Lab assignment 2: Multilayer perceptron for classification problems

Academic year 2021/2022

Subject: Introduction to computational models
4th course Computer Science Degree (University of Córdoba)

20th October 2021

Abstract

This lab assignment serves as familiarisation for the student with neural network computational models applied to classification problems, in particular, with the multilayer perceptron implemented in the previous lab assignment. On the other hand, it is required to implement the *off-line* version of the training algorithm. The student must implement these modifications and check the effect of different parameters over a given set of real-world datasets, with the aim of obtaining the best possible results in classification. Delivery will be made using the task in Moodle authorized for this purpose. All deliverables must be uploaded in a single compressed file indicated in this document. The deadline for the submission is **31st October 2021**. In case two students submit copied assignments, neither of them will be scored.

1 Introduction

The work to be done in this lab assignment consists on adapting the back-propagation algorithm implemented in the previous lab assignment to classification problems. Concretely, a probabilistic meaning will be given to this algorithm by means of two elements:

- Use of the *softmax* activation function in the output layer.
- Use of the cross-entropy error function.

Furthermore, the *off-line* version of the algorithm will also be implemented.

The student should develop a programme able to train a model with the aforementioned modifications. This programme will be used to train models able to classify as accurate as possible a set of databases available in Moodle. Also, an analysis about the obtained results will be included. **This analysis will greatly influence the qualification of this assignment.**

In the statement of the assignment, indicative values are provided for all parameters. However, it will be positively evaluated if the student finds other values for these parameters able to achieve better results. The only condition is that the maximum number of iterations for the outer loop can not be modified (established to 1000 iterations for the XOR problem and *divorce*, and 500 iterations for the *noMNIST* dataset).

Section 2 describes a series of general guidelines when implementing the back-propagation algorithm. Section 3 explains the experiments to be carried out once the algorithm's modifications are implemented. Finally, section 4 specifies the files to be delivered for this assignment.

2 Implementation of the back-propagation algorithm

Follow the instructions on the class slides in order to add the following characteristics to the algorithm implemented in the previous lab assignment:

1. *Softmax function*: The possibility to use the *softmax* function in the neurons of the output layer should be incorporated, being its output defined as:

$$net_j^H = w_{j0}^H + \sum_{i=1}^{n_{H-1}} w_{ji}^H out_i^{H-1}, \tag{1}$$

$$out_j^H = o_j = \frac{\exp(net_j^H)}{\sum_{l=1}^{n_H} \exp(net_l^H)}. \tag{2}$$

   You should implement the model optimised by the number of weights, in this way, for the last output (output n_H), $net_{n_H}^H = 0$ will be considered. This allows us to reduce the number of weights, in this sense, we can discard the weights $w_{n_H 0}^H, w_{n_H 1}^H, \ldots, w_{n_H n_{H-1}}^H$. Although it is possible to avoid the allocation of one neuron on the output layer, it is recommended to establish to `NULL` all the vectors associated to that neuron. Use $net_{n_H}^H = 0$ when `NULL` is found in the forward-propagation (only for the last layer and when using *softmax* function) and ignore the neuron in the rest of the methods.

2. *Error function based on the cross-entropy*: The possibility to use the cross-entropy as error function must be introduced, this is:

$$L = -\frac{1}{N} \sum_{p=1}^{N} \left(\frac{1}{k} \sum_{o=1}^{k} d_{po} \ln(o_{po}) \right), \tag{3}$$

   where N is the number of patterns of the database, k is the number of outputs, d_{po} is set to 1 if the pattern p belongs to the o class (and 0 otherwise) and o_{po} is the probability value obtained by the model for the pattern p and the class o.

3. *Working mode*: Apart from working in *on-line* mode (previous lab assignment), the algorithm should include the possibility of working in *off-line* mode or *batch*. This is, for each training pattern (inner loop), the error will be computed and the change accumulated, but we will not adjust the network weights. Once all the training patterns are processed (and the changes accumulated), then the weights will be adjusted and the stopping condition of the outer loop will be checked (in the case that the stopping condition is not satisfied, we will start again by the first pattern). Remember to average the derivates during the weight adjustment for the *off-line* mode, as it is explained in the slides.

4. The rest of the algorithm characteristics (use of the *training* files and *test* files), *stopping condition*, *copies of the weights* and *seeds for the random numbers*) will be the same specified in the previous lab assignment. However, for this assignment, it is highly recommended to take the default values for the learning rate and the following momentum factors: $\eta = 0.7$ and $\mu = 1$, adjusting them if necessary until convergence is achieved.

