Lab assignment 3: Radial basis functions neural networks

Academic year 2021/2022

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

16th of November of 2021

Resumen

This lab assignment serves as familiarisation for the student with radial basis functions (RBF) neural networks. In this way, a RBF neural network will be developed, using Python and the `scikit-learn` library [1]. In this sense, the assignment will also serve as familiarisation with external libraries, widely used in the machine learning field. The student must implement the algorithm and analyse the effect of different parameters over a given set of real-world datasets. 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 **23rd November**. 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 in implementing a RBF neural network with a training stage divided into three steps:

1. Application of a clustering algorithm which will be used to establish the centres of the RBF (input-to-hidden-layer's weights).
2. The RBF radium adjustment is done by means of a simple heuristic (distance average to the rest of the centres).
3. Hidden-to-output's weights learning:
   - For regression problems, using the Moore-Penrose's pseudo-inverse.
   - For classification problems, using a logistic regression linear model.

The student should develop a Python's script able to train a RBF neural network with the aforementioned characteristics. 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.

Section 2 describes a series of general guidelines when implementing the training algorithm for RBF neural networks. Section 3 explains the experiments to be carried out once the algorithm is implemented. Finally, section 4 specifies the files to be delivered for this assignment.

[1] `http://scikit-learn.org/`

2. Implementation of the RBF neural network training algorithm

2.1. Model's architecture to be considered

The RBF neural network models should have the following architecture:

- An input layer with as many neurons as input variables the dataset has.
- A hidden layer with a number of neurons specified by the user. It is important to highlight that, in the two previous lab assignment, the number of hidden layer was variable. However, for this lab assignment, we are going to consider just one hidden layer. The type of all the neurons in the hidden layer will be RBF (in contrast to the sigmoidal neurons used in the previous lab assignments).
- An output layer with as many neurons as output variables the dataset has.
    - When considering regression datasets, all the output neurons will be linear (i.e. similar to the sigmoidal neurons without the application of the $\frac{1}{1+e^{-x}}$ transformation).
    - When considering classification datasets, all the output neurons will be softmax. The softmax function is already implemented by the logistic regression algorithm used for adjusting the weights of the output layer.

2.2. Weights adjustment

The instructions given in the class slides should be followed so that the training is carried out as follows:

1. Application of a clustering algorithm that will serve to establish the centres of the RBF (input-to-output layer weights). For classification problems, the centroid initialisation will be random and stratified, n_1 patterns[2]. For regression problems, n_1 will be randomly selected. After initialising the centroids, the `sklearn.cluster.KMeans` class will be used, with only one centroid initialisation (`n_init`) and a maximum of 500 iterations (`max_iter`).
2. To adjust the radium of the RBF, a simple heuristic will be applied (the half of the distance average to the rest of the centres). This is, the radium of the j-th neuron will be[3]:

$$\sigma_j = \frac{1}{2\cdot(n_1-1)}\sum_{i\neq j}\|c_j - c_i\| = \frac{1}{2\cdot(n_1-1)}\sum_{i\neq j}\sqrt{\sum_{d=1}^{n}(c_{jd}-c_{id})^2}. \tag{1}$$

3. Learning the weights from hidden-to-output layer.
    - For regression problem, it is done using the Moore-Penrose pseudo-inverse. This is:

$$\boldsymbol{\beta}^{\mathrm{T}}_{((n_1+1)\times k)} = (\mathbf{R}^{+})_{((n_1+1)\times N)}\,\mathbf{Y}_{(N\times k)} = \tag{2}$$

$$= \left(\mathbf{R}^{\mathrm{T}}_{((n_1+1)\times N)} \times \mathbf{R}_{(N\times(n_1+1))}\right)^{-1}\mathbf{R}^{\mathrm{T}}_{((n_1+1)\times N)}\mathbf{Y}_{(N\times k)} \tag{3}$$

    where $\mathbf{R}$ is the matrix containing the outputs of the RBF neurons, $\boldsymbol{\beta}$ is a matrix containing a vector of parameters for each of the outputs to be predicted, and $\mathbf{Y}$ is a matrix with the target outputs. To perform these operations, we will use the matrix functions of `numpy`, which is a dependence of `scikit-learn`.

