Práctica 3: Redes neuronales de funciones de base radial

Asignatura: Introducción a los modelos computacionales, 4º Grado Ingeniería Informática (Escuela Politécnica Superior de Córdoba - Universidad de Córdoba)

Trabajo realizado por:

-Antonio Gómez Giménez (32730338G)
i72gogia@uco.es

Índice:

1. Descripción de entrenamiento de las redes RBF:

Principalmente, para llevar a cabo el entrenamiento de las redes RBF, nos hemos basado en el siguiente algoritmo (propuesto en la práctica):

Algoritmo de entrenamiento RBF *off-line*

Inicio

1. centroidesIniciales ← aleatoriamente n_1 patrones (regresión) o n_1 patrones de forma estratificada (clasificación).
2. centroides ← K-medias(**X**,n_1,centroidesIniciales) // **X** *es el conjunto de entradas para todos los patrones*
3. σ_j ← (media de las distancias al resto de centroides)/2.
4. Construir la matriz $\mathbf{R}_{(N\times(n_1+1))}$, donde $\mathbf{R}_{ij} = out_j^1(\mathbf{x}_i)$ para $j \neq (n_1+1)$, y $\mathbf{R}_{ij} = 1$ para $j = (n_1+1)$.
5. **Si** clasificación
   1. pesosSalida ← aplicarRegresionLogistica(**R**,*eta*)
6. **Si** regresión
   1. pesosSalida ← calcularPseudoInversa(**R**)

Fin

Primeramente, buscamos hacer el **clustering** usando ***kmeans*** pero, para ello, es necesario sacar los centroides iniciales. En el caso de regresión no hace falta buscarlos, con el parámetro *init='random'*, se escogen aleatoriamente pero para clasificación es necesario que nosotros creemos los centroides iniciales de forma estratificada, esto lo conseguiremos con la función **train_test_split**.
Aprovechamos el **clustering** para sacar también las distancias entre los centros y los patrones de entrada.
Posteriormente con la función **calculate_radii** sacamos el radio de los centroides, este valor va a ser la media entre las distancias entre los centroides entre dos.
Una vez tenemos estos valores calculamos la *matriz R*, para ello usamos la función **calculate_r_matrix** donde le pasamos la distancia de los patrones a centroides y radio. En esta función se aplica la fórmula de la RBF y aparte añadimos al final una columna de 1 que representará al sesgo.
Una vez calculada la *matriz R*, resta calcular los coeficientes, esto variará dependiendo de si es un problema de *clasificación* o de *regresión*:
-Si es **regresión**, llamaremos a la función **LogisticRegression**, donde los devolverá los coeficientes respecto la *matriz R* y los valores de salida.
-Si es **clasificación**, aplicamos la matriz de *Moore Penrose*.

Una vez tenemos estos coeficientes, dependiendo de si es **regresión** o **clasificación**, calcularemos sus respectivos *MSE* y *CCR*, para ello es necesario sacar los coeficientes de test (necesitaremos la *matriz R* de test).
Por último, cabe destacar, que para **clasificación** es necesario pasar el vector de etiquetas a binario, ya que si no se hace, para el *MSE* le damos más valor de error a unas etiquetas que a otras y esto no es correcto.

2. Experimentos y análisis de resultados:

Para los experimentos se han utilizado las siguientes bases de datos donde función seno, quake y parkinson's se basan en problemas de regresión mientras que noMNIST y divorce de clasificación:

-**Función seno**: esta base de datos está compuesta por 120 patrones de entrenamiento y 41 patrones de test. Ha sido obtenido añadiendo cierto ruido aleatorio a la función seno. Consta de una variable de entrada y una variable de salida.

-**Base de datos quake**: esta base de datos está compuesta por 1633 patrones de entrenamiento y 546 patrones de test. Se corresponde con una base de datos en la que el objetivo es averiguar la fuerza de un terremoto (medida en escala sismológica de Richter). Como variables de entrada, utilizamos la profundidad focal, la latitud en la que se produce y la longitud, de tal forma que son 3 variables de entrada y una variable de salida.

-**Base de datos parkinsons**: esta base de datos está compuesta por 4406 patrones de entrenamiento y 1469 patrones de test. 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. Consta de 19 variables de entrada y dos variables de salida.

