Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Especialidad: Computación
Cuarto curso. Primer cuatrimestre

Introducción a los modelos computacionales.

Práctica 3: Redes neuronales de funciones de base radial.

Antonio Ariza García
DNI
i62argaa@uco.es

Curso académico 2020-2021
Córdoba, 24 de diciembre de 2020

Índice

Índice de figuras

Índice de tablas

List of Algorithms

1. Introduccion a redes neuronales de funcio-ness Base Radial

La arquitectura de las redes RBF está formada por una capa de entrada, una capa oculta y una capa de salida, está última será la suma ponderada de las salidas de la RBFs en caso de Regresión y será de tipo Softmax en caso de Clasificación. Las redes neuronales de funciones de base Radial (RBF) se basan en una aproximación local.

- Las neuronas de capa oculta son funciones de base radial: Son funciones locales. Dividen el espacio mediante hiperesferas, una por cada neurona de capa oculta, en las cuales se define un centro y un radio.
- Al contrario de las redes de tipo perceptrón multicapa, donde las neuronas son de capa oculta son funciones de tipo sigmoide: Estas son funciones de proyección.

A continuación en la Figura 1 se puede comprobar las diferencias.

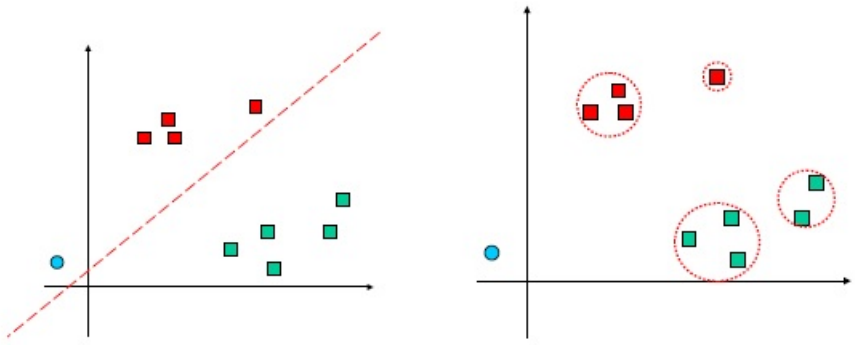

Figura 1: Izq: Función proyección, Der: Función local

Las funciones RBF dependen de la distancia que hay entre la entrada y un centro, cada RBF guarda un centro como referencia, y cada vez que se le presenta un patrón nuevo, se calcula la distancia a dicho centro. Para calcular la distancia de un patrón entre centros se usará los radios.

- Si la distancia es pequeña, la RBF se activa (su salida es 1).
- Si la distancia es grande, la RBF no se activa (su salida es 0).

- Si el radio es pequeño, la activación solo se producirá cuando el patrón esté muy cerca del centro.
- Si el radio es grande, la activación se producirá a más distancia.

¿Como ajustamos los parámetros?, aunque las funciones RBF son derivables, son bastantes complejas y tienen un coste computacional alto, por lo tanto emplearemos un entrenamiento Híbrido: parte no supervisada (clustering) y parte supervisada (regresión logística o inversión de una matriz).

2. Implementación Algoritmo RBF

2.1. Arquitectura de los modelos

- Una capa de entrada con tantas neuronas como variables tenga la base de datos considerada.
- Una capa oculta con un número de neuronas a especificar por el usuario. Siempre tendremos solo una capa oculta, con todas las neuronas de tipo RBF.
- Una capa de salida con tantas neuronas como variables de salida tenga la base de datos considerada.
    - Si la base de datos es de regresión: Todas las neuronas serán de tipo lineal.
    - Si la base de datos es de clasificación, todas las neuronas de la capa de salida serán de tipo Softmax.

