Hand Written Digit Recognition Using Elman Neural Network

J.V.S.Srinivas,
DRK IST, Hyderabad.
srinivasjvs@gmail.com

P. Premchand
OU,UCE, HYD

Abstract- Objective of this work is to recognize the hand written digits represented in black-and-white rectangular pixel displays using Elman neural network (ENN). ENN is one of the simplest supervised multi layer neural networks. The performance parameters like speed-up, and processing time are evaluated for sequential implementation with and without adaptive learning rate.

Key words- Handwritten Digit Data Set, ENN, Backpropagation(BP) algorithm, Adaptive learning rate.

I. INTRODUCTION

In the recent years handwriting number recognition is one of the challenging research problem. Many approaches such as minimum distance, decision tree and statistics have been developed to deal with handwriting number recognition problems. Alceu de Britto et al.[11] proposed an approach for recognizing the handwritten numeral strings that relies on the two-stage HMM-based method. The main objective for our system was to recognize isolated Arabic digits exist in different applications. The studies were conducted by using Semeion Handwritten Digit Data Set taken from UCI machine learning repository[8]. The dataset contains 1593 handwritten digits from around 80 persons that were scanned, stretched in a rectangular box 16x16 in a gray scale of 256 values. Then each pixel of each image was scaled into a Boolean (1/0) value using a fixed threshold. Each person wrote on a paper all the digits from 0 to 9, twice.

Elman neural network is feed forward network with an input layer, a hidden layer, an output layer and a special layer called context layer. The output of each hidden neuron is copied into a specific neuron in the context layer. The value of the context neuron is used as an extra input signal for all the neurons in the hidden layer one time step later. In an Elman network, the weights from the hidden layer to the context layer are set to one and are fixed because the values of the context neurons have to be copied exactly. Furthermore, the initial output weights of the context neurons are equal to half the output range of the other neurons in the network. The Elman network can be trained with gradient descent back propagation and optimization methods.

II. MATHEMATICAL MODEL FOR ENN

In this paper, we consider a fully connected ENN trained using BP algorithm. Let \( N_0 \) be the number of neurons in the input layer. Similarly \( N_j \), \( j=1,2,3 \) be the number of neurons in hidden, output and context layers respectively. The different phases in the learning algorithm and corresponding processing time are discussed in the following sections.

A. Sequential Implementation

The BP algorithm is a supervised learning algorithm, and is used to find suitable weights, such that for a given input pattern \( (X^0) \), the network output \( (Y^2) \) should match with the desired output \( (t) \). The details of algorithm and time taken to process are discussed below.

Elman Sequential Algorithm

Step 0: Initialize weights (set to small random values)
Step 1: While stopping condition is false do steps 2-9
   Is_First_Time = TRUE
   First iteration is a special case.
Step 2: For each training pair do steps 3-8
   Feed forward
   Step 3: Each input unit \( x_k = y_k^0 \) (k = 1,........\( N_0 \)) receives input signal \( x_k \) and broadcasts this signal to all units in the layer above
   (the hidden units)
   Step 4: Each hidden unit \( y_j^1 \) (j = 1,2,........\( N_1 \)) sums its weighted input signals and context signals. Then applies the activation function to compute its output signal.
   If (Is_First_Time) then
   \[ y_j^1 = f \left( \sum_{k=0}^{N_0} w_{jk}^1 y_k^0 \right) \]
   else
   \[ y_j^1 = f \left( \sum_{k=0}^{N_0} w_{jk}^1 y_k^0 + \sum_{h=1}^{N_1} w_{jh}^3 y_h^3 \right) \]
Step 5: Each output unit \( y_i^2 \) \((i = 1, 2, \ldots, N_2)\) sums its weighted input signals and applies the activation function to compute its output signal.

\[
y_i^2(t) = f \left( \sum_{j=0}^{N_1} w_{ij}^2 y_j^1 \right)
\]

where \( y_0^1 = \text{bias} \) and \( w_{0j}^2 \) is the bias.

**Backpropagation**

Step 6: Each output unit \( y_i^2 \) \((i = 1, 2, \ldots, N_2)\) receives a target pattern corresponding to the input training pattern and computes error \( e_i \) and error in formation term \( \delta_i^2 \)

\[
e_i = (y_i^2 - y_i^2(t=v)) i = 1, 2, \ldots, N_2 \quad \text{and} \quad \delta_i^2 = e_i f'(y_i^2(t=v)) i = 1, 2, \ldots, N_2
\]

It calculates its weight correction terms

\[
\Delta w_{ij}^2 = \eta \delta_i^2 y_j^1 \quad i = 1, 2, \ldots, N_1
\]

and sends \( \delta_i^2 \) \((i = 1, 2, \ldots, N_2)\) to the units in the layer below.

