Maastricht University

Advanced concept in machine learning
Assignment 1

Authors
Luca Brugaletta
Glauco Lorenzut

Supervisor
Kurt Driessens

November 3, 2020

1 Introduction

In this assigment the task was to implement a neural network with 1 input layer, 1 hidden layer and 1 output layer. The hidden layer has 3 nodes while the other 2 layers have 8 nodes.

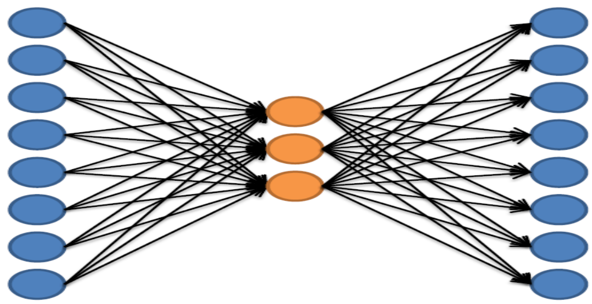


Figure 1: Neural Network Structure

2 Implementation

In this section are presented the choices made for our implementation.

2.1 Weights & Biases

Weights are initialized with small random numbers as we did with biases.

There are 2 ways to consider biases. The first option is to consider them as an additional node with a value of 1 for both, input and hidden layers. The other option is to add the bias vector to the dot product of layers and weights. In our case, we chose the second possibility. Thus, we created 2 different bias vectors initialised to a small random number close to 0. Then we updated them using the gradient descent in the same way as we did with the weights.

2.2 Forward propagation

To make the forward propagation we took our ideas from the ACML course lecture, especially "Backprop_Help.pdf", and Andrew NG course on Coursera[2].

2.3 Cost function & back propagation

The cost function and the backpropagation are part of the course Neural networks and deep learning.[1]
This is how our cost function is made:

$$J = -Y log A^{[2]} + (1 - Y) log(1 - A^{[2]}) \tag{1}$$

3 Results

The first experiment regarded the variation of the learning rates. We used this set of learning rates $\{0.01, 0.05, 0.1, 0.5, 1, 2, 3, 5, 10\}$ and the same values of random weights and biases were used to calculate the results for every learning rate. It results that too small learning rates struggled to obtain the same good results of the bigger ones and the best performance was given by learning rates around 1-3. Another important consideration is that small learning rates slightly decrease while big ones have a really variable trend with a lot of noise.

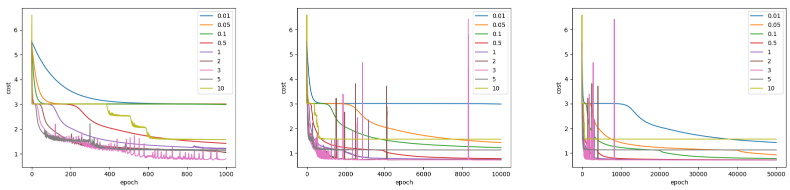
After different tries we observed that in our structure learning rates are influenced by epoch's number. In fact, with a smaller number of epochs (e.g 1000) lower learning rates cannot achieve the same immediate results of the higher rates. So taking into account bigger number epochs such as 10000 or 50000 the local minima are reached before by higher learning rates. They also present a quite steady or really noisy trend, depending from random weights and biases, although, lower learning rates still continue to decrease. The noisy trend of some learning rate has to be considered as the action of skipping possible local minimum present in its immediately descent.

As already said in the previous section, our weights and biases are initialised with random values. However, we decided to see how much the random parameters were influencing the behaviour of our neural network. So, a seed was introduced to see if it is possible to say that our structure gets affected by the randomness of the values. It follows that, as we expected, the initial random values play a big role. This could have also affected the neural network convergence.

The training time was about 0.15, 1 and 5 seconds for 1000, 10000 and 50000 epochs respectively. Tests performed with an i7-4710HQ CPU.

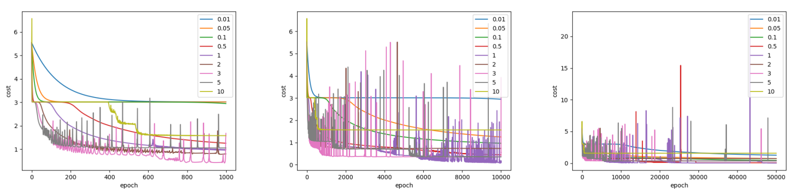
4 Conclusions

Even considering the randomness of the initial values, we can conclude that the implemented neural network reflects what we wanted to achieve. Taking into account the Figures 2 and 3 it is possible to see a common pattern where all the learning rates tend to push down the cost. During the prediction performed with the input value of [0,0,0,1,0,0,0,0] we obtained an output of [0,0,0, 0.997,0.001,0,0,0.001] using a learning rate of 3. The results are rounded using 3 decimals.



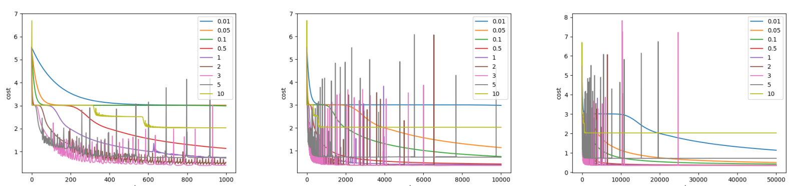
(a) 1000 epochs (b) 10000 epochs (c) 50000 epochs

Figure 2: Numpy seed 1



(a) 1000 epochs (b) 10000 epochs (c) 50000 epochs

Figure 3: Numpy seed 2



(a) 1000 epochs (b) 10000 epochs (c) 50000 epochs

Figure 4: Numpy seed 3

References

[1] deeplearning.ai. Backpropagation intuition (optional). `https://www.coursera.org/learn/neural-networks-deep-learning/lecture/6dDj7/backpropagation-intuition-optional`.

[2] Andrew NG. Machine learning). `https://www.coursera.org/learn/machine-learning/home/welcome`.