Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Especialidad: Computación
Cuarto curso. Primer cuatrimestre

Introduction to computational models.

Lab Assignment 3: Radial basis functions neural networks.

Javier Herrero Porras
i72hepoj@uco.es

Academic Course 2020-2021
Córdoba, November 22, 2020

Contents

List of Figures

List of Tables

1 Radial Basis Function

In this practice I have worked with a new type of neuronal network, based on Radial Basis Functions.

Basically, the function that implements RBF decides the output by computing the distance between the input vector (the pattern) and the vector stored in it (centres). The distance to consider if RBF is activated or not is considered the radius. If it is very small, the activation only occurs if the pattern is very close, and vice versa.

A Radial Basis Function Neural Network (RBFNN) is trained by a clustering (unsupervised part) and a logistic regression or matrix inversion (supervised part):

- Clustering is used to adjust centres in the network, and locate one RBF in each cluster detected for training patterns. In our case, we are using K-means to establish centres. Its important to know that results of clustering depends on the initialisation of the centroids so, in classification we are going to select in a stratified manner, and for regression we are going to select them in a random way.
- The next step is adjust the radius for each RBF. To do this, we are going to take half of the average distance to the rest of centroids.
- Last step for training is adjust the output layer weights. Depending if we are considering a classification or regression problem, we will apply logistic regression or Moore Penrose pseudo-inverse matrix, to obtain the coefficients of weights of the output layer.

2 Datasets

To make the experiments, 5 different datasets have been used to train and test the neural network. They are divided into regression datasets:

- *Sin-function dataset*, where training dataset consists of 120 patterns with 1 input and 1 output. Also, test dataset has 41 patterns.
- *Quake dataset*, where training dataset consists of 1633 patterns with 3 inputs and 1 output. Also, test dataset has 546 patterns.

- *Parkinson dataset*, where training dataset consists of 4406 patterns with 19 inputs and 2 outputs. Also, test dataset has 1469 patterns.

And also classification datasets:

- *Divorce dataset*, where training dataset consists of 127 patterns with 54 inputs and 1 output. Also, test dataset has 43 patterns.
- *Parkinson dataset*, where training dataset consists of 900 patterns with 784 inputs and 6 outputs. Also, test dataset has 300 patterns.

3 Parameters

In order to make the experiments analysed in this report, these values have been considered:

- *Ratio rbf*, to indicate the radium of RBF neurons with respect the total number of patterns in training: $\{0.05; 0.15; 0.25; 0.50\}$.
- *Eta*, in classification problems, which is the value of the regularisation factor for logistic regression. If it is a small value then the regularisation will be strong: $\{1; 0.1; 0.01; ...\ 10e\text{-}10\}$. By default its value is 0.01.
- *Regularisation used*, which is a parameter used to decide if the neural network uses L1 or L2 regularisation.

4 Experiments and result discussion

4.1 First experiment

In this section, I have to change the network architecture. As we know, an RBFNN has a specific architecture, composed by an input layer, a hidden layer with RBF neurons and an output layer. So in this experiment I can only change the number of neurons of the hidden layer (which is set by the parameter *Ratio rbf*, with the values specified in the previous section). Thus, the result obtained for classification datasets are shown in Table 1.

Dataset	Hidden Neurons	MSE$_{train}$	MSE STD$_{train}$	MSE$_{test}$	MSE STD$_{test}$
Divorce	**0.05**	**0.000001**	**0.000001**	**0.000005**	**0.000006**
Divorce	0.15	0.000001	0	0.000012	0.00001
Divorce	0.25	0.000001	0.000001	0.003806	0.004967
Divorce	0.50	0.000001	0	0.004097	0.005155
noMNIST	**0.05**	**0.029813**	**0.001446**	**0.029197**	**0.001336**
noMNIST	0.15	0.000269	0.000147	0.04334	0.004471
noMNIST	0.25	0.000033	0.000006	0.039814	0.004175
noMNIST	0.50	0.000019	0.000002	0.035101	0.001836

