University of Córdoba

Computer engineering
Computer Science Engineering Degree
4th course

Introduction to computational models

Lab Assignment 2: Multilayer perceptron for classification problems

Valentín Gabriel Avram Aenachioei
03524931C
p92avavv@uco.es

Academic year 202-2023
Córdoba, June 9, 2023

Contents

List of Tables

List of algorithms

In this paper, I will set out how I have done the second lab assignment of the subject, the Multilayer Perceptron for classification problems, explaining the Neural Network model used, its architecture and layer organization, how its algorithm works, and the pseudocode explaining the implementation of the learning process. As a conclusion, I will show, as experiments, ways to improve the error values given as a guideline, using different architectures for each dataset.

The main goal of this lab assignment is the subsequent analysis, emphasising this over the implementation of the code.

1 Description of the model

The neural network model implemented is a Multilayer Perceptron, explained in the theoretical classes and similiar to the one implemented in the first lab assignment. The multilayer perceptron is a simple C structure with 4 parameters, a learning rate, a momentum factor, the number of neurons, and a vector of layer structures. The multilayer perceptron structure contains the functions of the back-propagation algorithm.

Each layer strucuture has only two parameters, the number of neurons on each layer, and a vector of neuron strucutres. The main component of the architecture are the neuron structures, containing their own deltas, weights stored in a vector, and output. The neurons in the hidden and output layers will apply a sigmoidal function, and will have a bias.

The structures of neurons, layers, and the multilayer perceptron is already given.

There are multiple layers, an input layer, an output layer, and in between, there are $\boldsymbol{l}$ hidden layers given by the user. Each hidden layer has $\boldsymbol{h}$ nodes, given by the user, being the neurons of the input and output layer determined by the problem itself. The default values are 1 hidden layer with 5 neurons, using a learning rate of 1 and a momentum factor of 0.9.

The model can work in online mode, which means that in every iteration of the online back-propagation algorithm applied to the training set, the error will be computed and the weights of each neuron will be update according to the error. In the first iteration, the weights of the neurons will be established randomly.

Two datasets will be used, a training dataset and a testing dataset. The neural network will train using the training dataset, applying $\boldsymbol{i}$ iterations for the back-propagation algorithm, and it will test the results with the testing dataset. The algorithm can stop earlier if the network it is not improving after 50 iterations of the algorithm.

Also, it is posible to normalize the data from both datasets before using them.

There are three main differences between this and the last lab assignment. First of all, the neural network can work in offline mode, using the number of training patterns as batch size, although it can still working in the offline mode as in the first assignment.

Furthermore, the neural network can work with a softmax as output function for the last layers instead of using a sigmoidal function, and using the cross entropy as error function, instead of MSE, for both training and testing. These features can be combined.

2 Pseudocodes

The back-propagation algorithm does $\boldsymbol{n}$ iterations of ***performEpoch*** function, used for both online and offline versions of the training algorithm:

The pseudocode 1 shows the performEpoch function.

Algorithm 1 Backpropagation

```
1:  procedure ONLINE EPOCH
2:      if Online mode then
3:          for all hidden and output layers i do
4:              for all neurons j in layer i do
5:                  for all neurons k +1 in layer i − 1  do
6:                      Δ^i_jk ← 0
7:      for all inputs i do
8:          out^0_j ← i_j
9:      forwardPropagation()
10:     backpropagateError()
11:     accumulateChange()
12:     if Online mode then
13:         weightAdjustment()
```

We will online set the deltas to zero in the online mode, in the offline mode the deltas will be setted to zero in the training function.

2.1 Forward Propagation

The pseudocode 2 shows the forward propagation function, where the output of the neurons are calculated and propagated, from the first layer to the last one. There are two different propagations, depending on the ouput function used.

