Udacity Machine Learning Engineer Nanodegree Program - Capstone Proposal

Capstone Proposal - Digit Recognizer

Taihwa, Song

Date: March 7, 2020

Domain Background

The traditional way of documentations were typically done on a piece of paper. This is extremely cumbersome because it is not only inefficient to make extra copies but also difficult to search words from text. With the widespread use of personal computers and smartphones in the 21st century, many documents are now digitized. This Modern way of storing documents made it easy for researchers to share findings across the world with a single click of a mouse.

Most recent documents can be easily searchable because they are already digitized to begin with; however, the legacy/traditional documents are still kept on a piece of paper. This makes it difficult for modern people to find relevant traditional documents and many efforts were put into turning legacy papers into digital documents. One method of turning legacy texts into digitized texts is leveraging reCAPTCHA [1]. This technology is an advanced version of CAPTCHA[2] that utilizes some portion of text-based CAPTCHA tests to translate an image of a hand-written word into a digital word. Another technique is to utilize machine learning to translate a set of documents into a digital copy. This technique is called handwriting recognition.

The recurrent neural network and deep feedforward neural networks (researched by a group of researchers from Jurgen Schmidhuber at the Swiss AI Lab IDSIA) were proven to be effective against the handwriting recognition problems [3].

Problem Statement

The primary goal of this project is to predict digits from handwritten images. Recognizing digit is the most fundamental approach of handwriting recognition. This is a supervised classification problem where the model is expected to predict a number ranging from 0 to 9 by looking at an image with 28 pixels in height and 28 pixels in width, for the total of 784 pixels in total. The performance of the model can be computed by comparing the predicted number with the actual number.

Dataset and inputs

The origin of the data comes from MNIST and the data was downloaded from Kaggle, specifically "Digit Recognizer" competition challenge [4]. The data files train.csv contains 42,001 lines (where the first line is header and the rest of numerical data). Each line contains 785 columns. The first column is label: the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated images. Each pixel column in the training set has a name like pixel_x, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i * 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixel_x is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

Solution statement

Provided that this is a classification problem, there are a number of difficult approaches to tackle this problem.

Pre-Data Processing

Image resizing

Depending on the person who wrote the digit, the size of the digit may vary. Every image in the data will be resized to maximally use all pixels. This could be easily accomplished using a scikit-image library.

Binary feature transformation

The training data are gray scale images ranging from 0 to 255. Depending on the type of pen that the person used to write the digit, the darkness of a pixel may vary due to the property of the pen: thickness of the tip and ink type. To better make the model to utilize light-gray pixels, each pixel will be translated into a binary: 0 if pixel is empty and 1 if pixel has at least some darkness in it.

PCA

Given that the dimensionality of the data is high (255), it is often recommended to reduce the dimensionality through PCA (Principal component analysis). It is a statistical method to translate a high dimensional data into smaller dimensional data while maintaining the high variance. This is a useful technique for this problem because it can remove pixels that are rarely used. For example, the first pixel (located at the left top) is often blank because all digit images are already centered and rarely any digits require the first pixel to be filled in. PCA will recognize that the pixel is rarely used (such that it has low variance across different digits) and it will remove the pixel which will result in dimensionality reduction.

Data split between training and testing

Given that there are 42,000 data, 80% of data will be used for training and 20% will be used for testing. This is chosen because 42,000 is neither a small nor a big number. In this case, 80 to 20 ratio split is a good choice.

Training Algorithm

A number of different learning algorithms will be used to testify the result: Logistic Regression, K Nearest Neighbours, Neural Network. Logistic Regression is one of the simpliest training algorithm that could be used to justify the baseline of the model accruacy. Neural network is another algorithm that can be used to explain the deeper layer of patterns.

