Application of Neural Network In Handwriting Recognition

Shaohan Xu, Qi Wu, and Siyuan Zhang
Stanford University
353 Serra Mall
Stanford, CA 94305 USA
{shao2, qiwu, siyuan}@cs.stanford.edu

***Abstract*—This document describes the application of machine learning algorithms to solving the problem of handwriting recognition. Two models were explored, namely Naïve Bays and Artificial Neural Network, and ANN was found to generate more accurate recognitions. By setting up our model and training on the MNIST database of handwritten digits, we were able to achieve recognition of handwritten digits with decent accuracy on testing data obtained from Stanford students as well as data from MNIST database. Possible future improvements of this task are discussed in the end.**

Keywords—handwritten character recognition; Naïve Bays; Artificial Neural Network

I. Introduction and Motivation

Handwriting recognition can be divided into two categories, namely on-line and off-line handwriting recognition. On-line recognition involves live transformation of character written by a user on a tablet or a smart phone. In contrast, off-line recognition is more challenging, which requires automatic conversion of scanned image or photos into a computer-readable text format.

Motivated by the interesting application of off-line recognition technology, for instance the USPS address recognition system, and the Chase QuickDeposit system, this project will mainly focus on discovering algorithms that allow accurate, fast, and efficient character recognition process. The report will cover data acquisition, image processing, feature extraction, model training, results analysis, and future works.

II. Data Acquisition

The MNIST database of handwritten digits[i] is used as training sets. The MNIST database has a training set of 60,000 examples, and a test set of 10,000 examples. 30 sets of data - telephone numbers and zip codes - were randomly collected among Stanford students for testing purposes.

III. Image Processing

We use 28 by 28 pixels to represent all the images. All the digits are size-normalized without changing the shape to fit in the fixed size. Collected testing images need to be scanned before being processed. The training dataset contains individual digits, while the testing images need to be partitioned into separated digits for further processing.

Fig. 1. Image Processing Flow

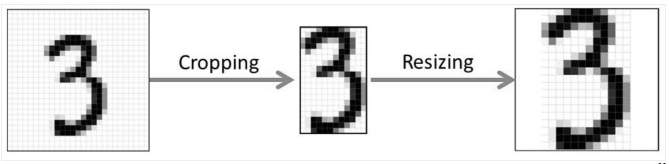


Image processing is applied to the dataset using Matlab[ii]. List below illustrate the basic steps:

- Read image: Reads the image into Matlab as an array of RGB values.
- Convert image to gray scale: Convert the array to a gray scale array.
- Image cropping: Get rid of the blank spaces near four edges of the image.
- Resizing: Resize the image to be 28*28 pixels without changing the shape of the digit. This usually means that the digit would fill all 28 pixels in one dimension, while leaving blank some pixels in the other dimension.

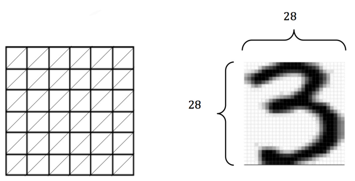
For testing images that consist of more than one digit, the digits need to be separated prior to cropping and resizing. Digit separation is done through these steps in Matlab:

- Edge detection
- Image dilation
- Image filling
- Blobs analysis (Bounding Boxes and Centroids)

IV. Feature Extraction

For each training example, i.e. the digit image: $I^{(i)}$, the feature, $x^{(i)}$, were selected to be all the gray scale pixel values of an image. Here the x's are 784 dimensional vectors in R^{784}, and I's are 28 by 28 matrixes. Feature vector was arranged so that the pixel values were extracting diagonally, and starting from the upper-left to the bottom-right of a matrix.[iii]

Fig. 2. Diagonal Based Feature Extraction of A Gray Scale Image



Naïve Bayes and Neural Network models are used to determine the dimension, extraction sequence, and derivation of features based on the training errors of each model. More details about Naïve Bays and Neural Network models will be introduced later in model training section.