Algorithm 2 Forward propagation of the ouputs

1: **procedure** FORWARDPROPAGATION
2: **if** Sigmoidal in output layer **then**
3: **for all** layer i **do**
4: **for all** neurons j in layer $i-1$ **do**
5: $out_j^i \leftarrow \frac{1}{1+exp(-(w_{j0}^i+\sum_{k=1}^{n_{i-1}} w_{jk}^i out_k^{i-1}))}$
6: **if** Softmax in output function **then**
7: **for all** output neuron j **do**
8: $net_j^H \leftarrow w_{j0}^H + \sum_{i=1}^{n_{H-1}} w_{ji}^H out_i^{H-1}$
9: $out_j^H \leftarrow \frac{exp(net_j^H)}{\sum_{l=1}^{n_H} exp(net_j^H)}$

2.2 Backpropagation of the error

The pseudocode 3 shows the backpropagation of the output error function, where the output error of the neurons are backpropagated, from the last layer to the first one. This pseudocode can be optimized, but this is the representation of the code made.

Algorithm 3 Backpropagation of the output error

procedure BACKPROPAGATEERROR
  if Sigmoidal output function using MSE **then**
    for all output neuron i **do**
      $\delta_i^{layers-1} \leftarrow -(target_i - out_i^{layers-1})(out_i^{layers-1})(1 - out_i^{layers-1})$
    for i from layers - 1 to 1 **do**
      for all neurons j in layer i **do**
        $\Delta_j^i \leftarrow ((\sum_{k=1}^{n_{i+1}} w_{kj}^{i+1}\delta_k^{i+1})out_j^i(1 - out_j^i))$
  if Sigmoidal output function using CCR **then**
    for all output neuron i **do**
      $\delta_i^{layers-1} \leftarrow -(\frac{target_i}{out_i^{layers-1}})(out_i^{layers-1})(1 - out_i^{layers-1})$
    for i from layers - 1 to 1 **do**
      for all neurons j in layer i **do**
        $\Delta_j^i \leftarrow ((\sum_{k=1}^{n_{i+1}} w_{kj}^{i+1}\delta_k^{i+1})out_j^i(1 - out_j^i))$
  if SoftMax output function using MSE **then**
    for all output neuron i **do**
      $\delta_i^{H-1} \leftarrow -(target_i - out_i^{lH-1})(out_i^{H-1})(I(i == j) - out_i^{H-1})$
    for i from layers - 1 to 1 **do**
      for all neurons j in layer i **do**
        $\Delta_j^i \leftarrow ((\sum_{k=1}^{n_{i+1}} w_{kj}^{i+1}\delta_k^{i+1})out_j^i(1 - out_j^i))$
  if SoftMax output function using CCR **then**
    for all output neuron i **do**
      $\delta_i^{layers-1} \leftarrow -(\frac{target_i}{out_i^{layers-1}})(out_i^{layers-1})(I(i == j) - out_i^{layers-1})$
    for i from layers - 1 to 1 **do**
      for all neurons j in layer i **do**
        $\Delta_j^i \leftarrow ((\sum_{k=1}^{n_{i+1}} w_{kj}^{i+1}\delta_k^{i+1})out_j^i(1 - out_j^i))$

2.3 Adjust Weights

The pseudocode 4 shows the weight adjustment function, where the weights of the network are updated, from the first to the last layer.

Algorithm 4 Adjustment of weights

```
1: procedure WEIGHTADJUSTMENT
2:     for all layers i do
3:         for all neurons j in layer i do
4:             for all neurons k in layer i − 1 do
5:                 if Online mode then
6:                     w^i_jk ← w^i_jk − (ηΔw^i_jk + μ(ηΔw^i_jk(t − 1)))
7:                 else
8:                     w^i_jk ← w^i_jk − ((ηΔw^i_jk)/N + (μ(ηΔw^i_jk(t − 1)))/N)
9:                 w^i_j0 ← w^i_j0 − (ηΔw^i_j0 + μ(ηΔw^i_j0(t − 1)))
```

2.4 Test classification

The pseudocode 5 shows the test classification function, where the network is tested with a dataset and the CCR is obtained. The accumulate Change function is the same as it was in the previous assignment, it will not be explained.

Algorithm 5 Test Classification

```
1: procedure WEIGHTADJUSTMENT
2:     for all training patterns do
3:         feedInputs()
4:         forwardPropagate()
5:         getOutputs()
6:         if Class predicted is correct then
7:             correct ← +1
8:     return correct / NumberTrainingPatterns
```

With the *performEpoch* 1 function and the five functions it calls inside, the neuronal network does its learning process.

