Project Nature and Content - Computer Vision

Sonny Desai

January 2020

Abstract

This project deals with creating a neural network and five experiments which are conducted on the MNIST data set. The aim of the project is to explore the MNIST data set by using dense neural networks and hidden layers. The five experiments conducted were: 1) we group the 60,000 activation values of the hidden node for the (original) set of training images by the 10 predicted classes and visualize these sets of values, 2) For each of the 60,000 images, the output of the two hidden nodes are plotted using a scatterplot. We color code the points according to which of the 10 classes the output of the two nodes predicts, 3) explore with more hidden nodes, 4) Use PCA decomposition to reduce the number of dimensions of our training set of 28x28 dimensional MNIST images from 784 to 154, 5) use a Random Forest classifier to get the relative importance of the 784 features (pixels) of the 28x28 dimensional images in training set of MNIST images and select the top 70 features (pixels). These experiments evaluated how different hidden layers and nodes affected the results of the DNN in correctly identify and predicting the MNIST images. The PCA further tested the DNN with limited dimensions, and the Random Forest tested the importance of the 784 features.

Introduction

Artificial neural networks have been around for nearly a century but are being currently applied as artificial intelligence and machine learning has become more and more prominent. Neural networks are based upon how human brains function and go through decisions and data similar to that of a human brain. A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture. A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis. The main purpose of this research problem is to use the aforementioned five experiments to test the DNN with differing hidden layers and nodes to produce the best model. Each experiment reveals a different model, and after comparing results which include PCA dimension reduction and a random forest classifier, the best model can be determined.

Literature Review

Another scholarly article related to the experiments and purpose of this project is "Digit Identification using Deep Neural Networks in TensorFlow" by Pankaj Kumar. This article covers the authors own efforts to create a DNN with the MNIST dataset. He similarly split his data into test and training datasets with 10,000 and 60,000 images used respectively. His experiment had two hidden layers compared to our multiple tests. For his DNN, he used a Mini-Batch Gradient Descent for his training data which means that instead of sending whole data, he sent mini batches of data updates the model weights on them. The predictions from his model were similar to ours as well as being very accurate. He doesn't provide his accuracy rate, but from viewing his results, they looked almost 90% accurate. His model showed both training and testing accuracy increasing with each epoch – showing his model was learning. He notes that there is future work to do on his model, and although he did not cover all the elements of our different experiments, he had a similar result to our model predictions!

Methodology

As mentioned previously, our data comes from the MNIST dataset which was imported from the keras open source library in python. Pre-processing of the data was done before the creation and testing of the models. We break out the data into test and training datasets. First we converted the labels (integers 0 to 9) to 1D numpy arrays of shape (10,) with elements 0s and 1s with all the elements set to 0 except the one which the label belongs to. We did this with one-hot encoding, which we used when there are categorical variables where no ordinal relationship exists. In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value. Next, we reshaped the images from 2D arrays of shape (28,28) to 1D float32 arrays of shape (784,) and then rescale their elements to values between 0 and 1. Lastly, we scaled the data so that they are both normalized.

The models we use to experiment on the data and find the hidden layer are the DNN, a PCA (Principal Components Analysis), and a Random Forest Classifier. A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. We used a Sequential class defined in Keras to create our model. All the layers are Dense layers which means all the nodes of a layer would be connected to all the nodes of the preceding layer. We used stochastic gradient descent function and defined a loss faction. The loss needs to be minimized in order to have a higher model accuracy. We used the accuracy during training as a metric to keep track of as the model trains. In order to ensure that this is not a simple "memorization" by the machine, we evaluated the model on a test set.

Finally, we created a Random Forest Classifier (with the default 100 trees) and used it to find the relative importance of the 784 features (pixels) in the training set. We produced a heat map to visual the relative importance of the features. The last step was to select the 70 most important feature (pixels) from the training, validation and test images to test our 'best' model on.

Computational Experiment and Results

The DNN model for the first experiment was not very accurate, despite all of our data pre-processing, and the 50th epoch ended with an accuracy rate of 38%. The second experiment had the 50th epoch with accuracy of 69.82%. The third experiment had the 50th epoch with accuracy 99.92%. The fourth experiment had the 50th epoch with the accuracy score of 100%. The fifth experiment had an accuracy score for the 50th epoch of 97.76%

Discussion and Conclusions

The DNN models became more successful as the models added more nodes. It became increasingly the most accurate after the PCA dimension reducing, and slightly less accurate with the Random Forest Classifier. For experiment 1, the accuracy scores are the lowest with the 50th epoch only reaching 38%. This, especially in comparison with the accuracy of the other models, clearly shows that the small number of nodes (for experiment 1 there was only one node) led to very inaccurate and unreliable predictions. Experiment 2 increased the hidden layer nodes to 2 and had a significantly improved result for the accuracy score which nearly doubled after the 50th epoch. This shows that increasing the hidden layer nodes leads to better results for the DNN. For experiment 3, I chose to use the same 78f input nodes but 200 layer nodes which had a great impact upon the results. The accuracy was improved to a 99%, following the trend shown from experiment 2, that additional nodes led to more accurate results. Experiment 4 showed included a PCA dimension reduction from 784 to 154. Using only 154 input nodes, and 154 layer nodes, the model had a 100% accuracy prediction. This is likely due to the reduction of dimensions, but still maintaining a high level of layer nodes. Lastly, for experiment 5, we found the top 70 pixels after running the random forest classifier. We used the classifier to first find the importance of the 784 features and then picked the top 70 from both the training and test datasets. This model had a diminished accuracy result from experiment 3 and 4 highs, with the result dropping to 97%. This was likely due to the input nodes being cut to 70, however the accuracy prediction was still higher than experiment 1 and 2. This experiment had 154 layer nodes showing the importance of those nodes affect on the accuracy of results.

In conclusion, the results of the 5 experiment show that the hidden layer nodes and input nodes are very important to the results. When the experiments had higher layer nodes, that accuracy of the prediction was vastly greater than those with lower amounts. This was also true with the input nodes, as experiment 5 showed a lower result with lower nodes than experiment 3. Experiment 4 had the highest accuracy prediction, likely due to the PCA reduction in dimensions.