American International Journal of Computer Science and Information Technology
Volume.8, Number 4;October-December, 2023;
ISSN: 2837-388X| Impact Factor:
https://zapjournals.com/Journals/index.php/aijcsit
Published By: Zendo Academic Publishing

HANDWRITTEN DIGIT RECOGNITION USING MULTILAYER PERCEPTRON AND BACKPROPAGATION

Dr. Emily Katherine Anderson[1] and Dr. Nadim Georges Khoury

Article Info

Keywords: Handwriting Recognition, Neural Networks Java Application, Back Propagate Neural Network, Supervised Learning

Abstract

Handwriting recognition, a longstanding challenge in computer applications, has witnessed significant advancements with the advent of neural networks. These versatile networks find applications in various domains, including handwriting and voice recognition, as well as complex decision-making through machine learning. This project presents a Java application that employs a Neural Network to process image data, converting it into a 24 by 24 matrix on a pixel-by-pixel basis. The data is represented with values ranging from 0 to 255. The dataset is further organized, segregating it digit-wise and storing it in distinct sheets within a single Excel file located in local storage. Subsequently, the dataset is split into two distinct subsets: the training dataset and the test dataset. To optimize performance, 180 training data points are used for each digit, resulting in a total of 1800 data rows for training. The pixel values are sequentially fed into the implemented Back Propagate Neural Network Java application, developed using Java 1.8, and trained through supervised learning. During training, error is computed for the given expected output, facilitating the adjustment of the two weight matrices - one with dimensions 784 by 200, and the other with dimensions 200 by 10.

1. Introduction

Handwriting recognition has been one of the challenging computer applications for decades until Neural networks come to the table. Neural network has lot of application like handwriting recognition, voice recognition, and complex decision-making using machine learning. In this Project we develop java application implementing Neural Network to feed image data converted to 24 by 24 Matrix pixel-wise. All these data have values starting from 0 to 255.we divided this data digit wise and place in separate excel sheets in same excel file located in local storage. Then we divide this data set to two separate datasets as training dataset and Test dataset. Because of the performance issue we use only 180 trained data for each digit and total of 1800 data rows for training. Then we feed all this pixel's row by row to implemented Back Propagate Neural Network

[1] Department of Computer Science, California State University, 25800 Carlos Bee Blvd, Hayward, CA 94542, USA

java application developed in java 1.8 and train Using Supervise learning. In Training what we do is for given Expected output we calculate error and adjust the two weight matrices which is 784 by 200 and 200 by 10.

2. Literature Review

2.1 Backpropagation Learning:

Leung and Haykin introduced a complex backpropagation algorithm for application in signal processing and communication problems. The backpropagation algorithm provides a popular method for the design of a multilayer neural network to include complex coefficients and complex signals so it can properly be applied to solve complex problems (Leung &Haykin, 1991).

Benvenuto and Piazza introduced a recursive algorithm for updating the coefficients of the neural network structure for complex signals in which multiple complex activation functions had been tested (Benvenuto & Piazza, 1992). Leonard and Kramer proposed the application of neural networks such as backpropagation to chemical engineering problems such as malfunction diagnosis (Leonard & Kramer, 1990). Riedmiller and Braun introduced a learning algorithm for multilayer feed forward networks to solve the issue of the inherent disadvantages of pure gradient-descent approach in backpropagation by introducing a local adaptation of the weight-updates according to the behavior of the error function (Riedmiller & Braun, 1993). Yu, Chen and Cheng have proposed a dynamic approach on the learning rate of the backpropagation algorithm to increase the efficiency (Yu, Chen, & Cheng, 1995).

2.2 Handwritten character recognizing:

The handwritten character recognition is one of the most challenging task in pattern recognition. Kato et al. introduced a precise system for handwritten Chinese and Japanese character recognition by extracting directional element feature from each character image, transformation based on partial inclination detection is used to reduce undesired effects of degraded images (Kato et al., 1999). Chen et al. proposed a framework for handwritten character recognition with appropriate sample generation, training scheme and convolutionary neural network for the properties of handwritten characters (Chen et al., 2015). Ayyaz, Javed, and Mahmood adopted multiclass Support Vector Machine (SVM) classification with hybrid feature extraction for handwritten character recognition. Their algorithm identifies the type and location of some elementary strokes in the character. The strokes to be looked for comprise horizontal, vertical, positive slant and negative slant lines–as we observe that the structure of any character can be approximated with the help of a combination of simple straightline strokes. The strokes are identified by correlating different segments of the character with the chosen elementary shapes (Ayyaz, Javed, & Mahmood, 2016).