[2]For this, the `sklearn.model_selection.train_test_split` method can be used. It performs one or more stratified dataset partitions, this is, keeping the ratio of patterns belonging to each class in the original dataset `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

[3]Consider using the functions `pdist` and `squareform` of `scipy` to obtain the distances matrix

- For classification problems, it is done using a logistic regression linear model. Using the `sklearn.linear_model.LogisticRegression` class, providing a value for the C parameter in order to apply regularisation. Note that in this library what we are specifying is the cost value C (importance of the approximation error versus the regularisation error), in such a way that $\eta = \frac{1}{C}$. We will use the L2 regularisation[4] and the `liblinear` optimisation algorithm.

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. For the regression problems, only the MSE will be shown. However, for classification problems, the CCR (the percentage of correct classified patterns) will be shown, as well as the MSE[5].

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

- *Sin-function dataset*: This dataset is composed of 120 training patterns and 41 testing patterns. It has been obtained by adding some random noise to the sin function (see Figure 1).

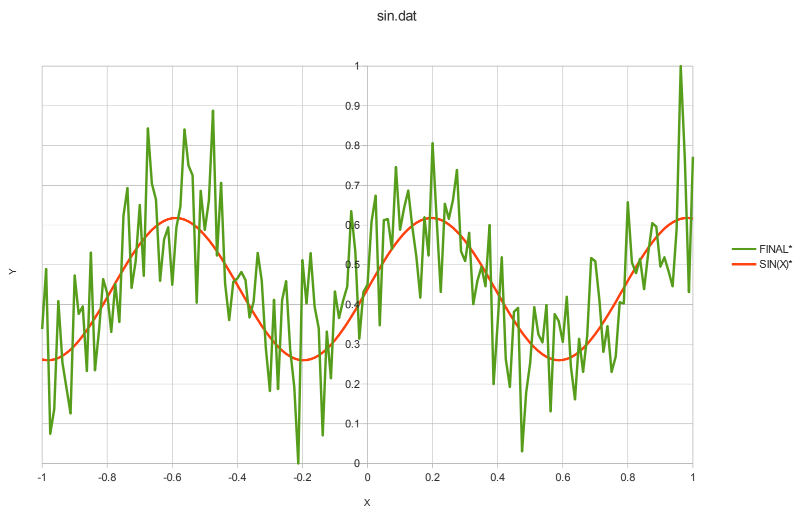

Figura 1: Data representation of the data included in the sin-function estimation problem.

- *Quake dataset*: this dataset is composed by 1633 training patterns and 546 testing patterns. It corresponds to a database in which the objective is to find out the strength of an earthquake (measured on the Richter scale). As input variables, we use the depth of focus, the latitude at which it occurs and the longitude [6].

- *Parkinson dataset*: this dataset is composed by 4406 training patterns and 1469 testing patterns. It contains, as inputs or independent variables, a series of clinical data from patients

[4]`https://msdn.microsoft.com/en-us/magazine/dn904675.aspx`

[5]In classification, the MSE should be obtained in the same way as done for the previous lab assignment, this is, converting the class labels to binary values and comparing them against the predicted probabilities, which could be obtained by the `predict_proba` method.

[6]see `https://sci2s.ugr.es/keel/dataset.php?cod=75` to seek more information.

with Parkinson's disease, including biometric measurement data from their voice. Furthermore, as output or dependent variables, it includes the motor value and the UPDRS (Unified Parkinson's Disease Rating Scale) [7].

And two classification datasets:

- *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)[8].

- *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 $[-1{,}0; +1{,}0]$[9]. 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 2 represents a subset of 180 training patterns, whereas figure 3 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.

