Exercises of ANN classifiers

Pablo García Fernández
(pablo.garcia.fernandez2@rai.usc.es)

Sunday 16th January, 2022

Abstract

The aim of this report is to present the work carried out during the execution of the exercises of the unit 6 assignment, as well as the obtained results. In addition to this report, we also attach the developed .py code, the obtained figures and a .txt file with the results.

Exercise 2: MLP and ELM (default configuration) using the whole dataset as training and test set.

As in previous reports the starter point are the hepatitis and wine datasets. This exercise asks us to run MLP and ELM classifiers using the **whole dataset as training and test set** and, more importantly, to analyze the effect of increasing the number of neurons. To achieve this second goal, we repeat the execution of both classifiers with different layers and neurons (layers only in MLP). This is not a tuning process, it is just running the classifier with different configurations on the whole data.

"Tab. 1" shows the obtained kappas under different configurations using a MLP classifier. H1, H2 and H3 columns refers to hidden layers of neurons. Rows indicate the number of neurons.

Table 1: Analyzing the effect of neurons/layers. MLP is ran over the whole dataset.

Hepatitis dataset				*Wine dataset*			
Configuration			**Kappa (%)**	**Configuration**			**Kappa (%)**
H1	**H2**	**H3**		**H1**	**H2**	**H3**	
10			14.1	10			56.2
20			68.7	20			94.9
30			0.00	30			92.3
10	10		0.00	10	10		87.0
20	20		98.0	20	20		57.9
30	30		89.8	30	30		100
10	10	10	0.00	10	10	10	0.00
20	20	20	87.3	20	20	20	100
30	30	30	100	30	30	30	88.1

"Tab. 2" shows the same but for an ELM classifier, where the only modifiable parameter is the number of neurons in the hidden layer.

Table 2: Analyzing the effect of neurons. ELM is ran over the whole dataset.

Hepatitis dataset		*Wine dataset*	
IK	**Kappa (%)**	**IK**	**Kappa (%)**
5	13.0	5	61.1
10	30.9	5	88.0
15	49.5	5	94.1
20	60.6	5	95.7
25	64.5	5	97.4
30	59.9	5	98.3
35	66.1	5	98.3
40	67.9	5	100

From both tables we can extract the same conclusion: by increasing the number of neurons or layers, we make the classification function more complex. This allows to better fit the points and therefore the training error (which is roughly what is being measured when training and testing over the whole dataset) decreases. However, this is not a good approximation because the model will tend to overfitting. That is, if we were to test it on new data (on which it was not trained) the results would be worse. Moreover, we can see how MLP tends to overfit more than ELM. If we look at hepatitis, the kappa obtained by MLP reaches 100% and for ELM only 67.9%. In both cases the best result is achieve with the most complex configuration.

As the statement asks us to show also the confusion matrix and the accuracy, the following table ("Tab. 3") shows it for the configuration [30,30,30] of MLP and [40] of ELM.

Table 3: MLP, ELM. Whole dataset as training and test. Most complex configurations

Hepatitis dataset

Classifier	**Metrics**				
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**
MLP	100	100	100	100	100
ELM	90.32	67.86	91.54	96.75	94.07

Wine dataset

Classifier	**Metrics**				
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**
MLP	100	100	-	-	-
ELM	100	100	-	-	-

"Fig. 1" presents the confusion matrices for the 4 above cases.

Finally, it is worth mentioning that since the training algorithms of these models have a random component (e.g., initialization of the weights), the results may vary between runs.

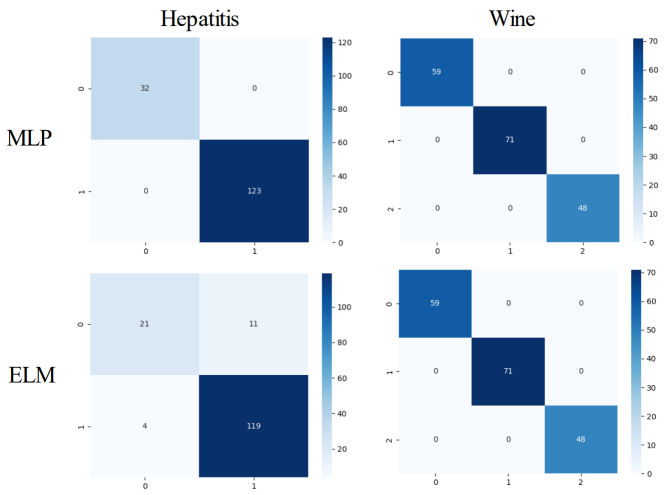

Figure 1: asd

Also note that the ELM implementation was based on the matlab code provided by the professor, changing the activation function to RELU and using softmax for the final output. Details can be seen in the attached *ELM.py* file.

Exercise 3: 4-folds cross validation with parameter tuning.

First we use the function `crea_folds()` to obtain the partitions. Then we train the classifier on the training partition under different configurations and evaluate the results on the validation partition. This process is performed using the cross-validation methodology already explained in previous reports. Once the best hyperparameters are obtained, we re-train the model using both training and validation sets and evaluate its performance on the never-used test partition (using again the 4-folds cross validation approach). We repeat this process for both MLP and ELM classifiers.