2.2. Algoritmo Entrenamiento RBF

Algorithm 1 Entrenamiento RBF Offline

centroidesIniciales ← aleatoriamente n_1 patrones (regresion) o n_1 patrones de forma estratificada (clasificacion)
centroides ← K-medias (X,n_1,centroidesIniciales)
σ_j ← (medias de las distancias al resto de centroides)/2
Construir la matriz $R_{(Nx(n1+1))}$, donde $R_{ij} = out_j^1(x_i)$ para $j \neq (n_1 + 1)$, y $R_{ij} = 1$ para $j = (n_1 + 1)$
if clasificacion **then**
    PesosSalida ← AplicarRegresionLogistica(R,eta)
else
    PesosSalida ← CalcularPseudoInversa(R)
end if

1. Si es clasificación sacamos centroides de forma estratificada, y si es regresión escogemos aleatoriamente n_1 patrones

2. Ajuste de centros mediante algoritmo de clustering más popular, el K-means, con tantos clusters como neuronas de la RBF.

3. Para ajustar los radios de la RBF tomaremos la mitad de la distancia media al resto de centroides. $\sigma_j = \frac{1}{2(n_1 - 1)} * \sum_{i \neq j} ||c_j - c_i||$

4. Construimos la matriz R, donde tenemos el valor de activación de cada neurona para cada patrón.

5. Si es Clasificación

    6. Aplicamos regresión logística, utilizamos una función a la que le pasamos la matriz R. (puede incluir regularización, parámetro η).

7. Si es Regresión

    8. Una vez construida la matriz R, aplicamos la pseudo-inversa de Moore Penrose.

3. Resultados Prácticos

Probaremos distintas configuraciones de la red neuronal y ejecutaremos cada configuración con cinco semillas (1, 2, 3, 4 y 5). A partir de los resultados obtenidos, se obtendrá la media y la desviación típica del error. Se deberá calcular el error MSE para el conjunto de entrenamiento y el error MSE para el conjunto de test.

3.1. Descripción de las Bases de Datos

Para valorar como funciona el algoritmo implementado en esta práctica, emplearemos un total de cinco bases de datos, 3 de ellas de regresión:

- Función seno: esta base de datos está compuesta por 120 patrones de train y 41 patrones de test. Ha sido obtenida añadiendo cierto ruido aleatorio a la función seno
- Base de datos quake: esta base de datos está compuesta por 1633 patrones de train 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.
- Base de datos parkinsons: esta base de datos está compuesta por 4406 patrones de train 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.

Y dos bases de datos de clasificación:

- Base de datos Divorce: Contiene 127 patrones de entrenamiento y 43 patrones de test. 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, todas las variables de entrada se consideran numéricas. Contiene 54 preguntas (variables entrada) y dos categorías.

- Base de datos noMNIST: Está compuesta por 900 patrones de entrenamiento y 300 patrones de test, y un total de 6 clases. Está formada por un conjunto de letras de la 'a' a la 'f' con diferentes tipografías o simbologías.

3.2. Descripción de los distintos parámetros

A continuación se explicarán cada parámetro que podrá recibir el programa:

- Argumento -t, $--train\_file$: Indica el nombre del fichero que contiene los datos de entrenamiento a utilizar. Sin este argumento, el programa no puede funcionar.
- Argumento -T, $--test\_file$: Indica el nombre del fichero que contiene los datos de test a utilizar. Si no se especifica este argumento, utilizar los datos de entrenamiento como test.
- Argumento -c, $--classification$: Booleano que indica si el problema es de clasificación. Si no se especifica, suponemos que es de regresión.
- Argumento -r, $--radio\_rbf$: Indica la razón (en tanto por uno) de neuronas RBF con respecto al total de patrones en entrenamiento. Si no se especifica, utilizar 0,1 capa oculta.
- Argumento -l, $--l2$: Booleano que indica si utilizaremos regularización de L2 en lugar de la regularización L1. Si no se especifica, supondremos que regularización L1.
- Argumento -e, $--eta$: Indica el valor del parámetro eta (η). Por defecto, utilizar $\eta = 1e-2$.
- Argumento -o, $--outputs$: Indica el número de columnas de salida que tiene el conjunto de datos y que siempre están al final. Por defecto, utilizar o=1.
- (Kaggle) Argumento -p, $--pred$: Booleano que indica si utilizaremos el modo de predicción.
- Argumento -m, $--model\_file$: Indica el directorio en el que se guardarán los modelos entrenados (sin flag p), o el fichero que contiene el modelo que se utilizará (en modo predicción, con flag p).

