

# PRÁCTICA 3: REDES NEURONALES RBF

Carlos de la Barrera Pérez. DNI:31010594Q. Email UCO: i12bapec@uco.es

Escuela Politécnica Superior

Introducción a los  
modelos  
computacionales.  
Asignatura de  
4ºCurso

## INDICE

1.	Descripción de cómo se lleva a cabo el entrenamiento RBF .....	2
2.	Descripción de las bases de datos utilizadas.....	3
3.	Descripción de los parámetros considerados .....	4
4.	Experimentos realizados .....	4
4.1.	Primera prueba .....	4
4.2.	Segunda prueba .....	6
4.3.	Tercera prueba .....	9
4.4.	Cuarta prueba.....	10
5.	Comparativa de NOMNIST con la práctica anterior .....	10

## INDICE DE FIGURAS

Figura 1.	Tabla Seno prueba 1.....	4
Figura 2	Tabla Quake prueba 1 .....	4
Figura 3	Tabla Parkinsons prueba 1 .....	5
Figura 4	Tabla Vote prueba 1 .....	5
Figura 5	Tabla noMNIST prueba 1.....	6
Figura 6	Tabla noMNIST prueba 2.....	6
Figura 7	Tabla Vote prueba 2 .....	7
Figura 8	Código para el cálculo de coeficientes .....	7
Figura 9	Diferencia de coeficientes Vote .....	8
Figura 10	Diferencia de coeficientes noMNIST .....	8
Figura 11	Código para K-Means++ .....	9
Figura 12	Tabla K-Means++ prueba 3.....	9
Figura 13	Tabla clasificación con regresión prueba 4 .....	10
Figura 14	Matriz de confusión TEST con RBF .....	10
Figura 15	Matriz de confusión TEST con Perceptrón Multicapa .....	10
Figura 16	Mal clasificados con Perceptrón Multicapa .....	11

## 1. Descripción de cómo se lleva a cabo el entrenamiento RBF

Para esta práctica partiremos de un script de Python con funciones a completar. Esto se llevará a cabo con la librería externa Sklearn, en la cual encontraremos los modelos y algoritmos necesarios para llevar a cabo el entrenamiento. No obstante, han sido necesario el uso de otras librerías matemáticas como Numpy, estas nos han permitido realizar los cálculos referentes a matrices. Por último, se ha usado la librería Pandas para el tratamiento de los ficheros en los que se encuentran los conjuntos de datos.

El entrenamiento lleva a cabo un proceso de clústering que utiliza el algoritmo K-Means, que nos servirá para colocar los centroides en las mejores posiciones. Será necesario entonces determinar una posición inicial de estos centroides, así que los determinaremos mediante un patrón del conjunto de entrenamiento. Al final, cada centroide acabará representando una clase.

Con los centroides iniciales calculados, se procede a calcular los radios de cada uno de ellos, esto nos permite saber a qué clase pertenece cada instancia en función de cual sea su centroide más cercano. Para esta distancia, se utilizará la métrica de la distancia euclídea. Tras esto, obtendremos una matriz de distancias de donde obtendremos los radios con el siguiente cálculo:  $\text{sum(filas)/2 * num\_rbf}-1$ .

Con los radios listos, es momento de ajustar los pesos. Aquí hay que tomar distintos caminos ya que el problema puede ser de clasificación o regresión.

- Si el problema es de clasificación se utilizará una regresión logística.
- Si el problema es de regresión se calcula una matriz pseudo-inversa con el método de Moore-Penrose. De donde se obtendrán los pesos de la última capa.

Además, no es necesario implementar la transformación Softmax ni Lineal, ya que estas vienen incluidas en los algoritmos que utilizaremos.