3 Experiments

In this section, different architectures will be tested, looking to improve the Correct Classification Rate (CCR) given as a guideline for each possible dataset.

In the following tables, μ represents the momentum factor and η represents the learing rate. The architecture will be represnted as $\{n : h : h : k\}$, where n represent the input layer, k represents the output layer and h represents the number of neurons on each hidden layer.

3.1 XOR Dataset

The XOR dataset represent the problem of non-linear classification of the XOR logical operation. It is the smallest dataset, and the same dataset will be used for training and for testing. In this dataset, we are trying to improve the given guideline :

$CCR_{train} = CCR_{test} = 50\%$

For the XOR dataset, we are going to use the best proven architecture from the last assignment, wich was $\{n : 100 : 100 : k\}$ using as parameters $\mu = 0.9$ and $\eta = 1$.

The table 1 shows the results of the experiments using the architecture and values for μ, η previously commented, using 1000 iterations. The improved results, using a different architecture to prove the correct working of the neural network is marked in **bold**, using the architecture $\{n : 32 : 32 : k\}$.

Learning mode	**Output**	**Error**	η	μ	**MSE**	**CCR**
Offline	Sigmoid	MSE	*1*	*0.9*	*0.00047*	*100*
Offline	Softmax	MSE	*1*	*0.9*	*0.15042*	*50*
Ofline	Softmax	Cross entropy	*1*	*0.9*	*0.20849*	*50*
Ofline	**Softmax**	**Cross entropy**	***1***	***0.9***	***0.08338***	***85***
Online	Sigmoid	MSE	*1*	*0.9*	*0.02509*	*95*
Online	Softmax	MSE	*1*	*0.9*	*0.12677*	*55*
Online	Softmax	Cross entropy	*1*	*0.9*	*0.17344*	*50*

Table 1: XOR dataset results

As it can be seen, the best results, both in offline and online mode, are

obtained using MSE as error function and sigmoid as output function.

This is because of the dataset used, the XOR dataset is not designed for classification problems, so the only plaussible results are the ones obtained using the MSE as error function.

3.2 idlp Dataset

Is a dataset containing 10 input medical variables related to liver and non-liver patients. It is the only dataset which inputs needs to be normalized. Two different datasets will be used, one for training and one for testing.

In this dataset, we are trying to improve the given guideline :

$CCR_{train} = 73.3\%$ and $CCR_{test} = 68.965\%$

For the ildp dataset, we need to find the best architecture. The table 2 shows the results of the given architectures, using the default values for μ and η .

Architecture	η	μ	MSE_{Train}	MSE_{Test}	CCR_{Train}	CCR_{Test}
$\{n : 4 : k\}$	1	0.9	0.18458	0.19007	72.0494	70.6897
$\{n : 8 : k\}$	1	0.9	0.1843	0.18976	71.9506	70.6897
$\{n : 16 : k\}$	1	0.9	0.18926	0.19621	71.9506	70.6897
$\{n : 64 : k\}$	1	0.9	0.18822	0.1937	71.9506	70.6897
$\{n : 4 : 4 : k\}$	1	0.9	0.19599	0.2002	71.8159	70.6897
$\{n : 8 : 8 : k\}$	1	0.9	0.19052	0.19523	71.8519	70.6897
$\{n : 16 : 16 : k\}$	1	0.9	0.17977	0.18812	71.6543	70.6897
$\{n : 64 : 64 : k\}$	1	0.9	0.17459	0.18738	71.4074	70.1149

Table 2: idlp dataset architectures

Once we have the best architecture, which is $\{n : 4 : 4 : k\}$ and using the best parameters for μ and η, in the table 3 we can see the result for the dataset in the different tests.

As it can be seen, the results both in offline and online mode, are quite similar. As this dataset is prepared for classification problems, the results are the expected ones. Using cross entropy, the results are the expected.

