Handwritten Digit Recognition using Neural Network

Roshni Musaddi [1], Anny Jaiswal [2], Pooja J [3], Mansvi Girdonia [4]
[1, 2, 3, 4] Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India.

Abstract – With the advancement of various domains in computer science and technology, smart devices have been invented in every field. Artificial intelligence has created a smart world with its invention. Digit and character recognition is a skill in demand in the industry for real time applications like signal processing, currency recognition, house number recognition, etc. In simpler words, a machine learning algorithm trains a machine according to the algorithm used and the machine is then tested using a test set. Many machine learning algorithms have been suggested to identify handwritten digits. In this paper, we have compared regularized logistic regression and ANN (Artificial Neural Network) to recognize handwritten digits with an accuracy of 94.9% and 97.5% respectively. Artificial Neural Network proved to be more efficient in terms of accuracy in recognizing handwritten digits. We used Octave 4.4.0 for training and testing the dataset which has 20x20 pixel images.

Index Terms – Artificial Intelligence, Machine Learning, logistic regression, digit recognition, regularization, vectorization, Artificial Neural Network.

1. INTRODUCTION

Machine Learning is an application in the field of AI that provides its system to learn and progress from its own experiences. We train our algorithm for a part of the dataset and then test the remaining dataset on that same algorithm to obtain accuracy. It has many real time applications. Handwritten digit recognition is one such application which is very useful when it comes to reading latent images, license number plate recognition, house number recognition, computerized bank check number readings, symbol processing, detect binary numbers, postal addresses, currency recognition, signature recognition and many other situations. This paper uses ANN to recognize handwritten digits. Artificial neural network is like a combination of neurons in our brain. Input is driven from one side (input layer) of our network and processed from one layer to another to finally give an output. We used Octave 4.4.0 which is a high-level programming language convenient for numerical data processing. In this paper we apply feedforward algorithm and use it to identify handwritten digits. Our dataset contains 5 thousand training samples with each example having 20 pixel x 20 pixel images of digits from zero to nine. Hence, we obtain a matrix of five thousand by four hundred pixels. Here, each row is a sample for training our algorithm. The network has three layers – input layer, hidden layer and output layer. The grayscale datasets have been extracted before implementing the algorithm. 70% of our datasets is the training set and 30% is our test set. Our network has an accuracy of 97.5% with test perform.

2. RELATED WORK

KhTohidul Islam[1] used ANN to anticipate handwritten digits. They have evaluated their experiments using MNIST dataset containing 42000 samples. These datasets contains 28 X 28 pixels images of low resolution. They have achieved 99.60% recognition accuracy.

Yawei Hou[2] used recognized MNIST datasets containing 60 thousand training samples and 10 thousand test samples. They have trained and tested the model with CNN and deep Neural Network containing double hidden layer BP neural network with different feature extraction. They have achieved the accuracy of 99.43% with CNN and an accuracy of 99.55% with deep Neural Network but if the learning rate is too large, the network training may not converge.

Tanya Makkar[3] have used CNN and KNN algorithm to find k nearest neighbour and set maximum label class of k to test data. On applying CNN the success achieved is an accuracy of 98.10% while KNN showed 96.20% accuracy. They have compared and showed that CNN has proved to obtain better results with a loss rate of 1.9% while KNN produces loss rate of 3.8%. The main drawback is that the computational cost is too high.

Renuka Kajale[4] tried to train the dataset with 79 samples. Two different assessments that were used for searching the features are originality and information. Recognition rate of about 98% is obtained using two classifiers on a test set with 1000 digits per class. This can be applied to tasks only for which image patterns can be segmented. They got the accuracy of about 70% . With 200 samples, the accuracy increased to 95 %.