"Tab. 4" shows the tuning step for the MLP classifier.

Table 4: MLP tunning step. 4-folds cross validation

Hepatitis dataset				*Wine dataset*			
Configuration			**Kappa (%)**	**Configuration**			**Kappa (%)**
H1	**H2**	**H3**		**H1**	**H2**	**H3**	
10			23.1	10			65.2
20			30.8	20			69.8
30			36.4	30			77.4
10	10		7.60	10	10		58.5
20	20		49.5	20	20		79.4
30	30		45.1	30	30		83.6
10	10	10	6.00	10	10	10	42.0
20	20	20	34.9	20	20	20	80.2
30	30	30	30.6	30	30	30	92.2

Since 2 hyperparameters have to be tuned, a 3D image should be used to see in image format the evolution of kappa. To simplify this, we present the data as a table. The best configuration for hepatitis is [20,20] (2 hidden layers with 20 neurons each). And for wine [30,30,30] (3 hidden layers with 30 neurons each).

"Fig. 2" shows the tuning step for the ELM classifier. The best number of neurons in the hidden layer for hepatitis is 30, and for wine is 15.

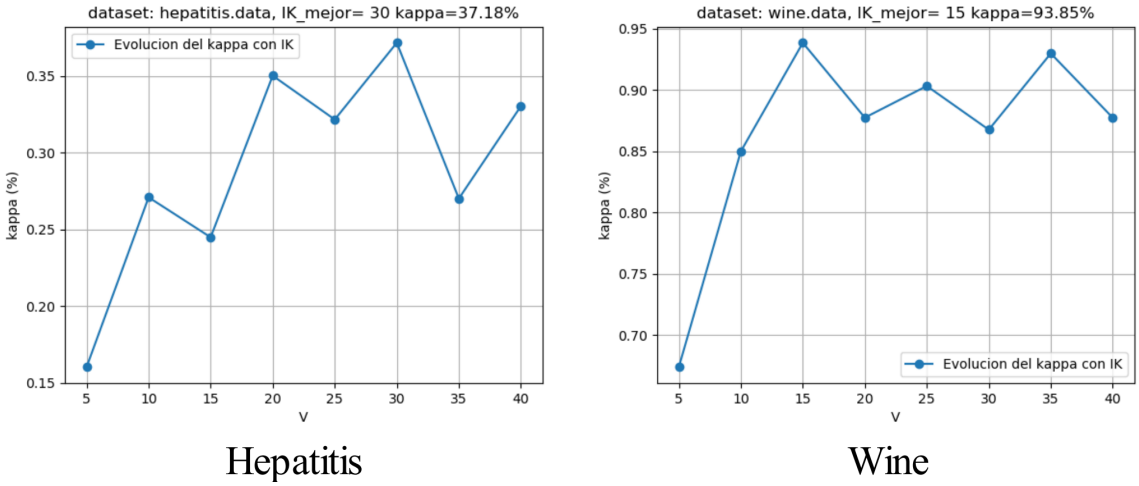

Figure 2: ELM tuning step. 4-folds cross validation

Finally, "Tab. 5" and "Fig. 3" show the test results for MLP and ELM classifiers using the hyperparameters configuration previously established as the best. We can see how ELM achieves significantly better results. Probably because it is easier to use. That is, to obtain similar values with MLP it would probably be necessary to tune also the optimization algorithm (since it is an iterative process that has no analytical solution and depends on the learning rate, the optimizer, etc.). In contrast, ELM does calculate analytically the optimal weight of the neurons of the hidden layer.

Table 5: MLP and ELM test. 4-folds cross validation with tuned hyperparameters

Hepatitis dataset

Classifier	**Metrics**				
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**
MLP	80.49	22.02	85.53	92.42	88.22
ELM	81.71	37.90	86.93	90.91	88.83

Wine dataset

Classifier	**Metrics**				
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**
MLP	82.65	71.10	-	-	-
ELM	93.88	90.66	-	-	-

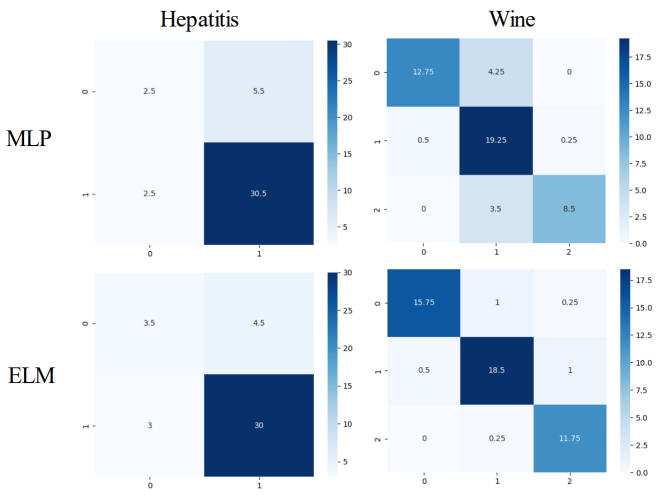

Figure 3: MLP and ELM test. 4-folds cross validation with tuned hyperparameters

Exercise 4: Use MLP and ELM in LBP and CooCur datasets

If we repeat the previous exercise (4-folds cross validation) with LBP and CooCur datasets (concatenating train and test sets before performing `crea_folds()`), we obtain the following results:

- "Tab. 6" and "Fig. 4": Hyperparameter tunning. Best configurations:
  - MLP CooCur: [30] (1 hidden layer with 30 neurons).
  - MLP LBP: [30] (1 hidden layer with 30 neurons).
  - ELM CooCur: 40 neurons.
  - ELM LBP: 40 neurons.
- "Tab. 7" and "Fig. 5": Test results. LBP and MLP achieve the best results. ELM stays close to MLP, with the advantage of a 100 times shorter computational cost.

Table 6: MLP tuning step. 4-folds cross validation

CooCur dataset				*LBP dataset*			
Configuration			**Kappa (%)**	**Configuration**			**Kappa (%)**
H1	**H2**	**H3**		**H1**	**H2**	**H3**	
10			53.3	10			70.3
20			59.7	20			75.7
30			65.2	30			80.0
10	10		48.3	10	10		67.8
20	20		60.8	20	20		76.3
30	30		61.9	30	30		79.2
10	10	10	47.2	10	10	10	63.6
20	20	20	57.9	20	20	20	69.1
30	30	30	61.3	30	30	30	77.0

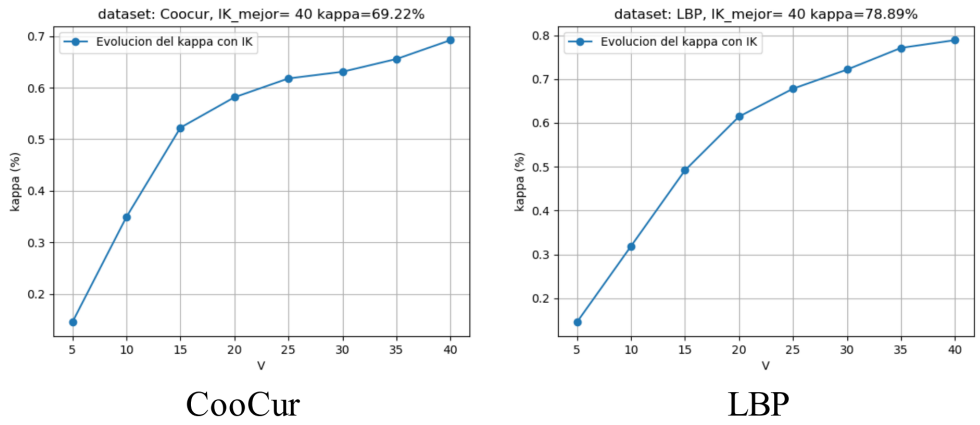

Figure 4: ELM tuning step. 4-folds cross validation

Table 7: MLP and ELM test. 4-folds cross validation with tuned hyperparameters

CooCur dataset

Classifier	**Results**				
	acc. (%)	**kappa (%)**	**Time tuning (s)**	**Time test (s)**	**Time total (s)**
MLP	72.92	72.41	15.17	2.13	17.29
ELM	69.79	69.22	0.03	0.01	0.04

LBP dataset

Classifier	**Results**				
	acc. (%)	**kappa (%)**	**Time tuning (s)**	**Time test (s)**	**Time total (s)**
MLP	83.91	83.61	14.79	2.10	16.88
ELM	77.66	77.24	0.03	0.01	0.04

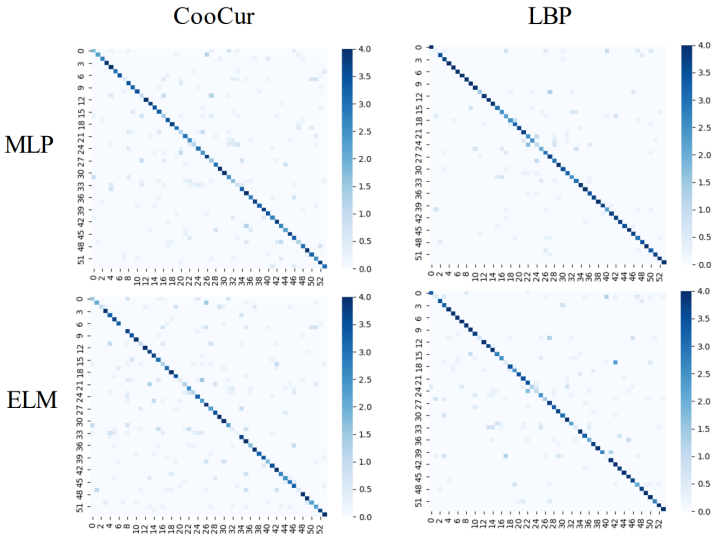

Figure 5: MLP and ELM test. 4-folds cross validation with tuned hyperparameters