TD 3 : Multi-class Classification and Neural Networks

Adam Goux--Gateau

31 October 2023

All the code is available here : `https://github.com/Gougaaate/Machine_Learning/tree/main/TDs/TD3`

1 Introduction

The purpose of this exercise is to perform a multi-class classification using a neural network. There will be only 3 layers in our example, including an input and an output. The goal is to detect hand-written numbers.

2 Propagation and Prediction

We are using a pretrained network (we already have the values of Θ). We perform the propagation through our layers with $\Theta^{(1)}$ and $\Theta^{(2)}$. We then use the sigmoid function and `argmax` function to get the class.

```
Training Set Accuracy: %f 97.52
Expected training Set Accuracy: 97.5%
```

Figure 1: Accuracy of the algorithm

Pretty good accuracy, we also have acces to examples of figures with predictions, where we can see that the number is effectively recognized.

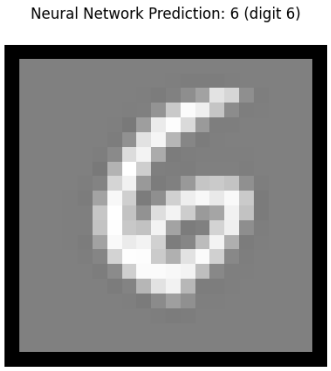

Figure 2: Number 6 recognized

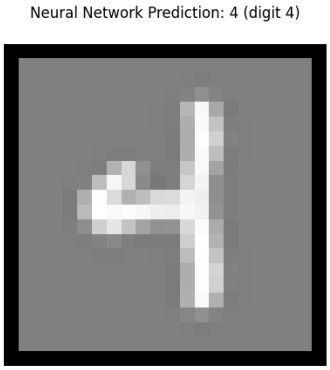

Figure 3: Number 4 recognized

3 Learning algorithm

In this section, we will attempt to train our models to access various parameters Θ. To do this, we will use a backpropagation algorithm (we used a forward prediction algorithm in the previous model). The first step is to implement the cost function. We will use the following function (without regularization parameters for now).

$$J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} \sum_{k=0}^{K-1} \Big[ ] - y_k^{(i)} \log \Big(h_\theta(x^{(i)})_k \Big) - (1 - y_k^{(i)}) \log \Big(1 - h_\theta(x^{(i)})_k \Big) \Big]$$

The parameters are divided into two matrices, $\Theta^{(1)}$ and $\Theta^{(2)}$, corresponding to the weights between the input layer and the hidden layer, as well as between the hidden layer and the output layer of the network. The evaluation of the function with the desired parameter is 0.287629, which is consistent with reality. The matrix y is constructed as follows: When a digit is detected, a 1 is placed in the row corresponding to that digit. Subsequently, we add the regularization parameter to our cost function. It is used to adjust overfitting or underfitting in our model. Thus, we try not to under-learn or over-learn from our data. The new function is therefore :

$$J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} \sum_{k=0}^{K-1} \Big[ ] - y_k^{(i)} \log \Big(h_\theta(x^{(i)})_k \Big) - (1 - y_k^{(i)}) \log \Big(1 - h_\theta(x^{(i)})_k \Big) \Big] +$$
$$\frac{\lambda}{2m} \left(\sum_{j=1}^{25} \sum_{k=1}^{400} \Theta_{j,k}^{(1)} + \sum_{j=1}^{10} \sum_{k=1}^{25} \Theta_{j,k}^{(2)} \right)$$

Finally, the cost with the chosen parameter is 0.383770, which is perfectly consistent once again. In the next section, we will implement the backpropagation algorithm. First, we need to create code that implements useful tools for our method, such as the derivative of the sigmoid function and the random initialization of parameters. Finally, the relative difference, which shows that our backpropagation algorithm is 2.28448e-11, which is again consistent with reality.

The visualization of the hidden layer is given by this figure :

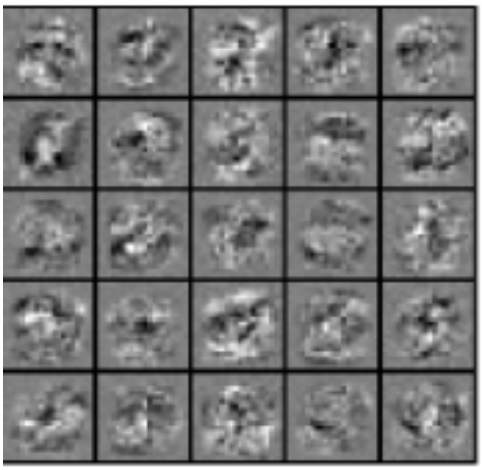

Figure 4: Hidden layer

The best parameters are $\lambda = 0.01$, and a maximum iteration of 100. That gives me a precision of 99.4%.

4 More questions

4.1 Question 1

The approach based on neural networks is a technique inspired by the functioning of the human brain. In a neural network, the elements known as neurons are organized into layers. Each neuron in one layer is connected to all neurons in the next layer. These connections are associated with weights that are adjusted during the learning process. There are two possible ways of operation: forward propagation and backpropagation. In the first method, data is passed through

the network from the input layer to the output layer. In the second method, errors are calculated at the network's output, and then adjustments are made in reverse to minimize the error. Here, we have chosen the backpropagation modeling because it provides better accuracy (99.32% as opposed to 97.5%). The parameters to be learned include the weights (model parameters learned during training) and the biases (in this case, we are using the first column of θ).

4.2 Question 2

The cost function is the one provided higher in the doc :

$$J(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} \sum_{k=0}^{K-1} \Big[ ] - y_k^{(i)} \log \Big(h_\theta(x^{(i)})_k \Big) - (1 - y_k^{(i)}) \log \Big(1 - h_\theta(x^{(i)})_k \Big) \Big] +$$
$$\frac{\lambda}{2m} \left(\sum_{j=1}^{25} \sum_{k=1}^{400} \Theta_{j,k}^{(1)} + \sum_{j=1}^{10} \sum_{k=1}^{25} \Theta_{j,k}^{(2)} \right)$$

We also sum on the number of classes to be recognized (10 in our example)