3. PORPOSED MODELLING

The training set is in .mat format which means that the dataset has been secured in an Octave matrix format, rather of a text scheme like a csv file. The input of 20 by 20 pixels is converted into 400 rows to form a vector. Each row has one training example in the matrix X. A matrix X containing 5000 by 400 input is obtained where each row constitutes the training set for our handwritten digit image. Another vector Y has 5000 dimensional vector which has labels for each of our training set. The digit zero is given the value ten as Octave

has no zero indexing. Therefore, a “0" digit is labelled as “10", while the digits “1" to “9" are labelled as “1" to “9" in the usual order. The code selects 100 rows randomly from the matrix X and passes these rows to the function displayData.

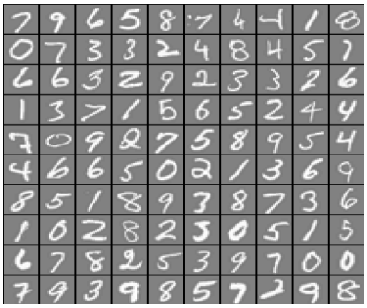

Fig 1: Training set randomly selected from X

This function maps every row to a grayscale image of 20 by 20 pixel and displays those images together. In order to make a multi-class classifier, we implemented multiple one-vs-all logistic regression models. Similar to binary classification, one-vs-all classification is advantageous when there is more classes than two. Each class is taken and compared with other classes to form a classifying model. Since there are ten classes, we train ten separate logistic regression classifiers. To make this training productive, it is important to assure that our code is vectorized properly. So we vectorized the cost function without any loops.

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \quad \text{and} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}.$$

Then, by computing the matrix product $X\theta$, we have

$$X\theta = \begin{bmatrix} — (x^{(1)})^T\theta — \\ — (x^{(2)})^T\theta — \\ \vdots \\ — (x^{(m)})^T\theta — \end{bmatrix} = \begin{bmatrix} — \theta^T(x^{(1)}) — \\ — \theta^T(x^{(2)}) — \\ \vdots \\ — \theta^T(x^{(m)}) — \end{bmatrix}.$$

Fig 2: Vectorizing the cost function

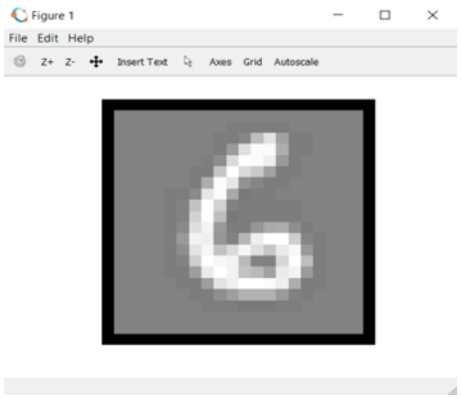

Fig 3: Input grayscale image for the number “six”

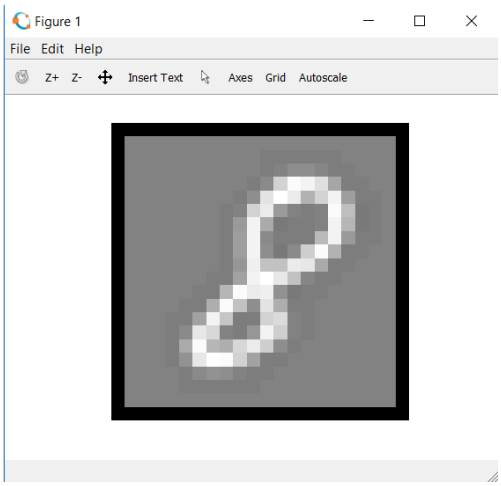

Fig 4: Input grayscale image for the number “eight”

The gradient is vectorized by the following equation:

$$\begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \right) \\ \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) \\ \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) \\ \vdots \\ \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x_n^{(i)} \right) \end{bmatrix}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \right)$$

$$= \frac{1}{m} X^T (h_\theta(x) - y).$$

where

$$h_\theta(x) - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(1)}) - y^{(m)} \end{bmatrix}.$$

$$\sum_i \beta_i x^{(i)} = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} = X^T \beta,$$

where the values $\beta_i = (h_\theta(x^{(i)}) - y^{(i)})$.