Table 1: MSE obtained in classification datasets

Also, it is important to measure the percentage of correct classified instances in classification problems. So, in table 2 we can see CCR obtained by the RBFNN for each classification dataset:

Dataset	Hidden Neurons	CCR$_{train}$	CCR STD$_{train}$	CCR$_{test}$	CCR STD$_{test}$
Divorce	**0.05**	**100**	**0**	**100**	**0**
Divorce	0.15	100	0	100	0
Divorce	0.25	100	0	99.53	0.93
Divorce	0.50	100	0	99.53	0.93
noMNIST	**0.05**	**88.11**	**0.5**	**88.67**	**0.87**
noMNIST	0.15	100	0	86.2	1.56
noMNIST	0.25	100	0	86.67	1.46
noMNIST	0.50	100	0	87.87	0.62

Table 2: CCR obtained in classification datasets

The following datasets I have worked with are regression ones. In the table 3 we can see the different results obtained. **NOTE:** In regression datasets, the CCR (correctly classified instances) is not taken into account, because its a measure only for classification problems.

Dataset	Hidden Neurons	MSE_{train}	MSE STD_{train}	MSE_{test}	MSE STD_{test}
SIN	**0.05**	**0.014034**	**0.000186**	**0.02251**	**0.00019**
SIN	0.15	0.01183	0.000172	0.112002	0.026099
SIN	0.25	0.01166	0.000322	0.143153	0.060406
SIN	0.50	0.011674	0.000243	0.152911	0.05686
QUAKE	**0.05**	**0.028462**	**0.000098**	**0.028229**	**0.00014**
QUAKE	0.15	0.026763	0.00017	0.032237	0.001837
QUAKE	0.25	0.026322	0.000118	0.034765	0.001114
QUAKE	0.50	0.026061	0.000111	0.036175	0.000973
SIN	0.05	0.021453	0.00025	0.024794	0.000289
SIN	**0.15**	**0.013979**	**0.000206**	**0.019781**	**0.000346**
SIN	0.25	0.010467	0.000056	0.020959	0.00144
SIN	0.50	0.005204	0.000079	0.057888	0.013506

Table 3: MSE obtained in classification datasets

If we see the previous tables, we can easily check that better results are obtained with a lower ratio of RBF neurons, because the ratio which gets the better results is generally 0.05, which indicates that the neural network is able to predict better with a lower number of neurons. This can be caused because of the problem complexity, as we saw in section 'Datasets'. Usually they contain a low number of training patterns and inputs, and the model is able to predict more correctly classified instances (CCR) in classification.

In table 2 we can see that divorce dataset is an easy problem to learn and predict, because the CCR Test measure is very high with all ratios considered. On the other hand, noMNIST dataset is more complex than divorce and we can validate that CCR results are close to 88-89%. This is a good example to verify the importance of having an appropriate ratio of hidden neurons.

Also in table 1 the MSE results of classification datasets are shown. For this practice, MSE is comparing the probabilities obtained with the model in a 1-Of-J coding (for example it is comparing a ideal pattern (1 0 0) with a predicted pattern (0.8 0.1 0.1)). In this table we can see that MSE for train dataset with a high ratio of hidden neurons is very low, caused by overfitting. Also we check that MSE for test dataset increases if we put a high ratio, so the model is learning very well the train dataset, but it is not able to generalise new patterns of test dataset.

In the next section we will use the concept of regularization in classification problems, that avoids overfitting in the model.

4.2 Second experiment

In the second experiment I have to work with classification datasets (divorce and noMNIST) in order to check the importance of value eta, which controls the regularization (L2 and L1) in logistic regression, which is a model that approximates the probability of belonging to a class by a softmax function. Results are shown in table 4.