It is recommended to implement the previous points, checking that everything work fine (at least with two datasets) before moving forward to the next point.

3 Experiments

We will test different configurations of the neural network and execute each configuration with five seeds (1, 2, 3, 4 and 5). Based on the results obtained, the average and standard deviation of the error will be obtained. Although the training is guided by the cross-entropy or the MSE, the programme must show the percentage of correct classified patterns (CCR), given that for classification problems this is the most appropriate performance measure. [1] The percentage of

[1]The bad thing is that it is not derivable and we can not use it to adjust weights.

correct classified patterns can be expressed as follows:

$$CCR = 100 \times \frac{1}{N} \sum_{p=1}^{N} \left(I(y_p = y_p^*) \right), \tag{4}$$

where N is the number of patterns of the dataset considered, y_p is the target class for the pattern p (this is, the index of the maximum value of the vector $\mathbf{d}_p$, $y_p = \arg\max_o d_{po}$, or what is the same, the index of the position with a 1) and y_p^* is the class obtained for the pattern p (this is, the index of the maximum value of the vector $\mathbf{o}_p$ or the output neuron that achieves the highest probability for the pattern p, $y_p^* = \arg\max_o o_{po}$).

To assess how the implemented algorithm works, we will run it on three different datasets:

- *XOR problem*: this dataset represents the problem of non-linear classification of the XOR. The same file will be used for train and test. As can be seen, this file has been adapted to 1-to-k codification, finding two outputs instead of one.
- *Divorce dataset*: *divorce* contains 127 training patterns and 43 test patterns. The dataset contains the answer to a series of questions belonging to surveys, with the aim of predicting the *divorce* of a partner. The answers to the questions are provided in the Likert scale with values from 0 to 4. All the input variables are numerically considered. Two examples of questions are as follows:
  - *23. I know my spouse's favourite food.*
  - *24. I can tell you what kind of stress my spouse is facing in her/his life.*

  The dataset contains a total of 54 questions (therefore, 54 input variables) and two categories (0 if there is no divorce, 1 if there is a divorce)[2].
- *noMNIST dataset*: originally, this dataset was composed by 200.000 training patterns and 10.000 test patterns, with a total of 10 classes. Nevertheless, for this lab assignment, the size of the dataset has been reduced in order to reduce the computational cost. In this sense, the dataset is composed by 900 training patterns and 300 test patterns. It includes a set of letters (from a to f) written with different typologies or symbols. They are adjusted to a squared grid of 28×28 pixels. The images are in grey scale in the interval $[-1.0; +1.0]$[3]. Each of the pixels is an input variable (with a total of $28 \times 28 = 784$ input variables) and the class corresponds to a written letter (a, b, c, d, e y f, with a total of 6 classes). Figure 1 represents a subset of 180 training patterns, whereas figure 2 represents a subset of 180 letters from the test set. Moreover, all the letters are arranged and available in Moodle in the files `train_img_nomnist.tar.gz` and `test_img_nomnist.tar.gz`, respectively.

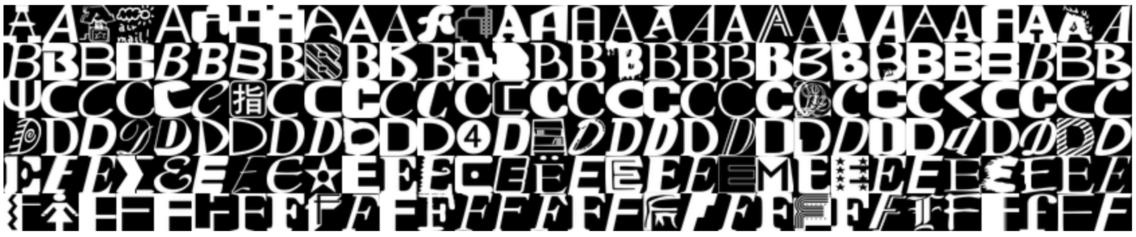
Figure 1: Subset of letters belonging to the training dataset.

[2]Check `https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set` for more information

[3]Check `http://yaroslavvb.blogspot.com.es/2011/09/notmnist-dataset.html` for more information.