Fig 5: Vectorizing the gradient

We added regularization to our cost function. Regularization is done in order to avert over-fitting of the data. Over-fitting occurs when our model tries too hard to adjust with the random data points that don't actually represent the input value. These data points are known as noises which lead to a less accurate model. Also, bias and variance are two terms that come into context when we say “over-fitting”. Bias is an error that occurs when we try to implement a model that is just too simple when compared to the real-time complex problems whereas, variance is the amount by which our estimate of output would vary if we would have estimated it using a different dataset. Basically, high variance means that minor changes in the input dataset would result in much larger changes in our output. Regularization reduces this risk of

over-fitting by discouraging the learning of a more flexible model. For regularized logistic regression, the cost function is defined as:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$

Fig 6: Cost function for regularized logistic regression

We didn't regularize Θ_0 as it is used for the biased value. Furthermore, partial derivative of the regularized logistic regression cost for Θ_j is described as:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad \text{for } j \geq 1$$

Fig 7: Partial derivative of regularized logistic regression cost

Now we implemented one versus all classification and train multiple logistic regression classifiers, for every K class in our dataset. In our model, K=10 for each digit from zero to nine, though our model should work perfectly fine for any value of K. We train one classifier for each class. In particular, our code returns all classifier parameters in the matrix $\Theta \in R^{K(N+1)}$, where each row containing theta correlates to the logistic regression parameters learned for one class. We can perform this with a "for" loop from 1 through K, training each classifier individually. The y argument of this function is a vector of tags from 1 through 10, where the digit "0" has been mapped to the tag 10 in order to avert confusions with zero indexing. After training our one versus all classifier, we used it to conclude the digit for a given image. For each input, we compute the probability of it belonging to each class applying the logistic regression classifier that we trained. Our prediction function will choose the class for which the related logistic regression classifier gains the maximum probability and returns the class tag of one to K as prediction for each input example. The accuracy for our training set is nearly 94.9%.

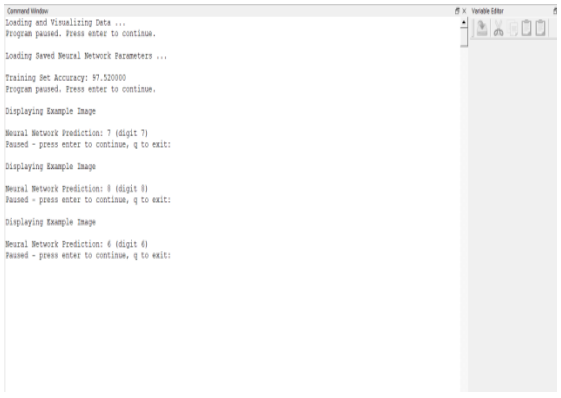

Fig 8: Accuracy obtained for regularized logistic regression

Since, logistic regression is a linear classifier and it cannot form more complicated hypotheses, we utilize a neural network and verify handwritten digits using the same training set as before.

A neural network is like a combination of neurons in our brain. Neurons take signals as input from the axon and gives output using a sigmoid function on the basis of weights. Similarly, the units in neural networks are layers of sigmoid functions where each unit or node is connected to all previous layer nodes. Weights are knowledge between two units from one layer to another. The neural network will be capable of representing complex models that form non-linear hypotheses. We used parameters from a neural network that we have already trained. Our goal is to implement the feed forward propagation algorithm to use our weights for prediction.

A feedforward algorithm is so called as it has no feedback, the inputs passes through every layer until it reaches the output. There are three layers. The input layer which takes in the input, the hidden layer which processes the data and finally the output layer which generates the output. The inputs are pixels of handwritten digits. Since the 20 x 20 sized images gives us 400 units of input layer (eliminating the additional bias unit which invariably outputs +1). The data for training was weighted into X and y. The parameters of theta1 and theta2 have dimensionality for a network with 25 units in the second layer and 10 units in the output layer (each for the ten digit classes).

We implemented the feed forward algorithm that measures $h_\Theta(x^{(i)})$ for each example i and returns the combined predictions. Identical to our one-vs-all classification approach, the prediction from our neural network will be the tag or label that has the maximum output $(h_\Theta(x))_k$. The accuracy obtained is about 97.5%.

