Exploring neural network optimisations using MNIST

Stefanos Stefanou(020363)

1 Abstract

Multilayer Neural Networks trained using back-propagation algorithm are the best example of a successful Gradient-Based Learning technique.In this report we will going to explore an application of an MN N , handwritten patterns recognition.We will tune our MN N application in order to be able to handle efficiently the dozens(or even hundreds) of neurons and hidden layers, using basic linear algebra rules

2 Introduction

2.1 Structure

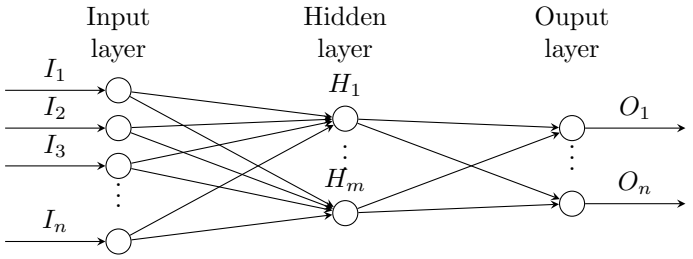
we have used an OOP approach to synthesize our solution, with 2 basic classes, NeuralNetwork and PerceptronMultilayer.NeuralNetwork class is a wrapper over multiple PerceptronMultilayer objects.it handles and supervises their learning by injecting them the proper data during learning

2.2 The Algorithm

In order to perform our experiments in a reasonable amount of time, a new approach was needed. so we we-wrote our Multilayer Neural Network engine , using matrix multiplications. this reduced the learning time significantly

2.2.1 Calculating a layer

let the following sigmoid-activated MNN



The calculation of a single layer can be done by multiplying the weights of the current layer with the transposed matrix inputs

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1m} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & w_{m3} & \dots & w_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i w_{1i} \\ \sum_{i=1}^{n} x_i w_{2i} \\ \vdots \\ \sum_{i=1}^{n} x_i w_{mi} \end{bmatrix}$$

the following python code implements this algorithm

```
def run(self, prev_layer_out):
  sum=np.dot(self.weights,prev_layer_out.T)
  result=sigmoid(sum)
  return result.T
```

2.2.2 Training the output layer

let the M targets, O be the outputs(after the activation function is applied) and I be the inputs of the output layer

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{bmatrix}, O = \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_m \end{bmatrix}, I = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ \vdots \\ i_n \end{bmatrix}$$

then, by defining $\odot$ to be Hadamard multiplication, the matrix of deltas in the output layer will be..

$$\Delta = (T-O)\odot(1-O)\odot O = \begin{bmatrix} (t_1-o_1)(1-o_1)o_1 \\ (t_2-o_2)(1-o_2)o_2 \\ \vdots \\ (t_m-o_m)(1-o_m)o_m \end{bmatrix} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_m \end{bmatrix}$$

let η be the MNN's learning rate, then the correction matrix will be

$$C = I(\Delta\eta)^T = \begin{bmatrix} \delta_1 i_1\eta & \delta_1 i_2\eta & \delta_1 i_3\eta & \dots & \delta_1 i_m\eta \\ \delta_2 i_1\eta & \delta_2 i_2\eta & \delta_2 i_3\eta & \dots & \delta_2 i_m\eta \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_n i_1\eta & \delta_n i_2\eta & \delta_n i_3\eta & \dots & \delta_n i_m\eta \end{bmatrix}$$

The python code implementing this algorithm is given below

```
def term_learn(self, i, o, t):
  t=t.reshape(1,self.output_len)
  o=t.reshape(1, self.output_len)
  Delta=np.array((t-o)*(1.0-o)*o)
  correction=i*Delta.T
  self.weights+=correction
```

2.2.3 Training the hidden layer

let the Δ_r, W_r be the deltas and the weight matrices of r'th layer respectively

$$\Delta_r = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_m \end{bmatrix} W_r = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1m} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & w_{m3} & \dots & w_{nm} \end{bmatrix}$$

Next, we define $E_r(i)$ to be the error of i'th neuron at r'th layer

$$E_r(i) = \sum_{j=1}^{n} \delta_{r+1}(j) * w_{r+1}(j,i)$$

It can be easily seen that...