Dataset	Eta	Regularisation	MSE_{train}	MSE STD_{train}	MSE_{test}	MSE STD_{test}
DIVORCE	1	L2	0.021194	0.00081	0.023025	0.000376
DIVORCE	0.1	L2	0.015424	0.001866	0.020498	0.000534
DIVORCE	0.01	L2	0.00778	0.00136	0.017003	0.001241
DIVORCE	0.001	L2	0.001432	0.000591	0.005938	0.001706
DIVORCE	0.0001	L2	0.000086	0.000049	0.000845	0.001182
DIVORCE	0.00001	L2	0.000003	0.000002	0.000204	0.000382
DIVORCE	0.000001	L2	0	0	0.00005	0.0001
DIVORCE	0.0000001	L2	0	0	0.00005	0.0001
DIVORCE	0.00000001	L2	0	0	0.000049	0.000098
DIVORCE	0.000000001	L2	0	0	0.000049	0.000098
DIVORCE	1	L1	0.019916	0.000735	0.021779	0.000448
DIVORCE	0.1	L1	0.005616	0.00119	0.014129	0.001708
DIVORCE	0.01	L1	0.000223	0.000122	0.000951	0.001361
DIVORCE	0.001	L1	0.000007	0.000006	0.00011	0.000199
DIVORCE	0.0001	L1	0.000001	0.000001	0.000004	0.000006
DIVORCE	0.00001	L1	0.000001	0.000001	0.000005	0.000006
DIVORCE	0.000001	L1	0.000001	0.000001	0.000004	0.000006
DIVORCE	0.0000001	L1	0.000001	0.000001	0.000004	0.000006
DIVORCE	0.00000001	L1	0.000001	0.000001	0.000004	0.000006
DIVORCE	0.000000001	L1	0.000001	0.000001	0.000004	0.000006

Table 4: MSE obtained in Divorce dataset

If we take a quick look at the table 4 we can review that if we give more importance to regularization (both L1 or L2) we get better results in terms of MSE. This means that probabilities given by the model are very close to the ideal ones. As we said in the previous experiment, this is an easy model, so the results obtained are very good. Also it is important to validate the effect of the regularization in CCR measure. After making the experiments for CCR, we can predict that CCR will be near 100%.For this dataset, results are shown in table 5:

Dataset	Eta	Regularisation	CCR_{train}	CCR STD_{train}	CCR_{test}	CCR STD_{test}
DIVORCE	1	L2	97.64	0	97.67	0
DIVORCE	0.1	L2	97.8	0.31	97.67	0
DIVORCE	0.01	L2	98.58	0.31	97.67	0
DIVORCE	0.001	L2	99.84	0.31	98.6	1.14
DIVORCE	0.0001	L2	100	0	100	0
DIVORCE	0.00001	L2	100	0	100	0
DIVORCE	0.000001	L2	100	0	100	0
DIVORCE	0.0000001	L2	100	0	100	0
DIVORCE	0.00000001	L2	100	0	100	0
DIVORCE	0.000000001	L2	100	0	100	0
DIVORCE	1	L1	97.64	0	97.67	0
DIVORCE	0.1	L1	99.21	0	97.67	0
DIVORCE	0.01	L1	100	0	100	0
DIVORCE	0.001	L1	100	0	100	0
DIVORCE	0.0001	L1	100	0	100	0
DIVORCE	0.00001	L1	100	0	100	0
DIVORCE	0.000001	L1	100	0	100	0
DIVORCE	0.0000001	L1	100	0	100	0
DIVORCE	0.00000001	L1	100	0	100	0
DIVORCE	0.000000001	L1	100	0	100	0

Table 5: CCR obtained in Divorce dataset

It is important to know the meaning of regularization in the logistic regression model. Regularization does not improve the performance on training data, but is used to train our model to make it able to generalise better on the test data, by preventing the algorithm from overfitting the training dataset [1].

