to be randomly picked within a 20x20x20 cube centered at 0.
- The covariance matrix is supposed to be a diagonal covariance matrix with the diagonal elements randomly picked from the range [0,10)
- Each label should have a random number of points in the range [1,100].
- Make a basic neural network model and train it using the data created.
- Plot the data using predicted labels and calculate the accuracy.

Covariance Matrix

The diagonal covariance matrix was generated by initially selecting three random values within the range of 0 to 10. These chosen values were then employed to form a diagonal matrix utilizing the np.diag function. An important observation from this process is that, because the matrix is diagonal, the variability in each dimension is entirely self-contained, implying that the normal distribution of each dimension operates independently from the others. You can view the code that executed this operation in the screenshot provided below.



```
def cov_matrix_gen():
    cov_rand = 10*np.random.rand(3)
    cov_matrix = np.diag(cov_rand)
    return cov_matrix
```

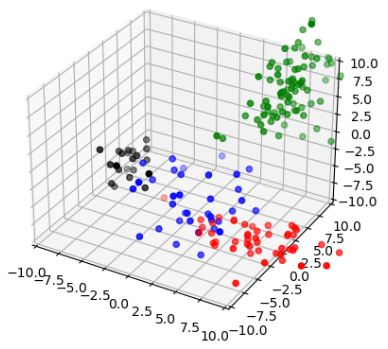
Figure 1: Covariance Matrix Generator

The Dataset

The dataset was generated to satisfy the previously mentioned criteria. Initially, a random mean was selected from a 20x20x20 box centered at the point (0, 0, 0). Using this mean and the randomly generated covariance matrix, a multivariate normal distribution consisting of 100 data points was generated. Subsequently, a random number of elements were chosen from this distribution, and each of them was assigned a label value of ”i.” A loop iterated through ”i” within the range [0,3]. In a broader context, this code sequentially produced data for each label while simultaneously plotting each label as soon as its data was generated

Plotting the Data

We modified the plot parameters to set the limits for each dimension to [-10, 10]. Additionally, we assigned specific colors to data points corresponding to different labels: 0 in red, 1 in green, 2 in blue, and 3 in black. You can see this visualization in the figure provided below.



Neural Net Model

Following the instructions provided, a simple neural network model was used similar to the MNIST example. The model contains two layers. First layer contains 128 neurons and uses relu as the activation function. The second layer has 4 neuron as the output layer should give out 4 probabilities, one for each class. This layer uses softmax as it's activation function. The model was trained using 75% of the data generated and 25% was used as the test data. The model was fitted for 10 epochs with batch size of 128. Over multiple runs, the minimum loss achieved was approximately 0.33 and the accuracy was around 90%. The data was plotted using the predicted labels, which can be seen in the figure below.

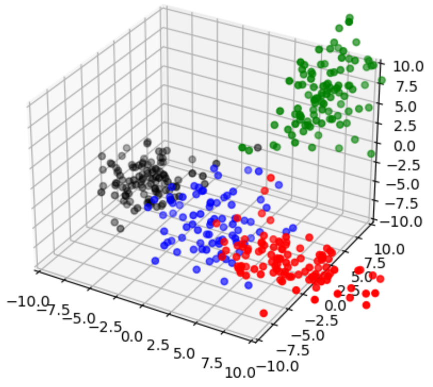


Figure 2: Data Plotted using Predicted Labels

Calculating Accuracy

The numbers used in this section are from one of the multiple runs. To calculate the accuracy the following formula was used:

$$Accuracy = CorrectPredictions * 100/SizeOfDataset \tag{1}$$

Accuracy was calculated for the test data as well as the whole dataset. For this run, The accuracy for test as well as the whole dataset was 94%. Although, the accuracy for the test data was usually sightly lower than the accuracy for the whole dataset in most runs.