3 Algorithm Formulation

3.1 Activation function

The sigmoid output range contains all numbers strictly between 0 and 1. Both extreme values can only be reached asymptotically. The computing units considered in this chapter evaluate the sigmoid using the net amount of excitation as its argument. Given weights w1. . . wn and a bias −θ, a sigmoidal unit computes for the input x1, . . ., xn the output.

$$\frac{1}{1+\exp\left(\sum_{i=1}^{n} w_i x_i - \theta\right)}.$$

3.2 Neural Network and Backpropagation

We will consider a network with n input sites, k hidden, and m output units. The weight between input site i and hidden unit j will be called $w^{(1)}_{i,j}$. The weight between hidden unit i and output unit j will be called $w^{(2)}_{i,j}$. The bias −θ of each unit is implemented as the weight of an additional edge. Input vectors are thus extended with a 1 component, and the same is done with the output vector from the hidden layer. Figure below shows how this is done. The weight between the constant 1 and the hidden unit j is called $w^{(1)}_{n+1,j}$ and the weight between the constant 1 and the output unit j is denoted by $w^{(2)}_{k+1,j}$

There are (n + 1) × k weights between input sites and hidden units and (k + 1) × m between hidden and output units. Let W1 denote the (n + 1) × k matrix with component $w^{(1)}_{i,j}$ at the i-th row and the j-th column. Similarly let W2 denote the (k + 1) × m matrix with components $w^{(2)}_{i,j}$. We use an over lined notation to emphasize that the last row of both matrices corresponds to the biases of the computing units.

The matrix of weights without this last row will be needed in the backpropagation step. The n-dimensional input vector o = (o1, . . . , on) is extended, transforming it to oˆ = (o1, . . ., on, 1). The excitation net of the j^{th}hidden unit is given by

$$net_j = \sum_{i=1}^{n+1} w_{ij}^{(1)} \hat{o}_i.$$

The activation function is a sigmoid and the output o (1) j of this unit is thus

$$o_j^{(1)} = s\left(\sum_{i=1}^{n+1} w_{ij}^{(1)} \hat{o}_i\right).$$

The excitation of all units in the hidden layer can be computed with the vector-matrix multiplication oˆW1. The vector o (1) whose components are the outputs of the hidden units is given by

$$\mathbf{o}^{(1)} = s(\hat{\mathbf{o}}\overline{\mathbf{W}}_1),$$

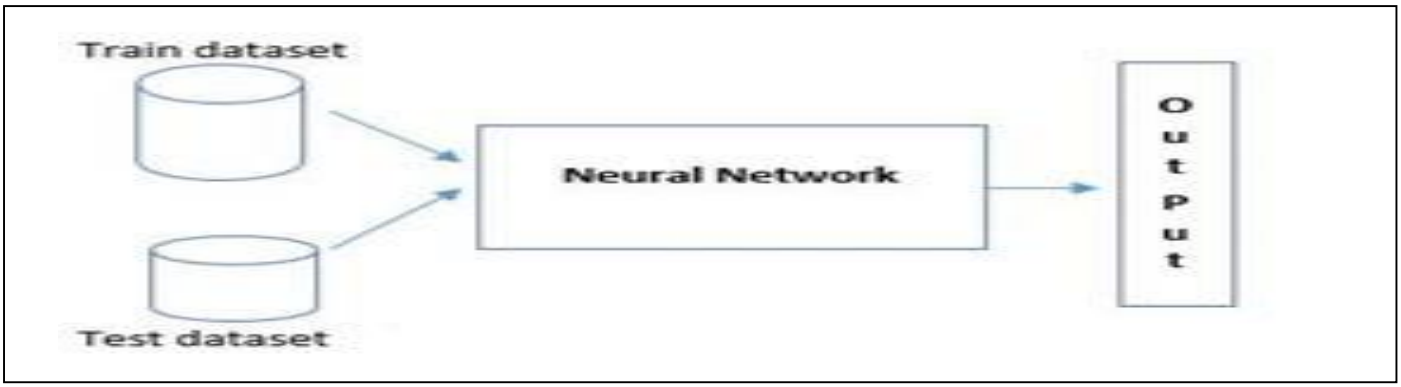
Using the convention of applying the sigmoid to each component of the argument vector. The excitation of the units in the output layer is computed using the extended vector oˆ (1)
= ($o^{(1)}1$, . . . , $o^{(1)}k$, 1). The output of the network is the m-dimensional vector $o^{(2)}$, where

$$\mathbf{o}^{(2)} = s(\hat{\mathbf{o}}^{(1)}\overline{\mathbf{W}}_2).$$

These formulas can be generalized for any number of layers and allow direct computation of the flow of information in the network with simple matrix operations

3.3 Training set and data set

The detail Map of the Handwrite Digit Recognition Neural Network Implemented Below.

