Machine Learning: Handwritten Digital Recognition Using a Simplified K-Nearest Neighbors Algorithm in Javascript.

Leon Do

***Abstract* – Hand written recognition is common exercise into artificial intelligence (AI). However, most programs are written in Python using prebuilt AI libraries. The purpose of this paper is to demonstrate machine learning in a different programming language (Javascript) without any AI libraries. In essence, writing from scratch.**

This paper will describe the process from a hand written image to binary numbers for the machine to predict.

I. Hand Written Image to NodeJS

The user will hand draw a number from 0 to 9 into an HTML 5 canvas (Figure 1).

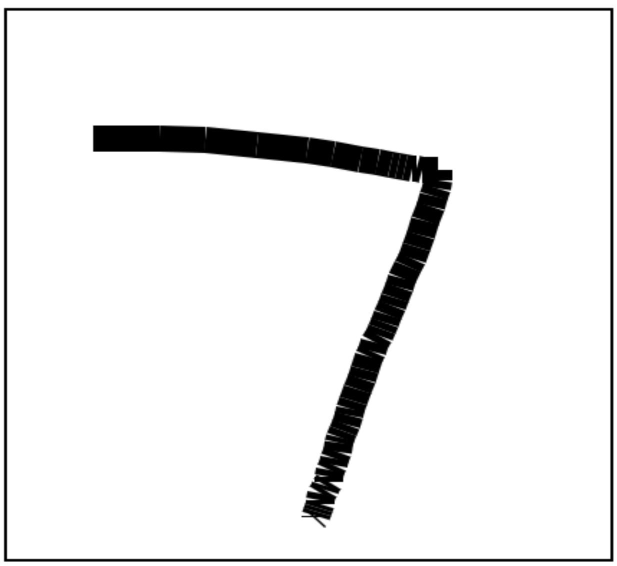

Figure 1

Once the user clicks assess, the image is converted into an Portable Network Graphics (.png) file then into a base64 string. The string is then posted to a NodeJS server.

II. Image Resize Then Converting to Binary

The NodeJS server receives the base64 string then converts it back to a png, then using an npm module called JIMP (JavaScript Image Manipulation Program) would convert to greyscale and resize the image to 28x28 px.

The 28x28 pixel size is specific for the dataset discussed later in the paper.

Each pixel (28*28 = 784) is then analyzed by grabbing all the pixels in an image and return the result as an array. Let's label this array `'test array'`.

White pixels are represented as 0 and black/grey pixels are presented as 1. The test array can be visualized in a grid as seen in Figure 2.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 2

To further standardize the data, the test array is then rearranged to remove the 'white' space on top (var top), as seen in Figure 3.

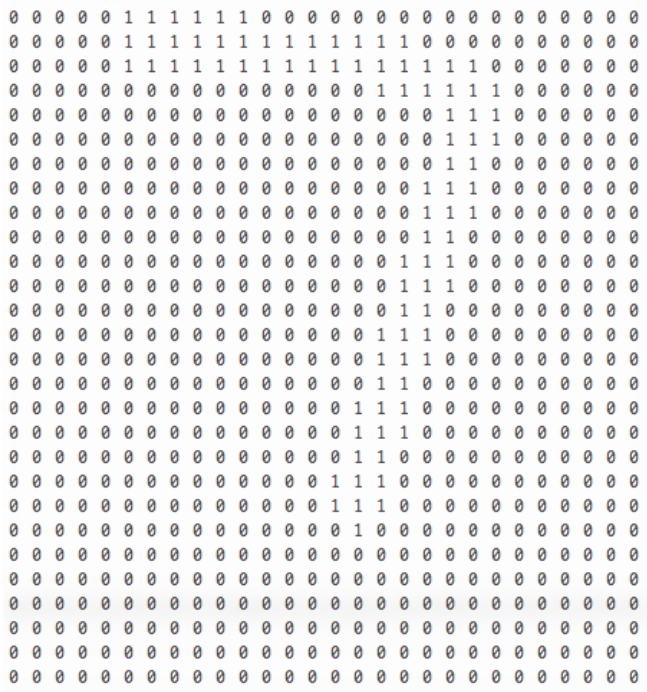

Figure 3

This can be done by using the function below:

```
for (var i = 0; i < arr.length; i++){
    if (arr[i] === 1){
        var top = Math.floor(i / 28) * 28;
        return top;
    }
}
```

The modified test array can then be compared to the database.

Database Overview

A fraction of the data is taken from the MNIST dataset[1] and stored into a Mongo database. The database consists of one array with approximately 2000 objects. Each object has a key of `answer` with value 0-9 and key `arr` with 784 0's and 1's.