In this practice, I have worked with L1 (or Lasso) and L2 (or Ridge), which allow us to reduce the coefficients of the model, because the changes made on the model are less sensitive than if we do not use it, and it makes the model avoiding the overfitting. This can be caused because in training data classes are not balanced and the model learn more from the common ones. If we try to generalise with all the classes of the problem, we will get poor results in the uncommon classes. Another reason that causes overfitting is that, in our problem we have lots of coefficients to estimate and the training data is very small, our model will tend to learn too much the coefficients for the data given.

A way to control this is the concept of regularization, which is a penalty that depends on the magnitude of all the coefficients. It is controlled by the parameter *eta*, which is a tuning parameter and it is set by the user. In this experiment we are going to check how it affects the results.

- L2 regularization:

$$L = \left(\frac{1}{N}\sum_{p=1}^{N}\left(\frac{1}{k}\sum_{o=1}^{k} d_j \ln(o_j)\right)\right) - \eta\left(\sum_{j=1}^{k}\sum_{i=0}^{n}\beta_{ji}^2\right) \quad (6)$$

- L1 regularization:

$$L = \left(\frac{1}{N}\sum_{p=1}^{N}\left(\frac{1}{k}\sum_{o=1}^{k} d_j \ln(o_j)\right)\right) - \eta\left(\sum_{j=1}^{k}\sum_{i=0}^{n}|\beta_{ji}|\right) \quad (7)$$

Figure 1: L1 & L2 regularization

In figure 1 [2] we see how we are going to apply regularization on the logistic model. The main difference between the two types of regularization is that L1 tends to reduce less important feature's coefficient to zero, so we consider that L1 removes some features. We will consider this idea later.

Also, in tables 6 and 7 we can check the results obtained when we apply L1 and L2 regularization for the best architecture obtained in the first experiment (ratio of hidden neurons = 0.05).

Dataset	Eta	Regularisation	MSE$_{train}$	MSE STD$_{train}$	MSE$_{test}$	MSE STD$_{test}$
noMNIST	1	L2	0.067368	0.001356	0.0662	0.001607
noMNIST	0.1	L2	0.045785	0.001033	0.043954	0.001592
noMNIST	0.01	L2	0.035546	0.001102	0.032803	0.001457
noMNIST	0.001	L2	0.031208	0.001264	0.029188	0.00151
noMNIST	0.0001	L2	0.029998	0.001397	0.029075	0.001369
noMNIST	0.00001	L2	0.029811	0.001443	0.029207	0.001323
noMNIST	0.000001	L2	0.02979	0.001448	0.029228	0.001321
noMNIST	0.0000001	L2	0.029786	0.001449	0.029232	0.001322
noMNIST	0.00000001	L2	0.029784	0.001448	0.029231	0.001325
noMNIST	0.000000001	L2	0.029784	0.001448	0.029234	0.001317
noMNIST	1	L1	0.054876	0.001214	0.052103	0.001606
noMNIST	0.1	L1	0.03487	0.001265	0.03185	0.001664
noMNIST	0.01	L1	0.030459	0.001369	0.02896	0.001364
noMNIST	0.001	L1	0.029874	0.001434	0.029147	0.00134
noMNIST	0.0001	L1	0.02982	0.001445	0.029192	0.001335
noMNIST	0.00001	L1	0.029813	0.001446	0.029197	0.001336
noMNIST	0.000001	L1	0.029813	0.001446	0.029198	0.001336
noMNIST	0.0000001	L1	0.029813	0.001446	0.029198	0.001336
noMNIST	0.00000001	L1	0.029813	0.001446	0.029198	0.001336
noMNIST	0.000000001	L1	0.029813	0.001446	0.029198	0.001336