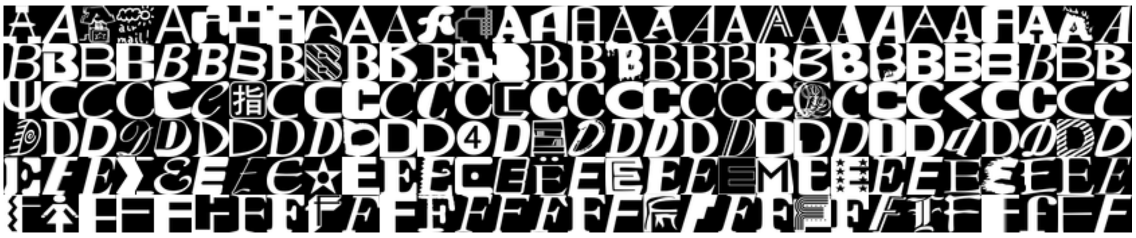

Figura 2: Subset of letters belonging to the training dataset.

The average and standard deviation of two measures (regression) or four measures (classification) should be computed:

- Regression: average and standard deviation of training and testing MSE.
- Classification: average and standard deviation of training and testing CCR, along with the average and standard deviation of training and testing MSE. The MSE requested for the classification task is the one obtained when comparing the target output (0 for the wrong classes and 1 for the correct ones) against the predicted probabilities by the model (this is known as Brier score[10]).

[7]Check `http://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring` to seek more information

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

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

[10]`https://en.wikipedia.org/wiki/Brier_score`

Figura 3: Subset of letters belonging to the test dataset.

At least, the following configurations should be tried:

- *Network architecture*:
  - For all the datasets, consider a number hidden neurons (n_1) equal to the $5\,\%$, $15\,\%$, $25\,\%$ and $50\,\%$ of the total number of patterns of the dataset. In this stage, for classification problems use L1 regularisation and $\eta = 10^{-5}$.
- For the classification problems, once decided the best architecture, try the following values for η: $\eta = 1$, $\eta = 0{,}1$, $\eta = 0{,}01$, $\eta = 0{,}001, \ldots, \eta = 10^{-10}$, along with the two types of regularisation (L2 y L1). What is happening? Compute the difference in number of coefficients for divorce and noMNIST dataset when the regularisation type is modified (L2 vs L1)[11].
- For both, regression and classification problems, compare the results obtained using the initialisation proposed for the `sklearn.cluster.KMeans` algorithm (using both the best architecture and the configuration for the logistic regression) according to the `'k-means++'` initialisation.
- Finally, for any of the classification problems, run the script considering the problem as if it was regression (i.e. the classification parameter is False and compute the CCR rounding the predictions to the closest integer). What is happening for this situation?

As a guideline, the training and generalisation errors achieved by a linear regression (using Weka) over the three regression datasets is shown:

- *sin dataset*: $MSE_{\text{train}} = 0{,}02968729$; $MSE_{\text{test}} = 0{,}03636649$.
- *Quake dataset*: $MSE_{\text{train}} = 0{,}03020644$; $MSE_{\text{test}} = 0{,}02732409$.
- *parkinsons dataset*: $MSE_{\text{train}} = 0{,}043390$; $MSE_{\text{test}} = 0{,}046354$.

Also, the training CCR and the test CCR achieved by a logistic regression (using Weka) over the two classification datasets is shown:

- *divorce dataset*: $CCR_{\text{entrenamiento}} = 90{,}5512\,\%$; $CCR_{\text{test}} = 90{,}6977\,\%$.
- *noMNIST dataset*: $CCR_{\text{entrenamiento}} = 80{,}4444\,\%$; $CCR_{\text{test}} = 82{,}6667\,\%$.

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

3.1. File format

The files containing the datasets will be CSV, in such a way that the values will be separated by commas. In this sense, there are no headers. In order to read the files properly, the function `read_csv` from `pandas` should be used.