This means each array corresponds with a number between 0-9. This database is used as the control. Note: `arr.length` returns 748 which matches the 28x28 pixel grid discussed previously. This length is important in calculating the distance using K Nearest Neighbors.

Figure 5a is an example from the database. The array maps out the number 7, which will look like Figure 5b.

Figure 5a

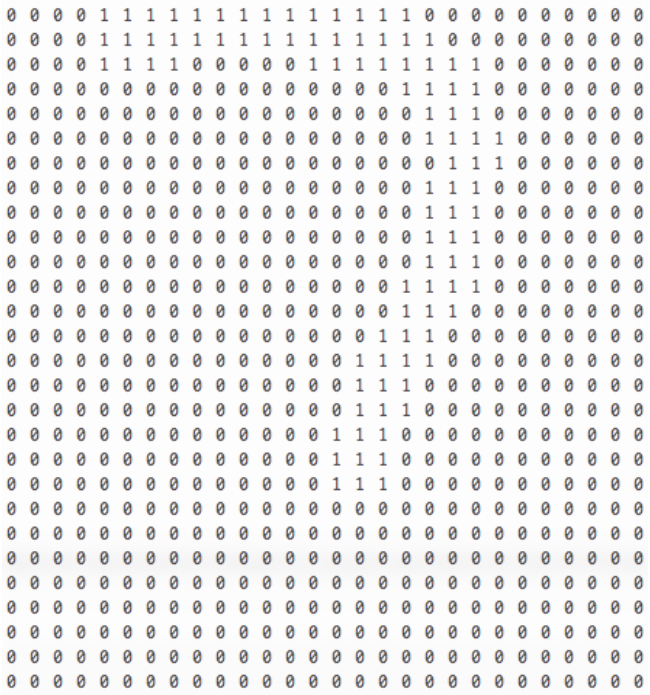

Figure 5b

K Nearest Neighbors Algorithm

Using a simplified k-nearest neighbors algorithm (Figure 6), the test array is compared with all of the arrays in the dataset using the Euclidean distance equation.

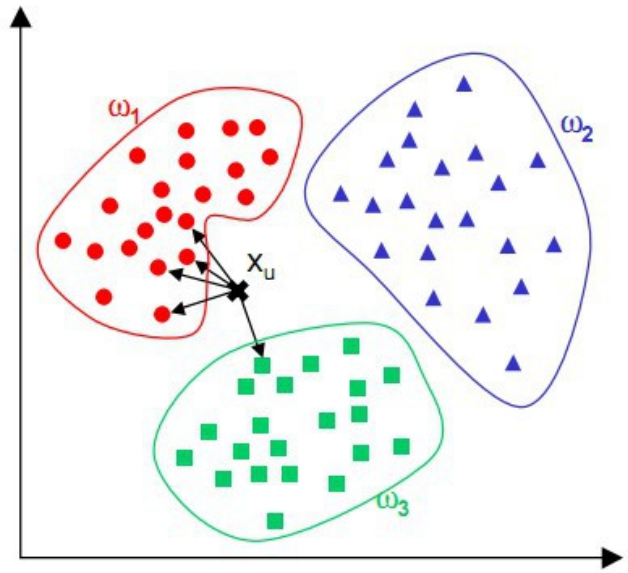

Figure 6. Two dimensional plane

Euclidean Distance Equation Overview

The Euclidean distance equation calculates the distance between point p and q. A two dimensional Euclidean plane is similar to Pythagorean theorem: $a^2 + b^2 = c^2$ or Figure 7.

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

Figure 7

For a three dimensional plane, distance can be calculated as: $a^2 + b^2 + c^2 = d^2$ or Figure 8.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

Figure 8

For n dimensions, distance can be calculated as shown in Figure 9.

$$\sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Figure 9

The Euclidean Distance Equation written in Javascript is written below where `arr1.length = 748`, or simply, we are calculating distance for a 748 dimensional plane.

```
function euclideanDistance(arr1, arr2){
        var answer = Math.pow(arr2.map(function(a,i){
            return Math.pow(arr1[i] - arr2[i], 2);
        }).reduce(function(a,b){
            return a+b;
        }),0.5)
        return answer;
    }
```

The euclidean distances are stored into an array called `distanceArr`. According to the k-nearest neighbors algorithm, the best match is one with the shortest distance (or smallest value) in distanceArr. The javascript snippet is:

```
var shortestDistanceIndex =
distanceArr.indexOf(Math.min.apply(Math,distanceArr))
```

The index is then matched with the database index to find the predicted answer.

References

[1] "THE MNIST DATABASE."*MNIST Handwritten Digit Database, Yann LeCun, Corinna Cortes and Chris Burges*. N.p., n.d. Web. 27 Mar. 2017.