CSc 84020 Neural Networks and Deep Learning
Homework 2: Classification and Prediction with DL Multi-Layer Perceptron
Andrea Ceres
Shao Liu

3a. Dataset Descriptive Statistics and Analysis

Our dataset is a subset of Google *Quick, Draw!*[1]. After experimentation with larger themed groupings of classes to discern processing time and performance, we have chosen drawings of critters having six classes (*ant*, *bee*, *butterfly*, *mosquito*, *scorpion* and *spider*) as our target labels. In each class, we have 25,000 grayscale images with a size of 28x28. For each image, the length of the feature vector is 28x28= 784 and the range of values is [0, 255] to be normalized to [0,1].

Because the images are quick drawings with lines, most of the pixel values are 0. Only pixels from lines have positive values. Moreover, most of these positive pixels have a large value around 250.

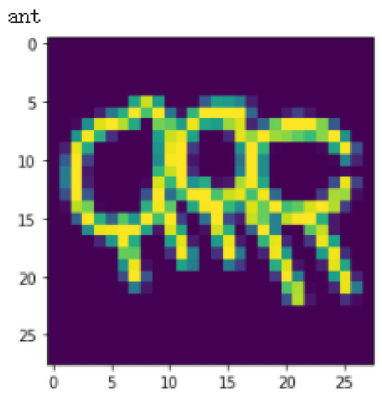

Fig.1

From *Fig.1* we can see that all the purple areas are 0 pixels and most of the line areas are 255. This situation makes the values very sparse and somehow difficult to deal with. *Fig.2* is the histogram of 28 pixels in the center row (image[14, :]). These pixels are the range [391:419] from the 1x784 feature vector. We can see from *Fig.2*, even in the center row of this image, most of the pixels are distributed close to 0 and 255. The variances of center attributes (pixels) are high at more than 100.

We also calculated the Pearson correlation for pairs of pixels. The Correlation Matrix Plot shows that the neighbor pixels have the most correlation values. This is obvious, because the drawn lines are continuous.

The Scatter Matrix and Density Plot also show the distribution of pixel values. Values are concentrated toward 0 and 255 areas. For training considerations, those 0 pixels may need to be dropped to eliminate their effect on the model, as they provide no information.

Classification

Because there are six classes and the dataset is large, we use a small sample from the dataset for Spot Check Algorithms (1000 images for each class). The sample dataset is split into 90% training and 10% Testing. Six models are each fit to the training data via ten-fold cross validation. We set the solver to *sag* and *C* to 0.01 for the Logistic Regression (LR) model. With this solver as well as the other solver options, the LR model does not converge during training. This may be because the values are too sparse. The accuracy for LR is 0.46.

From the algorithm comparison box plot (*Fig.3*), we can conclude that the K-Nearest Neighbors (KNN) model achieved the highest average accuracy with low variance. LDA achieved the next highest accuracy

[1] https://github.com/googlecreativelab/quickdraw-dataset

(0.498). However, the best and worst scores of LDA are very different where the lowest accuracy of LDA is around 0.45. Moreover, half of the scores are in a large range around the mean accuracy. This indicates that the accuracy of LDA has larger variance and might not work well on validation dataset.

After determining the trained models' accuracy from 10-folds training, we find the KNN model achieved the highest score (0.645). We use KNN to test on the validation data. The KNN model got 0.645 accuracy in training and 0.672 in validation. Higher accuracy on validation data may be because the validation dataset is small and the samples might fit to the model by coincidence. The relatively low accuracy is also caused by the small sample dataset we use. Nevertheless, we can use the accuracy of 0.672 as a baseline benchmark to which we will compare the results of our multilayer perceptron models.

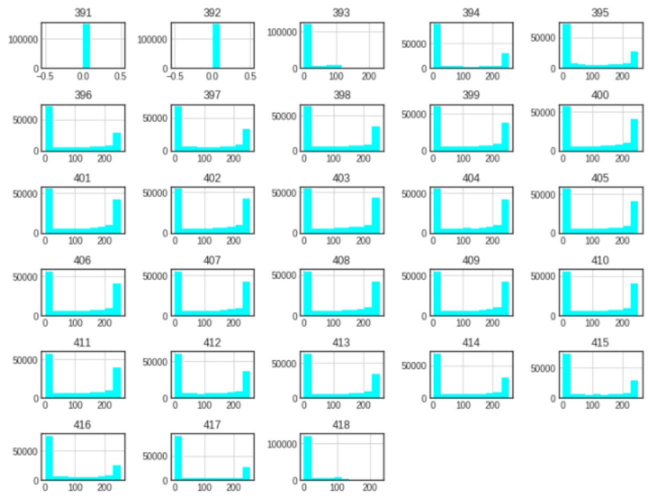

Fig.2

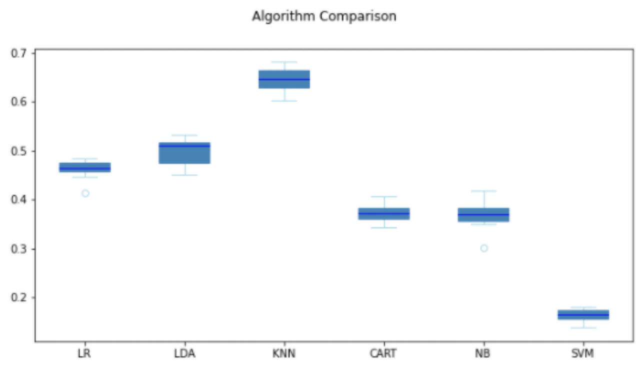

Fig.3