## 2. Descripción de las bases de datos utilizadas

- **Seno:**
  - Patrones train: 120
  - Patrones test: 41
  - Se ha obtenido añadiendo ruido a la función seno.
- **Quake:**
  - Patrones train: 1633
  - Patrones test: 546
  - Se corresponde con una base de datos en la que el objetivo es averiguar la fuerza de un terremoto (medida en escala sismologica de Richter). Como variables de entrada, utilizamos la profundidad focal, la latitud en la que se produce y la longitud.
- **Parkinsons:**
  - Patrones train: 4406
  - Patrones test: 1469
  - Contiene, como entradas o variables independientes, una serie de datos clínicos de pacientes con la enfermedad de Parkinson y datos de medidas biométricas de la voz, y, como salidas o variables dependientes, el valor motor y total del UPDRS.
- **Vote:**
  - Patrones train: 326
  - Patrones test: 109
  - Clases: 2
  - La base de datos incluye los votos para cada uno de los para cada uno de los candidatos para el Congreso de los EEUU, identificados por la CQA. Todas las variables de entrada son categóricas.
- **noMNIST:**
  - Patrones train: 200.000
  - Patrones test: 10.000
  - Clases: 6
  - . Esta´ formada por un conjunto de letras (de la a a la f) escritas con diferentes tipografías o simbologías. Están ajustadas a una rejilla cuadrada de ´  $28 \times 28$  pixeles. Las imágenes están en escala de grises en el intervalo  $[-1,0; +1,0]$ . Cada uno de los pixeles forman parte de las variables de entrada (con un total de  $28 \times 28 = 784$  variables de entrada) y las clases se corresponden con la letra escrita

Mientras las bases de datos Vote y noMNIST son de clasificación, el resto son de regresión.

### 3. Descripción de los parámetros considerados

- **Fichero train:** Fichero que contiene el conjunto de datos de entrenamiento.
- **Fichero test:** Fichero que contiene el conjunto de datos de test.
- **Número de RBF:** Numero de neuroas RBF que tendrá el modelo.
- **Clasificación:** Parámetro booleano que activará o desactivará el modo de clasificación.
- **Eta:** Tasa de aprendizaje en el proceso de entrenamiento.
- **L2:** Regularización de la regresión logística. Por un lado, L2 proporcionará pesos más pequeños, mientras que L1 hace una mayor “poda” de las variables haciendo que muchos pesos sean iguales a cero, los que no son cero no tienen por qué ser pequeños en valor absoluto.

### 4. Experimentos realizados

#### 4.1. Primera prueba

Para todas las bases de datos, considerar un numero de neuronas en capa oculta ( $n_1$ ) igual al 5 %, 15 %, 25 % y 50 % del número de patrones de la base de datos. En esta fase, para problemas de clasificación, utilizar regularización L1 y un valor para el parámetro  $\eta = 10^5$

BASE DE DATOS SENO					
	NUM RBF	5	15	25	50
	MEDIA MSE TRAIN	0.013798	0.011893	0.011772	0.011472
	DESV MSE TRAIN	0.000063	0.000081	0.000349	0.000344
	MEDIA MSE TEST	0.022310	0.109616	0.124051	0.195235
	DESV MSE TEST	0.000196	0.015957	0.053192	0.072867
	MEDIA CCR TRAIN	80.33%	78.33%	79.50%	80.83%
	DESV CCR TRAIN	0.67%	0.00%	1.13%	1.05%
	MEDIA CCR TEST	78.05%	65.85%	67.80%	69.27%
	DESV CCR TEST	0.00%	0.00%	2.39%	1.95%

Figura 1. Tabla Seno prueba 1

BASE DE DATOS QUAKE					
	NUM RBF	5	15	25	50
	MEDIA MSE TRAIN	0.028349	0.026902	0.026321	0.026151
	DESV MSE TRAIN	0.000061	0.000145	0.000115	0.000023
	MEDIA MSE TEST	0.028323	0.032791	0.034935	0.035324
	DESV MSE TEST	0.000274	0.002280	0.000953	0.000819
	MEDIA CCR TRAIN	94.54%	94.73%	94.73%	94.73%
	DESV CCR TRAIN	0.02%	0.00%	0.00%	0.00%
	MEDIA CCR TEST	94.86%	94.42%	93.80%	93.94%
	DESV CCR TEST	0.00%	0.27%	0.36%	0.12%

