Gurjus Singh

MSDS 458 Artificial Intelligence and Deep Learning

January 24th, 2021

Week 3: A.1 First Research/Programming Assignment MNIST Classification

**Abstract**

In this research I explored the MNIST dataset. This dataset contains handwritten number images. The goal of this research was to explore using multiclass classification using a neural network and supervised learning. I wanted to use some metrics to see how the Neural Network was classifying the images. I also performed dimensionality reduction using PCA and random forests to see if the neural network could still classify with great results.

**Introduction**

The purpose of this research was to use the MNIST dataset of handwritten digits 0-9 to do supervised learning multiclass classification using Neural Networks. The Neural Network architecture and complexity was explored as well as the activation values in the research. Five experiments were done, and one neural network experiment was which had the best results was chosen for the management recommendation. I also explored lowering the dimensions of the dataset using PCA, and Random Forests and wanted to see if the dataset had same or better results.

**Literature Review**

Much research has been done with the MNIST dataset. The dataset was originally introduced by LeCun et al. in 1998 (Baldominos et al., 2019, pg. 1). Lecun has also published research of models that have been used on the MNIST dataset (Baldominos et al., 2019, pg. 3). Models include Linear Classifiers which error rate for classification of MNIST dataset is 7.6 percent to 12 percent. Other models include K Nearest-Neighbor which has an error rate of 1.1 percent to 5 percent, nonlinear classifiers which have 5 percent error, Neural Networks have 1.6 percent to 4.7 percent and Convolution Neural Networks which have the lowest error rate at 0.7 percent to 1.7 percent (Baldominos et al., 2019, pg. 3). Other researchers that have worked with MNIST, have also expanded the dataset to include 50,000 more test images, fearful that their models were overfitting (Synced, 2019, para 2).

**Methods**

My research was first started by importing the packages in 1-1. The Most important packages to import was the numpy packages which was to manipulate arrays, tensor flow package which was used to use key metrics, and Keras package which was used to create the Deep Neural Network architecture, train the model. After the import statements, I made sure the Tensorflow package was 2.0 or above. I then used the **load\_dataset** function in Keras to load the MNIST dataset which was then split into **Train and Test** after loading. I then looked at the shapes of the Train and Test dataset which were **(60,000,784) and (10,000, 784).** I also observed the first 10 target labels of the dataset, which gave me a sense of what numbers were in the dataset. I then tried to determine most common numbers in the Train and Test datasets.

*1-1 Import Packages*

Text

Description automatically generated

Text

Description automatically generated

*1-2 Numbers in MNIST*

After seeing the most common number I used the matplotlib package to see the images of the numbers seen by the code and pictures in 1-2. The function imshow() in python helped me to show the images. From the initial observation I saw a 9 that looked like a G and a 5 that looked like a 6. After looking at the initial data, the next step was to do one hot encoding. This was used to convert each data target label into an array of 10 elements where the target value was represented by index and filled with an one if it matched, and 0s elsewhere. This was done using **to\_categorical** method. After one hot encoding it was time to normalize the MNIST dataset. This was done by dividing the pixel values by 255. This was to get the values between 0 and 1.

After the data preparation, data visualization, it was time to create the models. I did a total of 10 experiments. The first two models had 128 1st layer nodes, and 1 and 2 hidden layers respectively. On the third layer I used a total of 5 hidden layers. I used this architecture for experiments for 4 and 5, but what was interesting about experiments 4 and 5 was that I used PCA, a dimensionality reduction algorithm, which reduced the features from 784 to 154. I then reduced the features even further with Random Forests to 70-pixel columns and wanted to see if the accuracy, and results were great as well. For Experiments 5-10, I wanted to see what increasing the number of nodes would do in each of the two layers, so I increased to 64 in Experiment 6 to 512 in Experiment 10 in both layers.