-**Base de datos divorce**: divorce contiene 127 patrones de entrenamiento y 43 patrones de test. La base de datos contiene la respuesta a una serie de preguntas de un conjunto de encuestas en las que se pretende predecir si se va a producir un divorcio en la pareja. Las respuestas a las preguntas de la encuesta se proporcionan en escala de Likert con valores de 0 a 4. Todas las variables de entrada se consideran numéricas. La base de datos tiene un total de 54 preguntas (por lo tanto, 54 variables de entrada) y dos categorías (0 no hay divorcio, 1 hay divorcio).

-**Base de datos noMNIST**: esta base de datos está compuesta por 900 patrones de entrenamiento y 300 patrones de test. Está 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 28×28 píxeles. Las imágenes están en escala de grises en el intervalo [−1,0; +1,0].

Cada uno de los píxeles es una variable de entrada (con un total de 28×28 = 784 variables de entrada) y las clases se corresponden con la letra escrita (a,b,c,d,e y f, con un total de 6 clases).

Se han realizado 4 experimentos:

En el **primer experimento** se buscaba encontrar la mejor arquitectura para nuestras bases de datos. Por ello, para todas las bases de datos, consideramos un número de neuronas en capa oculta (n) igual al 5%, 15%, 25% y 50% del número de patrones de la base de datos(este es el parámetro ratio). En esta fase, para problemas de clasificación, utilizamos regularización L1 y un valor para el parámetro tasa de aprendizaje = 10e−5.

Para mayor claridad en las tablas se va a dividir en partes:

Función seno:

Ratio	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
0.05	0.013401	0.000046	0.020366	0.000235	0.00	0.00	0.00	0.00
0.15	0.012586	0.000006	0.032220	0.001097	0.00	0.00	0.00	0.00
0.25	0.012553	0.000045	0.033029	0.002627	0.00	0.00	0.00	0.00
0.50	0.012563	0.000051	0.032614	0.002720	0.00	0.00	0.00	0.00

Base de datos quake:

Ratio	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
0.05	0.028500	0.000052	0.028170	0.000159	0.00	0.00	0.00	0.00
0.15	0.028050	0.000067	0.028785	0.000094	0.00	0.00	0.00	0.00
0.25	0.028002	0.000042	0.029093	0.000036	0.00	0.00	0.00	0.00
0.50	0.027977	0.000023	0.029139	0.000045	0.00	0.00	0.00	0.00

Base de datos parkinsons:

Ratio	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
0.05	0.021721	0.000161	0.024636	0.000390	0.00	0.00	0.00	0.00
0.15	0.015479	0.000145	0.020408	0.000824	0.00	0.00	0.00	0.00
0.25	0.014252	0.000166	0.019868	0.000161	0.00	0.00	0.00	0.00
0.50	0.013503	0.000057	0.019580	0.000430	0.00	0.00	0.00	0.00

Base de datos divorce:

Ratio	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
0.05	0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00
0.15	0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00
0.25	0.000000	0.000000	0.009302	0.011393	100.00	0.00	99.07	1.14
0.50	0.000000	0.000000	0.009302	0.011393	100.00	0.00	99.07	1.14

Base de datos noMNIST:

Ratio	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
0.05	0.039407	0.001699	0.038444	0.002288	88.18	0.51	88.47	0.69
0.15	0.000000	0.000000	0.045778	0.005135	100.00	0.00	86.27	1.54
0.25	0.000000	0.000000	0.044444	0.004869	100.00	0.00	86.67	1.46
0.50	0.000000	0.000000	0.040000	0.002331	100.00	0.00	88.00	0.70

Como podemos observar en las tablas, una ver realizado este experimento, llegamos a las siguientes conclusiones:
-El uso de un valor no lo suficientemente alto en la función *pinv* a la hora de hacer la inversa, hace que algunos valores de media de test y media de train sean 0.

Aun así, esto es necesario ya que en el valor por defecto, el MSE de Test se dispara, esto es por la correlación existente entre los valores de entrada para solucionar este problema se usa la función dicha anteriormente.