Figura 2 Tabla Quake prueba 1

Estas bases de datos tienen pocos patrones (sobre todo la del Seno), con lo cual con 5 rbf basta para obtener un buen resultado. Ya que de subir el número de neuronas podemos estar incurriendo en un sobreentrenamiento no deseado. Esto lo podemos apreciar en el error de test. También se observa que la base de datos SENO puntúa bastante mal, con lo que quizás el modelo de red neuronal utilizado en esta práctica no sea el más adecuado para dicha base de datos.

BASE DE DATOS PARKINSONS					
	NUM RBF	5	15	25	50
	<b>MEDIA MSE TRAIN</b>	0.001656	0.000754	0.000408	0.000134
	<b>DESV MSE TRAIN</b>	0.000030	0.000017	0.000007	0.000004
	<b>MEDIA MSE TEST</b>	0.002018	0.001288	0.001045	0.001155
	<b>DESV MSE TEST</b>	0.000038	0.000069	0.000079	0.000108
	<b>MEDIA CCR TRAIN</b>	96.37%	97.50%	98.26%	98.93%
	<b>DESV CCR TRAIN</b>	0.21%	0.17%	0.09%	0.06%
	<b>MEDIA CCR TEST</b>	96.54%	97.06%	97.36%	97.49%
	<b>DESV CCR TEST</b>	0.20%	0.14%	0.34%	0.13%

Figura 3 Tabla Parkinsons prueba 1

Con esta base de datos tenemos una cantidad más significativa de patrones. Lo que indica que seguramente se necesiten más rbf para un modelo más ajustado. Conseguimos unos muy buenos resultados con 25 y 50 neuronas. Pero como el error de test es ligeramente menor con 25 neuronas, nos quedaremos con dicha cantidad para desempatar. Ya que están prácticamente igualadas con 25 y 50.

BASE DE DATOS VOTE					
	NUM RBF	5	15	25	50
	<b>MEDIA MSE TRAIN</b>	0.026196	0.004647	0.003793	0.003640
	<b>DESV MSE TRAIN</b>	0.003005	0.000514	0.000059	0.000007
	<b>MEDIA MSE TEST</b>	0.041162	0.063110	0.059041	0.058302
	<b>DESV MSE TEST</b>	0.008827	0.007491	0.004569	0.001275
	<b>MEDIA CCR TRAIN</b>	96.20%	99.39%	99.39%	99.39%
	<b>DESV CCR TRAIN</b>	0.31%	0.00%	0.00%	0.00%
	<b>MEDIA CCR TEST</b>	94.50%	92.48%	93.58%	93.58%
	<b>DESV CCR TEST</b>	1.92%	1.07%	0.82%	0.00%

Figura 4 Tabla Vote prueba 1

De nuevo nos encontramos ante una base de datos con pocos patrones. En esta ocasión se produce una situación similar a la comentada anteriormente. Y es que una vez pasadas las 5 neuronas, el error de test aumenta a la vez que lo hacen los CCR, lo que puede indicar sobreentrenamiento. Con lo cual nos quedaremos con 5 neuronas.

BASE DE DATOS NOMNIST					
	NUM RBF	5	15	25	50
	<b>MEDIA MSE TRAIN</b>	0.030243	0.005929	0.001480	0.000519
	<b>DESV MSE TRAIN</b>	0.001388	0.000516	0.000205	0.000047
	<b>MEDIA MSE TEST</b>	0.028968	0.033221	0.035078	0.031106
	<b>DESV MSE TEST</b>	0.001350	0.000989	0.004032	0.001084
	<b>MEDIA CCR TRAIN</b>	87.93%	98.91%	99.93%	100.00%
	<b>DESV CCR TRAIN</b>	0.58%	0.32%	0.05%	0.00%
	<b>MEDIA CCR TEST</b>	88.93%	88.20%	87.73%	88.73%
	<b>DESV CCR TEST</b>	0.93%	0.96%	1.83%	0.57%