Fig 9: Accuracy obtained for feedforward algorithm

An interactive arrangement will dispatch featured images from the training set, while the console outputs the predicted tag or label for each displayed image.

4. RESULTS AND DISCUSSIONS

The input for the digit "zero" can be given as:

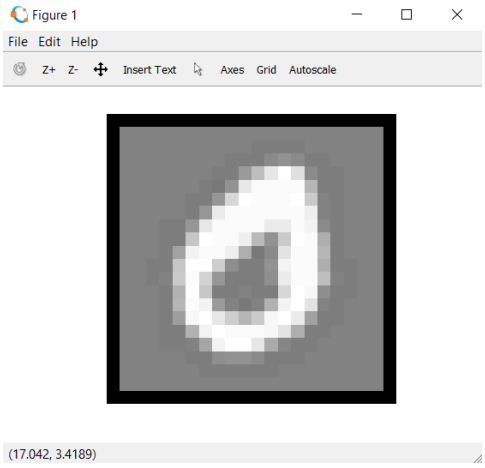

Fig 10: Input grayscale image for the number "zero"

As mentioned earlier, there is no zero indexing in OCTAVE. Therefore, the value obtained for zero was "10". The output is given below:

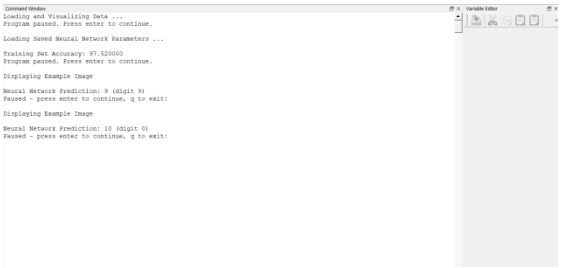

Fig 11: Value obtained for input "zero"

Another example shows the value for the digit 3. Input appears as a grayscale image of 20 x 20 pixel.

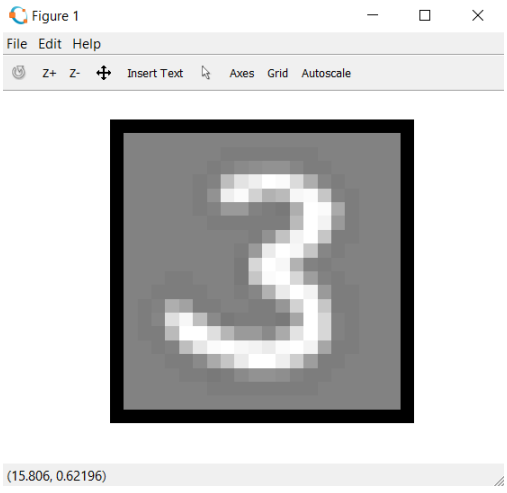

Fig 12: Input grayscale image for the number "three"

The output value for three is 3 itself. The screenshot of the output screen is given below.

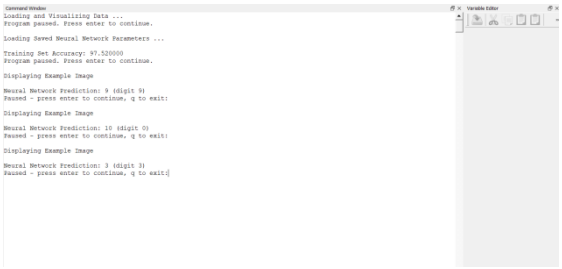

Fig 13: Value obtained for input "three"

5. CONCLUSION

The system architecture for the proposed model is described below. Here, the dataset is pre-processed into grayscale images and feature extraction is performed to reshape the features. Then we train the network according to our algorithm and map the images to its corresponding values. Output obtained is the value for each digit from zero through nine.