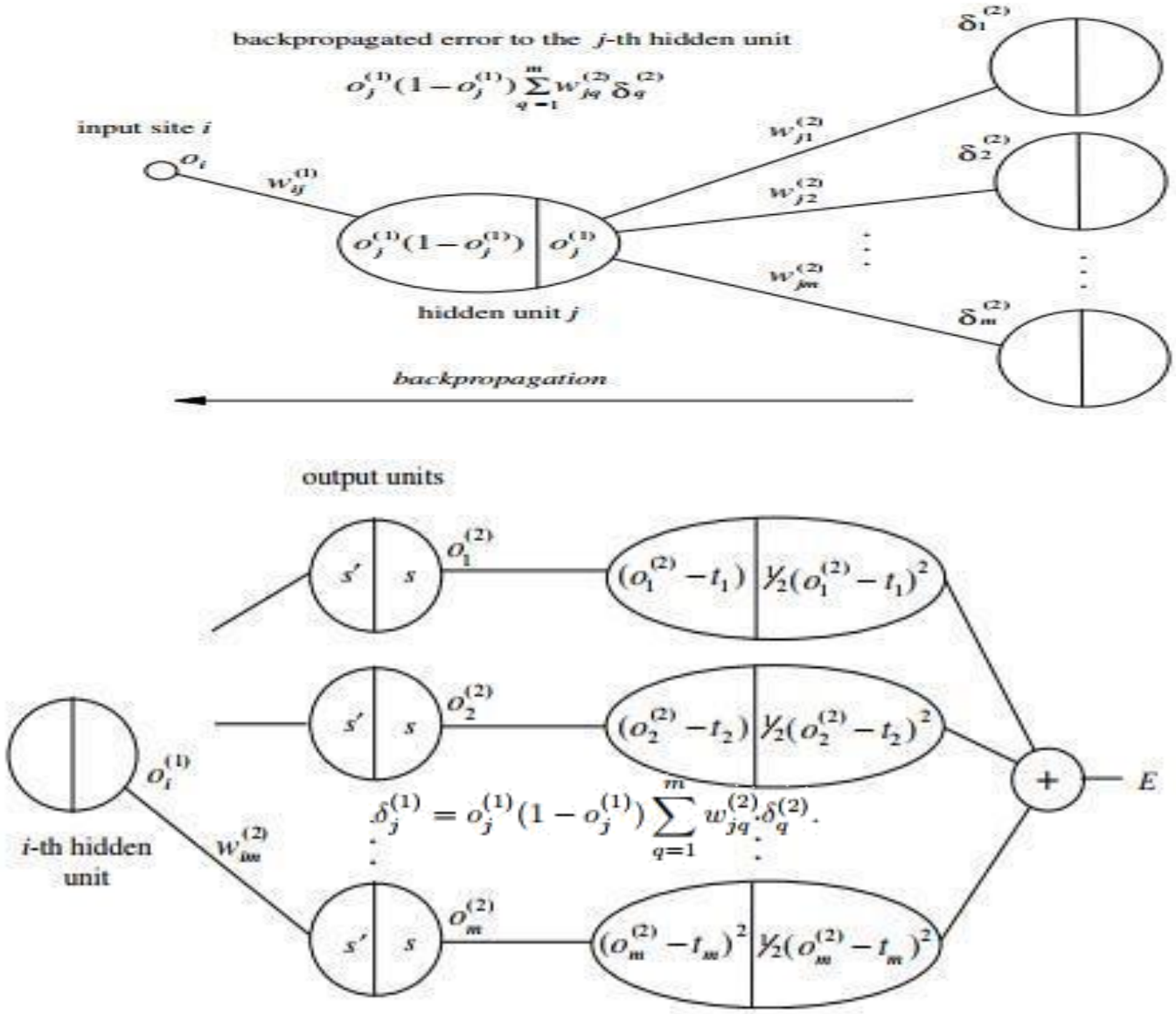


4. Algorithm Steps

After choosing the weights of the network randomly, the backpropagation algorithm described in the previous section is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

i) Feed-forward computation, ii) Backpropagation to the output layer, iii) Backpropagation to the hidden layer, iv) Weight updates.

The algorithm is stopped when the value of the error function has become sufficiently small



First step: Feed-forward

Computation the vector o is presented to the network. The vectors o (1) and o (2) are computed and stored. The evaluated derivatives of the activation functions are also stored at each unit.

Second step: Backpropagation to the output layer

We are looking for the first set of partial derivatives $\partial E/\partial w^{(2)}ij$. The backpropagation path from the output of the network up to the output unit j is shown in the diagram

Third step: **Backpropagation to the hidden layer**.