Figura 5 Tabla noMNIST prueba 1

Esta es la base de datos más grande y encontramos el caso de que, con 50 neuronas el CCR de train es del 100% y con 25 neuronas casi del 100%. Esto podría ser otro síntoma de sobreentrenamiento. Además, con tal cantidad de neuronas el tiempo de cómputo sería demasiado alto. Teniendo en cuenta los resultados mostrados, elegiremos 15 neuronas en capa oculta.

#### 4.2. Segunda prueba

Para los problemas de clasificación, una vez decidida la mejor arquitectura, probar los siguientes valores para  $\eta$ :  $\eta = 1$ ,  $\eta = 0,1$ ,  $\eta = 0,01$ ,  $\eta = 0,001$ , . . . ,  $\eta = 10^{-10}$ , junto con los dos tipos de regularización (L2 y L1). ¿Qué sucede? Calcula la diferencia en número de coeficientes en vote y noMNIST cuando modificas el tipo de regularización (L2 Vs L1)

BASE DE DATOS NOMNIST con 15 num rbf												
	ETA	1,00E+00	1,00E-01	1,00E-02	1,00E-03	1,00E-04	1,00E-05	1,00E-06	1,00E-07	1,00E-08	1,00E-09	1,00E-10
	<b>REGULARIZACION</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>	<b>L1</b>
	<b>MEDIA MSE TRAIN</b>	0.042464	0.026243	0.013773	0.006437	0.002082	0.000519	0.000148	0.000059	0.000031	0.000021	0.000019
	<b>DESV MSE TRAIN</b>	0.000234	0.000236	0.000462	0.000494	0.000184	0.000047	0.000015	0.000003	0.000003	0.000002	0.000001
	<b>MEDIA MSE TEST</b>	0.039545	0.027733	0.024031	0.024732	0.028544	0.031106	0.032684	0.033833	0.034440	0.034985	0.035107
	<b>DESV MSE TEST</b>	0.000297	0.000395	0.000221	0.000838	0.001388	0.001084	0.001311	0.001493	0.001697	0.001743	0.001946
	<b>MEDIA CCR TRAIN</b>	85.98%	92.02%	96.16%	98.67%	99.84%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	<b>DESV CCR TRAIN</b>	0.16%	0.26%	0.18%	0.14%	0.09%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	<b>MEDIA CCR TEST</b>	89.47%	91.73%	91.60%	90.20%	88.73%	88.73%	88.33%	88.13%	88.07%	87.93%	87.93%
	<b>DESV CCR TEST</b>	0.69%	0.49%	0.39%	0.62%	0.57%	0.57%	0.70%	0.58%	0.71%	0.68%	0.65%
	ETA	1,00E+00	1,00E-01	1,00E-02	1,00E-03	1,00E-04	1,00E-05	1,00E-06	1,00E-07	1,00E-08	1,00E-09	1,00E-10
	<b>REGULARIZACION</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>	<b>L2</b>
	<b>MEDIA MSE TRAIN</b>	0.035390	0.027215	0.020049	0.013838	0.008853	0.004929	0.002164	0.000759	0.000226	0.000058	0.000013
	<b>DESV MSE TRAIN</b>	0.000182	0.000317	0.000443	0.000427	0.000411	0.000357	0.000209	0.000078	0.000023	0.000007	0.000002
	<b>MEDIA MSE TEST</b>	0.034311	0.028351	0.025085	0.023793	0.024056	0.025681	0.028213	0.030693	0.032754	0.034244	0.035610
	<b>DESV MSE TEST</b>	0.000213	0.000122	0.000115	0.000274	0.000483	0.000737	0.001006	0.001188	0.001333	0.001491	0.001487
	<b>MEDIA CCR TRAIN</b>	88.69%	91.73%	93.87%	96.09%	97.69%	99.02%	99.82%	100.00%	100.00%	100.00%	100.00%
	<b>DESV CCR TRAIN</b>	0.26%	0.23%	0.34%	0.39%	0.37%	0.22%	0.05%	0.00%	0.00%	0.00%	0.00%
	<b>MEDIA CCR TEST</b>	90.87%	91.33%	92.00%	91.47%	90.87%	89.93%	89.20%	88.20%	88.20%	87.93%	87.80%
	<b>DESV CCR TEST</b>	0.27%	0.21%	0.21%	0.16%	0.34%	0.25%	0.50%	0.45%	0.45%	0.68%	0.45%

