

Práctica 3: Redes neuronales de función de base radial

Introducción a los modelos computacionales.

**Grado en Ingeniería Informática, Escuela Politécnica Superior de
Córdoba, Universidad de Córdoba.**

4º Curso.



**Escuela
Politécnica
Superior**

Realizado por: José Pérez-Parras Toledano

Dni: 31014082P

Email: i32petoj@uco.es

Índice:

1.	Descripción de los pasos a realizar para el entrenamiento de RBF.....	Página 2
2.	Experimentos y análisis de los resultados.....	Página 3
2.1.	Descripción de las bases de datos utilizadas.....	Página 3
2.2.	Breve descripción de los parámetros utilizados.....	Página 4
2.3.	Resultados obtenidos.....	Página 5
2.3.1.	Seno.....	Página 5
2.3.2.	Forest.....	Página 6
2.3.3.	CPU.....	Página 7
2.3.4.	Iris.....	Página 8
2.3.5.	Digits.....	Página 11
3.	Bibliografía.....	Página 14

1. Descripción de los pasos a realizar para llevar a cabo el entrenamiento de las redes RBF:

Para esta práctica hemos usado el script proporcionado por el profesor en python, que además hace uso de una librería externa llamada *sklearn*. La cual contiene distintos modelos y algoritmos para entrenar los modelos.

En nuestro caso hemos usado como entrenamiento el algoritmo “k-means” con el cual colocaremos los centroides en las posiciones más oportunas. Dichos centroides han de tener una posición inicial, la cual será un patrón de nuestro conjunto de entrenamiento. Pero nuestro patrón puede ser aleatorio o hacerlo nosotros mismos, escogiendo x centroides de cada clase para que todo esté más repartido. Puesto que esto es un algoritmo de clustering y cada centroide junto a un radio que calcularemos posteriormente representará una clase.

Una vez que tenemos los centroides iniciales, como ya hemos dicho ejecutaremos el algoritmo *k-means* el cual viene implementado en la librería *sklearn*. Que tras unas 300 iteraciones nos posicionará los centroides para que agrupen al mayor número de instancias cada uno de una forma eficiente.

Una vez tenemos los centroides en las posiciones deseadas, calculamos los radios de acción de cada uno de ellos, por los cuales sabremos a qué clase pertenece cada instancia en función a la distancia al centroide más cercano.

Dichos radios se calcularán con una simple distancia media de la distancia a cada uno de los demás centroides, además hemos de dividir dicha distancia por 2, puesto que se trata del radio y no del diámetro.

Una vez tenemos dichos radios, hemos de ajustar los pesos de la capa de salida, lo cual se hará de manera distinta dependiendo de si el problema a tratar es de regresión o de clasificación. Para los problemas de clasificación se usará una regresión logística, la cual también viene incluida en la biblioteca de *sklearn*. Mientras que para los problemas de regresión deberemos de calcular la pseudo-inversa de una matriz por el método de Moore-Penrose, para poder despejar una ecuación con matrices y así poder sacar los pesos de la última capa.

2. Experimentos y análisis de resultados:

1. Descripción de las bases de datos utilizadas.

- **Seno:**

Esta es una base de datos de regresión, que representa la función del seno. Está compuesta por 120 patrones de entrenamiento y 41 patrones de test. Además ha sido obtenida añadiendo cierto ruido aleatorio a la función seno.

- **Forest:**

Esta base de datos de regresión es un poco más amplia que la anterior puesto que contiene un mayor número de patrones tanto en entrenamiento como en test. Concretamente por 387 patrones de entrenamiento y 130 patrones de test. Contiene, además, como entradas o variables independientes, una serie de datos meteorológicos y otras variables descriptoras sobre incendios en bosques al norte de Portugal. Y como salida una resultará el área quemada.

- **Datos CPU:**

Esta base de datos de regresión será una intermedia entre las dos anteriores puesto que cuenta con 109 patrones de entrenamiento y 100 patrones de test. Contendrá como variables de entrada una serie de características de un procesador y salida el rendimiento relativo de dicho procesador.

- **Iris:**

Esta base de datos de clasificación está formada por una serie de patrones de tres tipos de plantas, en particular, clases de iris, las cuales son: *iris setosa*, *iris virginica* e *iris versicolor*. Estas 3 clases serán las que usaremos para clasificar. Estos patrones tienen 4 variables de entrada, según las cuales sabremos a qué clase pertenece, dichas variables son: longitud y ancho de los pétalos y los sépalos.

