PREDICTIVE ANALYTICS

Northeastern University

ALY6020, FALL 2020

ASSIGNMENT 4

NUID: 001084592

CRN: 71717

SUBMITTED BY: VISHRUTA MAKHIJA

EMAIL ID: makhija.v@northeastern.edu

SUBMITTED TO: PROF. JUSTIN GROSZ

NORTHEASTERN UNIVERSITY, COLLEGE OF PROFESSIONAL STUDIES

PREDICTIVE ANALYTICS

This week we are working on the MNIST dataset. It has a total of 70,000 datapoints divided into 42000 rows as training set and 28000 rows as test set. It is a subset of the NIST's accessible larger package. The MNIST dataset (Modified National Institute of Standards and Technology Database) is a massive handwritten number database that is generally used for image analysis. It is popularly used in machine learning. The goal of this week's assignment is to train a model that detects the handwritten digits correctly.

DATA PREPARATION

We initially checked the data for missing values and observed that there are no missing values in our train as well as test dataset. We then separated the train data into feature and target variables and named it as train_images and train_labels respectively. We then plotted a count plot of the train_labels data to check the distribution of the label column. We then normalized the train_images data having values in the range of 0 to 255 to a range of 0 to 1 by dividing all the values by 255, in order to make our model easier to train. Post normalization, we divided the train data (train_images and train_labels) into test and train dataset with 20% data classified as validation and 80% data classified as train data.

NEURAL NETWORK

We applied the Convolutional Neural Network (CNN) on our dataset. In comparison to other image classification algorithms, CNN uses very little image pre-processing. This implies that the network learns through filters that have been hand engineered in conventional algorithms. So, for image processing Computational neural network is the best fit. We built a model with three layers, in which two layers have 64 neurons and ReLU function and one layer has 10 neurons (because we have digits from 0 to 9) to classify and function as softmax. ReLU function adjusts

all the values to zero and all other values remain unchanged whereas the Softmax layer is usually the final output layer which takes a number of values as inputs and overpowers them into values ranging from 0 to 1, having sum as 1. Then we compiled a model and specified the loss function and optimizer. The loss function measure how well the model performed on the training data and then tries to improve on it using the optimizer. We used the optimizer "adam." Adam is an algorithm for optimization and can be used in place of classical stochastic gradient descent technique to iteratively adjust weights focusing on training data. For the loss function we used the categorical crossentropy, which is used for multiclass classification as we have more than 2 classes. These are the activities where just one out of the several possible categories will correspond to an instance and the model must determine which one. Also, we specified the metrics that we want to check and mentioned accuracy for the same. We then fit the data on the train data (X_train and y_train), where we converted the y_train data to 10-dimensional vector by using to_categorical function, for example, 3 can be represented [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]. We also specified the values for epochs and batch_size as 5 and 32 respectively. Epochs is the number of iterations for the entire data and batch size is the number of samples per gradient update for training. From the output received we can see that the loss and the accuracy of each epoch. We received an accuracy of 97.79% on the last epoch and also the loss gradually decreases from epoch 1 to epoch 5. We then evaluated our model on the validation data (X_test and y_test). Similar to the train data, we converted the y_test to 10-dimensional vector before evaluating the model on test. We received an accuracy of 96.36% on the validation data. We then made predictions on the first five digits of our validation dataset to check if our model predicts the labels correctly and observed the accurate predictions for the same.

```
Predicted Labels: [8 1 9 9 8]
Actual Labels: [8 1 9 9 8]
```

Figure 1: Actual and Predicted values of the Neural Network Model (Validation Dataset).

After performing the validation, we checked our model on unseen test data and visualized the predictions as seen in Figure 2.