Figura 6 Tabla noMNIST prueba 2

BASE DE DATOS VOTE CON 5 num rbf												
	ETA	1,00E+00	1,00E-01	1,00E-02	1,00E-03	1,00E-04	1,00E-05	1,00E-06	1,00E-07	1,00E-08	1,00E-09	1,00E-10
REGULARIZACION	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
MEDIA MSE TRAIN		0.035887	0.021388	0.010109	0.005161	0.003884	0.003640	0.003591	0.003584	0.003581	0.003580	0.003580
DESVMSE TRAIN		0.000276	0.000270	0.000415	0.000069	0.000022	0.000007	0.000002	0.000001	0.000001	0.000000	0.000000
MEDIA MSE TEST		0.040938	0.036776	0.042216	0.052391	0.056993	0.058302	0.059016	0.058997	0.059332	0.059521	0.059767
DESVMSE TEST		0.000865	0.000912	0.001195	0.002055	0.002221	0.001275	0.001169	0.001706	0.001650	0.001471	0.001456
MEDIA CCR TRAIN		95.28%	97.61%	98.47%	99.39%	99.39%	99.39%	99.39%	99.39%	99.39%	99.39%	99.39%
DESVMSE CCR TRAIN		0.15%	0.23%	0.19%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIA CCR TEST		95.78%	96.33%	95.41%	93.76%	93.58%	93.58%	93.58%	93.39%	93.76%	93.76%	93.76%
DESVMSE CCR TEST		0.45%	0.00%	0.58%	0.37%	0.00%	0.00%	0.00%	0.37%	0.37%	0.37%	0.37%
	ETA	1,00E+00	1,00E-01	1,00E-02	1,00E-03	1,00E-04	1,00E-05	1,00E-06	1,00E-07	1,00E-08	1,00E-09	1,00E-10
REGULARIZACION	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2
MEDIA MSE TRAIN		0.027544	0.021020	0.014941	0.009612	0.006185	0.004500	0.003843	0.003642	0.003593	0.003582	0.003580
DESVMSE TRAIN		0.000427	0.000296	0.000234	0.000305	0.000251	0.000121	0.000042	0.000010	0.000002	0.000000	0.000000
MEDIA MSE TEST		0.039935	0.037224	0.036130	0.036822	0.039853	0.045045	0.050509	0.054507	0.056728	0.057919	0.058710
DESVMSE TEST		0.000596	0.000612	0.000622	0.000616	0.000715	0.001060	0.001266	0.001336	0.001285	0.001222	0.001237
MEDIA CCR TRAIN		96.32%	97.12%	97.85%	98.83%	99.33%	99.39%	99.39%	99.39%	99.39%	99.39%	99.39%
DESVMSE CCR TRAIN		0.00%	0.15%	0.00%	0.12%	0.12%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIA CCR TEST		94.86%	96.15%	96.33%	96.33%	95.05%	94.50%	94.13%	93.58%	93.58%	93.58%	93.58%
DESVMSE CCR TEST		0.45%	0.37%	0.00%	0.00%	0.45%	0.00%	0.45%	0.00%	0.00%	0.00%	0.00%

Figura 7 Tabla Vote prueba 2

En estas pruebas se estudia como afecta el valor de eta y la regularización a la red.

