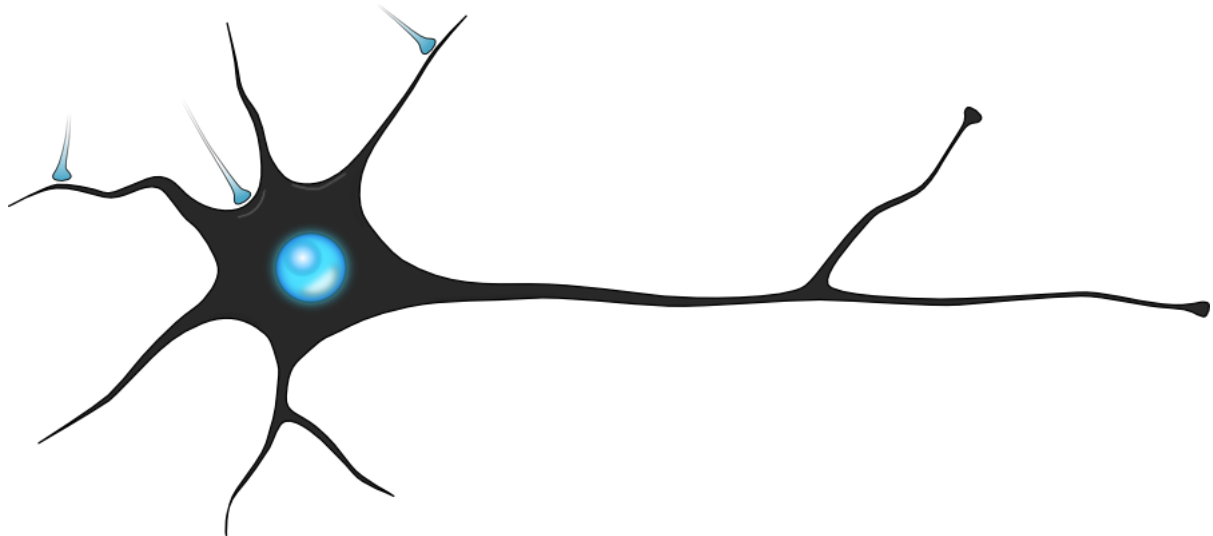




## PRÁCTICA 2: PERCEPTRÓN MULTICAPA PARA PROBLEMAS DE **CLASIFICACIÓN.**



Víctor Monserrat Villatoro  
i32moviv@uco.es  
45887876R



---

Descripción de las adaptaciones para problemas de clasificación.	2
Descripción en pseudocódigo de funciones que se han modificado.	3
Fase 1: propagación hacia delante.	4
Fase 2: propagación hacia atrás.	4
Retropropagación off-line.	5
Experimentos y análisis de resultados.	6
Descripción de las bases de datos utilizadas.	6
Problema XOR.	6
Iris.	7
Digits.	7
Descripción de los valores de los parámetros considerados.	7
Resultados obtenidos.	8
Problema XOR.	8
Iris.	9
Digits.	11
Análisis de resultados.	11
Problema XOR.	11
Iris.	12
Digits.	13
Referencias.	14



## 1. Descripción de las adaptaciones para problemas de clasificación.

Para poder representar, de forma adecuada, problemas de clasificación necesitamos realizar una interpretación probabilística al perceptrón multicapa. Para ello, vamos a utilizar la representación 1-de-k, una representación probabilística de una serie de eventos.

La representación 1-de-k consiste en lo siguiente:

- Se tienen  $k$  clases.
- Se tienen  $k$  eventos: el patrón pertenece a una de las clases del problema.
- La salida deseada es predecir que el patrón pertenece a la clase correcta con la mayor probabilidad.

De esta forma, modelamos y predecimos la probabilidad de que un patrón pertenezca a cada una de las clases.

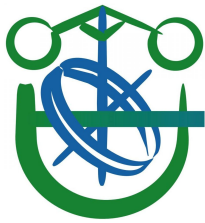
Aún así, necesitamos que las salidas de la red neuronal sean consistentes desde un punto de vista probabilístico, es decir, que la suma de probabilidades de que un patrón pertenezca a cada una de las clases sea la total.