Además contamos con que son amplios ficheros con 112 patrones de entrenamiento y 38 patrones de test.

- **Digits:**

Esta base de datos de clasificación está compuesta por dígitos escritos por 80 personas distintas, los cuales han sido binarizados en un espacio de 16x16 píxeles, lo que nos da un total de 256 variables de entrada.

Gracia a dichas variables de entrada podremos saber de qué cifra se trata, del 0 al 9, lo que nos da un total de 10 clases para nuestro modelo. Además contamos con amplios ficheros con una gran suma de patrones, teniendo 1274 patrones para entrenamiento y 319 patrones para test, por lo que podremos entrenar muy bien nuestra red neuronal, pero también hemos de tener en cuenta que dichas cifras escritas a mano difieren mucho unas de otras porque cada persona tiene una forma distinta de escribir, por lo que puede que nuestra red neuronal falle un poco.

2. Breve descripción de los parámetros utilizados.

En este apartado describiremos los parámetros utilizados para entrenar los distintos modelos en las distintas pruebas que hemos realizado sobre las bases de datos.

Dichos parámetros son:

- **fichero_train**: Con este parámetro le indicamos al *script* que fichero de entrenamiento ha de cargar y usar para dicha ejecución.
- **fichero_test**: Con este parámetro le indicamos al *script* que fichero de *test* ha de cargar y utilizar para dicha ejecución. Con este se probará el modelo entrenado en dicha ejecución y nos dará unos resultados en cuanto a dichas instancias de generalización.
- **num_rbf**: Con este parámetro indicamos el número de neuronas RBF que tendrá nuestro modelo. Dicho número de neuronas coincide con el número de centroides inicial que usaremos para el entrenamiento y posterior clasificación de los patrones.
- **clasificación**: Con este parámetro indicaremos si vamos a usar clasificación o regresión para calcular las salidas de nuestro modelo. Tendremos que cambiarlo dependiendo de la base de datos a analizar, puesto que algunas son de regresión y otras de clasificación como Iris.
- **eta**: Este parámetro si se viene usando en las anteriores prácticas y representa la tasa de aprendizaje que se usa en el entrenamiento, es decir, los saltos que va dando nuestro modelo, si queremos que sean más destacados o menos.
- **l2**: Este parámetro es nuevo en este tipo de modelo de RBF, puesto que representa la regularización en la regresión logística. Teniendo esta dos variantes: 'l1' y 'l2', 'l2' tiende a proporcionar pesos más pequeños. Mientras que 'l1' tiende a podar más variables haciendo que muchos pesos sean iguales a cero, aunque los que no son cero no tienen porque ser pequeños en valor absoluto.

3. Resultados obtenidos.

Se van a realizar distintas pruebas o experimentos sobre las distintas bases de datos que tenemos para esta práctica. Además distinguiremos las bases de datos de regresión de las de clasificación, puesto que sus experimentos y datos obtenidos serán distintos.

Para ejecución se probará con distintas semillas, en concreto con cinco semillas, que serán como diez ejecuciones diferentes dentro de la misma, para así poder obtener la media y desviación típica de los resultados. Para los problemas de regresión se obtendrá únicamente el MSE, mientras que para clasificación también se podrá obtener el CCR así como la matriz de confusión.

Se harán distintas pruebas variando el número de neuronas en capa oculta, es decir, el número de neuronas rbf. Una vez obtengamos cual ha sido la mejor arquitectura según el número de neuronas, probaremos a cambiar otros parámetros como eta o la regularización en los problemas de clasificación. Sin más dilación procederemos a exponer dichos resultados.

3.1. Seno

Con esta base de datos de regresión como ya hemos explicado anteriormente, realizaremos distintos experimentos variando el número de nodos en capa oculta. Dicho número de nodos en capa oculta será dependiendo del número de patrones de la base de datos de entrenamiento. Estas pruebas se harán con un 5%, 10%, 25% y 50%.

Para nuestro caso con 120 patrones en entrenamiento dicho número de neuronas serán:

5% = 6

10% = 12

25% = 30

50% = 60

Y una vez tengamos la mejor arquitectura cambiaremos los valores de eta, para obtener la mejor arquitectura y como será de regresión, el parámetro de regularización no se usará en dicha ejecución.