En el caso de NOMNIST el mejor resultado en test se obtiene con una eta de  $1e-2$  y L2 y lo mismo ocurre en el caso de la VOTE.

Aunque se ha determinado ese resultado como el “mejor”, tanto L1 como L2 resultan muy similares al menos en el caso de estas bases de datos. Por tanto, el parámetro eta resulta más determinante. Observamos que es en la parte más central-izquierda de las tablas donde están los resultados que tienen un buen resultado en test y que parecen no sobreentrenar.

En el caso donde eta vale 0, el operador recibe  $1/\eta$ . Con lo cual se estaría trabajando con infinito y esto llevaría mucho más tiempo de computo. De hecho, si se probase el rendimiento sería tan malo que no merece la pena tenerlo en cuenta.

Para el cálculo de los coeficientes se ha llevado a cabo la siguiente modificación en el código:

```
logreg1 = LogisticRegression(penalty='l2', C=1 / eta, fit_intercept=False, multi_class='auto',
                             solver='liblinear', max_iter=500)
# else:
logreg2 = LogisticRegression(penalty='l1', C=1 / eta, fit_intercept=False, multi_class='auto',
                             solver='liblinear', max_iter=500)

# Entrenated model
if train_outputs.shape[1] == 1:
    train_outputs = np.ravel(train_outputs)
logreg1.fit(matriz_r, train_outputs)
logreg2.fit(matriz_r, train_outputs)
print('L1')
print(logreg1.coef_.size)
print('L2')
print(logreg2.coef_.size)
print('DIF')
print(logreg1.coef_-logreg2.coef_)
raw_input()
return logreg1
```

Figura 8 Código para el cálculo de coeficientes



Para la base de datos vote obtenemos 160 coeficientes y la siguiente diferencia L1 vs L2:

```
[ [ 4.05569761e-01 -1.39677523e+00 -2.07659873e+00 -4.09986890e+00
    1.43693745e+00 -2.70895936e+00 5.19530549e+00 -1.22575373e+00
    -4.95821917e+00 1.34989154e+00 2.31539737e+00 -1.59866861e+00
    -3.03816573e+00 5.38965536e+00 -6.25814105e-01 5.25279387e-01
    2.87734315e+00 2.28313484e-01 2.08814588e+00 1.09692766e-01
    2.04283448e+00 6.67638587e+00 -2.82828859e+00 -5.02542680e+00
    -1.85101013e+00 2.96164478e+00 3.48272627e+00 -1.66945518e+00
    -2.56453787e+00 -7.09846653e-01 -2.12256252e+00 -6.84952400e-01
    -2.80774887e+00 3.50672795e+00 -9.66181261e+00 1.16779992e+00
    -3.67092757e+00 2.44355616e+00 2.55956294e+00 -1.05773706e+00
    1.65564977e+00 -3.70069361e-02 2.43966926e+00 -3.24723627e+00
    6.79392955e-01 -4.58351471e+00 -1.42590287e+00 -1.19197268e+00
```

Figura 9 Diferencia de coeficientes Vote

Para la base de datos noMNIST obtenemos 20706 coeficientes y la siguiente diferencia L1 vs L2:

```
[ [ -2.60221951 5.02644058 1.15910608 ... 4.1261887 0.57360058
    -22.38370871]
  [ 0.24782199 -3.83104782 -1.29110165 ... -0.29677712 2.12229734
    7.03310254]
  [ -1.65406264 3.05750593 2.44809593 ... -14.6413744 -5.13316114
    2.88043928]
  [ -10.04295968 -2.40955335 1.72377344 ... -0.94359975 6.3487444
    -11.82625472]
  [ 0.08461449 2.39119852 10.83655895 ... 3.47059611 4.41378372
    2.87167435]
  [ -2.70249059 -0.78791162 -8.32557302 ... -3.14294926 -4.79110907
    -1.59060257]]
```

Figura 10 Diferencia de coeficientes noMNIST

### 4.3. Tercera prueba

Para problemas de regresión y de clasificación, comparar los resultados obtenidos con la inicialización propuesta para el algoritmo `sklearn.cluster.KMeans` (usando la mejor arquitectura y la mejor configuración para la regresión logística) con respecto a la inicialización `k-means++`.

