Allison R. Deming

IST 718 Big Data Analytics

Week 9 Lab Exercise

12/05/2023

**Introduction**

Machine Learning is a subset of Artificial Intelligence and strives on completing specific tasks by prediction based on input and algorithms. Deep Learning is a subset of Machine Learning, which attempts to mimic our own brain’s network of neurons to learn. This is how image recognition works today. We can upload an image on social media platforms, and Facebook/Instagram is able to recognize the facial construction of someone in the background and make a prediction. Same with self-driving cars. These cars wouldn’t work if they car wouldn’t be able to recognize an image of a stop sign. It will study and read the stop sign and stop. For Humans, these images are second nature. We see a stop sign, which we know to stop based on either the color, the shape, or the word, “STOP” on a stop sign. This is how image recognition work. Image Recognition is the ability of computers to look at a photograph and understand what’s in the photograph. However, computers, need a million images of a stop sign to process and recognize a sign for a car. In this study, we will be looking at the Fashion-Most Dataset, a beginner’s data set filled with clothing images. We will be using two different algorithms to train the model of images to obtain predictions of the trained images, and then gather accuracy percentages to benchmark the algorithm and the prediction.

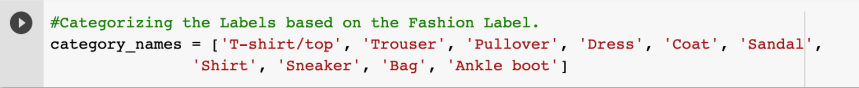
Information About The Data

The Fashion-MNIST dataset is made up of images from fashion chain Zalando. It contains a training set of 60,000 images and a test set of 10,000 images. Each image is a 28 x 28 grayscale image associated with the label from 10 categories. Those labels are from 0-9 and those labels are associated with T-Shirt/Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, sports Shoe, Bag, and Ankle Boot. It is a drop-in replacement of the MNIST data set which contained a list of handwritten digits between zero and nine. Below is the Label and the description of how the data is organized into the Fashion-MNIST dataset.

A screen shot of a computer screen

Description automatically generated**Categorizing the Dataset**

The dataset used for this study was downloaded from the Fashion-MNIST repository directly. No additional data cleaning steps were performed other than creating the labels for each image.

Inside this list, we have the corresponding label names with the numbers associated with the fashion item. The next thing wants to do is grab those category names and labels and make sure that those images were labeled correctly. We grabbed 16 different images to test it out and the output of those images is in the picture below.

A collage of different shoes

Description automatically generated**1st Algorithm – Keras Neural Network**

So, to explain what is going on with Kera’s Neural Network, we take one 28 by 28 image, row by row flatten it. Whatever the length of the rows goes in our image, that will be the number of times we repeat this process and flatten them out. At the end of this flattening process, we have one long row which has 784 numbers. We start off with 784 input nodes because these are the number of pixels each image contains and each of these nodes corresponds to a pixel of the image. and we have two hidden layers with 128 nodes and 64 nodes with “ReLU” in brackets which stands for Rectified Linear Unit which activates the function. The final output has 10 nodes, which corresponds to our 10 categories.

A diagram of a network

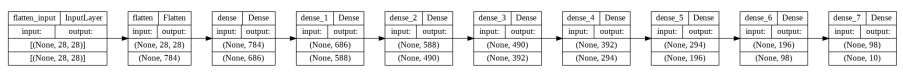
Description automatically generatedEach pixel has an RGB number that determines the shade of the color which will aid Keras neural Network based on the position of these colors. Anytime the algorithm detects a color, it is able to process the image based on the changes of the numbers in a certain prediction to make a prediction.

A screenshot of a computer screen

Description automatically generatedNow that it’s known how the algorithm work, we can use Keras to flatten our model, create the layers for the algorithm, and show the loss of data and the accuracy of the predictions based on our model. Below is the code on how the layers and nodes were created.

A screenshot of a computer program

Description automatically generatedBelow is a visual representation of the layers and their nodes, showcasing the initial mode for accuracy.

In this example, there were 9 layers created, and the last layer contains 10 nodes for the prediction model.

A table of data with numbers and letters

Description automatically generated with medium confidence

According to our first model, the accuracy was about 93.2% with a loss of 18%. In the next model, it was decided to lessen the number of layers, and lower the number of Epochs in the model to see what kind of effect lowering the model would have on the accuracy.

A screenshot of a computer program

Description automatically generated

We can see that the accuracy and loss are almost identical in our model. This algorithm was multiple times, and the model tends to plateau at around 90% Confirming our suspicion that when we have fewer layers and nodes, we will have less accurate predictions, but in this case, since the Fashion-MNIST dataset is easy to run in this algorithm, it’s easier to make these predictions with a simple image recognition dataset.

A diagram of a computer code

Description automatically generated with medium confidence**2nd Algorithm - Naïve Bayes (Gaussian)**

Naive Bayes is a machine learning method that is used to predict the likelihood that an event will occur given evidence that's present in the data. It assumes all the predictors are independent of one another. It can be used for multiclass classification and is one of the simpler Algorithms to use in Python. For our Naïve Bayes Algorithm, the Gaussian model is selected. The Gaussian model is best used for images because it makes predictions from normally distributed features.

A screenshot of a computer code

Description automatically generated

As stated before, the simplicity of the Gaussian model makes it appealing to new programmers. The first and only requirement for the model is to shape the train and test data sets to 784 pixels. Once done, the Gaussian model was fit with the train images and labels. Then, test the prediction of the model using our test images. A confusion matrix is used to calculate the accuracy of prediction labels and the actual test labels.

A graph with colored squares

Description automatically generated

**Accuracy of the model: 58.56%**

**Conclusion and Final Thoughts: Keras Neural Network vs. Naïve Bayes Algorithm – Gaussian**

In terms of which method is more accurate in the study, the Keras method outperforms the Gaussian model 93.20% to 58.56%. This outcome makes sense given the level of the complex differences between the two Algorithms. Keras, the most complex, allows us to construct our Neural Network to our preferences. We can modify the combination of hidden nodes, epochs, and images to the model to improve our model’s accuracy and reduce our loss. However, that is at the expense of computer performance to execute increasing preferences in the Keras Neural Network. Gaussian, the less complex, does not give us the ability to modify the parameters of the Algorithm (i.e., increase the number of hidden nodes, or epochs). Provide training data to the Gaussian model, predict an outcome with test data, and verify the accuracy. What is gained in the simplicity of the computer’s performance, is lost in the model’s accuracy? At the end of the day, it comes down to balancing the expectations. Do we the most accurate outcome, then expect to wait sometime for the Algorithm to complete. Or do we want the quickest response, then expect that the accuracy will suffer.