Course Project - Natural Image Classification

Pattern Recognition and Machine Learning

CSL2050

Raunak Gandhi (B19CSE117), Pareek Vivek Manishkumar (B19EE061),
Pawandeep Suryavanshi (B19EE063)

Raunak Gandhi (Department of Computer Science and Engineering, Indian Institute of Technology, Jodhpur),
Pareek Vivek Manishkumar (Department of Electrical Engineering, Indian Institute of Technology, Jodhpur),
Pawandeep Suryavanshi (Department of Electrical Engineering, Indian Institute of Technology, Jodhpur).

***Abstract* - This document is an analytic report for the course project for the Pattern Recognition and Machine Learning course.**

I. INTRODUCTION

For the project, we use the CIFAR-10 data set for natural language classification. The data can be downloaded for the link - https://www.cs.toronto.edu/~kriz/cifar.html.

We have 10 classes with 6,000 samples each (total 60,000 samples) and among them, 10,000 samples are for testing and comparison between different models.

II. DATA UPLOADING

First, the data from the above mentioned link was downloaded and uploaded to google drive to make it easily accessible. The CIFAR-10 python version was downloaded from the website.

Then, google drive was mounted in the google colab notebook file. Required dependencies were imported.

The data was then uploaded in the colab file. The data contained pickle files and the files were unpickled in the notebook.

A function was written to unpickle a particular file and convert it to a pandas DataFrame for convenience. Only training data and label of the classes were kept in the DataFrame.

All six files (five training batches and a test batch) were unpickled using the said function and converted to DataFrame. All five training files were merged to get a single training file.

Each data point contains 3073 values 3072 pixel values (1024 i.e. 32x32 for each red, blue and green) and one label value (true class of the data point)

III. Preprocessing

For preprocessing, first a data set with gray scale pixel values of given images was created. This was performed by taking the average of red, blue and green pixel values of each pixel of every image.

We get a gray scale data set with 1024 pixel value for each data point. This data was used at the end for the visualization of the classes of the data set.

Then, the data was divided into training, testing and validation data sets. 30,000 data points from the complete data were assigned to the training data set. Remaining 20,000 data points were assigned to the validation data set. This was done to avoid overfitting.

The data points in the test batch were allotted to the testing data set.

The data was then scaled using the StandardScaler model from sklearn.preprocessing library.

IV. Dimensionality Reduction

For dimensional reduction, Linear Discriminant Analysis was performed.

Linear Discriminants are used in classification problems where the inter class distance between different classes is increased, whereas the intraclass distance is decreased.

We fit the Linear Discriminant model from sklearn.linear_discriminant_analysis library over the training data. Then, the feature variables for training data, testing data and validation data were transformed using the model.

V. Feature Selection

For feature selection, SelectKBest model from the sklearn.feature_selection library.

SelectKBest method selects the k best features from the entire feature variable set.

Thus, SelectKBest removes all but k highest scoring features. Thus, the best 512 features among all other features were selected for creating a new data set.

The SelectKBest model was fitted over the training data . Then, the training, testing and validation feature variables were transformed using the trained model.

VI. Classifier Models

Decision tree Classifier, Adaboost Classifier, Naive Bayes Classifier and MultiLayer Perceptron Classifier were used for training over the training data and testing over the validation and testing data.

Performance of these models are compared below.

A. Decision Tree Classifier

Decision Trees are a non-parametric supervised learning method used for classification and regression.

The DecisionTreeClassifier model from sklearn.tree library was used for training, validation and testing. The max_depth was kept 15 to avoid overfitting.

Three kinds of models were trained, i.e. scaled data (transformed using StandardScalar), data with reduced dimensions (transformed using LinearDiscriminantAnalysis), and data with reduced features (transformed using SelectKBest).

The obtained results are:

	Training Accuracy	Validation Accuracy	Testing Accuracy
Scaled Data	0.770	0.271	0.275
LDA Transformed data	0.794	0.286	0.285
Data obtained from feature selection	0.676	0.229	0.238

* Accuracies are rounded off upto 3 decimal digits.

5 fold Cross validation was applied for each model over the training data.

The obtained results are:

Scaled Data	0.270	0.270	0.265	0.267	0.277
LDA Transformed data	0.438	0.443	0.431	0.440	0.455
Data obtained from feature selection	0.226	0.222	0.227	0.240	0.223

* Accuracies are rounded off upto 3 decimal digits.

B. Adaboost Classifier

Adaboost Classifier is an ensemble method that uses multiple weak learners to create a strong classifier. This method helps decrease overfitting.

The AdaBoostClassifier model from sklearn.ensemble library was used for training, validation and testing. The base estimator was kept as a DecisionTreeClassifier with max_depth as 3. Number of n_estimators was kept equal to 10.

Three kinds of models were trained, i.e. scaled data (transformed using StandardScalar), data with reduced dimensions (transformed using LinearDiscriminantAnalysis), and data with reduced features (transformed using SelectKBest).