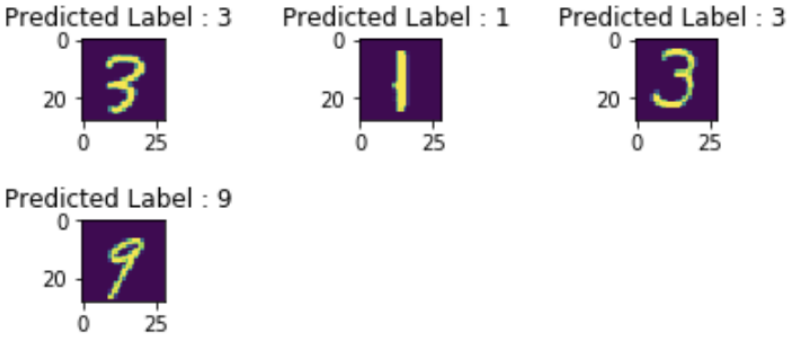

Figure 2: Actual and Predicted values of the Neural Network Model (Test Dataset).

From the Figure 2, we can see that our model correctly detects the handwritten images.

K-NEAREST NEIGHBORS

We then performed the K-Nearest Neighbors model on our MNIST data. We used the KNeighborsClassifier function for the KNN model. We first fitted the model on the train data (X_train and y_train) and then predicted on the validation data (X_test and y_test) and calculated the accuracy for all the models with different values of k with k as 1, 3, 5 and 10 displayed in Figure 3.

We observed that the accuracies of all the k values is almost the same having minute differences. But we observed the pattern that when the number of k increases the accuracy decreases. We observed the maximum accuracy to be 96.8% when k is equal to 1, followed by k equal to 3 having accuracy of 96.6%, then k equal to 5 and k equal to 10 with accuracies of 96.4% and 96.1% respectively.

	Model	Accuracy
0	KNN (k=1)	0.968690
1	KNN (k=3)	0.966667
2	KNN (k=5)	0.964881
3	KNN (k=10)	0.961667

Figure 3: KNN Model accuracies with different values of k.

COMPARISON BETWEEN CNN AND KNN

We observed that the accuracies of both the CNN and the KNN models are almost similar that is 96.6% on the data and the accuracy of the KNN model with k value of k=1 is 96.8%. We also observed that the Computational Neural Network model predicts the handwritten labels correctly as represented by Figure 1 and Figure 2. Out of the two models, CNN is better for image classification than the KNN model. **The metrics that differentiates between the KNN and the CNN Model is the run time.** The major disadvantage of the KNN model is its application in such kinds of datasets where there are a large number of variables, that is, KNN does not work well with voluminous data. More the data fed into the KNN model, more time it consumes to run the model and provide prediction outputs. Whereas the CNN model gives the prediction results quite fast. Convolutionary neural network has become influential in different tasks in image recognition and attracts curiosity in different fields such as medical research and its growing interest and scope in radiology. The CNN consist of multiple building blocks such as convolution layers, pooling layers and completely connected layers and are built using back propagation algorithm to learn spatial hierarchies of variables automatically and adaptively. CNNs are very efficient in reducing the number of variables without sacrificing the consistency of the models. Images have high dimensionalities as each pixel is treated as a feature, which can be effectively handled by the CNN

model and which at the same time is highly time consuming for the KNN model. Moreover, in addition to image processing, CNN have also achieved recognition in text analysis.

CONCLUSION

Though both the CNN and KNN model have almost the same accuracy, the CNN model as explained in the above section is better for image recognition when compared to the KNN model.

The pixel elements that are considered while identifying a number through CNN model are the edges and curves. CNN works in a similar way as our brain does. When we look at a picture of a dog, we identify the recognizable characteristics (for example: four feet) that help our brain classify it as a dog. Similarly, by scanning for low-level characteristics such as **edges and curves**, then processing it through various Convolutionary layers, the system is able to perform image classification or identification. To create high level features, such as paws in this case, the machine learning model makes use of the initially obtained low-level features.

REFERENCES

MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. (2020). Retrieved 21 October 2020, from http://yann.lecun.com/exdb/mnist/

Y. LeCun, Y., O. Russakovsky, J., V. Gulshan, L., A. Esteva, B., B. Ehteshami Bejnordi, M., P. Lakhani, B., . . . S. Ren, K. (1970, January 01). Convolutional neural networks: An overview and application in radiology. Retrieved October 21, 2020, from https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

Mishra, P. (2019, July 20). Why are Convolutional Neural Networks good for image classification? Retrieved October 21, 2020, from https://medium.com/datadriveninvestor/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8