En primer lugar, añadimos `k-means++` al código:

```
if clasificacion == True:
    centros = inicializar_centroides_clas(train_inputs, train_outputs, num_rbf)
    kmedias = KMeans(n_clusters=num_rbf, init='k-means++', n_init=1, max_iter=300, n_jobs=-1)
else:
    # Obtenemos num_rbf numeros aleatorios
    kmedias = KMeans(n_clusters=num_rbf, init='k-means++', max_iter=300, n_jobs=-1)
```

Figura 11 Código para K-Means++

Ahora podemos compararlo con los resultados obtenidos anteriormente:

NUM RBF	5	5	25	5	15	5	5	25	5	15
BASE DE DATOS	SIN	QUAKE	PARKINSONS	VOTE	NOMNIST	SIN	QUAKE	PARKINSONS	VOTE	NOMNIST
INICIALIZACION	RANDOM	RANDOM	RANDOM	CENTROS	CENTROS	K-MEANS++	K-MEANS++	K-MEANS++	K-MEANS++	K-MEANS++
MEDIA MSE TRAIN	0.013798	0.028349	0.000408	0.026196	0.005929	0.013870	0.028351	0.000454	0.027964	0.006764
DESV MSE TRAIN	0.000063	0.000061	0.000007	0.003005	0.000516	0.000014	0.000030	0.000008	0.003094	0.000721
MEDIA MSE TEST	0.022310	0.028323	0.001045	0.041162	0.033221	0.022567	0.028286	0.001187	0.036038	0.032443
DESV MSE TEST	0.000196	0.000274	0.000079	0.008827	0.000989	0.000030	0.000205	0.000070	0.007022	0.001165
MEDIA CCR TRAIN	80.33%	94.54%	98.26%	96.20%	98.91%	79.67%	94.57%	98.21%	95.64%	98.40%
DESV CCR TRAIN	0.67%	0.02%	0.09%	0.31%	0.32%	0.41%	0.05%	0.07%	0.71%	0.38%
MEDIA CCR TEST	78.05%	94.86%	97.36%	94.50%	88.20%	78.05%	94.83%	97.33%	95.78%	87.87%
DESV CCR TEST	0.00%	0.00%	0.34%	1.92%	0.96%	0.00%	0.07%	0.19%	1.24%	0.69%

Figura 12 Tabla K-Means++ prueba 3

La inicialización por K-Means++ debería seleccionar los centros automáticamente de manera que se acelere la convergencia.

Podemos apreciar que la diferencia entre ambas inicializaciones es tan escasa, que en los problemas que estamos tratando no afecta para nada. Incluso se han obtenido puntuaciones ligeramente superiores en test con las inicializaciones aleatorias y de centros. Esto puede deberse a que quizás K-Means++ converja un poco más prematuramente que los métodos iniciales.

#### 4.4. Cuarta prueba

En alguno de los problemas de clasificación, probar a lanzar el script considerando el problema como si fuera un problema de regresión (es decir, incluyendo un False en el parámetro clasificación y calculando el CCR redondeando las predicciones hasta el entero más cercano). ¿Qué sucede en este caso?

NUM RBF	25	50
BASE DE DATOS	VOTE	NOMNIST
MEDIA MSE TRAIN	0.019695	0.178518
DESV MSE TRAIN	0.000684	0.007441
MEDIA MSE TEST	0.046399	0.684503
DESV MSE TEST	0.002859	0.005175
MEDIA CCR TRAIN	97.73%	88.91%
DESV CCR TRAIN	0.31%	0.28%
MEDIA CCR TEST	93.76%	64.60%
DESV CCR TEST	0.37%	1.31%

Figura 13 Tabla clasificación con regresión prueba 4

Ejecutando ambas bases de datos de clasificación como regresión nos damos cuenta de que sobre todo la NOMNIST sufre una bajada de puntuación importante con respecto a la VOTE (aunque esta también la sufre). Claramente no es la forma adecuada de tratar este tipo de problemas ya que lo mejor sería una salida Softmax para clasificar de la mejor manera posible. Sin embargo, en la base de datos VOTE, al tener solo dos clases, es posible que sin aplicarle clasificación pueda dar buenos resultados.