Table 6: MSE obtained in noMNIST dataset

Dataset	Eta	Regularisation	CCR_{train}	CCR STD_{train}	CCR_{test}	CCR STD_{test}
noMNIST	1	L2	77.71	0.54	77.47	1.6
noMNIST	0.1	L2	83.24	0.76	86.67	1.28
noMNIST	0.01	L2	86.4	0.72	88.73	0.9
noMNIST	0.001	L2	87.73	0.46	89.13	0.96
noMNIST	0.0001	L2	88.04	0.52	88.87	0.93
noMNIST	0.00001	L2	88.09	0.48	88.6	0.77
noMNIST	0.000001	L2	88.11	0.42	88.6	0.77
noMNIST	0.0000001	L2	88.11	0.42	88.6	0.77
noMNIST	0.00000001	L2	88.11	0.42	88.6	0.77
noMNIST	0.000000001	L2	88.11	0.42	88.6	0.77
noMNIST	1	L1	79.82	1.02	82.8	1.61
noMNIST	0.1	L1	86.56	0.71	88.73	0.71
noMNIST	0.01	L1	87.8	0.52	88.93	0.77
noMNIST	0.001	L1	88.04	0.52	88.73	0.9
noMNIST	0.0001	L1	88.09	0.54	88.67	0.87
noMNIST	0.00001	L1	88.11	0.5	88.67	0.87
noMNIST	0.000001	L1	88.11	0.5	88.67	0.87
noMNIST	0.0000001	L1	88.11	0.5	88.67	0.87
noMNIST	0.00000001	L1	88.11	0.5	88.67	0.87
noMNIST	0.000000001	L1	88.11	0.5	88.67	0.87

Table 7: CCR obtained in noMNIST dataset

The first conclusion we obtain if we take a look at the previous tables is that if parameter *Eta* decreases, the results obtained in terms of MSE are better (considering the test results). What is the cause of that fact?

If we read documentation of logistic regression model on Python, the parameter eta is "Inverse of regularization strength". So, if *Eta* is lower then the regularization is stronger and test MSE and CCR are lower too. On one hand, if we consider Divorce results in figure 2 (expressed at logarithm scale), we can see that L1 gets better results and also this method has a higher variation than L2 (the improvement of Test MSE is very high between 0.1 and 0.01).

Its also important to say that with a low value of *Eta* (means that regularization is strong in the model) the differences are very small, because the model is stable and there are not differences in both results.

Why is this happening? As we said before, L1 tends to remove the least significant features of the model so, for this dataset we can conclude that removing some features has positive effects in the model, because it is simpler than L2 model and the results obtained are better. Also, we can check this by the results in CCR and the conclusions we can get for the divorce dataset are the same.

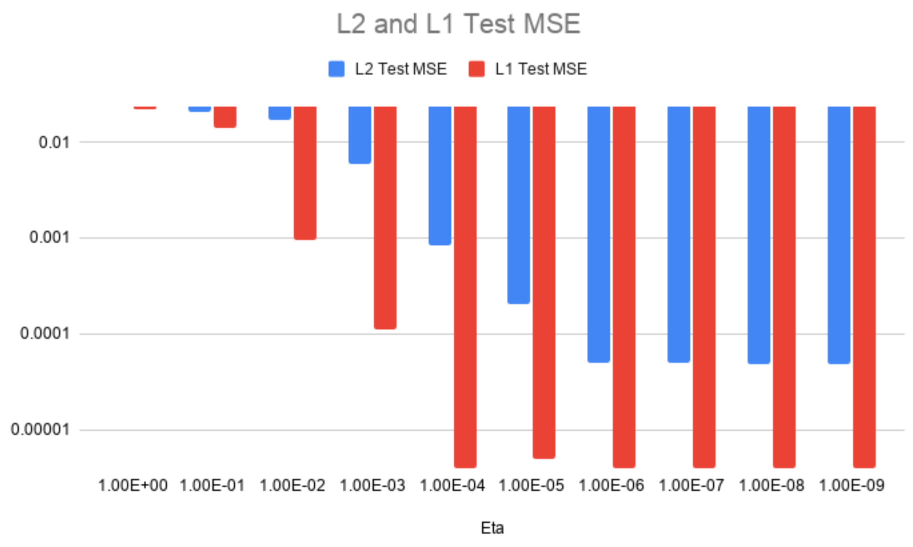

Figure 2: L1 & L2 regularization MSE test Divorce

This fact can also be seen by considering the difference in number of coefficients of the logistic regression instance. In table 8 we can validate this idea, where we can see that L1 usually considers a lower number of coefficients (greater than 10^{-5}) with high values of *Eta*. When values are lower, as we said before, there are not significant differences between both types of regularization.