Número de neuronas	6	12	30	60
Media MSE train	0,013522	0,01249	0,010361	0,01029
Desv MSE train	0,000046	0,000028	0,000007	0,000164
MEdia MSE test	0,021221	0,036226	1,848221	1,496992
Desv MSE test	0,000145	0,003009	0,494484	1,136243

He marcado la mejor arquitectura, que casualmente ha sido la de un menor número de neuronas en capa oculta. Como se puede comprobar muy fácilmente con estos datos se ha obtenido un gran sobreentrenamiento para 30 y 60 neuronas, cuyos modelos obtienen muy poco error en entrenamiento pero un gran error en generalización. Entonces siendo esta una base de datos bastante sencilla con pocos patrones, podemos suponer que al aumentar el número de neuronas en capa oculta estamos sobreentrenando nuestro modelo hasta límites bastante imprecisos y destructivos para nuestro modelo.

3.2. Forest

Como hemos dicho anteriormente para la base de datos de Seno, esta base de datos de Forest también se trata de una base de datos de regresión y por lo tanto no haremos uso del atributo de normalización. Como en el apartado anterior, haremos distintos experimentos variando el número de neuronas en capa oculta, con los mismo porcentajes que los citados anteriormente, pero en este caso al tener un mayor número de patrones, también han aumentado dichas neuronas RBF en nuestro modelo, siendo las siguientes:

5% = 19

10% = 38

25% = 96

50% = 193

Los datos de las neuronas no eran enteros, así que he redondeado al número anterior, he aquí los resultados obtenidos en estos experimentos:

Número de neuronas	19	38	96	193
Media MSE train	0,001514	0,001446	0,001258	0,001027
Desv MSE train	0,000007	0,000074	0,000073	0,000075
MEdia MSE test	0,009007	0,009033	0,009825	0,0118
Desv MSE test	0,000039	0,000166	0,000355	0,00046

En este caso también nos ha dado mejor resultado la mejor arquitectura con un menor número de neuronas en capa oculta, además de ser mucho más simple, puesto que cuantas más neuronas en capa oculta más complejo se vuelve nuestro modelo. También se puede ir observando un sobreentrenamiento conforme vamos aumentando la complejidad de nuestro modelo, puesto que se va viendo cómo se reduce el error en el conjunto de entrenamiento, mientras que va aumentando el error en el conjunto de generalización. Aunque estos cambios no son tan drásticos como ocurrían en la base de datos de seno, sí se puede observar. Este cambio más lento puede deberse a la complejidad de la base de datos, puesto que está compuesta de un mayor número de patrones, tanto para entrenamiento como para test y además cuenta con un mayor número de entradas, lo que hace que se tengan que asignar un mayor número de pesos y que nuestro modelo sea más complejo, afectando menos el número de neuronas en capa oculta.

3.3. Datos CPU

Seguimos con la última base de datos de **regresión**, aunque esta base de datos tiene un menor número de patrones que la de Forest, es de las que más entradas tiene. Por lo tanto y como ya venimos haciendo para las otras bases de datos, intentaremos encontrar el mejor modelo, haciendo las pruebas necesarias cambiando el número de neuronas en capa oculta, el cual coincide con el número de rbfs de nuestro atributo. Dichas variaciones son iguales a las anteriores, teniendo los siguientes porcentajes, algunos de estos números se han redondeado al entero anterior.

5% = 5

10% = 10

25% = 27

50% = 54

Teniendo dichos números de neuronas, podemos proceder a realizar las pruebas y mostrarlas simultáneamente, cuyos resultados obtenidos han sido los siguientes:

Número de neuronas	5	10	27	54
Media MSE train	0,004264	0,001756	0,000459	0,000263
Desv MSE train	0,000027	0,001218	0,000036	0,000053
MEdia MSE test	0,004581	0,003438	0,002246	0,004166
Desv MSE test	0,000076	0,000983	0,000115	0,002979

En esta base de datos si nos hemos encontrado algo distinto a las dos anteriores de regresión y es que hemos llegado hasta el 25% de neuronas del número total de patrones y aún nuestro modelo no había sobreentrenado, como se puede ver para el 50% si ha comenzado ya la sobreentrenación, obteniendo mejores resultados en el conjunto de entrenamiento pero bastante peores en el conjunto de test.

Esto no nos sirve, puesto que queremos que nuestro modelo sea bueno en generalización con patrones que no ha conocido nunca, por ello no debe de sobreentrenar. Dejando esto de lado también es el que nos ha dado mejores resultados en entrenamiento, así que estaba claro que arquitectura de red escoger, aunque fuera más compleja que la de 5% y 10%, pero al tratarse de una base de datos tan pequeña, el coste computacional apenas es apreciable por lo que nos podemos permitir dicha complejidad en la red.