The obtained results are:

	Training Accuracy	Validation Accuracy	Testing Accuracy
Scaled Data	0.337	0.321	0.323
LDA Transformed data	0.528	0.324	0.323
Data obtained from feature selection	0.260	0.245	0.253

* Accuracies are rounded off upto 3 decimal digits.

5 fold Cross validation was applied for each model over the training data.

The obtained results are:

Scaled Data	0.318	0.321	0.312	0.323	0.318
LDA Transformed data	0.502	0.512	0.508	0.530	0.516
Data obtained from feature selection	0.247	0.240	0.254	0.249	0.242

* Accuracies are rounded off upto 3 decimal digits.

C. Naive Bayes Classifier

Naive Bayes Classifier uses probability to find the posterior probability for a data point being in a particular class. The posterior probability is calculated assuming that all features are independent.

The GaussianNB model from sklearn.naive_bayes library was used for training, validation and testing.

Three kinds of models were trained, i.e. scaled data (transformed using StandardScalar), data with reduced dimensions (transformed using LinearDiscriminantAnalysis), and data with reduced features (transformed using SelectKBest).

The obtained results are:

	Training Accuracy	Validation Accuracy	Testing Accuracy
Scaled Data	0.288	0.294	0.297
LDA Transformed data	0.564	0.348	0.342
Data obtained from feature selection	0.231	0.227	0.234

* Accuracies are rounded off upto 3 decimal digits.

5 fold Cross validation was applied for each model over the training data.

The obtained results are:

Scaled Data	0.292	0.286	0.278	0.292	0.289
LDA Transformed data	0.557	0.563	0.558	0.577	0.561
Data obtained from feature selection	0.236	0.236	0.224	0.230	0.224

* Accuracies are rounded off upto 3 decimal digits.

D. Multi-Layer Perceptron

Multi-Layer Perceptron Classifier uses gradient descent to find optimum values of weights and biases that are initialised randomly. Learning rate was set to 0.01 where maximum accuracy was obtained. The value of maximum iterations was increased to 500 as the values of weights and biases did not converge for the default value of 200 iterations.

The MLPClassifier model from sklearn.neural_network library was used for training, validation and testing.

Three kinds of models were trained, i.e. scaled data (transformed using StandardScalar), data with reduced dimensions (transformed using LinearDiscriminantAnalysis), and data with reduced features (transformed using SelectKBest).

The obtained results are:

	Training Accuracy	Validation Accuracy	Testing Accuracy
Scaled Data	0.555	0.428	0.429
LDA Transformed data	0.583	0.348	0.341
Data obtained from feature selection	0.661	0.293	0.304

* Accuracies are rounded off upto 3 decimal digits.

5 fold Cross validation was applied for each model over the training data.

The obtained results are:

Scaled Data	0.424	0.418	0.418	0.424	0.422
LDA Transformed data	0.556	0.562	0.555	0.568	0.563
Data obtained from feature selection	0.293	0.285	0.288	0.306	0.290

* Accuracies are rounded off upto 3 decimal digits.

VII. Visualisation

The pixel data for different classes was visualised by plotting the images using the pixel values.

Coloured images were plotted using the pixel values for different colours in the original data.

Decolourised images were plotted using the gray scale data that was developed by taking the average of pixel values of all three colours, as mentioned earlier in the report.

One image from each class was plotted using the imshow function of the matplotlib.pyplot library.

The class and the class number of each image was also printed to compare the results.

VIII. Conclusion

The data was analysed, uploaded, and preprocessed. Models were trained for different classifiers.

Maximum validation accuracy observed when MLPClassifier was trained for scaled data. The observed validation accuracy was 0.428.

Maximum training accuracy observed when MLPClassifier was trained for scaled data. The observed validation accuracy was 0.429.

Maximum individual accuracy for cross validation was observed when GaussianNB was trained for LDA transformed data. The observed accuracy was 0.577.

Maximum average accuracy for cross validation was observed when GaussianNB was trained for LDA transformed data. The observed accuracy was 0.563.

All models were compared and data was visualized. This helped to understand the working of image related data in the real world and also helped to get a better understanding about how Machine Learning is useful for real world applications.

IX. Contribution

The complete machine learning pipeline was thoroughly discussed among the team members. Every decision was made unanimously.

The work was divided for writing the code.

Raunak Gandhi (B19CSE117) - Wrote the code for training, validation, testing and cross validation for all models for different data sets.

Pareek Vivek Manishkumar (B19EE061) - Wrote the code for visualising the data by plotting coloured and decolourised images for each class using matplotlib.

Pawandeep Suryavanshi (B19EE063) - Wrote the code for importing the data and unpickling the pickled files, and converting it to DataFrame.

The work was divided only for writing the code. Discussion about the data and planning was done by all team members at length.