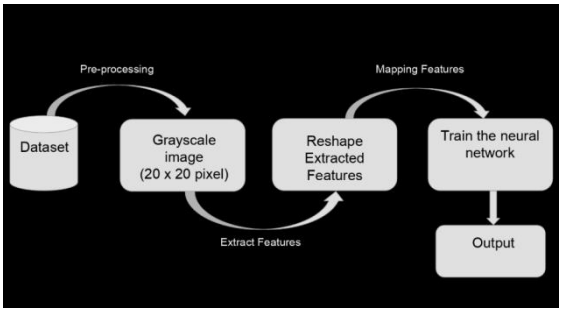

Fig 14: System architecture

This paper uses Artificial Neural Network to identify or recognize handwritten digits in Octave as it proved to be more efficient than regularized logistic regression. Regularization is done to avert over-fitting of our data. We have used feed forward algorithm to train the data and then test the data which contains 5000 samples each having 20 X 20 pixels of grayscale images. Our model takes less execution time and produces an accuracy rate of 97.5%.

REFERENCES

[1] KhTohidul Islam, Ghulam Mujtaba, Dr. Ram Gopal Raj, Henry Friday Nweke. "Handwritten Digits Recognition with Artificial Neural Network". 2017 International Conference on Engineering Technology and Technopreneurship (ICE2T).

[2] Yawei Hou,Huailin Zhao. "Handwritten Digit Recognition Based on Depth Neural Network" . 978-1-5090-6664-3/17/$31.00 ©2017 IEEE

[3] Tanya Makkar. "Analogizing Time Complexity of KNN and CNN in Recognizing Handwritten Digits".

[4] Renuka Kajale, Soubhik Das, Paritosh Medhekar. "Supervised machine learning in intelligent character recognition of handwritten and printed nameplate". 978-1-5386-3852-1/17/$31.00 ©2017 IEEE

[5] Zhenzhen, Guan, et al. "Intelligent recognition for surface roughness based on microscopic image texture characters." Measurement, Information and Control (MIC), 2012 International Conference on. Vol. 1. IEEE, 2012.

[6] Meher, Sukadev, and Debasish Basa. "An intelligent scanner with handwritten odia character recognition capability." Sensing Technology (ICST), 2011 Fifth International Conference on. IEEE, 2011.

[7] Jian, Zhang, Fan Xiaoping, and Huang Cailun. "Research on characters segmentation and characters recognition in intelligent license plate recognition system." Control Conference, 2006. CCC 2006. Chinese. IEEE, 2006.

[8] Hussain, Rafaqat, et al. "Recognition based segmentation of connected characters in text based CAPTCHAs." Communication Software and Networks (ICCSN), 2016 8th IEEE International Conference on. EEE, 2016.

[9] Mehta, Honey, Sanjay Singla, and Aarti Mahajan. "Optical character recognition (OCR) system for Roman script & English language using Artificial Neural Network (ANN) classifier." Research Advances in Integrated Navigation Systems (RAINS), International Conference on. IEEE, 2016.

[10] Singh, Dipti, et al. "An application of SVM in character recognition with chain code." Communication, Control and Intelligent Systems (CCIS), 2015. IEEE, 2015.

[11] Verma, Vivek Kumar, and Pradeep Kumar Tiwari. "Removal of Obstacles in Devanagari Script for Efficient Optical Character Recognition." Computational Intelligence and Communication Networks (CICN), 2015 International Conference on. IEEE, 2015.

[12] Chen, Datong, HervéBourlard, and J-P. Thiran. "Text identification in complex background using SVM." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 2. IEEE, 2001.

[13] Khaustov, P. A., V. G. Spitsyn, and E. I. Maksimova. "Algorithm for optical handwritten characters recognition based on structural components extraction." Strategic Technology (IFOST), 2016 11th International Forum on. IEEE, 2016.

[14] X. Yang and J. Pu, "MDig: Multi-digit Recognition using Convolutional Nerual Network on Mobile", in Proc. Yang2015 MDigMR, 2015, pp. 110.

[15] "K.A. Patel and V. Gupta, "Review on Handwritten Digits Recognizing System", Intl. J. of Advanced Research in Computer Science and Management Studies, vol. 3, no. 4, pp. 94-101, 2015.