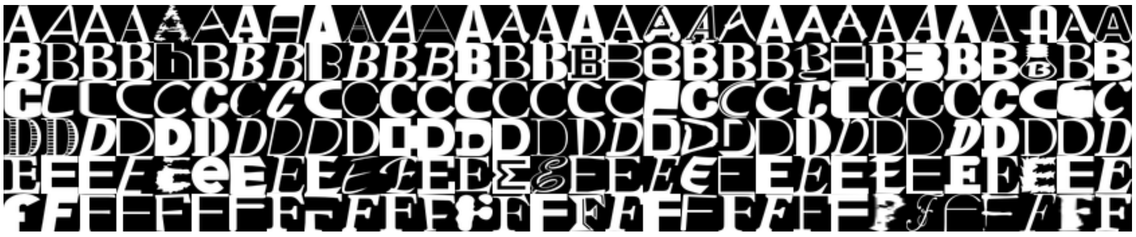

Figure 2: Subset of letters belonging to the test dataset.

A table for each dataset must be built, comparing the average and standard deviation of the following four measures:

- Training and test errors. The MSE or the cross-entropy will be used, according to the decision made by the user to adjust the weights.
- Training and test CCR.

The momentum factor must always be used. It is highly recommended to use the values $\eta = 0.7$ y $\mu = 1$, adjusting them if necessary until convergence is achieved. At least, the following configuration must be tested:

- *Network architecture*: For this first run, use the cross-entropy error function and the *softmax* activation function in the output layer, using the *off-line* version of the algorithm. Do not use the validation set ($v = 0.0$) and deactivate the decreasing factor ($F = 1$).
  - For the XOR problem, use the architecture achieving the best performance in the previous lab assignment.
  - For the *divorce* and *noMNIST* problems, 8 different architectures (one or two hidden layers with 4, 8, 16 o 64 neurons) must be tested.
- Once decided the best architecture, test the following combinations (with the *off-line* algorithm, $v = 0.0$ and $F = 1$):
  - MSE error function and *sigmoidal* activation function in the output layer.
  - MSE error function and *softmax* activation function in the output layer.
  - Cross-entropy error function and *softmax* activation function in the output layer.
  - Do not try the combination of cross-entropy error function and *sigmoidal* activation function in the output layer, given that it will lead to a bad performance (explain why).
- Once decided the best combination (achieved using the *off-line* version of the algorithm, $v = 0.0$ y $F = 1$), compare the results against the *on-line* version of the algorithm.
- Finally, find the best values for the parameters v and F in the ranges $v \in \{0.0; 0.15; 0.25\}$ and $F \in \{1, 2\}$.
- NOTE1: for the XOR dataset, $v = 0.0$ (no validation) will be considered.
- NOTE2: note if when we activate the validation ($v = 0.15$ o $v = 0.25$), the number of average iterations decreases with respect to not considering validation ($v = 0.0$). This implies a lower computational cost and, therefore, it is an advantage.

Attention: Depending on the error function, it could be necessary to adapt the values for the learning rate (η) and the momentum factor (μ).

As a guideline, the training CCR and the test CCR achieved by a logistic regression (using Weka) over the three datasets is shown:

- *XOR problem*: $CCR_{\text{train}} = CCR_{\text{test}} = 50\%$.
- *Divorce dataset*: $CCR_{\text{train}} = 90.5512\%; CCR_{\text{test}} = 90.6977\%$.
- *noMNIST dataset*: $CCR_{\text{train}} = 80.4444\%; CCR_{\text{test}} = 82.6667\%$.

The student should be able to improve this error values with some of the configurations.

3.1 File format

The datasets files will follow the same format than the previous assignment. Note that for this lab assignment, all the files have multiple outputs (one for each class).

4 Assignments

The files to be submitted will be the following:

- Report in a `pdf` file describing the programme implemented, including results, tables and their analysis.
- Executable file and source code.

4.1 Report

The report for this lab assignment must include, at least, the following content:

- Cover with the lab assignment number, its title, subject, degree, faculty department, university, academic year, name, DNI and email of the student
- Index of the content with page numbers.
- Description of the neural network models used (architecture and layer organisation) (**1 page maximum**).
- Pseudocode description of the back-propagation algorithm and all those relevant operations. The pseudocode must necessarily reflect the implementation and development done and not a generic description extracted from the slides or any other source. (**3 pages maximum**).
- Experiments and results discussion:
  - Brief description of the datasets used.
  - Brief description of the values of the parameters considered.
  - Results obtained, according to the format specified in the previous section.
  - Discussion/analysis of the results. The analysis must be aimed at justifying the results obtained instead of merely describing the tables. Take into account that this part is extremely decisive in the lab assignment qualification. The inclusion of the following comparison items will be appreciated:
    * Test confusion matrix of the best neural network model achieved for the *noMNIST* database.
    * Also for *noMNIST*, analyse the errors, **including the images of some letters for which the model mistakes**, to visually check if they are confusing.
    * Convergence charts: they reflect, on the x-axis, the iteration number of the algorithms, and, in the y-axis, the CCR on the training set and/or on the test set (it can also include the CCR on the validation set).

