Digit Recognizer

Project Description, Details, and Goals

The goal of this Digit Recognizer project is to allow us to learn computer vision fundamentals using the famous MNIST (Modified National Institute of Standards and Technology) data. We shall be using machine learning algorithms to correctly identify digits from a dataset of tens of thousands of handwritten images. The topics we shall be practicing are computer vision fundamentals which include simple neural networks and methods that deal with regression and classification such as SVM (Support Vector Machine) and KNN (K-nearest neighbors). Ultimately, the end goal is to take an image of a handwritten single digit, and determine what that digit is. For every single image ID in our test set, we should predict the correct label.

Details about the Data

With the digit recognizer, the data given for us to work with are training and testing data sets of multiple images. Within the training data set, there are 42000 images, each with 784 pixels. Included in the training data set is the label column which indicates the digit of the hand drawn image. While similar to the training set, the testing data includes 28000 images with 784 pixels with the addition that this data set doesn't include the label column from the testing data set. Each individual pixels are their own column in the data set where their values range from 0 to 255. The lower the pixel value the brighter the pixel, while a higher pixel count makes the pixel darker. With the training data, we'll be using all of the 784 pixel columns as features and drawn digits from the column label as the labels itself. After setting up all of the features and labels we'll be needing from these datasets, we'll be able to apply algorithms that will help recognize a number from the hand drawn images.

The developed methods, algorithms, and tools to address the project's requirements

In order to visualize the data, we implemented matplotlib.pyplot which is plotting library which produces quality figures in a variety of hard-copy formats and interactive environments to display the hand drawn image by using the pixel values from the features. We also implemented MXNet which automates common workflows, so standard neural networks can be expressed concisely in just a few lines of code that allows us to accelerate any numerical computation.

Some of the methods/algorithms we addressed are random forest, decision tree, K-nearest neighbors, and logistic regression along with with their appropriate classifiers. The KNN method of classification is one of the simplest methods in machine learning its most basic level of machine learning, it is essentially classification by finding the most similar data points in the training data, and making an educated guess based on their classifications. Although simple to understand and implement, the KNN method has seen wide application. Random forest is a supervised learning algorithm. It creates a forest and makes tries to make it as random as possible which can be used for both classification and regression problems, used by many current machine learning systems.

Decision Trees are a very commonly used tool in data mining to achieve a particular goal or target at each node, widely used in machine learning. However, most of the time trained with the “bagging” method. The idea of the “bagging” method is that of a combination of learning models to increase the overall result.

Logistic regression is a very powerful algorithm it looks at the relationship between a training variables and testing variables. Provided the right representation of the labels and features it can return very accurate results, which makes it a very accurate machine learning algorithm. We begin our project by converting the .csv files provided by the Kaggle competition page into data frames and then we used that data to train and test.

The developed code and final results

Method	Accuracy | Execution Time
Random Forest (n_estimators = 100, bootstrap = True, random_state=3)	96.12% | 16.3s
Random Forest (n_estimators = 700, bootstrap = True, range_state=2	96.41% | 1m 52.1s
Random Forest (n_estimators = 500, bootstrap = True,random_state=4) with Voting ”Bagging”	96.44% |1m 20.8s
K Nearest Neighbor (k = 3, test_size=0.1, random_state=42)	96.76% | 3m 7s
Decision Tree (test_size=0.25, random_state=4)	85.32% | 7.3s
Logistic Regression (test_size=0.25, random_state=4)	90.45% | 1h 25m 17s
Logistic Regression (test_size=0.1, random_state=2)	90.95% | 42 m 33s

Problems

Some of the problems encountered in this project was limited memory resources and the amount of data available to train our algorithms. Since the program takes the data from reading a image pixel by pixel and then plotting it to represent a number, the program takes a while to run and sometimes you would even get a error saying insufficient memory. Also, to further improve our accuracy scores we would need more training data sets to train our algorithms. One solution to increase the efficiency of this program might be to normalize the testing and training data set before using them. Another problem we encountered while working on this project was creating an Area under Curve (AUC) and plotting the ROC curve. The issue was that it took a really long time to run and after a couple of hours, it still wasn't able to generate the Area under Curve and ROC curve.

Team Responsibilities

Among the members of our team, Simon is responsible for testing the machine learning methods. Kevin is responsible for configuring the simple neural network. Andy is responsible for setting up the code to take inputted images and recognizing the image to display a proper output. Angelo is responsible for researching and applying computer vision fundamentals. Christopher is responsible for the overall quality control of our project, research and data analysis.