Podemos usar la función softmax, que es una función de normalización, asegurando así que las salidas estén entre 0 y 1 y que la suma será siempre 1.

La función softmax es una aproximación plausible y derivable a la función máximo. Se calcula dividiendo la exponencial del valor de cada neurona de la última capa entre la suma de todas estas exponenciales.

La función exponencial asegura que vamos a tratar valores positivos y exagera las salidas para que el resultado se parezca a la función máximo.

El denominador de la función normaliza estos valores positivos, ya que es la suma de todos ellos. Así, conseguimos cumplir la restricción para valores probabilísticos, que sumen en total 1.



2. Descripción en pseudocódigo de funciones que se han modificado.

El algoritmo de retropropagación aplicado en una red neuronal prealimentada, concretamente de la clase perceptrón multicapa, se divide en tres fases, de las cuáles se han tenido que modificar las dos primeras.

2.1. Fase 1: propagación hacia delante.

```
c ← 2
Mientras c ≤ numeroCapas-isSoftmax
    Para Cada neurona n ∈ capa c Hacer
        net ← 0
        Para Cada neurona m ∈ capa c-1 Hacer
            net ← net + pesomn · salidam
        Si sesgo = verdadero
            net ← net + pesom+1n
        salidan ← 1 ÷ 1+e-net
    c ← c + 1
Si isSoftmax = verdadero
    sumNet = 0
    Para Cada neurona n ∈ capa c Hacer
        net ← 0
        Para Cada neurona m ∈ capa c-1 Hacer
            net ← net + pesomn · salidam
        Si sesgo = verdadero
            net ← net + pesom+1n
        salidan ← 1 ÷ 1+e-net
        exponenciales[n] ← exponencial(net)
        sumNet ← sumNet + exponenciales[n]
    Para Cada neurona n ∈ capa c Hacer
        salidan ← exponenciales[n]/sumNet
```



## 2.2. Fase 2: propagación hacia atrás.

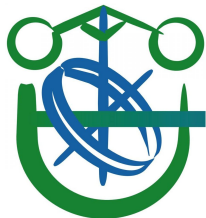
```
c ← numeroCapas
Si isSoftmax = falso
    Para Cada neurona n ∈ capa c Hacer
         $\delta_n \leftarrow -(\text{objetivos}_n - \text{salida}_n) \cdot \text{salida}_n \cdot (1 - \text{salida}_n)$ 
Sino
    Para Cada neurona n ∈ capa c Hacer
        sum = 0
        Para Cada neurona m ∈ capa c Hacer
            Si i = j
                cond = 1
            Sino
                cond = 0
            sum = sum + (objetivo[m] - salidam) · salidan · (cond - salidam)
         $\delta_n \leftarrow -\text{sum}$ 
c ← numeroCapas - 1
Mientras c > 0
    Para Cada neurona n ∈ capa c Hacer
        s ← 0
        Para Cada neurona m ∈ capa c+1 Hacer
            s ← s + pesonm ·  $\delta_m$ 
         $\delta_n \leftarrow s \cdot \text{salida}_n \cdot (1 - \text{salida}_n)$ 
    c ← c - 1
```

Además se han tenido que modificar la simulación de red y el entrenamiento porque se ha añadido el algoritmo off-line.



### 2.3. Retropropagación off-line.

```
inicializar()
c ← 2
Mientras c <= numeroCapas
    Para Cada neurona n ∈ capa c Hacer
        Para Cada neurona m ∈ capa c-1 Hacer
             $u\Delta\text{peso}_{mn} \leftarrow \Delta\text{peso}_{mn}$ 
             $\Delta\text{peso}_{mn} \leftarrow 0$ 
        c ← c + 1
p ← 1
Mientras p <= numeroPatrones
    alimentarEntradas(entradasp)
    propagarHaciaDelante()
    propagarHaciaAtrás(objetivosp, funcionError)
    calcularAjustePesos()
    p ← p + 1
actualizarPesos()
```