3.4. Iris.

Esta es la primera base de datos que nos hemos encontrado en la práctica que será de clasificación, es decir, tendremos que cambiar los parámetros de nuestro programa puesto que haremos un modelo clasificador, que nos dará unas salidas esperadas que serán la clase a la que se cree que pertenece cada patrón. Para este modelo realizaremos las mismas pruebas que para las bases de datos anteriores, pero además a la mejor arquitectura que obtengamos en las primeras pruebas le realizaremos más cambiando el valor de eta. Y como no faltaría decir, también tenemos que parámetro del tipo de regularización a aplicar, I1 o I2. Dicho parámetro también se cambiará para la mejor arquitectura obtenida y se observará que diferencias existen.

Además, diferente a las bases de datos anteriores, en esta se mostrarán también el CCR además del MSE, dicho CCR es el porcentaje de acierto que ha tenido nuestro clasificador en cuanto a los patrones, es decir, cuanto más alto más patrones bien clasificados hemos obtenido, por lo tanto observaremos dicho resultado por encima del MSE, priorizando en él para el conjunto de generalización.

Ahora mostraremos los primeros experimentos para encontrar la mejor arquitectura de red variando el número de neuronas en capa oculta, dichas pruebas han sido con un valor de eta de 10^{-5} y con el método de regularización I1:

Número de neuronas	5	11	28	56
Media MSE train	0,030357	0,016071	0	0
Desv MSE train	0,012111	0,003571	0	0
MEdia MSE test	0,1	0,036842	0,105263	0,105263
Desv MSE test	0,010526	0,026837	0	0
Media CCR train	96,964286	98,392857	100	100
Desv CCR train	1,21113	0,357143	0	0
Media CCR test	90	96,315789	89,473684	89,473684
Desv CCR test	1,052632	2,683694	0	0

Como ya hemos ido viendo en las anteriores bases de datos, pero aquí si que se puede comprobar perfectamente es el sobreentrenamiento realizado con 28 y 56 neuronas, puesto que han tenido un CCR perfecto sin ningún fallo en el conjunto de entrenamiento pero obteniendo peores resultados que otras configuraciones para el conjunto de test.

Por ello he seleccionado como la mejor arquitectura para dichos parámetros la de 11 neuronas en capa oculta, puesto que no llega a ser demasiado compleja y además obtiene un buen valor de CCR para el conjunto de generalización con un 96%

Ahora veremos cómo afecta el valor de eta para dicha configuración de la red, además también variaremos el parámetro de regularización entre L1 y L2.

Cómo vamos a realizar bastantes pruebas para estos cambios he decidido sólo incluir el CCR en el conjunto de entrenamiento, puesto que si no la tabla sería prácticamente ilegible.

eta	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
L1	95,789474	94,736842	95,789474	94,736842	93,684211	94,210526	93,684211	94,736842	94,736842	97,368421
L2	98,947368	95,263158	94,736842	94,736842	94,210526	94,736842	92,105263	94,210526	95,789474	94,736842

En estas pruebas he encontrado que los mejores resultados han sido con un valor de eta de 0,1 y la regularización L2, casi hemos llegado al 99% de CCR en el conjunto de generalización lo cual es bastante alto.

Comentando sobre estos datos obtenidos se puede ver como la diferencia entre las dos regularizaciones aumenta más en cuanto a valores muy pequeños (parte derecha de la tabla) los cuales hacen que L1 se ponga por delante en cuanto a rendimiento sobre L2, y para valores “grandes” (parte izquierda de la tabla) donde es L1 el que ha obtenido mejores resultados que L2. Mientras que si nos fijamos en la parte central de la tabla para valores cercanos a eta de 10^{-5} , apenas hay diferencias apreciables.

Esto se hace bastante visible para eta 10^{-4} , para el que hemos sacado exáctamente el mismo resultado tanto para la regularización de L1 como L2.

Hay un **caso particular** para eta y es cuando esta vale 0, puesto que al operador se le pasa $1/\text{eta}$, por lo que habría que trabajar con infinito, lo cual conlleva un coste computacional bastante alto y probado una ejecución con dicha eta, aún esperando durante al menos 10 minutos no se había completado ninguna ejecución de ninguna semilla, por lo que he decidido obviar dicho valor para este parámetro.