- Bibliographic references or any other material consulted in order to carry out the lab assignment different to the one provided by the lecturers (if any).

Although the content is important, the presentation, including the style and structure of the document will also be valued. The presence of too many spelling mistakes can decrease the grade obtained.

4.2 Executable and source code

Together with the report, the executable file prepared to be run in the UCO's machines (concretely, test using `ssh` on `ts.uco.es`) must be included. In addition, all the source code must be included. The executable should have the following characteristics:

- Its name will be `la2`.
- The programme to be developed receive twelve arguments on command line (that could appear in any order). [4] The first nine arguments have not changed with respect to the previous assignment. The last three arguments incorporate the modifications included in this assignment:
  - Argument `t`: Indicates the name of the file that contains the training data to be used. This argument is compulsory, and without it, the program can not work.
  - Argument `T`: Indicates the name of the file that contains the testing data to be used. If it is not specified, training data will be used as testing data.
  - Argument `i`: Indicates the number of iterations for the outer loop. If it is not specified, use 1000 iterations.
  - Argument `l`: Indicates the number of hidden layers of the neural network. If it is not specified, use 1 hidden layer.
  - Argument `h`: Indicates the number of neurons to be introduced in each hidden layer. If it is not specified, use 5 neurons.
  - Argument `e`: Indicates the value for the eta (η) parameter. By default, use $\eta = 0.1$.
  - Argument `m`: Indicates the value for the mu (μ) parameter. By default, use $\mu = 0.9$.
  - Argument `v`: Indicates the ratio of training patterns to be used as validation patterns. By default, use $v = 0.0$.
  - Argument `d`: Indicates the value for the decreasing factor (F in the slides) to be used for each of the layers. By default, use $F = 1$.
  - Argument `o`: Boolean that indicates if the *on-line* version is applied. By default, use the *off-line* version.
  - Argument `f`: Indicates the error function to be used (0 for the MSE and 1 for the cross-entropy). By default, use MSE.
  - Argument `s`: Boolean that indicates if the *softmax* function is used for the output layer. By default, use the sigmoidal function.
- Optionally, another argument could be included to save the configuration of the trained model (it would be necessary to obtain the predictions for the Kaggle competition):
  - Argument `w`: Indicates the name of the file in which the configuration will be stored and the value of the weights of the trained model.
- An example of execution can be seen in the following output:

[4] Use the function `getopt()` from `libc` to process the input sequence.

```

i02gupep@NEWTS:~/imc/workspace/la2/Debug$ ./la2 -t ../train_xor.dat -T ../test_xor.
    dat -i 1000 -l 1 -h 16 -e 0.7 -m 1 -f 1 -s
**********
SEED 1
**********
Iteration 1      Training error: 0.366974        Validation error: 0
Iteration 2      Training error: 0.428928        Validation error: 0
Iteration 3      Training error: 0.359314        Validation error: 0
Iteration 4      Training error: 0.404694        Validation error: 0
Iteration 5      Training error: 0.354386        Validation error: 0
Iteration 6      Training error: 0.38935         Validation error: 0
Iteration 7      Training error: 0.351464        Validation error: 0
Iteration 8      Training error: 0.376896        Validation error: 0
Iteration 9      Training error: 0.348207        Validation error: 0

....

Iteration 997    Training error: 0.000854865     Validation error: 0
Iteration 998    Training error: 0.000853851     Validation error: 0
Iteration 999    Training error: 0.000852839     Validation error: 0
Iteration 1000   Training error: 0.000851829     Validation error: 0
NETWORK WEIGHTS
===============
Layer 1
------
1.890014 -1.701798 1.897538
0.330608 -0.183608 -0.265321
-0.494545 0.501866 -0.647277
-0.127540 0.794900 0.055426
-1.714361 1.917311 1.665996
1.877923 1.585010 1.831484
-2.962570 2.817746 -2.966236
-2.119120 -2.356071 2.169989
-2.635774 -2.427666 -2.633887
-1.783463 2.066134 1.723217
-0.212003 1.037069 0.177721
-1.790252 2.016734 1.726581
2.405366 2.624863 -2.454070
2.159760 1.917194 2.151348
-2.390844 2.232019 -2.398387
0.712774 0.364864 0.987507
Layer 2
------
-2.034694 0.645946 0.169628 -0.859297 -2.580101 2.436878 4.976043 3.776574
    -4.019719 -2.962101 -0.721652 -2.916534 -4.314177 3.248817 3.412149 1.108400
    0.247830
Desired output Vs Obtained output (test)
=========================================
1 -- 0.998009 0 -- 0.00199144
0 -- 0.00141374 1 -- 0.998586
1 -- 0.998492 0 -- 0.00150804
0 -- 0.00189549 1 -- 0.998105
We end!! => Final test CCR: 100
**********
SEED 2
**********
Iteration 1      Training error: 0.373712        Validation error: 0
Iteration 2      Training error: 0.446452        Validation error: 0
Iteration 3      Training error: 0.382052        Validation error: 0
Iteration 4      Training error: 0.403883        Validation error: 0
Iteration 5      Training error: 0.383328        Validation error: 0

....

Iteration 997    Training error: 0.00091654      Validation error: 0
Iteration 998    Training error: 0.000915438     Validation error: 0
Iteration 999    Training error: 0.000914339     Validation error: 0
Iteration 1000   Training error: 0.000913242     Validation error: 0
NETWORK WEIGHTS
===============
Layer 1
------
-2.587098 2.578320 -2.619753
-0.759566 -0.149755 -0.002021
2.047701 -2.066665 -2.047086
1.337941 1.483398 -1.350814
1.836233 -1.846479 1.891185
-2.385468 -2.426681 2.388022
0.701236 -0.490246 -0.277441
-1.694265 1.704642 -1.750630
-0.416997 0.865281 0.703919
-2.532052 -2.493683 -2.503934
-2.877679 2.886804 2.877883
0.194349 0.803944 -0.534277
-2.335808 2.323417 -2.366562
2.216600 -2.226776 -2.218230
-0.963446 0.411973 -0.568539
1.548762 1.482188 1.418917
Layer 2
------
4.304145 0.188449 2.996299 -1.668890 -2.590935 3.737036 0.263283 2.188783 -0.513814
     -4.092572 -5.431650 -0.199261 3.583364 3.350478 0.350937 1.630442 -0.783647
Desired output Vs Obtained output (test)
=========================================
1 -- 0.99812 0 -- 0.0018796
0 -- 0.0018402 1 -- 0.99816
1 -- 0.998286 0 -- 0.00171411
0 -- 0.00186534 1 -- 0.998135
We end!! => Final test CCR: 100
**********
SEED 3
**********

....

**********
SEED 4
**********

....

**********
SEED 5
**********

....

Iteration 995    Training error: 0.000996214     Validation error: 0
Iteration 996    Training error: 0.000995021     Validation error: 0
Iteration 997    Training error: 0.00099383      Validation error: 0
Iteration 998    Training error: 0.000992642     Validation error: 0
Iteration 999    Training error: 0.000991457     Validation error: 0
Iteration 1000   Training error: 0.000990274     Validation error: 0
NETWORK WEIGHTS
===============
Layer 1
------
1.924121 -1.907170 1.880177
-2.877435 -2.894421 -2.923764
-0.465113 -0.098912 0.013589
2.301898 -2.325585 -2.286912
2.269386 -2.242283 2.208006
-0.260069 -0.097105 -0.120448
```

```
0.117679 -0.533281 -0.353352
-0.936595 -0.984372 -1.101798
-1.456342 1.467005 1.490475
-1.701050 -1.676890 1.751372
-0.054140 0.301263 -0.964476
2.406749 2.388844 -2.422281
-0.296602 -1.008889 0.356140
-2.691590 2.674935 -2.642923
0.907242 1.057448 1.125275
2.963765 -2.980305 -2.951356
Layer 2
------
-2.556302 -5.030181 0.300620 3.561749 -3.354764 1.023757 0.480647 -1.176288
    -1.501465 2.462988 0.125039 -3.878418 0.627243 4.759606 1.263119 5.504183
    0.443612
Desired output Vs Obtained output (test)
=========================================
1 -- 0.997981 0 -- 0.00201925
0 -- 0.00193022 1 -- 0.99807
1 -- 0.99795 0 -- 0.00204989
0 -- 0.00191499 1 -- 0.998085
We end!! => Final test CCR: 100
WE HAVE FINISHED WITH ALL THE SEEDS
FINAL REPORT
*************
Train error (Mean +- SD): 0.000950977 +- 8.47617e-05
Test error (Mean +- SD): 0.000950977 +- 8.47617e-05
Train CCR (Mean +- SD): 100 +- 0
Test CCR (Mean +- SD): 100 +- 0

____________________________________________________________________________________

i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.
    dat -i 500 -l 1 -h 4 -e 0.7 -m 1 -f 1 -s

....

FINAL REPORT
*************
Train error (Mean +- SD): 0.069922 +- 0.00569867
Test error (Mean +- SD): 0.123385 +- 0.0128565
Train CCR (Mean +- SD): 89 +- 0.906084
Test CCR (Mean +- SD): 80.6667 +- 1.5456

____________________________________________________________________________________

i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.
    dat -i 500 -l 1 -h 8 -e 0.7 -m 1 -f 1 -s -v 0.25 -d 2

....

FINAL REPORT
*************
Train error (Mean +- SD): 0.0698994 +- 0.00589789
Test error (Mean +- SD): 0.11215 +- 0.00631624
Train CCR (Mean +- SD): 87.9111 +- 1.02426
Test CCR (Mean +- SD): 80.4667 +- 1.84992


____________________________________________________________________________________


i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.
    dat -i 500 -l 1 -h 4 -e 0.7 -m 1 -f 0 -s

....

