# Lab assignment 2: Multilayer perceptron for classification problems

Academic year 2023/2024

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

18th October 2023

**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 **5th November 2023**. 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 *ProPublica COMPAS*, 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.
- *ProPublica COMPAS*: This dataset is about the performance of COMPAS algorithm, a statistical method for assigning risk scores within the United States criminal justice system created by Northpointe. It was published by ProPublica in 2016 [2], claiming that this risk tool was biased against African-American individuals (we will deal with this in the following assignment). In this dataset, they analyzed the COMPAS scores for "risk of recidivism" so each individual has a binary "recidivism" outcome, that is the prediction task, indicating whether they were rearrested within two years after the first arrest (the charge described in the data). We reduced the original dataset from 52 to 9 attributes similarly to the original dataset: sex, age, age_cat, race, juv_fel_count, juv_misd_count, juv_other_count, priors_count, c_charge_degree. The prediction variable is whether the individual will be rearrested in two years or not.
  1. sex: binary sex.
  2. age: numerical age.
  3. age_cat: categorical (age $< 25$, age $\geq 25$ and age $< 45$, age $\geq 45$).
  4. race: binary attribute (0 means 'white' and 1 means 'black').
  5. juv_fel_count: a continuous variable containing the number of juvenile felonies.
  6. juv_misd_count: a continuous variable containing the number of juvenile misdemeanors.
  7. juv_other_count: a continuous variable containing the number of prior juvenile convictions that are not considered either felonies or misdemeanors.
  8. priors_count: a continuous variable containing the number of prior crimes committed.
  9. c_charge_degree: Degree of the crime. It is either M (Misdemeanor), F (Felony), or O (not causing jail time).
- *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 $28 \times 28$ 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.

[2] `https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm`

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

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

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

All the databases are normalized. **Outputs are never normalised in classification**.

**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.
  - For the XOR problem, use the architecture achieving the best performance in the previous lab assignment.
  - For the *ProPublica* 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):
  - $MSE$ 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), compare the results against the *on-line* version of the algorithm.

**Attention**: Depending on the error function, it could be necessary to adapt the values for the learning rate ($\eta$) and the momentum factor ($\mu$).

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*: $CCR_{\text{training}} = CCR_{\text{test}} = 50\%$.
- *ProPublica dataset*: $CCR_{\text{training}} = 67.7348\%; CCR_{\text{test}} = 66.8514\%$.
- *noMNIST dataset*: $CCR_{\text{training}} = 80.4444\%; CCR_{\text{test}} = 82.6667\%$.

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.

- 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 $eta$ ($\eta$) parameter. By default, use $\eta = 0.1$.
  - Argument `m`: Indicates the value for the $mu$ ($\mu$) parameter. By default, use $\mu = 0.9$.
  - 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.
  - Argument `n`: Boolean indicating that the data (training and test) will be normalised after reading. Inputs will be normalised to the interval $[-1, 1]$ (outputs are not normalised in classification).
- 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.