Step 7: Each hidden unit \( y_j^2 \) \((j = 1, 2, \ldots, N_1)\) sums its delta inputs from units in the layer above to find corresponding error information term

\[
\delta_j^2 = f' \left( \sum_{k=0}^{N_2} w_{jk}^2 \right) \quad j = 1, 2, \ldots, N_1
\]

It calculates its weight correction terms

\[
\Delta w_{jk}^2 = \eta \delta_j^2 y_k \quad j = 1, 2, \ldots, N_1
\]

where \( j = 1, 2, \ldots, N_1 \) and \( k = 1, 2, \ldots, N_0 \)

\[
\Delta w_{jk}^2 = \eta \delta_j^2 y_k \quad j = 1, 2, \ldots, N_1 \quad \text{and} \quad k = 1, 2, \ldots, N_0
\]

Step 8: Each output unit \( y_i^2 \) \((i = 1, 2, \ldots, N_2)\) updates its weight

\[
w_{ij}^2 = w_{ij}^2 + \eta \delta_i^2 y_j^0
\]

Similarly each hidden unit \( y_i^1 \) \((i = 1, 2, \ldots, N_1)\) updates its weight related to both input layer and context layer.

\[
w_{ij}^1 = w_{ij}^1 + \eta \delta_i^1 y_j^0 \quad k = 1, 2, \ldots, N_0
\]

\[
w_{ih}^3 = w_{ih}^3 + \eta \delta_i^3 y_h^0 \quad h = 1, 2, \ldots, N_1
\]

Step 9: Error matrix \( E = [E_1, E_2, \ldots, E_p] \) where \( E_i = [e_{i1}, e_{i2}, \ldots, e_{iN_2}] \)

Calculate \( E^2 = \frac{1}{2} \sum_{i=1}^{N_2} \sum_{j=1}^{p} e_{ij}^2 \).

If \( E^2 < \) Threshold value, store \( w_{ij}^2 \) and \( w_{il}^2 \) and stop

else go to step 2.

Let \( t_m \), \( t_a \), and \( t_{ac} \) be time taken for one floating point multiplication, addition, and calculation of activation value respectively. The time taken to complete the forward phase \( (T_1) \) is given by

\[
T_1 = N_1 m_0 (N_0 + N_1 + N_2) + t_{ac} (N_1 + N_2) + 2 t_a \quad \text{where} \quad m_0 = t_a + t_m
\]

The time taken to complete the error back propagation phase is represented by \( T_2 \) and is calculated as

\[
T_2 = N_2 (1 + N_1) m_0 + N_1 t_m + (N_1 + N_2) t_{ac}
\]

The time taken to update the weight matrix between the three layers is represented by \( T_3 \) and it is equal to

\[
T_3 = 2 N_1 (N_0 + N_1 + N_2) m_0 + N_1 (N_0 + N_1 + N_2) t_m + N_1 t_{ac} + N_2 m_0 + 2 t_a + 2 t_{ac} (N_1 + N_2) + N_2 m_0 (1 + N_1)
\]

The total processing time \( (T_{seq}) \) for training a single pattern in one iteration is the sum of the time taken to process the three phases and is given as

\[
T_{seq} = T_1 + T_2 + T_3 = (N_0 + N_1 + N_2) (2 N_1 m_0 + N_0 t_m) + N_1 N_2 m_0 + t_m + 2 t_a + (N_1 + N_2) + t_{ac} + (N_1 + 1) m_0
\]

In case of sequential algorithm implementation with adaptive learning rate, we dynamically modify the learning rate depending on the weight changes.

Step 9 of sequential algorithm is modified as follows.

Error matrix \( E = [E_1, E_2, \ldots, E_p] \) where \( E_i = [e_{i1}, e_{i2}, \ldots, e_{iN_2}] \)

Calculate \( E^2 = \frac{1}{2} \sum_{i=1}^{N_2} \sum_{j=1}^{p} e_{ij}^2 \).

If \( E^2_{\text{new}} < E^2_{\text{old}} \) realize weights and \( \eta = \eta * 1.2 \)

else discard weights and reset \( \eta \) to its original value.
III. EXPERIMENTAL STUDY & RESULTS

In this paper, we implemented the sequential algorithm with adaptive and non-adaptive learning rate. We observed that the number of iterations to train the network is less in case of algorithm with adaptive learning rate. Thus the training time is also less. As one hidden layer is adequate for a large number of applications, in the present project work the algorithm is developed for neural nets with one hidden layer and a context layer.

IV. CONCLUSIONS

In this paper, we implemented BP algorithm to train Elman network with single hidden layer and a context layer. The analytical performance of the algorithm is compared with the algorithm with adaptive learning rate as shown in fig-2 & 3.

V. ACKNOWLEDGEMENTS

We acknowledge UCI machine learning repository for giving the opportunity to use the Semeion Handwritten Digit Data Set to train and test our model.

### Table: Comparison of Serial implementations on Elman Neural Network

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Metric</th>
<th>Value(Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>Time</td>
<td>21</td>
</tr>
<tr>
<td>Sequential algorithm with adaptive learning rate</td>
<td>Time</td>
<td>15</td>
</tr>
</tbody>
</table>

REFERENCES

[6] Semeion Research Center of Sciences of Communication, via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3, 525030 Mairano (Brescia), Italy
2010.