Ahora vamos a aplicar un **modelo de regresión** a esta base de datos de clasificación, lo que haremos será redondear las salidas al entero más próximo y esa será su clase, así calcularemos el CCR de dicho experimento, para esto experimento usaremos las arquitecturas que usamos al principio, variando únicamente el número de nodos puesto que es lo que afecta en regresión.

Número de neuronas	5	11	28	56
Media MSE train	0,083862	0,02832	0,021994	0,013978
Desv MSE train	0,000597	0,000397	0,001129	0,001218
Media MSE test	0,050431	0,030085	0,031621	0,049814
Desv MSE test	0,00095	0,003452	0,00387	0,02853
Media CCR train	88,214286	96,607143	97,321429	98,75
Desv CCR train	1,041241	0,357143	0,564692	0,437409
Media CCR test	92,105263	95,263158	97,368421	94,210526
Desv CCR test	0	1,052632	0	6,315789

Estos son los salidas obtenidas usando el modelo de regresión a este problema de clasificación, el cual no está preparado para mostrarnos un CCR, puesto que las salidas no son las clases, pero redondeándolas hemos obtenido dichos resultados.

Al final con un alto número de neuronas en capa oculta y haciendo de nuestro modelo algo complejo este sobreentrena, pero aún así tarda más que en clasificación y además hemos obtenido mejores resultados, mejorando en 1 el CCR de test de las pruebas realizadas con clasificación, aunque en clasificación podemos variar eta y la regularización lo que nos ha hecho llegar a mejores modelos, aún así el resultado obtenido es bastante bueno. Yo antes de realizar las ejecuciones esperaba que tuviera bastante peores resultados, puesto que el problema está pensado para clasificación y no para regresión.

3.5. Digits

Esta es la otra base de datos de clasificación, la cual es más extensa que la de iris y para la que haremos más pruebas, puesto que mostraremos la matriz de confusión para poder ver cuales han sido los patrones mal clasificados en el conjunto de test. Además para esta base de datos contamos con las imágenes por lo que será mucho más visual y se podrá comprobar por qué se ha equivocado el modelo y si es que la tipografía utilizada o la letra del que escribió el número no es bastante precisa.

Empezaremos por buscar cual es la arquitectura más adecuada para esta base de datos como ya hemos hecho en los experimentos de las otras bases de datos, para ello empezaremos por variar el número de neuronas en capa oculta, para después empezar a variar los parámetros de eta y la regularización.

Aunque el problema para esta base de datos es los recursos que consume y el tiempo de procesamiento que necesita entrenar dicho modelo, por lo que intentaremos escoger siempre el modelo más simple dentro de los mejores resultados, para poder ahorrar en tiempo de cómputo.

He aquí los datos de las pruebas obtenidas para distintas configuraciones de este modelo, basándose en distintos porcentajes del número de patrones del conjunto de entrenamiento para las neuronas en capa oculta:

Número de neuronas	63	127	318	637
Media MSE train	0,233909	0	0	0
Desv MSE train	0,071636	0	0	0
Media MSE test	1,677116	1,436364	1,264577	1,336677
Desv MSE test	0,236987	0,255576	0,192973	0,042226
Media CCR train	98,806907	100	100	100
Desv CCR train	0,401467	0	0	0
Media CCR test	91,097179	93,22884	94,420063	94,670846
Desv CCR test	0,581418	0,469173	0,234587	0

En esta tabla si se puede ver como con el aumento de las neuronas en capa oculta y por consiguiente el aumento de la complejidad de nuestro modelo, pero esto ha ido que nos vaya dando mejores resultados tanto en entrenamiento como en generalización, alcanzando casi un 95% de CCR sin desviación típica alguna entre las distintas semillas. Además de obtener un 100% en el conjunto de entrenamiento, pero lo que es mejor aún sin estar sobreentrenado puesto que la aceptación en el conjunto de test seguía aumentando.

Ahora vamos a realizar distintas pruebas variando el valor de eta y de la regularización para posteriormente quedarnos con la mejor configuración y ver en qué patrones se ha equivocado para analizarlos.

También hemos cogido esa configuración debido a que es más simple que la de 637 neuronas y los resultados obtenidos apenas varían.

Cogiendo la arquitectura anterior y aplicando las mismas variaciones de eta y de la regularización que las que ya aplicamos a la base de datos iris. Como ya hicimos en la base de datos de iris y para no enredar demasiado esta matriz de datos, solamente incluiremos el CCR medio de test para cada configuración y nos basaremos en ello para la explicación de los resultados.