[11]The coefficients are in the `coef_` attribute of the logistic regression object. Consider that if the absolute value of a coefficient is lower than 10^{-5}, then the coefficient is null

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 steps for the RBF training stage (**1 page 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. Analysing the errors, **including the images of some letters for which the model mistakes**, to visually check if they are confusing. Comparison between the confusion matrix obtained for this assignment against the one obtained in the previous lab assignment.
    - Computational time needed for the training step for `noMNIST` dataset and comparison against the computational time spent in the previous lab assignment.
- 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 script developed should receive the following command-line arguments[12].

- Argument `-t`, `--train_file`: 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`, `--test_file`: 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.

[12]To process the input sequence, the `click` library will be used.

- Argument `-c`, `--classification`: Boolean that indicates whether it is a classification problem. If it is not specified, we will suppose that it is a regression problem.
- Argument `-r`, `--ratio_rbf`: Indicates the radium (by one) of RBF neurons with respect to the total number of patterns in training. If not specified, use $0{,}1$.
- Argument `-l`, `--l2`: Boolean that indicated if L2 regularisation is used, instead of L1. If it is not specified, L1 will be used.
- Argument `-e`, `--eta`: Indicates the value for the *eta* (η) parameter. By default, use $\eta = 1e-2$.
- Argument `-o`, `--outputs`: Indicates the number of output columns of the dataset (always placed at the end). By default, use $o = 1$.
- (Kaggle) Argument `-p`, `--pred`: Boolean that indicates if the prediction mode is used.
- (Kaggle) Argument `-m`, `--model_file`: Indicates the directory in which the trained models are saved (in the training mode, without the flag p) or the file containing the model that will be used (in the prediction mode, with the flag p).
- Argument `--help`: It shows the help of the program (use the one automatically generated by the `click` library)

An example of execution can be seen in the following output[13]:

```
i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py --help
Usage: rbf.py [OPTIONS]

  5 executions of RBFNN training

  RBF neural network based on hybrid supervised/unsupervised training. We
  run 5 executions with different seeds.

Options:
  -t, --train_file TEXT  Name of the file with training data.
  -T, --test_file TEXT   Name of the file with test data.  [required]
  -c, --classification   The problem considered is a classification problem.
                         [default: False]
  -r, --ratio_rbf FLOAT  Ratio of RBF neurons (as a fraction of 1) with
                         respect to the total number of patterns.  [default:
                         0.1]
  -l, --l2               Use L2 regularization instead of L1 (logistic
                         regression).  [default: False]
  -e, --eta FLOAT        Value of the regularization parameter for logistic
                         regression.  [default: 0.01]
  -o, --outputs INTEGER  Number of columns that will be used as target
                         variables (all at the end).  [default: 1]
  -p, --pred             Use the prediction mode.  [default: False]
  -m, --model TEXT       Directory to save the model (or name of the
                         file to load the model, if the prediction mode is
                         active).
  --help                 Show this message and exit.



i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -t ./csv/train_divorce.csv -T ./csv/
    test_divorce.csv -c -r 0.15 -e 0.001 --l2
```

