ACML Assigment

Neural Network Implementation

Lavinia Pulcinella (i6233926) Enrique Barrueco (i6242336)

November 2020

Contents

1 Neural Network Implementation

The present report relies on the MainNN_default.ipynb python code which implements a 3 layer Neural Network. The implementation was performed with the solely auxiliary help of the numpy library.

2 Hyper-parameters

For the given setup, all activation functions used for forward propagation and weight updates which are in turn involved in the back-propagation algorithm have been easily derived by hand.
The main issue involved tuning the learning rate (α) and the weight decay (λ) parameters. For an easy implementation default values for both α and λ were set to 0.8 and 0.0001 respectively.

2.1 Learning Rate α

A first main investigation on the learning rate parameter was performed (with weight decay λ fixed to 0.0001) in terms of convergence of the total error in the output layer and its effect on the convergence of the Neural Network for a specific value of α.

In order to compare how the network learns to replicate the identity matrix pattern with different leaning rates, we plotted the errors for each iteration (i.e. $\delta^{(3)}$) using a range of α values. Just as the theory would suggest, we found that the network learns faster using higher values of α. In contrast, as the value decreases convergence takes longer to occur.

For the smallest values of α, the error ended up not converging at all to the minimum threshold of 1 (with an upper bound of 10000 iterations). On the contrary, if we used this network on a real world scenery we might observe that the highest values of α leads to a more unstable model that outputs sub-optimal solutions.

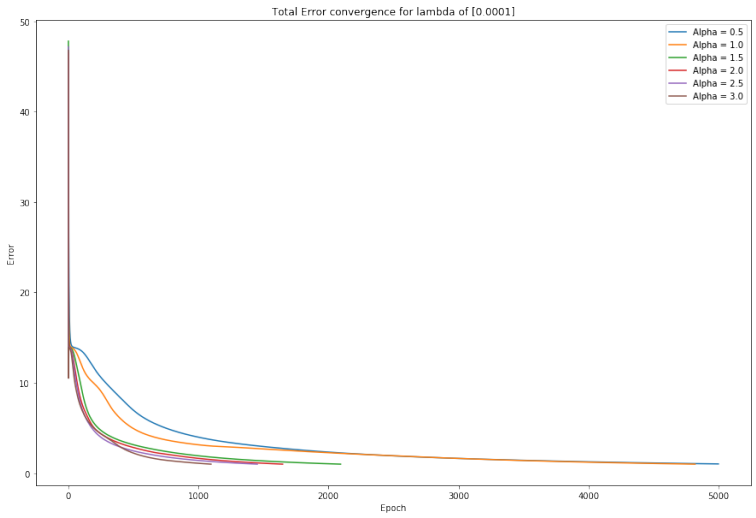

Figure 1: Learning to convergence with different alphas

2.2 Weight Decay λ

Smaller values of α can sometimes lead to better results. However, those values result in a higher number of iterations until convergence. Thus, using a regularization term like λ helps reduce the amount of time needed for convergence.

The weight decay parameter (also referred to as L2-regularization) has the primary objective to avoid model over-fitting (i.e. high variance). λ plays a role in the back-propagation algorithm by affecting the weight estimates by penalizing large weights over smaller ones. That is, since back-propagation is focused on reducing the error component on each node in the output layer (whose sum is the total error of the model) by reducing the values of the weights also the activation on each node will be reduced which in turn leads to the decrease of the activation and thus of the error.

2.3 Parameter tuning

When the learning rate is large enough, the "steps" toward optimization are larger. In this case, the regularization parameter λ might not have a strong impact on the estimation procedure. When the learning rate is smaller, however, since λ aims at *shrinking* the weights, it can help to achieve a quicker convergence.

Grid Search To decide which value of both hyper-parameters to use in order to train the network one could try to train and evaluate every possible combination of λ and α and select the one with the lowest total error. However, this approach adjusts the network in an isolated way without taking into account past evaluations of parameter combinations. Some of which can be useful and some that can rapidly be known to be unfruitful making their evaluation useless.

Bayesian Optimization A more sophisticated approach would use Bayesian optimization to focus on evaluating areas of the parameter distribution with the highest chance of bringing a reduction of the total error in contrast with previous iterations. That way the number of evaluations necessary is reduced, and better generalisation performance is achieved on the test set in comparison with grid search. However, because the size of the search space over which these parameters are evaluated in this shallow neural network is not so large, we did not implement it.

Thus, we opted for the less optimal solution of using grid search for finding the combination of α and λ that had the lowest total error. The results are showed in the first entry of Table 1.

Error	Epochs	Alpha	Lambda
0.994992	3093	3.4	0.0001
0.999377	884	3.4	0.0001
0.999657	1860	1.6	0.0001
0.999918	1445	2.8	0.0001
0.999941	1146	2.2	0.0001
0.999976	2562	1.0	0.0001
1.001708	10000	0.4	0.0001
1.198349	10000	1.0	0.0010
1.244136	10000	1.6	0.0010
1.316303	10000	2.2	0.0010
1.347205	10000	0.4	0.0010
1.857539	10000	2.8	0.0010
4.104367	10000	0.4	0.0050
5.754034	10000	1.0	0.0050
6.609012	10000	1.6	0.0050
7.551836	10000	2.2	0.0050
8.620945	10000	2.8	0.0050
9.016395	10000	3.4	0.0050

Table 1: Convergence for of different α and λ values

3 Weights Interpretation

The Neural Network at hand includes 8 input, 3 hidden and 8 output nodes for each corresponding layer. The input and hidden layer also include a bias node each allowing to *"keep the transferring alive"* even in the case of null weights for the general weights. Thus we have a total of 27 weights from input to hidden layer (9×3) and 32 weights from the hidden to the output layer (4×8).

The objective of the weights is to assign a level of "importance" to each specific neuron. However, the back propagation algorithm first employs forward propagation, then *goes backward* to compute the errors (the δs) in the nodes and then updates the weights according to the just computed errors. Thus:

- In the **forward pass** the weights assess the importance of the input value in the respective layer. The weights can (of course) be either positive or negative and their sign accounts for the positive (resp. negative) effect that the input has on the output.
- In **back-propagation** the weights are assessing the importance of the error of the activation values. Thus, prioritizing some over others. In better terms, the larger errors will be assigned a higher weight in order to have a larger impact (on the future weight updates) of the specific node.
- Weights are affected also by the **regularization (Weight update)** parameter, which is aimed at lowering them. Thus, when a weight is positive (and large) the regularization parameter λ has a higher effect.