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

- An example of execution can be seen in the following output:

```

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
Iteration 2       Training error: 0.394686
Iteration 3       Training error: 0.357086
Iteration 4       Training error: 0.366077
Iteration 5       Training error: 0.349135
Iteration 6       Training error: 0.351968
Iteration 7       Training error: 0.343165
Iteration 8       Training error: 0.343468
Iteration 9       Training error: 0.338281
Iteration 10      Training error: 0.337465

....

Iteration 996     Training error: 0.00102801
Iteration 997     Training error: 0.00102678
Iteration 998     Training error: 0.00102554
Iteration 999     Training error: 0.00102431
Iteration 1000    Training error: 0.00102309
NETWORK WEIGHTS
===============
Layer 1
------
1.932040 -1.737134 1.926088
0.326163 -0.146441 -0.182055
-0.490515 0.503036 -0.586897
-0.163416 0.775937 0.142567
-1.648976 1.859996 1.620257
1.865697 1.598651 1.845814
-2.915683 2.762668 -2.906463
-2.120769 -2.339651 2.162988
-2.523937 -2.326467 -2.534853
-1.837088 2.108107 1.788306
-0.238769 1.013190 0.253558
-1.826408 2.054005 1.775930
2.317019 2.528668 -2.357329
2.145112 1.922300 2.156072
-2.327206 2.158176 -2.320141
0.708153 0.410257 1.074749
Layer 2
------
-2.145596 0.658908 0.172077 -0.855432 -2.428730 2.420198 4.861758 3.750326
    -3.734557 -3.039142 -0.680678 -2.974963 -4.075474 3.229442 3.288709 1.130524
    0.243718
Desired output Vs Obtained output (test)
=========================================
1 -- 0.997629 0 -- 0.00237091
0 -- 0.00168976 1 -- 0.99831
1 -- 0.99817 0 -- 0.00183032
0 -- 0.00228516 1 -- 0.997715
We end!! => Final test CCR: 100
**********
SEED 2
**********
Iteration 1       Training error: 0.373712
Iteration 2       Training error: 0.402586
Iteration 3       Training error: 0.368335
Iteration 4       Training error: 0.369898
Iteration 5       Training error: 0.361395
Iteration 6       Training error: 0.358421
Iteration 7       Training error: 0.356269
```

```
Iteration 8        Training error: 0.353413
Iteration 9        Training error: 0.352642
Iteration 10       Training error: 0.350569

....

Iteration 997      Training error: 0.00111981
Iteration 998      Training error: 0.00111843
Iteration 999      Training error: 0.00111706
Iteration 1000     Training error: 0.0011157
NETWORK WEIGHTS
===============
Layer 1
------
-2.561168 2.538790 -2.589251
-0.770332 -0.212399 0.075913
1.886265 -1.926663 -1.888277
1.233641 1.415421 -1.243881
1.888271 -1.880845 1.937680
-2.398075 -2.444980 2.397720
0.675080 -0.449499 -0.187134
-1.626142 1.626569 -1.680457
-0.432686 0.870079 0.746575
-2.467137 -2.423554 -2.444506
-2.905327 2.927639 2.908822
0.178809 0.807696 -0.499326
-2.263792 2.236110 -2.289030
2.141228 -2.166324 -2.146262
-0.961022 0.368665 -0.519352
1.671652 1.605829 1.578243
Layer 2
------
4.218926 0.256066 2.682362 -1.515218 -2.665360 3.794517 0.199035 2.051653 -0.533876
     -3.897894 -5.539269 -0.179128 3.387229 3.200096 0.315796 1.884826 -0.745766
Desired output Vs Obtained output (test)
=========================================
1 -- 0.997695 0 -- 0.00230545
0 -- 0.0022113 1 -- 0.997789
1 -- 0.997915 0 -- 0.00208488
0 -- 0.00231397 1 -- 0.997686
We end!! => Final test CCR: 100
**********
SEED 3
**********

....

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

....

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

....

Iteration 996      Training error: 0.00120944
Iteration 997      Training error: 0.00120797
Iteration 998      Training error: 0.0012065
Iteration 999      Training error: 0.00120504
Iteration 1000     Training error: 0.00120358
NETWORK WEIGHTS
===============
Layer 1
```

```
------
1.900581 -1.882362 1.857397
-2.842675 -2.859711 -2.889926
-0.465074 -0.098927 0.013740
2.268086 -2.294572 -2.254044
2.243438 -2.213363 2.180051
-0.256642 -0.105276 -0.120501
0.114388 -0.534062 -0.351653
-0.911573 -0.958489 -1.080537
-1.433847 1.444481 1.471491
-1.661529 -1.637269 1.715131
-0.052632 0.301068 -0.963939
2.370791 2.352868 -2.385332
-0.288566 -1.009871 0.350465
-2.658619 2.639775 -2.608652
0.887022 1.042201 1.112214
2.932930 -2.951392 -2.921607
Layer 2
------
-2.509621 -4.920168 0.292810 3.478740 -3.292094 1.022310 0.478927 -1.150676 -1.462800 2.385706 0.120623 -3.790206 0.620506 4.659517 1.236610 5.406729 0.433783
Desired output Vs Obtained output (test)
=========================================
1 -- 0.997546 0 -- 0.00245364
0 -- 0.00234317 1 -- 0.997657
1 -- 0.99751 0 -- 0.00249031
0 -- 0.0023299 1 -- 0.99767
We end!! => Final test CCR: 100
WE HAVE FINISHED WITH ALL THE SEEDS
FINAL REPORT
*************
Train error (Mean +- SD): 0.00114768 +- 0.000104197
Test error (Mean +- SD): 0.00114768 +- 0.000104197
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 -f 1 -s

....
FINAL REPORT
*************
Train error (Mean +- SD): 0.0751722 +- 0.00948445
Test error (Mean +- SD): 0.124133 +- 0.0127144
Train CCR (Mean +- SD): 86.7778 +- 4.27236
Test CCR (Mean +- SD): 78.4 +- 5.96937

__________________________________________________________________________________

i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.dat -i 500 -l 1 -h 4 -e 1 -m 2 -f 1 -s

....

FINAL REPORT
*************
Train error (Mean +- SD): 0.048567 +- 0.00473395
Test error (Mean +- SD): 0.113218 +- 0.0124186
Train CCR (Mean +- SD): 92.1333 +- 0.535182
Test CCR (Mean +- SD): 83.6667 +- 1.31233


__________________________________________________________________________________