[13]To make the developed code to work in the UCO machines, the packages `click` and the last version of the package `scikit-learn` should be installed, using the following commands:
`pip install scikit-learn --user --upgrade`
`pip install click --user --upgrade`

```
-----------
Seed: 1
-----------
Number of RBFs used: 19
Training MSE: 0.000112
Test MSE: 0.001935
Training CCR: 100.00 %
Test CCR: 100.00 %
-----------
Seed: 2
-----------
Number of RBFs used: 19
Training MSE: 0.000099
Test MSE: 0.018344
Training CCR: 100.00 %
Test CCR: 97.67 %
-----------
Seed: 3
-----------
Number of RBFs used: 19
Training MSE: 0.000135
Test MSE: 0.007341
Training CCR: 100.00 %
Test CCR: 97.67 %
-----------
Seed: 4
-----------
Number of RBFs used: 19
Training MSE: 0.000130
Test MSE: 0.017479
Training CCR: 100.00 %
Test CCR: 97.67 %
-----------
Seed: 5
-----------
Number of RBFs used: 19
Training MSE: 0.000352
Test MSE: 0.011236
Training CCR: 100.00 %
Test CCR: 97.67 %
******************
Summary of results
******************
Training MSE: 0.000166 +- 0.000094
Test MSE: 0.011267 +- 0.006183
Training CCR: 100.00 % +- 0.00 %
Test CCR: 98.14 % +- 0.93 %

# In the following examples, CCRs are 0 because is a regression problem
i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -t ./csv/train_parkinsons.csv -T ./csv/
    test_parkinsons.csv -r 0.5 -o 2
-----------
Seed: 1
-----------
Number of RBFs used: 2203
Training MSE: 0.005435
Test MSE: 0.061848
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 2
-----------
Number of RBFs used: 2203
Training MSE: 0.005209
Test MSE: 0.055629
Training CCR: 0.00 %
Test CCR: 0.00 %
```

```
-----------
Seed: 3
-----------
Number of RBFs used: 2203
Training MSE: 0.005230
Test MSE: 0.051494
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 4
-----------
Number of RBFs used: 2203
Training MSE: 0.005305
Test MSE: 0.060224
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 5
-----------
Number of RBFs used: 2203
Training MSE: 0.005250
Test MSE: 0.051680
Training CCR: 0.00 %
Test CCR: 0.00 %
******************
Summary of results
******************
Training MSE: 0.005286 +- 0.000081
Test MSE: 0.056175 +- 0.004266
Training CCR: 0.00 % +- 0.00 %
Test CCR: 0.00 % +- 0.00 %

i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -t ./csv/train_parkinsons.csv -T ./csv/
    test_parkinsons.csv -r 0.15 -o 2
-----------
Seed: 1
-----------
Number of RBFs used: 660
Training MSE: 0.013441
Test MSE: 0.019442
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 2
-----------
Number of RBFs used: 660
Training MSE: 0.014156
Test MSE: 0.019407
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 3
-----------
Number of RBFs used: 660
Training MSE: 0.014024
Test MSE: 0.020129
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 4
-----------
Number of RBFs used: 660
Training MSE: 0.014096
Test MSE: 0.019187
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
```

```
Seed: 5
-----------
Number of RBFs used: 660
Training MSE: 0.014192
Test MSE: 0.020314
Training CCR: 0.00 %
Test CCR: 0.00 %
******************
Summary of results
******************
Training MSE: 0.013982 +- 0.000276
Test MSE: 0.019696 +- 0.000442
Training CCR: 0.00 % +- 0.00 %
Test CCR: 0.00 % +- 0.00 %

i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -t ./csv/train_sin.csv -T ./csv/test_sin.
    csv -r 0.15 -o 1
-----------
Seed: 1
-----------
Number of RBFs used: 18
Training MSE: 0.012100
Test MSE: 0.104196
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 2
-----------
Number of RBFs used: 18
Training MSE: 0.011401
Test MSE: 0.200121
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 3
-----------
Number of RBFs used: 18
Training MSE: 0.011954
Test MSE: 0.102267
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 4
-----------
Number of RBFs used: 18
Training MSE: 0.012082
Test MSE: 0.083309
Training CCR: 0.00 %
Test CCR: 0.00 %
-----------
Seed: 5
-----------
Number of RBFs used: 18
Training MSE: 0.011961
Test MSE: 0.092522
Training CCR: 0.00 %
Test CCR: 0.00 %
******************
Summary of results
******************
Training MSE: 0.011899 +- 0.000257
Test MSE: 0.116483 +- 0.042481
Training CCR: 0.00 % +- 0.00 %
Test CCR: 0.00 % +- 0.00 %

# Here we are running classification as is it was regression
i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -t ./csv/train_divorce.csv -T ./csv/
```

```
    test_divorce.csv -r 0.15
-----------
Seed: 1
-----------
Number of RBFs used: 19
Training MSE: 0.016020
Test MSE: 0.020228
Training CCR: 97.64%
Test CCR: 97.67%
-----------
Seed: 2
-----------
Number of RBFs used: 19
Training MSE: 0.014577
Test MSE: 0.020006
Training CCR: 98.43%
Test CCR: 97.67%
-----------
Seed: 3
-----------
Number of RBFs used: 19
Training MSE: 0.014949
Test MSE: 0.018446
Training CCR: 98.43%
Test CCR: 97.67%
-----------
Seed: 4
-----------
Number of RBFs used: 19
Training MSE: 0.012619
Test MSE: 0.021317
Training CCR: 98.43%
Test CCR: 97.67%
-----------
Seed: 5
-----------
Number of RBFs used: 19
Training MSE: 0.016418
Test MSE: 0.021326
Training CCR: 97.64%
Test CCR: 97.67%
******************
Summary of results
******************
Training MSE: 0.014917 +- 0.001332
Test MSE: 0.020265 +- 0.001059
Training CCR: 98.11% +- 0.39%
Test CCR: 97.67% +- 0.00%
```