Eta	Significant coefficients L1	Significant coefficients L2
$10e0$	3	7
$10e-1$	4	7
$10e-2$	4	7
$10e-3$	6	7
$10e-4$	7	7
$10e-5$	7	7
$10e-6$	7	7
$10e-7$	7	7
$10e-8$	7	7
$10e-9$	7	7

Table 8: Significant coefficients Divorce dataset

If we make the analysis for noMNIST dataset, the conclusions obtained are the same. Note that this dataset is not as simple as divorce dataset. It has more input variables and a higher number of patterns so the complexity of the model will be higher, and the predictions will be not so accurate as the ones obtained in divorce dataset, as we can check in tables 6 and 7.

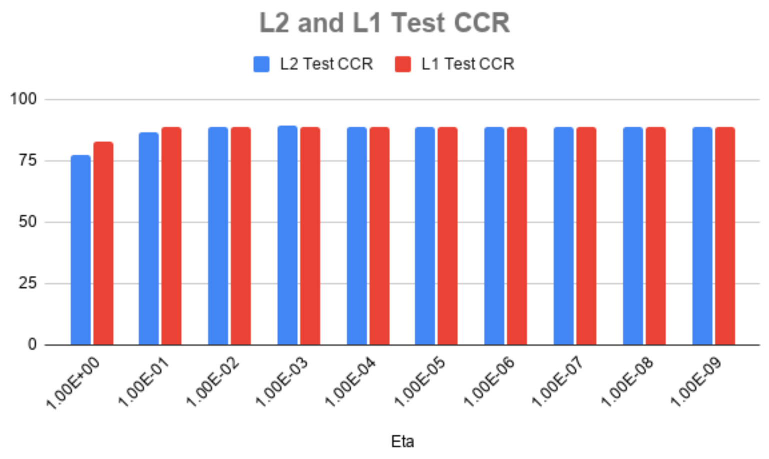

Figure 3: L1 and L2 regularization CCR test noMNIST

For example, if we consider the results obtained for Test CCR with both types of regularization (L1 and L2) in the picture 3, we can conclude that differences are not very significant between L1 and L2. But in fact, the best CCR was obtained with L2 regularization (89.13% with $Eta = 10^{-3}$). This lead us to conclude that for this dataset is better with L2 regularization, because the model obtained is more complex and removing some features (like L1) is not good for the model performance. We can also see this idea in table 11.

Also, in the previous image I noticed that the more regularization strength does not implies the best results, because the best results were obtained with a relatively low of *Eta* (0.001). Then, the results obtained when giving more importance to regularization are worse but they are stabilised. So the parameter *Eta* is essential to get a model which is able to generalise new patterns in an appropriate way.

Eta	Significant coefficients L1	Significant coefficients L2
$10e0$	110	276
$10e-1$	190	276
$10e-2$	256	276
$10e-3$	275	276
$10e-4$	276	276
$10e-5$	276	276
$10e-6$	276	276
$10e-7$	276	276
$10e-8$	276	276
$10e-9$	276	276

Table 9: Significant coefficients noMNIST dataset

In the previous table I can verify the ideas I declared for Divorce dataset. At the beginning L1 takes less features of the problem, but it tends to take the same features as L2 when regularization becomes stronger in the logistic regression model, and also the final differences between both CCR results are minimal, so we conclude that regularization is very important in our datasets, because it prevents overfitting and it leads our models to generalise new patterns correctly. On the other hand, is very important to set an appropriate value of the regularization strength because we can even get worse results, although it tends to stabilise when *Eta* is very low.