- Argumento –help: Mostrar la ayuda del programa (automáticamente librería click).

3.3. Prueba de distintas arquitecturas

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

3.3.1. BD Seno

Ratio_rbf	**MSE train**	**MSE test**
5 %(6)	0.014034 +- 0.000186	0.022510 +- 0.000190
15 %(18)	0.011830 +- 0.000172	0.112010 +- 0.026119
25 %(30)	0.011660 +- 0.000322	0.143285 +- 0.060435
50 %(60)	0.011675 +- 0.000243	0.152624 +- 0.056728

Tabla 1: Base de datos seno

Conforme vamos aumentando el número de neuronas de capa oculta, el error de Test también aumenta, la mejor arquitectura es la primera, la cual tiene 6 neuronas en capa oculta (5 % ratio_rbf).

3.3.2. BD Quake

Ratio_rbf	**MSE train**	**MSE test**
5 % (82)	0.028431 +- 0.000100	0.028261 +- 0.000130
15 %(245)	0.026759 +- 0.000176	0.032262 +- 0.001896
25 %(408)	0.026324 +- 0.000118	0.034716 +- 0.001028
50 %(816)	0.026061 +- 0.000111	0.036177 +- 0.000974

Tabla 2: Base de datos Quake

En la base de datos Quake sucede lo mismo que en la base de datos seno, conforme vamos aumentando ratio_rbf aumenta también el error, en este caso nos quedamos con la arquitectura que tiene 82 neuronas en capa oculta (5 % ratio_rbf).

3.3.3. BD Parkinsons

Ratio_rbf	**MSE train**	**MSE test**
5 % (220)	0.001669 +- 0.000064	0.002029 +- 0.000053
15 %(661)	0.000751 +- 0.000017	0.001247 +- 0.000045
25 %(1102)	0.000410 +- 0.000014	0.001083 +- 0.000102
50 %(2203)	0.000135 +- 0.000003	0.001275 +- 0.000165

Tabla 3: Base de datos Parkinsons

Para la base de datos Parkinsons, necesitamos un ratio_rbf mayor que en las anteriores, un 15 % ratio_rbf, lo que conlleva a 1102 neuronas en capa oculta. Aunque en realidad no hay mucha diferencia en test con las demás arquitecturas.

3.3.4. BD Divorce

Ratio_rbf	**MSE train**	**MSE test**	**CCR train**	**CCR test**
5 % (6)	0.000001	0.000002	100 %	100.0 %
15 % (19)	0.000001	0.001674	100 %	99.53 %
25 % (32)	0.000001	0.002126	100 %	100.0 %
50 % (64)	0.000001	0.005453	100 %	99.53 %

Tabla 4: Base de datos Divorce

Para el caso de base de datos de clasificación además del MSE también debemos tener en cuenta el CCR, para la BD divorce nos quedamos con la arquitectura de 6 neuronas en capa oculta (5 % ratio_rbf).

3.3.5. BD noMNIST

Ratio_rbf	MSE train	MSE test	CCR train	CCR test
5 % (45)	0.030195	0.0030288	87.93 %	88.53 %
15 % (135)	0.000655	0.0421280	99.96 %	86.67 %
25 % (225)	0.000043	0.0377220	100.0 %	87.33 %
50 % (450)	0.000020	0.0036403	100.0 %	87.93 %

Tabla 5: Base de datos noMNIST

Para la base de datos noMNIST también nos quedamos con ratio_rbf 5 % que son 45 neuronas en capa oculta con CCR en test del 88.53 %.