4.3. [OPTIONAL] Save the model to a file.

During the training stage, the script can save the model trained as a pickle[14]. This will allow to use the trained model to predict the outputs of the **Kaggle** dataset.

To save the model, it is necessary to use the `-m` parameter. An execution example is as follows:

```
i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -t train.csv -T test.csv -l -c -r 0.01 -m
    model
-----------
Seed: 1
-----------
Number of RBFs used: 118
Training MSE: 0.152570
Test MSE: 0.155294
```

[14]https://docs.python.org/3/library/pickle.html

```
Training CCR: 31.97 %
Test CCR: 28.87 %
-----------
Seed: 2
-----------
Number of RBFs used: 118
Training MSE: 0.152697
Test MSE: 0.155242
Training CCR: 31.70 %
Test CCR: 28.21 %
-----------
Seed: 3
-----------
Number of RBFs used: 118
Training MSE: 0.152596
Test MSE: 0.155267
Training CCR: 31.88 %
Test CCR: 28.58 %
-----------
Seed: 4
-----------
Number of RBFs used: 118
Training MSE: 0.152599
Test MSE: 0.155124
Training CCR: 31.87 %
Test CCR: 28.79 %
-----------
Seed: 5
-----------
Number of RBFs used: 118
Training MSE: 0.152681
Test MSE: 0.155183
Training CCR: 31.51 %
Test CCR: 28.78 %
******************
Summary of results
******************
Training MSE: 0.152629 +- 0.000051
Test MSE: 0.155222 +- 0.000061
Training CCR: 31.78 % +- 0.16 %
Test CCR: 28.65 % +- 0.24 %
```

Once the execution is finished, there will be a folder named "model" containing 5 pickles. Each one corresponds with the generated model for each seed. In order to obtain the predictions, one of these 5 pickles should be chosen.

```
i02gupep@NEWTS:~/imc/workspace/la3$ ls model/
 1.pickle  2.pickle  3.pickle  4.pickle  5.pickle
```

4.4. [OPTIONAL] Obtaining the predictions for Kaggle.

Once the model is saved to a pickle, it is possible to obtain the output predictions for the Kaggle dataset. For this, `-m` and `-p` parameters should be used. Below is an example:

```
i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -T kaggle.csv -p -m model/2.pickle
Id,Category
0,4
1,4
2,3
3,4
4,4
5,1
6,3
7,4
```

```
8,0


 ...

13859,0
13860,4
13861,2
13862,0
13863,3
13864,3
13865,0
13866,2
13867,3
13868,3
13869,0
13870,0
13871,1
13872,4
13873,4
13874,3
13875,4
```

The output can be redirected to a `csv` file:

```
i02gupep@NEWTS:~/imc/workspace/la3$ ./rbf.py -T kaggle.csv -p -m modelo/2.pickle >
    submission.csv
```

This file is ready to be uploaded to Kaggle.