$$W_{r+1}^T \Delta_{r+1} = \begin{bmatrix} \sum_{j=1}^{n} \delta_{r+1}(j) * w_{r+1}(j,1) \\ \sum_{j=1}^{n} \delta_{r+1}(j) * w_{r+1}(j,2) \\ \vdots \\ \sum_{j=1}^{n} \delta_{r+1}(j) * w_{r+1}(j,m) \end{bmatrix} = \begin{bmatrix} E_r(1) \\ E_r(2) \\ \vdots \\ E_r(m) \end{bmatrix}$$

so ,our delta matrix will become

$$\Delta_r = W_{r+1}^T \Delta_{r+1} \odot (1-O) \odot O = \begin{bmatrix} E_r(1)(1-o_1)o_1 \\ E_r(2)(1-o_2)o_2 \\ \vdots \\ E_r(m)(1-o_m)o_m \end{bmatrix} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_m \end{bmatrix}$$

The rest of the procedure is identical to section 2.2.2, the following code implements this algorithm

```
def mid_learn(self,i,next_layer,o):
 Wr_p_1=next_layer.weights
 Dr_p_1=next_layer.Delta
 self.Delta=Wr_p_1.T*Dr_p_1
 self.Delta=self.Delta*(1-o)*o
 self.Delta=i*self.Delta.T*self.learning_rate
 self.weights+=self.Delta
```

2.3 Technologies

In our application we have used a number of new technologies in order to cope with the natural complexity of the given task.We completely rewrite our Multilayer Neural Network engine using python3.6, in order to have available the following, as a result

2.3.1 numpy

numpy is a mathematics library for python, it offers fast matrix multiplication and dot product implementations, vital for our approach in MN N's

2.3.2 matplotlib

matplotlib is a ploting library for python, we used to visualise our data and results

2.3.3 pickle

pickle is a fast csv loader for python , it is vital to our applications performance

3 User Application

The MNIST Database(Modified National Institute of Standards and Technology)consists of 60.000 train and 10.000 test examples of handwritten digits.It is a standardised test in machine learning and is used as common ground for researches to compare their results. Our dataset is contained in 2 csv files, training and testing data.Every line of these files consists of an 28x28 image, given as exactly 784 numbers , between 0(white pixel) and 255(black pixel). An additional number at the beginning of each line is for the numbers label

3.1 Preprocessing

3.1.1 Normalised Labels

We need to perform "Binarization" to MNIST's labels, this can be done using one-hot representation. Each line of every csv file starts with a number $k[0 <= k <= 9]$ and will be converted as follows

1	2	...	9
(100000000)	(01000000)	...	(000000001)

3.1.2 Normalised Pixels

We will convert the default grayscale interval to a range [0,1], by dividing each pixel value with 255. This becomes huge problem though, because it allows for zero inputs, something that if occur can prevent the neuron from learning, so we need to eradicate zeros as inputs, by applying the following function in every pixel.

$$f(x) = \frac{x * 0.99}{255} + 0.01$$

3.2 Initialization

As it turns out, initialization is a very important factor in our application success, we cant choose random initialization weights. Weight matrices should be chosen randomly but not arbitrary.By choosing a random normal distribution we have broken possible symmetries, which may affect negatively the learning process.

let n be the number of inputs in our network, our initialization procedure draws random numbers from a truncated normal distribution as follows

$$X \sim \frac{1}{\sigma} \frac{\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})},$$

$$a = -\frac{1}{\sqrt{(n)}}, b = \frac{1}{\sqrt{(n)}}, \mu = 0, \sigma = 1$$

3.3 Fragmentation of data

MNIST Database has already provide us with a basic fragmentation of our data. 60.000 train and 10.000 test elements, but extensive care has been given in order minimise the uncertainty factor and not over-train our MNN[1].

As reference [1] finds, "the fraction of patterns reserved for the validation set should be inversely proportional to the square root of the number of free adjustable parameters"

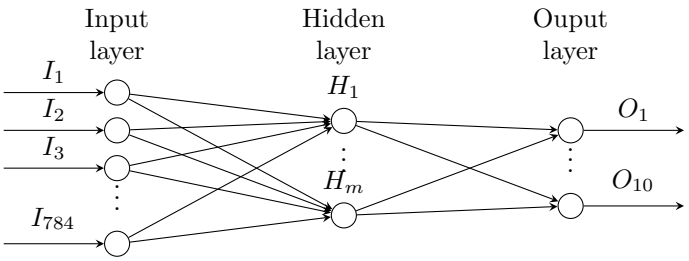
train	60000
test	4000
validation	6000

3.4 Structure of MNN

In order to ensure the optimal output, multiple experiments will be performed, using various hidden layer configurations, as a common ground though, the following rules will be followed in every experiment.

- every pixel will become a free adjustable parameter, in the input layer

- as handwritten digits is a purely categorical variable, the output layer will always consists of 10 Neurons, one for each digit category



4 Results

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum." "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

5 Discussion

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum." "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

6 Conclusion

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum." "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

7 References

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum." "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."