### 5. Comparativa de NOMNIST con la práctica anterior

Obtenemos la siguiente matriz de confusión:

[	47	0	0	1	0	2]
[	4	42	0	2	2	0]
[	0	1	46	0	2	1]
[	1	6	0	42	0	1]
[	0	0	4	0	42	4]
[	2	1	0	1	2	44]]

Figura 14 Matriz de confusion TEST con RBF

Y a continuación se muestra la obtenida en la práctica anterior (nótese que se encuentra traspuesta).

	41	1	0	1	0	1	
	3	40	1	4	4	2	
	0	5	44	1	2	1	
	5	3	2	44	0	2	
	1	1	1	0	41	3	
	0	0	2	0	3	41	

Figura 15 Matriz de confusión TEST con Perceptrón Multicapa

Aunque la matriz parece similar, lo mejor es comprobar las letras que se han clasificado mal en ambos casos, y comprobar cuando los mal clasificados son las letras más confusas.

Por ejemplo, en la práctica anterior estas fueron algunas de las mal clasificadas:



Figura 16 Mal clasificados con Perceptrón Multicapa


Mientras que en esta ocasión tendríamos estos ejemplos:

Como vemos en la matriz, hay cuatro B que se han confundido con una A. Una de ellas es esta





, en este caso, parece que dicho símbolo se hubiera podido confundir con cualquier cosa.

Otra de las B confundidas con A es esta  que también puede ocurrir debido al “ruido que

tiene la imagen. Por otro lado, esta B  ha sido confundida con una D.

Otra de las letras más confundidas es la E con la C, que también tiene cuatro errores. Uno de

estos ejemplos sería  y . Que son imágenes que se salen de la forma habitual que suele tener la letra E. Algo que también provoca, que algunas imágenes de E se confundan con la F.

En el caso de las menos confusas encontramos algunas similares a las de la práctica anterior.

Al final nos damos cuenta de que este modelo ha tendido a equivocarse sobre todo en las letras más confusas y difíciles de reconocer salvo alguna excepción. Mientras que el modelo de la práctica anterior tuvo más fallos en letras fáciles de reconocer.

También es importante hablar del tiempo de computo. En el caso de esta práctica, la NOMNIST que es la base de datos más extensa tarda 6.363154 segundos de media en completar el algoritmo con 50% de num RBF. Si probamos con 99% de num RBF, el tiempo medio tan solo asciende a 7.343762 segundos.

Nada que ver con la práctica anterior donde el tiempo de computo podía ascender a los 30 minutos fácilmente y utilizando conjunto de validación, que debería disminuir el número de iteraciones.

Claro está que hay otros factores que intervienen, por ejemplo, el equipo donde se esté ejecutando el algoritmo. Pero hay otros que podemos considerar mucho más importantes, como que en esta práctica se emplean menos iteraciones o hay solo una capa oculta, o que la base datos es algo más reducida.

Pero el factor más importante es que anteriormente se utilizaba el algoritmo de retro propagación del error, el cual es muy costoso debido, por ejemplo, a los cálculos de las derivadas y la cantidad de bucles que influyen. Además de haber sido programado desde cero y sin mucho tiempo para prestarle atención a la optimización del código.

Mientras que en esta ocasión se están utilizando cálculos más simples en los que intervienen matrices y métricas como la distancia euclídea. Que evidentemente tienen menos coste

computacional. Y también se debe tener en cuenta que se hace uso de las librerías de Python. Unas librerías que han sido muy testeadas y, por tanto, están muy bien optimizadas.