-En los problemas de regresión(seno, quake y parkinson) nos fijamos en el MSE dado por la media de Test, siendo los más bajos aquellos que su ratio es de un 5% excepto Parkinsons que su ratio es de 50%.
En el caso de clasificación(divorce y noMNIST) nos fijamos en el CCR dado, siendo el mayor CCR en el que el ratio es de 5%. Por ello, llegamos a la conclusión de que un menor ratio tiende a dar mejores soluciones, pero esto no tiene por qué ya que en gran parte depende de la propia base de dato, un claro ejemplo es la base de datos Parkinsons que da mejores resultados a medida que se aumenta el ratio de neuronas en capa oculta.
Quizás esto se debe a que al haber gran cantidad de centroides, gran cantidad de patrones se asocian a centroides que no les corresponden, por culpa de la abundancia de los mismos.
Aun así, tener pocos centroides no significa ser mejor, ya que si hay pocos centroides, la mayoría de patrones pueden ir al mismo centroide por la poca cantidad de los mismos. En el caso de clasificación, podría ser que un patrón que pertenecía a cierta clase se clasifique de incorrecta forma por este problema.

-Cabe destacar, que se pueden apreciar casos de sobreentrenamiento, uno de ellos es en la base de datos noMNIST, donde la media de entrenamiento del CCR a partir del ratio 0.15 da 100% de CCR, pero a la hora de generalizar en Test la mejor solución era aquella que entrenaba peor.

En el **segundo experimento,** para los problemas de clasificación, una vez decidida la mejor arquitectura, se probaron los siguientes valores para la tasa de aprendizaje: tasa de aprendizaje = 1, tasa de aprendizaje = 0,1, tasa de aprendizaje = 0,01, tasa de aprendizaje = 0,001, . . ., tasa de aprendizaje = 10−10, junto con los dos tipos de regularización (L2 y L1). También se calculó la diferencia en número de coeficientes en divorce y noMNIST cuando modificamos el tipo de regularización (L2 Vs L1).

La tasa de aprendizaje dicta cuánto va a aprender nuestro algoritmo, este valor es el que pasamos a invertido a la función logisticRegresion como valor C, en problemas de clasificación.
A la función anterior se le pasa si se aplica L2 o L1, dependiendo de si es uno u otro convergen de distinta forma.

Base de datos divorce:

L1: (eta va desde 1(primero) hasta 0,0000000001(último))

Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)	Coeficientes
0.014173	0.003150	0.023256	0.000000	98.58	0.31	97.67	0.00	4.000000
0.000000	0.000000	0.004651	0.009302	100.00	0.00	99.53	0.93	5.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	5.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000

L2: (eta va desde 1(primero) hasta 0,0000000001(último))

Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)	Coeficientes
0.023622	0.000000	0.023256	0.000000	97.64	0.00	97.67	0.00	7.000000
0.022047	0.003150	0.023256	0.000000	97.80	0.31	97.67	0.00	7.000000
0.015748	0.004980	0.023256	0.000000	98.43	0.50	97.67	0.00	7.000000
0.006299	0.003150	0.023256	0.000000	99.37	0.31	97.67	0.00	7.000000
0.000000	0.000000	0.004651	0.009302	100.00	0.00	99.53	0.93	7.000000
0.000000	0.000000	0.004651	0.009302	100.00	0.00	99.53	0.93	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000
0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00	7.000000

Base de datos noMNIST:

L1: (eta va desde 1(primero) hasta 0,0000000001(último))

Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)	Coeficientes
0.067630	0.003383	0.059111	0.004629	79.71	1.01	82.27	1.39	100.000000
0.045185	0.002197	0.037778	0.001859	86.44	0.66	88.67	0.56	186.000000
0.041185	0.002190	0.038000	0.002266	87.64	0.66	88.60	0.68	250.000000
0.039630	0.001932	0.038222	0.002288	88.11	0.58	88.53	0.69	275.000000
0.039481	0.001731	0.038444	0.002288	88.16	0.52	88.47	0.69	276.000000
0.039407	0.001699	0.038444	0.002288	88.18	0.51	88.47	0.69	276.000000
0.039407	0.001699	0.038444	0.002288	88.18	0.51	88.47	0.69	276.000000
0.039407	0.001699	0.038444	0.002288	88.18	0.51	88.47	0.69	276.000000
0.039407	0.001699	0.038444	0.002288	88.18	0.51	88.47	0.69	276.000000
0.039407	0.001699	0.038444	0.002288	88.18	0.51	88.47	0.69	276.000000

L2: (eta va desde 1(primero) hasta 0,0000000001(último))

Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)	Coeficientes
0.076222	0.001452	0.079111	0.005938	77.13	0.44	76.27	1.78	276.000000
0.056741	0.002227	0.046889	0.004629	82.98	0.67	85.93	1.39	276.000000
0.045556	0.002916	0.038222	0.002863	86.33	0.87	88.53	0.86	276.000000
0.041407	0.001865	0.036222	0.002863	87.58	0.56	88.64	0.86	276.000000
0.039926	0.002007	0.037778	0.002534	88.02	0.60	88.67	0.76	276.000000
0.039481	0.001683	0.038222	0.002288	88.16	0.50	88.53	0.69	276.000000
0.039481	0.001683	0.038444	0.002288	88.16	0.50	88.47	0.69	276.000000
0.039481	0.001683	0.038222	0.002288	88.16	0.50	88.53	0.69	276.000000
0.039481	0.001683	0.038444	0.002288	88.16	0.50	88.47	0.69	276.000000
0.039481	0.001683	0.038222	0.002288	88.16	0.50	88.53	0.69	276.000000
0.039481	0.001683	0.038444	0.002288	88.16	0.50	88.47	0.69	276.000000

Sorprendentemente, la tasa de aprendizaje, cuando se disminuye hasta cierto punto, no se realiza mejora, sino más bien se queda constante, esto ocurre ya que se aprende tan poco en el entrenamiento que prácticamente no cambia.
Entre L1 y L2 da mejores resultados L1, esto se debe a que ciertos coeficientes convergen, esto se observa mejor si nos fijamos en la última columna donde se muestra la media de los coeficientes dados en test.
El parámetro L permite realizar regularización, este mecanismo nos permite lograr que el máximo número de parámetros de la matriz de coeficientes tienden a valores muy pequeños. Por ello, el parámetro eta, establece la importancia que se le da a la regularización. Por consiguiente y como podemos observar, a menor tasa de aprendizaje menor importancia a la regularización y menos parámetros de la matriz convergen, siendo así el número de coeficientes más elevado.
Respecto a la comparativa entre L1 y L2, L1 da menos coeficientes y esto tiene sentido, ya que, en L1 el peso si es mayor se penaliza más (es constante), mientras que en L2 si es mayor se penaliza menos (se encuentra al cuadrado), entonces aquellos valores cercanos al cero es más fácil que convergen en el caso de L1 y esto es lo que podemos observar en el caso de nuestras tablas.

En el **tercer experimento** para los problemas de regresión y de clasificación, comparamos 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++'.
Para los resultados obtenidos se han utilizado las mejores arquitecturas para cada base de datos, para la regresión, el valor de la tasa de aprendizaje es indiferente, ya que, solo se usa en clasificación. Los resultados obtenidos son los siguientes:

dataset	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
sin	0.013401	0.000046	0.020366	0.000235	0.00	0.00	0.00	0.00
quake	0.028500	0.000052	0.028170	0.000159	0.00	0.00	0.00	0.00
parkinsons	0.013503	0.000057	0.019580	0.000430	0.00	0.00	0.00	0.00
divorce	0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00
noMNIST	0.045185	0.002197	0.037778	0.001859	86.44	0.66	88.67	0.56

dataset	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
sin	0.013400	0.000020	0.020527	0.000072	0.00	0.00	0.00	0.00
quake	0.028463	0.000018	0.027950	0.000112	0.00	0.00	0.00	0.00
parkinsons	0.013923	0.000038	0.020661	0.000103	0.00	0.00	0.00	0.00
divorce	0.000000	0.000000	0.000000	0.000000	100.00	0.00	100.00	0.00
noMNIST	0.046000	0.001679	0.036889	0.004298	86.20	0.50	88.93	1.29

Como podemos observar, los cambios son prácticamente insignificantes, si acaso, es un poco mejor en todas las bases de datos excepto en parkinsons el uso de k-means++.
Esto se debe a que el uso de este parámetro permite inicializar los centroides de manera inteligente, permitiendo así una mejora de los resultados, por ello, mejora un poco el uso de k-means++ respecto a los valores aleatorios en regresión o los centroides inicializados de manera uniformemente estratificada por nosotros.