Nos quedan los siguientes datos:

eta	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
L1	94,294671	94,294671	94,231975	94,420063	94,169279	94,294671	94,043887	94,796238	93,981191	94,357367
L2	93,730408	94,420063	94,545455	93,981191	94,733542	94,796238	94,169279	94,796238	94,169279	94,231975

Como podemos apreciar los valores obtenidos son bastante parecidos, aunque parece que L1 nos ha dado mejores resultados en la mayor parte de los casos, puesto que por ejemplo L2 empieza con un CCR de 93%, aunque es cercano a 94 pero L1 ya comienza directamente por encima del 94% para un valor de eta “grande”.

El mejor resultado obtenido ha sido con eta = 10^{-8} y hemos obtenido el mismo valor tanto para la regularización I1 como para la regularización I2.

Para este análisis sí hemos ejecutado eta=0, pero en vez de 0 para que C sea infinito he usado un valor muy pequeño de eta como 10^{-20} , el cual nos hará el apañó como 0, puesto que con infinito el coste computacional era demasiado alto. Y estos han sido los resultados:

Valor de eta	10^{-20}
L1	94,858934
L2	94,796238


Estos resultados son los mejores obtenidos hasta ahora. Cabe destacar que en casi todas estas pruebas realizadas para los distintos valores de eta, se ha obtenido una perfección en el entrenamiento, por lo que se podría suponer que se está sobreentrenando pero esto no es así, puesto que podíamos ver como variaba el CCR de test, aumentando. Si este únicamente disminuyera si habríamos caído en un umbral de sobreentrenamiento.


Ahora vamos a mostrar la matriz de confusión con el mejor modelo obtenido hasta ahora y analizar dónde se ha equivocado nuestro modelo para ver si dichas imágenes son realmente extrañas debido a la letra del autor o es que nuestro modelo se ha equivocado.


Clase Real Vs Predicha	0	1	2	3	4	5	6	7	8	9
0	32	0	0	0	0	0	0	0	1	0
1	0	30	1	0	1	0	0	0	0	1
2	0	0	32	0	0	0	0	0	0	0
3	0	0	0	31	0	0	0	0	0	1
4	1	0	0	0	30	0	0	1	0	0
5	0	0	0	0	0	31	0	0	0	0
6	0	0	0	0	0	0	32	0	0	0
7	0	0	0	0	1	0	0	29	1	1
8	0	0	0	0	0	0	0	0	30	1
9	0	0	1	1	0	1	0	1	0	27


Esta es la matriz de confusión con la clase real vs la predicha, como se puede observar en la mayoría de las clases únicamente hay un par de fallos, que tal vez sean por el tipo de letra usado que puede que incluso a nosotros nos cueste reconocer. Aunque también cabe destacar el gran acierto en 3 de las 10 clases, obteniendo en ellas un 100% de acierto, dichas clases han sido el 2, 5 y 6. A mí personalmente me parece sorprendente el 6, puesto que es fácilmente confundible con el número 8 si la escritura no es correcta, aunque analizando también el número 8, ese solo ha tenido un único fallo.


Siendo la clase que más ha fallado la 9, confundiéndose con la 2, 3, 5 y 7. A continuación pondremos algunos ejemplos de los errores, para ver si son fácilmente reconocibles por nosotros mismos.

El primer error en el que todos los modelos han fallado, es la confusión de este 0 con un 8 . Aunque cabe destacar que se parece más a un 8 que a un 0.

En esta por ejemplo si es un claro fallo del modelo: . Puesto que ha dicho que se trataba de un 9, cuando en esa si se puede apreciar mejor que se trata de un 8.

También podemos encontrarnos patrones como el siguiente: . En el que se ha de deducir que es un 1. Nuestro modelo lo ha clasificado como un 4. Pero esto es claramente un error bastante común puesto que a mi mismo me cuesta reconocer qué es.

En este caso , se puede apreciar un 1 pero nuestro modelo lo ha clasificado como 9. Este sí podría ser mejorable puesto que no es tan dudoso como los demás.

Y como último ejemplo, tenemos este 9 , que ha sido clasificado como un 7. Aunque como hemos podido observar en los demás ejemplos y en este mismo, casi todos los

patrones mal clasificados son de dudosa procedencia en cuanto a la manera de escribirlos y resulta difícil su clasificación hasta para un ojo humano.

3. Bibliografía:

- Presentación de la práctica.
- Documentación de la práctica.
- C. M. Bishop. Pattern Recognition and Machine Learning, Springer, 2006.