3.4. Para problemas de clasificación, variación de η y regularización L1 y L2

Para los problemas de clasificación, una vez decidida la mejor arquitectura, probamos los siguientes valores de η: $\eta = 1, \eta = 0.1, \eta = 0.01, \eta = 0.001, \ldots, \eta = 10^{-10}$

η	**Regularización**	**MSE train**	**MSE test**	**CCR train**	**CCR test**
1.0	L1	0.019964	0.021861	97.64 %	97.67 %
0.1	L1	0.005912	0.014801	99.21 %	97.67 %
0.01	L1	0.000233	0.000924	100.0 %	100.0 %
0.001	L1	0.000007	0.000049	100.0 %	100.0 %
1e-4	L1	0.000001	0.000002	100.0 %	100.0 %
1e-5	L1	0.000001	0.000003	100.0 %	100.0 %
1e-6	L1	0.000001	0.000003	100.0 %	100.0 %
1e-7	L1	0.000001	0.000003	100.0 %	100.0 %
1e-8	L1	0.000001	0.000003	100.0 %	100.0 %
1e-9	L1	0.000001	0.000003	100.0 %	100.0 %
1e-10	L1	0.000001	0.000003	100.0 %	100.0 %
1.0	L2	0.021211	0.022794	97.64 %	97.67 %
0.1	L2	0.016878	0.020989	97.64 %	97.67 %
0.01	L2	0.008642	0.018296	98.43 %	97.67 %
0.001	L2	0.001705	0.006904	100.0 %	98.14 %
1e-4	L2	0.000103	0.000665	100.0 %	100.0 %
1e-5	L2	0.000003	0.000061	100.0 %	100.0 %
1e-6	L2	0.000000	0.000015	100.0 %	100.0 %
1e-7	L2	0.000000	0.000002	100.0 %	100.0 %
1e-8	L2	0.000000	0.000001	100.0 %	100.0 %
1e-9	L2	0.000000	0.000001	100.0 %	100.0 %
1e-10	L2	0.000000	0.000001	100.0 %	100.0 %

Tabla 6: Variaciones de η y regularización en BD Divorce

η	**Regularización**	**MSE train**	**MSE test**	**CCR train**	**CCR test**
1.0	L1	0.054578	0.051867	80.49 %	82.80 %
0.1	L1	0.035240	0.032531	86.42 %	89.13 %
0.01	L1	0.030858	0.030006	87.87 %	88.87 %
0.001	L1	0.030250	0.030242	87.87 %	88.60 %
1e-4	L1	0.030195	0.030288	87.93 %	88.53 %
1e-5	L1	0.030191	0.030287	87.93 %	88.53 %
1e-6	L1	0.030190	0.030287	87.93 %	88.53 %
1e-7	L1	0.030190	0.030288	87.93 %	88.53 %
1e-8	L1	0.030190	0.030288	87.93 %	88.53 %
1e-9	L1	0.030190	0.030288	87.93 %	88.53 %
1e-10	L1	0.030190	0.030288	87.93 %	88.53 %
1.0	L2	0.067085	0.065913	77.67 %	77.47 %
0.1	L2	0.045423	0.043739	83.36 %	86.40 %
0.01	L2	0.035838	0.033371	86.22 %	89.00 %
0.001	L2	0.031632	0.030141	87.40 %	89.07 %
1e-4	L2	0.030373	0.030173	87.82 %	88.67 %
1e-5	L2	0.030187	0.030310	87.98 %	88.53 %
1e-6	L2	0.030167	0.030327	88.00 %	88.53 %
1e-7	L2	0.030165	0.030324	88.00 %	88.60 %
1e-8	L2	0.030164	0.030325	87.98 %	88.53 %
1e-9	L2	0.030164	0.030331	88.00 %	88.53 %
1e-10	L2	0.030166	0.030324	88.02 %	88.60 %

Tabla 7: Variaciones de η y regularización en BD noMNIST

Podemos comprobar en L1 que a la hora de variar η por encima de 0.001 apenas hay diferencia, tenemos prácticamente los mismo datos. He seleccionado regularización L1 con $\eta = 0.1$ que obtiene un 89.13 % en test, aunque también tenemos regularización L2 con $\eta = 0.001$ con un 89.07 %. En cuanto a la regularización comprobamos que quitando $\eta = 1$ y $\eta = 0.1$ obtenemos mejores resultados con L2, aunque son mínimos.