For compiling the Deep Neural Network models, the loss function I used was categorical cross entropy. A loss functions helps the neural network to find the right predictions. It uses a distance score to train the neural network (Chollet, 2017, pg. 10). The optimizer which I used was rmsprop. The optimizer is used in Back Propagation to adjust the weights so the loss function can lower its distance between predictions and target values (Chollet, 2017, pg. 11). The way back propagation works is by applying the “chain rule” to different parameters and in order to find out how much each parameter in the neural network contributed to the final answer (Chollet, 2017, pg. 52).

**Results**

**Table

Description automatically generated**

*Experiments figure 1-3*

I created a series of ten experiments in using hidden nodes from 1-512, and 1st layer number of nodes from 128-512. I wanted to find out what number of nodes got the accuracy just about right? I compared my results and most of the time my models were overfitting. Particularly, experiments 1 and 2 were underfitting as seen in 1-3 which had hidden nodes of one and two, and first level nodes of 128. As I increased the level of neurons for the hidden layer, I got overfitting as seen in experiments 3 and 4 in figure 1-3 with hidden nodes of 5. Experiment 5 did not overfit, but I wanted to see if train and test accuracy could be improved. With experiment 6 I did improve train accuracy to 98.51 percent and test accuracy to 97.35 percent. Experiments 7-10 also started to overfit as I increased number of nodes again in both layers as I saw from the epochs as seen in 1-4.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Examples of Overfitting Epochs from Experiments 7-10 1-4

After comparing my accuracies of my models and the differences between the train and test accuracies I determined that Experiment 6 which had **64 neurons for both layers** was my best model. I chose 6 over experiment 5 because of the higher of accuracies. I then looked at the confusion matrix for my best model and saw that it was in fact getting the majority of predictions right as seen in the diagonals in 1-5, 1-6. I also looked at the plot of the epochs and made sure the model was not overfitting as seen in 1-7. There was a tiny gap, which meant model was doing decent and was not diverging largely like the other models.

A picture containing text

Description automatically generated

Confusion Matrix using Train Set for Exp.6 1-5

Table

Description automatically generated with low confidence

Confusion Matrix using Test Set for Exp.6 1-6

Graphical user interface, chart, line chart

Description automatically generated

Plot Experiment 6 1-7

I also looked at the 1-8 boxplots which gave me a sense of the range of activation values. I only saw some overlapping between class 1 and 2. Most of the classes had a unique range of activation values and their boxplots were not overlapping which is what is ideally what I wanted.

Chart, box and whisker chart

Description automatically generated

1-8 boxplots Experiment 6

In 1-9 I then looked at the scatterplot to see how many classes could be predicted through 2 pixels and I found 4-5 classes could be predicted as seen in 1-9.

Chart, scatter chart

Description automatically generated

Experiment 6 2 pixel scatter plot 1-9

**Conclusions**

In this research, I learned how train deep neural networks on the MNIST dataset. The MNIST dataset as described above, is a set of handwritten number images, which the neural network could recognize. In order to train the model, it required preparing the data into training, validation and test sets. It required normalizing the data from 0 to 1, and it required one hot encoding for target values in the data. This was a supervised learning research method as it used target variables to train the neural network. After training the models, I evaluated the models, and found most of them were overfitting, and underfitting**.** I found Experiment 6 to be the best which used 64 nodes in 2 layers to fit the data; **therefore, for the management recommendation I recommend the 64 nodes in 2 layers neural network architecture to train using the MNIST data as it had a 98.51 percent training accuracy and a 97.35 percent test accuracy.** I also looked at the activation box plots and small minimum overlap which is ideal as that mean it is distinguishing between each image and each range of activation values in unique. In the future I would like to see how I can get the neural network to predict 100 percent accuracy, while not overfitting.

References

Baldaminos, A., Saez, Y., & Isasi, P. (n.d.). A survey of handwritten character recognition with MNIST and EMNIST. *Applied Sciences*.

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications Company.

Synced. (2019, June 19). MNIST reborn, restored and expanded: Additional 50K training samples. Retrieved January 20, 2021, from https://syncedreview.com/2019/06/19/mnist-reborn-restored-and-expanded-additional-50k-training-samples/