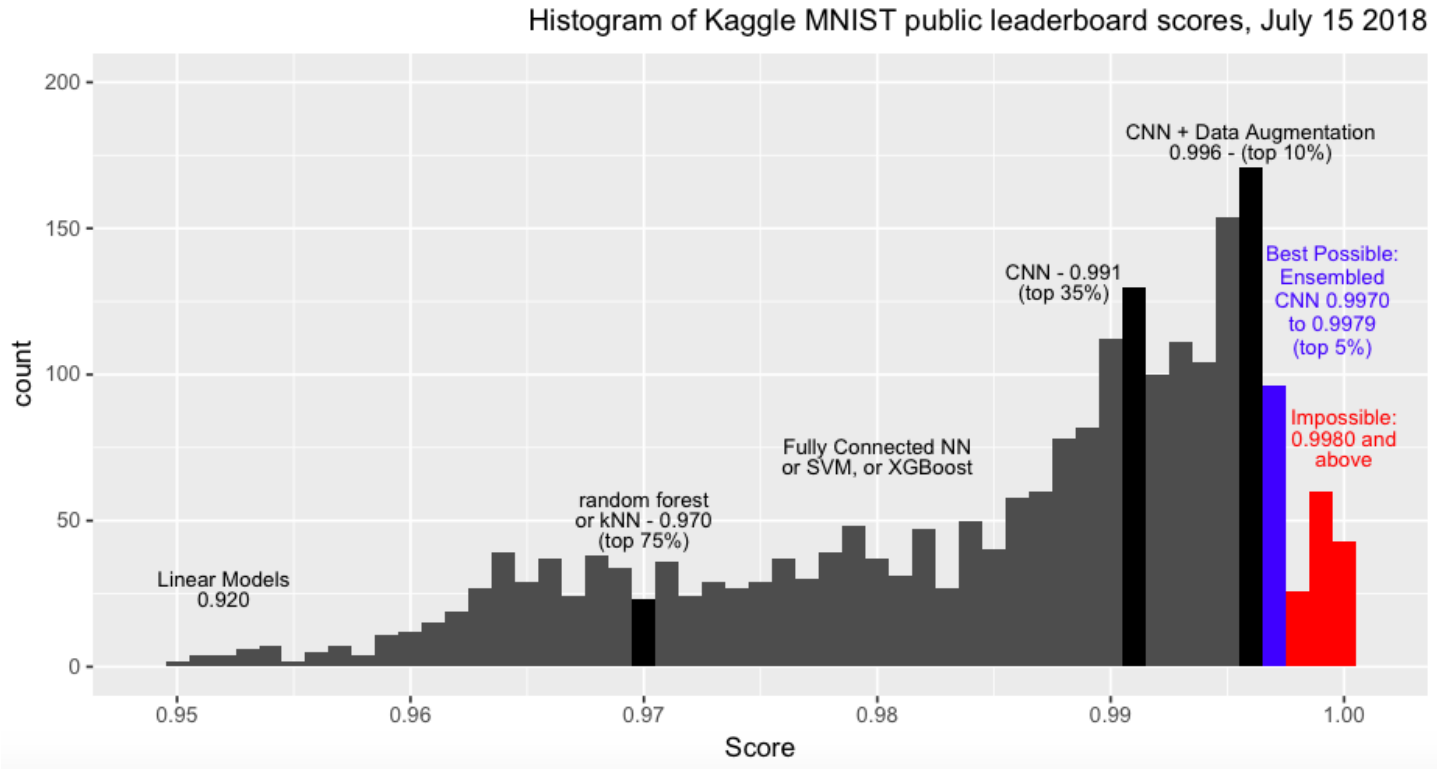
Hyper parameter tuning

Hyperparameter tuning techniques like Grid Search algorithm will be used to optimize the final model to find an optimal hyperparatmers for the learning algorithm.

Benchmark model

The histogram of Kaggle MNIST public leaderboard scores [4] will be used for the benchmarking model. The digit handwriting recognition problem is a publically known competition and many teams participated to share their model performances. The performance outcomes from the community is a good baseline to compare for this project. The result of the benchmark model is the accuracy score of the model. The following is the accuracy score of benchmarking models for each training algorithm:

- Linear regression: 0.92 (see 'Linear Model' benchmark score below)
- KNN: 0.97 (see 'KNN' model benchmark score below)
- Neural Network: 0.98 (See 'Fully Connected NN' benchmark score below)



Evaluation Metrics

Accuracy score will be used as an evaluation metrics. Accuracy score is calculated by finding the ratio between the sum of true positives and true negatives, and the total count of dataset. In the case of mutli-class classification problem like this project, true positive and the true negative are essentially the same; therefore, the accuracy will be calculated by finding the ratio between the sum of true positive positive and the total count of dataset. The outcome of the accuracy is a float number between 0 and 1 where number closer to 1 shows a better performance.

Project Design

Throughout the project, work will be done on a jupyter notebook. The initial data will be loaded as numpy arrays and Pandas dataframe for data inspection purposes via visualization. Raw data will be translated into actual images so that we better understand the shape of the data. Once we are comfortable with the data, pre-processing including digit size re-scaling, binary feature transformation and principal component analysis will be applied. Once the data is ready, the data will be splitted into 2 sets: training and testing. The training set will contain 80% of the original data and the testing set will contain 20% of the original data. Once the data split is completed, different learning algorithms without tuning will be applied to see the base model performances (accuracy). By comparing the baseline models, a model that works the best will be picked based on the performance. Once a good model is chosen, a hyper parameter tuning will be applied to boost up the performance scores.

References

1. https://en.wikipedia.org/wiki/ReCAPTCHA
2. https://en.wikipedia.org/wiki/CAPTCHA
3. https://en.wikipedia.org/wiki/Handwriting_recognition
4. https://www.kaggle.com/c/digit-recognizer