3.5. Comparación inicialización KMeans aleatorio con respecto a KMeans++

El esquema de inicialización 'kmean++' que se implementado en scikit-learn inicializa los centroides para que estén (generalmente) distantes entre sí, lo que conduce a unos resultados demostrablemente mejores que la inicialización aleatoria.

	Base datos	MSE train	MSE test
Aleatorio	Sin	0.014034	0.022510
	Quake	0.028431	0.028261
	Parkinsons	0.000410	0.001083
Kmean++	Sin	0.013827	0.022296
	Quake	0.028318	0.028362
	Parkinsons	0.001612	0.001929

Tabla 8: Comparación entre iniciación aleatoria y kmeans++ en BD regresión

Como podemos ver la diferencia entre 'Aleatorio' y 'Kmeans++' es mínima, para la BD 'Sin' es la única que conseguimos una mejora, para 'Quake' y 'Parkinsons' aumenta el error, esto puede ser debido a que aunque inicialice los centroides de una mejor manera, nada nos asegura que los clusters encontrados sean más correctos.

	Base datos	MSE train	MSE test	CCR train	CCR test
Aleatorio	Divorce	0.000233	0.000924	100.0 %	100.0 %
	noMNIST	0.035240	0.030006	86.42 %	89.13 %
Kmean++	Divorce	0.006112	0.014361	99.06 %	97.67 %
	noMNIST	0.034445	0.030409	86.04 %	88.87 %
	noMNIST	0.030306	0.028208	87.69 %	89.20 %
	noMNIST	0.029731	0.028371	87.69 %	89.53 %

Tabla 9: Comparación entre iniciación aleatoria y kmeans++ en BD clasificación

En cuanto a la BD de clasificación tenemos un peor CCR test, esto puede ser debido a que kMeans++ no tiene en cuenta las clases de los patrones, y lo hace de forma no estratificado. Sin embargo realizando más pruebas con un parámetro $\eta = 0.01$ y $\eta = 0.001$ conseguimos una mejora del 0.40 % con respecto 'Aleatorio'.

3.6. Usando Regresión en BD Clasificación, noMNIST

Usando regresión en problemas de clasificación, en concreto solamente quitando la bandera -c a la hora de ejecutar y calculando el CCR redondeando las predicciones hasta el entero más cercano.

	Base datos	**MSE train**	**MSE test**	**CCR train**	**CCR test**
	Divorce	0.000233	0.000924	100.0 %	100.0 %
Clasificación	noMNIST	0.035240	0.030006	86.42 %	89.13 %
	Divorce	0.018558	0.020998	97.64 %	97.67 %
Regresión	noMNIST	0.829759	0.895238	51.91 %	50.47 %

Tabla 10: Regresión BD noMNIST

Comprobamos como el resultado en 'Divorce' es muy parecido aunque es algo inferior. Para 'noMNIST' el resultado es mucho peor, está configuración no tiene mucho sentido.

3.7. Resultados en noMNIST

Volvemos lanzar la configuración con la mejor arquitectura (tabla 7), regularización L1 con $\eta = 0.1$ y ratio_rbf del 5 %

$$\begin{Bmatrix} 44 & 2 & 1 & 1 & 0 & 2 \\ 1 & 46 & 0 & 2 & 1 & 0 \\ 1 & 0 & 45 & 0 & 2 & 2 \\ 3 & 4 & 0 & 42 & 0 & 1 \\ 0 & 2 & 2 & 0 & 44 & 2 \\ 1 & 0 & 0 & 1 & 2 & 46 \end{Bmatrix}$$

Figura 2: Matriz de confusión noMNIST