Features determination processes:

a) Dimension: after normalizing each image, although the size and format are the uniform for each of them, the size of the matrix, or the total number of features, still need to be determined for optimization purposes. This task is done by trial and error methods, and the results are summarized in the table given below:

TABLE I. TRAINING ERROR VS. FEATURE DIMENSION USING NAÏVE BAYS MODEL

Feature Dimension	Matrix Size	Training Error
35	7 by 5	32.4%
784	28 by 28	25.0%
3,500	70 by 50	24.9%
10,000	100 by 100	24.9%

As can be seen, too few of features will compromise accuracy, and too many features will create redundancy, which increases run time significantly. Therefore the total numbers of features were chosen so that the accuracy is maximized and redundant information is minimized.

b) Extraction Sequence: Under what sequence the features are extracted from the data influences the accuracy of classification as well. Three approaches - vertical, horizontal, and diagonal - were investigated to explore their effects on the classification accuracy. Experiments were performed for both Naive Bays and Neural Network models to ensure its validation, and the diagonal extraction sequence is adopted eventually. The training error for each case is shown below:

TABLE II. TRAINING ERROR VS. FEATURE EXTRACTION SEQUENCE

Feature Extraction Sequence	Training Error
Vertical	95.2%
Horizontal	96.7%
Diagonal	98.9%

c) Feature Derivation: three types of feature-extraction approaches were investigated and summarized below:

- Hole Detection: after running Naïve Bays, we found the algorithm was doing badly on distinguishing resembled numbers (e.g. '5' and '6', and '4' and '9'). To improve the performance of recognition, a convenient feature that was able to detect whether there is a close circle in the picture was added. For example, in most human written numbers, '6' has a circle while '5' doesn't, '9' has a circle while '4' doesn't. The modified algorithm did do better on distinguishing 4s and 9s, however, the overall performance turned out to be worse.

- Zoning: to further decrease the total number of features input into the model, after processing, the image matrix was portioned into 4 by 4 sub matrixes. Each of 49 sub matrixes was treated separately. The sum operation was first performed on each diagonal line of a sub matrix, and then averaged to obtain a single value as the representation of the entire zone. Total numbers of features were able to drop from 784 to 49, while the training error increased by about 5%.

- Symmetry: similar to hole detection, a feature determining whether the image is vertically and horizontally symmetric is added to better distinguish resembled digits. However, with this feature being added, the training error was increased by around 10%.

As a result, only processed raw data was extracted and used as inputs for model training due to the fact that derived features compromise the overall performance of classification.

V. MODEL TRAINING

Based on features extracted from the example, Naïve Bays model was firstly used to obtain a quick and rough understanding of the data and features. Then neural network was applied to generate a more accurate recognition.

A. Naïve Bayes

The nature of the problem motivated us to apply Naive Bayes algorithm first to get a rough idea of the problem, which turns to be inacceptable with almost half of the time generating a wrong estimation. However with some modifications, the training error could be increased by around 15%.

1) Model modification

a) Introduces Laplace smoothing: although character recognition is different from spam email classification problem, the case where the pixel of all examples for a given position is zero still exists. Therefore, laplace smoothing was introduced to eliminate the problem of probabilities being estimated to be zero. For all training examples labeled as d, the probability of a particular binary pixel j being 1 is computed and represented as [2]:

$$\phi_{j|y=d} = \frac{\sum 1\left(x_j^{(i)} = 1\ and\ y^{(i)} = d\right) + 1}{\sum 1(y^{(i)} = d) + k} \tag{1}$$

b) Adding punishment: punishment was added to correct the case, where for a given position, the probability of the pixel value being equal to one is high and the corresponding pixel in the test example is found to be zero, so that the model can better differentiate two resemble digits. This apprach is to rescale the gray scale values from [0, 1] to [-0.5, 0.5] instead.

B. Nueral Network [1]

In this section, we describe the construction of our neural network model with the emphasis on the design of network and feature extractors. For the input layer, the feature for each node is the gray scale feature ranging from 0 to 1 representing how black the pixel is. So according to the model we selected, we have 784 input nodes. In the hidden layer, we tuned the number of nodes and find 700 nodes with sigmoid function as feature extractor works best for our model, which is shown in Fig. 3.

Just like in the Naïve Bayes model, where we projected our feature space from [0, 1] to [-0.5, 0.5] for punishment, the feature space was projected from [0, 1] to [-1, 1] for the intuition that if a pixel that has always seen to be very close to black or white in the training set is observed to be the opposite color in the test set, we subtract a little when calculating the score because they are less likely to be the same number.