Mode	**Output**	**Error**	μ	η	MSE_{Test}	MSE_{Train}	CCR_{Train}	CCR_{Test}
Offline	Sigmoid	MSE	*1*	*0.9*	*0.17447*	*0.18496*	*71.308*	*70.114*
Offline	Softmax	MSE	*1*	*0.9*	*0.18696*	*0.19249*	*71.9506*	*70.6897*
Offline	Softmax	CCR	*1*	*0.9*	*0.18696*	*0.19249*	*71.9506*	*70.6897*
Online	Sigmoid	MSE	*1*	*0.9*	*0.17447*	*0.18496*	*71.3086*	*70.1149*
Online	Softmax	MSE	*1*	*0.9*	*0.18696*	*0.19249*	*71.9506*	*70.6897*
Online	Softmax	CCE	*1*	*0.9*	*0.26098*	*0.28003*	*71.6049*	*70.2299*

Table 3: idlp dataset results

3.3 noMNIST dataset

This dataset is composed of a set of letters written with different typologies or symbols. The dataset is already normalized. Two different datasets will be used, one for training and one for testing.

In this dataset, we are trying to improve the given guideline :

$CCR_{train} = 80.4\% and CCR_{test} = 82.6\%$

For the noMNIST dataset, we need to find the best architecture. The table 4 shows the results of the given architectures, using the default values for μ and η . {center

Architecture	η	μ	MSE_{Train}	MSE_{Test}	CCR_{Train}	CCR_{Test}
{n : 4 : k}	*1*	*0.9*	*0.07391*	*0.12237*	*87.53*	*80.2*
{n : 8 : k}	*1*	*0.9*	*0.0593*	*0.10999*	*89.911*	*81.5333*
{n : 16 : k}	*1*	*0.9*	*0.05776*	*0.1038*	*90.088*	*81.46*
{n : 64 : k}	*1*	*0.9*	*0.05384*	*0.10158*	*90.6*	*81.733*
{n : 4 : 4 : k}	*1*	*0.9*	*0.054*	*0.10374*	*91.4*	*81.13*
{n : 8 : 8 : k}	*1*	*0.9*	*0.054*	*0.10374*	*91.4*	*81.13*
{n : 16 : 16 : k}	*1*	*0.9*	*0.02932*	*0.09594*	*95.6889*	*84.4*
{n : 64 : 64 : k}	*1*	*0.9*	*0.17366*	*0.18738*	*71.654*	*70.229*

Table 4: noMNIST dataset architectures

Once we have the best architecture, which is *{n : 16 : 16 : k}* and using the best parameters for μ and η, in the table 5 we can see the result for the dataset in the different tests.

As it can be seen, the results both in offline and online mode, are quite different, but the results are the expected ones. Using the offline learning

Mode	**Output**	**Error**	μ	η	MSE_{Test}	MSE_{Train}	CCR_{Train}	CCR_{Test}
Offline	Sigmoid	MSE	*1*	*0.9*	*0.0531*	*0.0597*	*82.8*	*77.4*
Offline	Softmax	MSE	*1*	*0.9*	*0.044*	*0.0538*	*86.4*	*80.733*
Offline	Softmax	CCR	*1*	*0.9*	*0.0267*	*0.0957*	*95.9778*	*84.533*
Online	Sigmoid	MSE	*1*	*0.9*	*0.0499*	*0.0625*	*82.511*	*78.2667*
Online	Softmax	MSE	*1*	*0.9*	*0.18635*	*0.18769*	*29.0667*	*28.86*
Online	Softmax	CCE	*1*	*0.9*	*0.18635*	*0.18769*	*29.0667*	*28.8667*

Table 5: noMNIST dataset results

mode, the results are good enough, where we can observe an improval of the result when we use the softmax function as output function the results improve.

While using the online learning mode, the softmax function does not seem to improve, it only give us good results while using MSE as error function and a sigmoidal function as output function.

3.4 Experimental conclusions

As experimental conclusion, we can observe that the results depends on the dataset, each one is completely different of the others.

The most remarkable point is how using a sigmoidal output function with a cross entropy error function, the network does not seem to work properly. That can be explained understanting that the cross entropy only has a probabilistic sense, so it only can work properly with a probabilistic output function, as the softmax function.