```

```

i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.
    dat -i 500 -l 1 -h 4 -e 1 -m 2 -f 0 -s

....

FINAL REPORT
*************
Train error (Mean +- SD): 0.0390912 +- 0.00594294
Test error (Mean +- SD): 0.0544008 +- 0.00626936
Train CCR (Mean +- SD): 85.0222 +- 3.20243
Test CCR (Mean +- SD): 77.5333 +- 4.09336

____________________________________________________________________________________


i02gupep@NEWTS:~/imc/la2/Debug$ ./la2 -t ../train_nomnist.dat -T ../test_nomnist.
    dat -i 500 -l 1 -h 8 -e 0.1 -m 2 -f 1 -s -o
....

FINAL REPORT
*************
Train error (Mean +- SD): 0.0425087 +- 0.0151428
Test error (Mean +- SD): 0.151445 +- 0.0202287
Train CCR (Mean +- SD): 91.8222 +- 3.62144
Test CCR (Mean +- SD): 84 +- 4.26224

____________________________________________________________________________________

i02gupep@NEWTS:~/imc/workspace/la2/Debug$ ./la2 -t ../train_compas.dat -T ../
    test_compas.dat
...
*************
Train error (Mean +- SD): 0.228885 +- 0.000211591
Test error (Mean +- SD): 0.230951 +- 0.000302507
Train CCR (Mean +- SD): 68.4249 +- 0.18029
Test CCR (Mean +- SD): 67.6718 +- 0.47264
```

### 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 ../val.dat -i 1000 -l
    1 -h 20 -e 1.5 -m 1 -f 1 -s -w weights.txt

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

...

```

```
**********
SEED 2
**********

...

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

...

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

...

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

...

FINAL REPORT
*************
Train error (Mean +- SD): 0.110211 +- 0.000890431
Test error (Mean +- SD): 0.162645 +- 0.00309483
Train CCR (Mean +- SD): 41.475 +- 1.07674
Test CCR (Mean +- SD): 23.3 +- 1.98746
```

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

```
i02gupep@NEWTS:~/imc/practica1/Debug$ ./la2 -T ../test_X.dat -p -w weights.txt >
    sample_submission.csv
i02gupep@NEWTS:~/imc/practica1/Debug$ head sample_submission.csv
Id,Category
0,13
1,4
2,9
3,12
4,10
5,8
6,11
7,9
8,9
i02gupep@NEWTS:~/imc/practica1/Debug$ tail sample_submission.csv
1702,7
1703,14
1704,9
1705,9
1706,8
1707,8
1708,2
1709,11
1710,8
1711,13
```