Now we want to compute the partial derivatives $\partial E/\partial w^{(1)}ij$. Each unit j in the hidden layer is connected to each unit q in the output layer with an edge of weight $w^{(2)}j,q$, for q = 1,…, m. The back propagated error up to unit j in the hidden layer must be computed considering all possible backward paths

The back propagated error can be computed in the same way for any number of hidden layers and the expression for the partial derivatives of E keeps the same analytic form\

Fourth step: weight updates

After computing all partial derivatives, the network weights are updated in the negative gradient direction. A learning constant□□ defines the step length of the correction. The corrections for the weights are given by

$$\Delta w_{ij}^{(2)} = -\gamma o_i^{(1)} \delta_j^{(2)}, \quad \text{for } i = 1, \ldots, k+1; j = 1, \ldots, m,$$

$$\Delta w_{ij}^{(1)} = -\gamma o_i \delta_j^{(1)}, \quad \text{for } i = 1, \ldots, n+1; j = 1, \ldots, k,$$

5. List of software and dataset used

1. Java 1.8: Java version 1.8 used as the Programming language.
2. Excel 2017 used as data storage.
3. Image Pixel Matrix Download from MNIST Dataset (LeCun, Cortes, & Burges, n.d.)
4. org.apache.poi Excel Library to read Excel data

We used Eclipse IDE as Programming Platform and Windows 10 as the Operating system. Since java is Platform independent programming language runs on Jvm this application has cross platform execution capabilities. We didn't test this application in any Other OS except windows)

6. Results

The results of applying the testing data set is demonstrated in Tables 1 and 2. Table 1 the shows if the result of each digits from 0 to 9 given to the network and it shows what output network has been settled after running through the algorithm. In Table 2 overall performance of the network on all inputs (digits 0-9) are shown, furthermore it shows that the network will return 91% of the handwritten digits properly.

7. Conclusion and Future Work

In this study, a version of backpropagation learning algorithm along with feedforward multilayer perceptron was used to recognize the digits 0 to 9. The performance of the network had been tested using the MNIST handwritten database. The performance of the network is discussed and overall performance over digits 0 to 9 shows a 91% accuracy rate. We are planning to extend this work to the Latin-based and non-Latin-based handwriting character recognition datasets.

Table 1. Testing Results for the digits 0-9 using the proposed network

Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0 Test for No: 0 Answer : 0	Test for No: 1 Answer : 0 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1 Test for No: 1 Answer : 1	Test for No: 2 Answer : 1 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2 Test for No: 2 Answer : 2	Test for No: 3 Answer : 2 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3 Test for No: 3 Answer : 3	Test for No: 4 Answer : 3 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4 Test for No: 4 Answer : 4

Test for No: 5 Answer : 4 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5 Test for No: 5 Answer : 5	Test for No: 6 Answer : 5 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6 Test for No: 6 Answer : 6	Test for No: 7 Answer : 6 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7 Test for No: 7 Answer : 7	Test for No: 8 Answer : 7 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8 Test for No: 8 Answer : 8	Test for No: 9 Answer : 8 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9 Test for No: 9 Answer : 9

Table 2. Accuracy of the proposed algorithm with respect to different digits

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9	Average
100%	90%	90%	90%	90%	90%	90%	90%	90%	90%	**91%**

References

Ayyaz, M. N., Javed, I., & Mahmood, W. (2016). Handwritten character recognition using multiclass svm classification with hybrid feature extraction. Pakistan Journal of Engineering and Applied Sciences.

Benvenuto, N., & Piazza, F. (1992). On the complex backpropagation algorithm. IEEE Transactions on Signal Processing, 40(4), 967-969.

Chen, L., Wang, S., Fan, W., Sun, J., & Naoi, S. (2015, November). Beyond human recognition: A CNN-based framework for handwritten character recognition. In 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR) (pp. 695-699). IEEE.

Kato, N., Suzuki, M., Omachi, S. I., Aso, H., &Nemoto, Y. (1999). A handwritten character recognition system using directional element feature and asymmetric Mahalanobis distance. IEEE transactions on pattern analysis and machine intelligence, 21(3), 258-262.

LeCun Y., Cortes C. & Burges C. (n.d.). MNIST handwritten digit database, Retrieved from: http://yann.lecun.com/exdb/mnist/

Leonard, J., & Kramer, M. A. (1990). Improvement of the backpropagation algorithm for training neural networks. Computers & Chemical Engineering, 14(3), 337-341.

Leung, H., &Haykin, S. (1991). The complex backpropagation algorithm. IEEE Transactions on signal processing, 39(9), 2101-2104.

Riedmiller, M., & Braun, H. (1993, March). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Proceedings of the IEEE international conference on neural networks (Vol. 1993, pp. 586-591).

Yu, X. H., Chen, G. A., & Cheng, S. X. (1995). Dynamic learning rate optimization of the backpropagation algorithm. IEEE Transactions on Neural Networks, 6(3), 669-677.