2021/6

IMPLEMENTATION OF A NEURAL NETWORK FROM SCRATCH FOR HANDWRITTEN DIGIT RECOGNITION TASK

BY MAHVASH SIAVASHPOUR
PRINCIPLES OF COMPUTATIONAL INTELLIGENCE
AMIRKABIR UNIVERSITY OF TECHNOLOGY

INTRODUCTION

One of the most famous machine learning and vision tasks is handwritten digit recognition. One can use multiple libraries to do this job, but we want to implement a neural network from scratch and train it for this specific task. Neural networks are powerful tools that can be trained and used to label or predict some unknown data. The power of neural networks was not discovered until recently in a contest named ImageNet. Since this discovery, we can observe a huge growth in modern techniques for adapting these neural networks in different areas.

ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs), usually called neural networks (NNs), are computing systems inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain.

Nodes and Weights

In the simples format, a neural network is a set of layers where each layer contains some weights and nodes. Each node of layer i is connected to each node of layer i+1. The connection between these nodes is established via a weight associated with each couple of nodes. Moreover, each node itself has a function called the activation function that creates the node's output.

Let a_i be the activations of nodes (neurons) in layer i and w_i be the weights connecting each of these neurons to a specific neuron in layer i+1. The input of the activation function of the neuron in layer i+1 is:

$$w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b_1$$

Where b_2 is the bias for that neuron.

The network learns by adjusting its weights and biases. The overall output of the network helps adjust these parameters by a method called backpropagation. What we are looking for is to minimize the error function using gradient descent. For the last layer of our network we can define an error function as bellow:

$$z_j^{(L)} = \cdots + w_{jk}^{L}\, a_k^{(L-1)} + \cdots$$

$$a_j^{(L)} = \sigma(z_j^{(L)})$$

$$Cost = \sum_{j=0}^{n_L - 1} (a_j^{(L)} - y_j)^2$$

The derivation of cost function relative to the parameters of the last layer is:

$$\frac{\partial\, Cost}{\partial\, w_{jk}^{(L)}} = \frac{\partial\, Cost}{\partial\, a_j^{(L)}} \times \frac{\partial\, a_j^{(L)}}{\partial\, z_j^{(L)}} \times \frac{\partial\, z_j^{(L)}}{\partial\, w_{jk}^{(L)}} = (2a_j^{(L)} - 2y_j) \times \sigma'(z_j^{(L)}) \times a_k^{(L-1)}$$

$$\frac{\partial\, Cost}{\partial\, b_j^{(L)}} = \frac{\partial\, Cost}{\partial\, a_j^{(L)}} \times \frac{\partial\, a_j^{(L)}}{\partial\, z_j^{(L)}} \times \frac{\partial\, z_j^{(L)}}{\partial\, b_j^{(L)}} = (2a_j^{(L)} - 2y_j) \times \sigma'(z_j^{(L)}) \times 1$$

$$\frac{\partial\, Cost}{\partial\, a_k^{(L-1)}} = \sum_{j=0}^{n_L - 1} \left(\frac{\partial\, Cost}{\partial\, a_j^{(L)}} \times \frac{\partial\, a_j^{(L)}}{\partial\, z_j^{(L)}} \times \frac{\partial\, z_j^{(L)}}{\partial\, a_k^{(L-1)}}\right) = \sum_{j=0}^{n_L - 1} ((2a_j^{(L)} - 2y_j) \times \sigma'(z_j^{(L)}) \times w_{jk}^{(L)})$$

In each epoch of training the network we do these calculations and the final gradient is the average of all of the gradients in the set.