```

```
FINAL REPORT
*************
Train error (Mean +- SD): 0.0607302 +- 0.00429784
Test error (Mean +- SD): 0.0688165 +- 0.00518643
Train CCR (Mean +- SD): 77.1778 +- 4.43652
Test CCR (Mean +- SD): 72.0667 +- 4.29082

___________________________________________________________________________________


i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.
    dat -i 500 -l 1 -h 8 -e 0.7 -m 1 -f 1 -s -v 0.25 -d 2 -o
....

FINAL REPORT
*************
Train error (Mean +- SD): 0.205463 +- 0.0543647
Test error (Mean +- SD): 0.225942 +- 0.057252
Train CCR (Mean +- SD): 68.6815 +- 4.17005
Test CCR (Mean +- SD): 66.6667 +- 5.07171

___________________________________________________________________________________

i02gupep@NEWTS:~/imc/workspace/la2/Debug$ ./la2 -t ../train_divorce.dat -T ../
    test_divorce.dat -i 1000 -l 1 -h 8 -e 0.7 -m 1 -f 0 -s
...
FINAL REPORT
*************
Train error (Mean +- SD): 0.000646809 +- 0.000112738
Test error (Mean +- SD): 0.0212406 +- 0.000947102
Train CCR (Mean +- SD): 100 +- 0
Test CCR (Mean +- SD): 97.6744 +- 0
```

4.3 [OPTIONAL] Obtaining the predictions for Kaggle

The same executable of the assignment will allow obtaining the predictions for a given dataset. This output must be saved in a `.csv` file that must be uploaded to Kaggle to participate in the competition (check the file format of `sampleSubmission.csv` on Kaggle). This prediction mode uses different parameter than those mentioned previously:

- Argument `p`: Flag indicating that the program will run in prediction mode.
- Argument `T`: Indicates the name of the file containing the test data to be used (`test_kaggle.dat`).
- Argument `w`: Indicates the name of the file containing the configuration and the values for the weights of the trained model that will be used to predict the outputs.

Below is an example of how the training mode is executed using the parameter `w`, which saves the configuration of the model.

```
i02gupep@NEWTS:~/imc/workspace/la2/Debug$ ./la2 -t ../train.dat -T ../test.dat -i 1000 -
    l 1 -h 4 -e 0.7 -m 1 -f 1 -s -w weights.txt

**********
SEED 1
**********

...

**********
SEED 2
**********

...
```

```

**********
SEED 3
**********

...

**********
SEED 4
**********

...

**********
SEED 5
**********

...

FINAL REPORT
*************
Train error (Mean +- SD): 0.061885 +- 0.00883649
Test error (Mean +- SD): 0.0627513 +- 0.00875778
Train CCR (Mean +- SD): 83.04 +- 1.93203
Test CCR (Mean +- SD): 81.97 +- 1.8529
```

Below is an example of the output using the prediction mode:

```
i02gupep@NEWTS:~/imc/practica1/Debug$ ./la2 -T ../kaggle.dat -p -w weights.txt
Id,Category
0,9
1,5
2,2
3,4
4,8
5,3

...

1994,3
1995,0
1996,1
1997,1
1998,4
1999,6
```