4.3 Third experiment

In this experiment I have to change the configuration of KMeans. It is a method used to adjust the centres in the network, by detecting the group of patterns in the input space and locate one in each clusters. For classification, centroids are initialised in a random and stratified way, and for regression it was randomly.

Method *sklearn.cluster.KMeans* has a parameter to indicate the method for centroids initialization. In this case I will use *k-means++*, which selects initial cluster centres in a smart way to speed up convergence (optimizes the step where we randomly select centroids).

It is important to say that this method has a higher relative computational cost than random initialization, but the convergence is reached more quickly.

The steps K-Means++ follows are[3]:

- The first cluster is chosen uniformly from the data points.
- Then, it computes the distance from the first cluster (D(x)) to each data point (x).
- Next, it chooses a new cluster with a probability of x being proportional to $D(x)^2$. That means that we take the farthest from the initial centroid.
- Repeat steps 2 and 3 until k clusters have been chosen.

The final results (with best configuration and architecture achieved in the previous sections) obtained in this experiment are shown in table 10.

Dataset	MSE_{test}	MSE STD_{test}	CCR_{train}	CCR STD_{train}
Divorce	0.000001	0.000001	100	0
noMNIST	0.029026	0.000886	89.53	0.62
Sin	0.022296	0.00051	–	–
Parkinsons	0.025201	0.000354	–	–
Quake	0.028311	0.000214	–	–

Table 10: Results K-Means++

The main advantage of this method is that despite taking more time in initial selection of clusters but the convergence is achieved faster. In the previous table we can see that, in classification datasets there is an improvement based on CCR measures. On the other hand, in regression datasets the improvement is not as high as classification.

4.4 Fourth experiment

The last experiment consists of running our program with a classification dataset but considering it as a regression one. In this case I have rounding the predictions to the closest integer. For example, with *Divorce dataset* I have obtained the following results:

Ratio RBF	MSE$_{train}$	MSE$_{test}$	CCR$_{train}$	CCR$_{test}$
0.05	0.023622	0.023256	0.98	0.98
0.15	0.012598	0.023256	0.99	0.98
0.25	0.004724	0.023256	1.00	0.98
0.50	0.0	0.023256	1.00	0.98

Table 11: Significant coefficients noMNIST dataset

In the previous table we can check that model predictions are very accurate in this way, because we are omitting decimal part of the probability predictions so the final predictions are the same that the desired ones. Also as we said in previous sections, if we increase the ratio of RBF then the model will tend to overfit training data. With this configuration the MSE obtained for train data is 0, so this model will predict perfectly the train data, but keeping the accuracy on the test data.

4.5 Optional experiment 1

In this optional experiment, I will make the confusion matrix of the best neural network model obtained during the previous sections. So, the best model has the following features:

- Ratio RBF : 0.05
- Eta: 0.001
- Regularization: L2
- Centroid initialization: k-means++

This model has obtained a test CCR = 89.53%. For this model I will obtain the test confusion matrix. An example of one confusion matrix obtained during the execution of this model is shown in table 12.

	A	B	C	D	E	F
A	45	1	1	0	0	3
B	0	47	0	2	1	0
C	1	0	46	1	1	1
D	3	3	0	43	0	1
E	0	2	2	0	45	1
F	2	1	0	0	2	45

Table 12: Test Confusion matrix noMNIST

Also in table 13 we can check the results obtained in the test confusion matrix for the previous lab assignment (MLP).

	A	B	C	D	E	F
A	43	2	2	3	0	0
B	4	40	1	2	2	1
C	1	1	45	0	2	1
D	1	2	2	43	1	1
E	0	4	3	0	40	3
F	1	3	1	1	3	41

Table 13: Test Confusion matrix noMNIST in MLP

If we compare the two previous tables we notice that for MLP the results were worse than RBFNN because the CCR obtained with MLP was 85% and for RBFNN is 89%. For example, the best class for MLP was C (CCR = 90%), but in RBFNN the CCR obtained for class C is 92%. The differences between both classifiers can be checked in previous tables. We conclude that RBFNN works better for noMNIST dataset, because it classifies better the instances in their desired classes.