Para el **último experimento**, lanzamos el algoritmo 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). Los resultados obtenidos fueron los siguientes:

dataset	Media (Train)	Desviación típica (Train)	Media (Test)	Desviación típica (Test)	Media (Train CCR)	Desviación típica (Train CCR)	Media (Test CCR)	Desviación típica (Test CCR)
divorce	0.022805	0.000622	0.024330	0.000384	97.64	0.00	97.67	0.00
nomnist	0.848358	0.016673	0.908674	0.024351	49.42	2.14	49.40	1.74

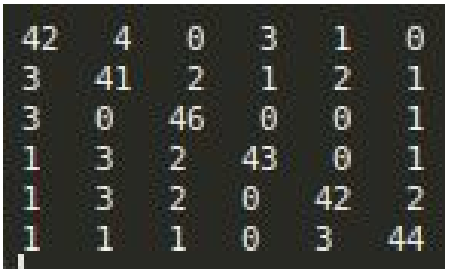
Como podemos ver, y era de esperarse, los resultados obtenidos son peores que los vistos anteriormente. Esto es completamente lógico, ya que estamos aplicando regresión a un problema de regresión, por ello los resultados a priori deberían ser peores que al realizar clasificación en un problema de clasificación.
Cabe destacar que en la base de datos divorce, el CCR obtenido es bastante alto aun habiendo realizado regresión, esto se debe a que divorce es una base de datos sencilla.

3. Resumen:

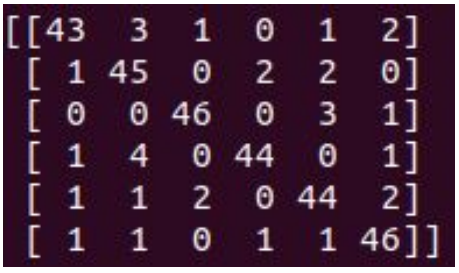
Una vez vistos estos experimentos y habiendo sacando conclusiones de los mismos, se va a proceder a realizar unos refuerzos sobre la práctica vista:

-Comparación de matrices de confusión entre los dos mejores modelos vistos para la base de datos noMNIST de cada respectiva práctica:

- Práctica anterior:



- Práctica actual (SEED 4)



Como podemos observar, las matrices de confusión son muy parecidas, es decir, la mayoría de tipos de errores son idénticos, esto quiza pueda ser por lo visto en la práctica anterior, que haya patrones que son ruido en la base de datos que vuelvan peor al clasificador o que se encuentre algún valor erróneo o difícil de clasificar en los datos de test.

Para ver qué patrones han fallado, únicamente es necesario ver que tenía que salir y qué valor hemos obtenido, unos ejemplos de fallos serían los siguientes:

El patrón 8 es una A (0), pero nuestro clasificador lo etiqueta como una F (5), veamos el patrón para ver si tiene sentido.

Para nosotros se ve claro, más o menos, que es una A pero ciertamente, para una máquina es comprensible que falle.

Otro patrón que falla es el 288 que es una F y lo confunde con una B, veamos ese patrón.

Este patrón sin embargo, se ve claramente que es una F, si acaso, ha podido fallar por falta de patrones similares en entrenamiento donde el palo de abajo de la F se acerque más al palo de arriba de la F, ya que quizá por eso lo confunde con una B al estar demasiado juntos.

Para finalizar se va a comprobar cómo aplicar redes neuronales de base radial es mucho más rápido que aplicar redes de percepción multicapa. Para ello veremos el tiempo computacional necesario para entrenar la base de datos noMNIST y compararla con lo que tarda en la práctica anterior.

```
antoniogg@antoniogg-SATELLITE-L50D-C:~/Escritorio/IMC/Practica2/skeletonLA2IMC/l
a2/Debug$ time ./la2 -t ../../../datasetsLA2IMC/basesDatosPr2IMC/dat/train_nomni
st.dat -T ../../../datasetsLA2IMC/basesDatosPr2IMC/dat/test_nomnist.dat -i 1000
-l 1 -h 8 -e 0.9 -m 1.5 -f 1 -s > prueba.txt

real    5m22,176s
user    5m20,963s
sys     0m0,380s
antoniogg@antoniogg-SATELLITE-L50D-C:~/Escritorio/IMC/Practica3$ time python3 ./
rbf.py -t ./datasetsLA3IMC/csv/train_nomnist.csv -T ./datasetsLA3IMC/csv/test_no
mnist.csv -c -r 0.05 -e 0.1 > prueba.txt

real    0m10,049s
user    0m15,426s
sys     0m7,078s
```

Como era de esperarse, es muchísimo más rápido, siendo el caso anterior 5 minutos y 22 segundos mientras que en el caso actual son 10 segundos.