Fig. 3. Artificial Nueral Network (ANN) Strucature

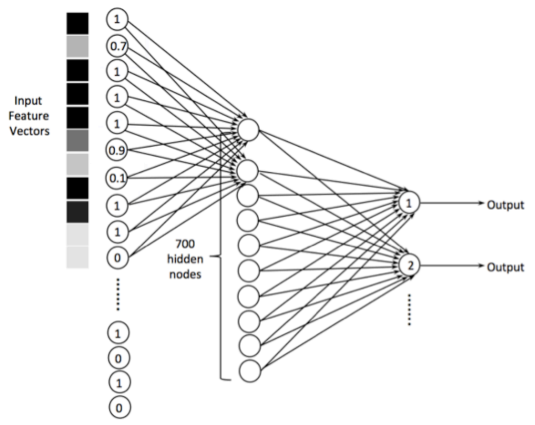
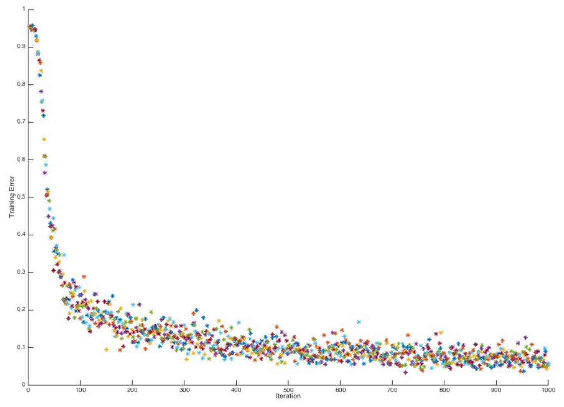


Fig. 4. Training Error vs. Number of Iterations



VI. Results Analysis

The diagonal based feature extraction of the processed raw data is used to train the model. The processed raw data is sufficient enough to capture most features of an image so that no extra or derived features are needed.

Classification error was simply calculated as the number of misclassified digits divided by the total number of testing examples. The optimized training error and generalization error for both models has been summarized in the table given below.

TABLE III. Training Error Generalization Error For Both Models

	Training Error	Generalization Error
Naive Bays	5% (60,000 samples)	24.9% (450 samples)
Neural Network	1.2% (60,000 samples)	2.7% (450 samples)

As can be seen, the Naive Bayes model generated a training error of as low as 4.9%, while a relatively high generalization error of 42.8%. The large difference between the training error and generalization error is because that the Naive Bays model failed to capture the characteristics of an image if the model hasn't seen the image before. However, Naive Bays, as a quick estimator for the determination of features extraction, led us to the right direction and narrowed down the range of exploration. The Neural Network model, in contrast, yielded much smaller training and generalization errors of 1.2% and 2.7% respectively. Using processed raw data, the model was able to classify the majority of handwritten digits except for the digits that are fairly vague and even human reader cannot tell.

The figure shown above illustrates that as the number of iterations increases the training error decreases exponentially. When 2,000 iterations were run, the training error can be as low as 1.2%, and can be even lower if more iterations were run. However, the balance between classification accuracy and runtime needs to be considered during practice.

VII. Future Works

Due to the limitation of time, only handwritten digit recognition is implemented completely. English character recognition is implemented partially without being optimized yet. Also, the current segmentation technique can only detect characters that have clear boundaries. If two consecutive characters were touching each other, they will be treated as one single character. Moreover, when testing the randomly collected images, the ones that are written really large and thin have poor performance during classification. After scanning and resizing, the image lost lots of pixels during the shrinking process. Therefore to resolve some of the issues mentioned previously the following tasks are recommended:

- Optimize image processing to retain the original features of the image as much as possible no matter the size, thickness, and angle of the character.
- Investigate better segmentation techniques[iv] to allow successful separation of characters touching each other. (Might want to consider set a fixed width when searching for individual letters)
- Introduce character recognition for different languages, including English, Mandarin, etc.

References

[i] The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/
[ii] Žiga Zadnik, Handwritten character Recognition: Training a Simple NN for classification using MATLAB)
[iii] J.Pradeep, E.Srinivasan, and S.Himavathi, Diagonal Based Feature Extraction For Handwritten Character Recognition System Using Neural Network
[iv] O. Matan, J. Bromley, C. J. Burges, J. S. Denker, L. D. Jackel, Y. LeCun, E. P. Pednault, W. D. Satterfield, C. E. Stenard, T. J. Thompson, Reading Handwritten Digits: A Zip Code Recognition System