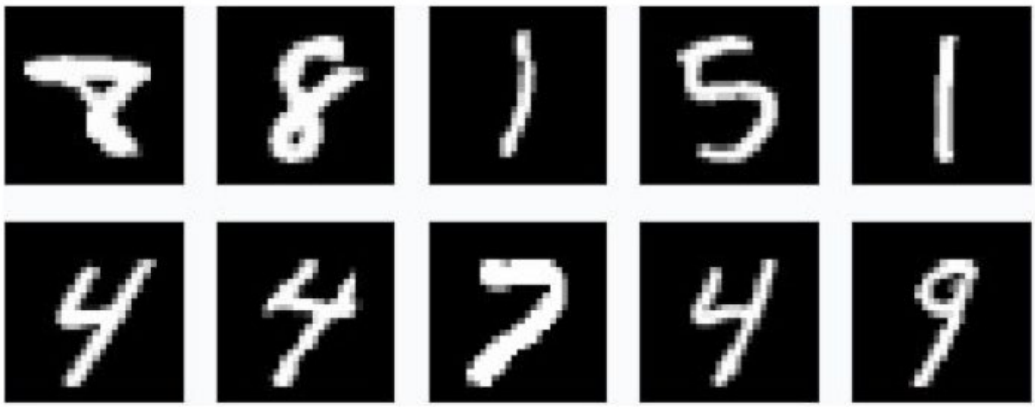
Implementation of the ANN

To implement this network, I used 28*28 = 784 neurons in the first layer, each representing a single pixel in the input image. There are also ten neurons in the last layer, one neuron for every digit. Finally, there are two hidden layers with sixteen neurons each.

I implemented the learning phase using the pseudo-code below:

```
Allocate W matrix and vector b for each layer.
Initialize W from standard normal distribution, and b = 0, for each layer.
Set learning_rate, number_of_epochs, and batch_size.
for i from 0 to number_of_epochs:
    Shuffle the train set.
    for each batch in train set:
        Allocate grad_W matrix and vector grad_b for each layer and initialize to 0.
        for each image in batch:
            Compute the output for this image.
            grad_W += dcost/dW for each layer (using backpropagation)
            grad_b += dcost/db for each layer (using backpropagation)
        W = W - (learning_rate × (grad_W / batch_size))
        b = b - (learning_rate × (grad_b / batch_size))
```

In order to train the network, we first need to get the appropriate dataset. I used the MNIST [1] dataset containing 60,000 handwritten digit images in the train section and 10,000 more in the test section.



As illustrated above, the digits are in a 28*28 pixel image with a black background. Each pixel has a value between 0 to 255 showing how bright it is. I divided these values by 255 to scale them between 0 to 1 and that is the input to the neuron responding to that pixel.

After acquiring the images, we can feed them into the ann using the feedforward method. After that we can calculate the cost function and adjust the weights using the backpropagation.

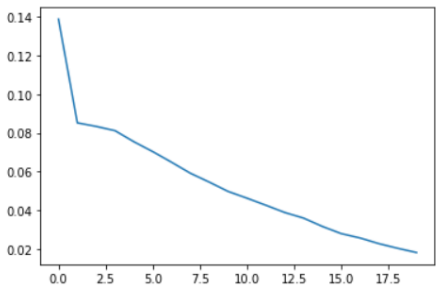
RESULTS AND CONCLUSION

For testing the network, the First thing is to create our wights and biases matrices randomly and to calculate the accuracy before doing ant training on the network. Next we train our network by feeding the first 100 images into the neural network from our train set and then training it by backpropagation.

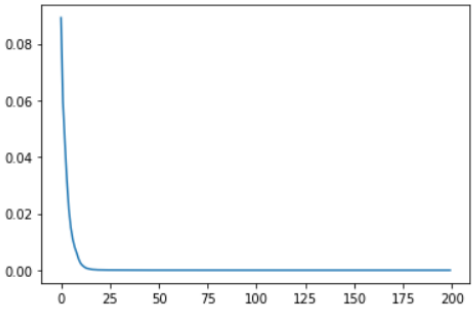
Here are the parameters I used to train the network, the accuracy and the results:

Learning Rate	Number of Epochs	Batch Size	Number of samples	Activation function	Accuracy before training	Wall time	Accuracy after training
1	20	10	100	Sigmoid	0.15	1min 11s	0.65

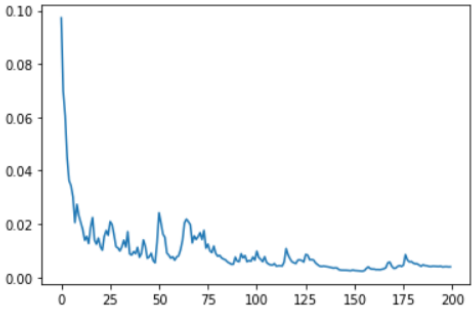
The following plot shows the cost function output in each epoch of training the ann:



I tested the results with 200 epochs and 500 samples and the final accuracy increased by more than 10 percent. I also did the same test with the hyperbolic tangent as the activation function. The ploting of each of these tests are illustrated below:



Sigmoid



Hyperbolic Tangent

REFERENCES

[1] http://yann.lecun.com/exdb/mnist/