To complete this experiment, I will extract and analyse some errors in predictions made by RBFNN for noMNIST dataset. For example, I will show 15 errors made on the test prediction:

- Pattern 16 was classified in 1 when was 0 class.
- Pattern 28 was classified in 2 when was 0 class.
- Pattern 29 was classified in 1 when was 0 class.
- Pattern 41 was classified in 5 when was 0 class.

- Pattern 69 was classified in 3 when was 1 class.
- Pattern 84 was classified in 4 when was 1 class.
- Pattern 91 was classified in 3 when was 1 class.
- Pattern 99 was classified in 3 when was 1 class.
- Pattern 140 was classified in 5 when was 2 class.
- Pattern 143 was classified in 4 when was 2 class.
- Pattern 177 was classified in 5 when was 3 class.
- Pattern 210 was classified in 1 when was 4 class.
- Pattern 233 was classified in 1 when was 4 class.
- Pattern 261 was classified in 0 when was 5 class.
- Pattern 293 was classified in 0 when was 5 class.

In the figure 4 we can see the image of this patterns to visually check if they are confusing. In general letters wrote with very thick contour (like 28,41,69,99,140 and 293) or thin contour (177, 210 and 261) and letters with strange shape (29,84,91,143 and 261) are incorrectly classified, because the model tends to confuse some pair of letters (A with F, B with E...).

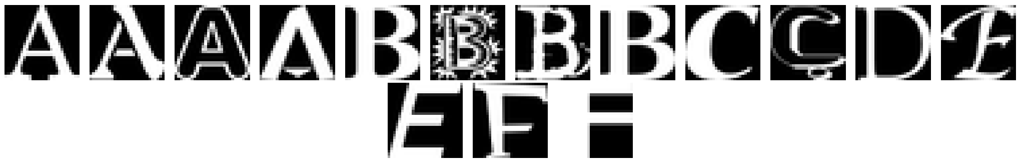

Figure 4: Incorrect Classified Patterns noMNIST

4.6 Optional experiment 2

In the second optional experiment, I have to compare the computational time needed for the training step for noMNIST dataset between RBF model and MLP model. Results for 6 different executions are shown in table 14.

NOTE: Every execution time is calculated as the mean of all seeds training times.

Execution	MLP time (s)	RBF time (s)
0	15.7983	0.52726
1	14.7769	0.86462
2	14.218	0.99778
3	15.6807	0.8559
4	15.4444	0.96168
5	15.8867	1.0854

Table 14: Significant coefficients noMNIST dataset

With these times measures we can easily check that MLP mean training time is 15.30 seconds while RBF mean training time is 0.8821 seconds, so RBF is 17 times faster than MLP in taking knowledge about a dataset and also, it has better results in prediction experiments. In resume, I can say after doing this experiments that RBF obtains better model for noMNIST problem, because the training step is faster and the results obtained are better.

References

[1] Regularization for Logistic Regression: L1, L2, Gauss or Laplace? URL: `https://www.knime.com/blog/regularization-for-logistic-regression-l1-l2-gauss-or-laplace#:~:text=Regularization%20can%20be%20used%20to%20avoid%20overfitting.&text=In%20other%20words%3A%20regularization%20can,to%20reduce%20the%20generalization%20error%3F`. [Online. Last consultation: 18-11-2020].

[2] Pedro Antonio Gutierrez. Radial basis function neural networks. URL: `https://moodle.uco.es/m2021/mod/resource/view.php?id=65297`. [Online. Last consultation: 19-11-2020].

[3] The Most Comprehensive Guide to K-Means Clustering You'll Ever Need. URL: `https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/`. [Online. Last consultation: 18-11-2020].