### 3. Experimentos y análisis de resultados.

#### 3.1. Descripción de las bases de datos utilizadas.

##### 3.1.1. Problema XOR.

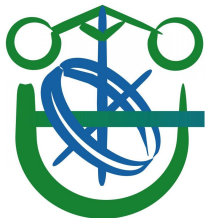
Esta base de datos representa el problema de clasificación no lineal del XOR. Está compuesta por 4 patrones, que se utilizaran tanto para entrenamiento como para test. Tiene 2 variables independientes, las entradas, y dos dependiente, las salidas. Estas salidas serán 1 si pertenecen a esa clase y 0 si no pertenecen a esta, por lo que sólo podrá tener un valor positivo una de las salidas. Esto representa la clase a la que pertenece.

##### 3.1.2. Iris.

Esta base de datos está compuesta por 112 patrones de entrenamiento y 38 patrones de test. Las clases son tres especies distintas de la flor iris, de manera que para cada flor se extraen cuatro medidas o variables de entrada (longitud y ancho de los pétalos y los sépalos, en cm). Las tres especies a distinguir son iris setosa, iris virginica e iris versicolor.

##### 3.1.3. Digits.

Esta base de datos está compuesta por 1274 patrones de entrenamiento y 319 patrones de test. Está formada por un conjunto de dígitos (del 0 al 9) escritos a mano por 80 personas distintas, y ajustados a una rejilla cuadrada de  $16 \times 16$  píxeles. Aunque las imágenes originales estaban en escala de grises, éstas fueron binarizadas, con un valor de umbral fijo 1. Cada uno de los píxeles forman parte de las variables de entrada (256 variables) y las clases se corresponden con el dígito escrito (0, 1, ..., 9, con un total de 10 clases).



### 3.2. Descripción de los valores de los parámetros considerados.

Para la la topología de la red neuronal se han considerado los siguientes valores.

- Número de capas ocultas. Se generarán redes de una y dos capas ocultas.
- Número de neuronas por capa oculta. Se van a generar cada una de las configuraciones anteriores con 5, 10, 50 y 75 neuronas.

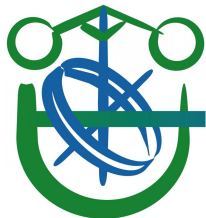
La capa de entrada tendrá tantas neuronas como entradas hay en la base de datos y la de salida como clases haya (representación 1-de-k).

La actualización de pesos podremos controlarla a través de los valores de  $\eta$  y  $\mu$ .

- Tasa de aprendizaje ( $\eta$ ). Controla el tamaño de los pasos que se van dando, que no sean muy grandes o muy pequeños. El valor deberá ser entre 0 y 1. Si este valor es demasiado grande, se pueden provocar oscilaciones y si es demasiado pequeño, se necesitaran muchas iteraciones. Se utilizará un valor de 0,1.
- Factor de momento ( $\mu$ ). Controla el efecto del momento, que hace que se pueda escapar de óptimos locales a través de la “inercia”. Deberá tener valores entre 0 y 1, se utilizará uno de 0,9.

Además se utilizará sesgo en las neuronas de nuestra red neuronal.





### 3.3. Resultados obtenidos.

#### 3.3.1. Problema XOR.

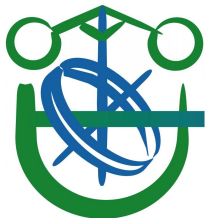
Usaremos una topología con dos capas ocultas de 100 neuronas cada una porque fue la mejor configuración obtenida.

	Error de entrenamiento		Error de test		CCR de entrenamiento		CCR de test	
	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$
-f 0	0.0555	0.0996	0.0555	0.0996	90	20	90	20
-f 0 -s	0.0028	0.0005	0.0028	0.0005	100	0	100	0
-f 1 -s	0.0027	0.0002	0.0026	0.0002	100	0	100	0

Como podemos ver la mejor configuración es con entropía y función de activación softmax. Estos resultados han sido obtenidos con el algoritmo off-line. Ahora veremos si obtenemos mejores resultados con el algoritmo on-line o el off-line.

	Error de entrenamiento		Error de test		CCR de entrenamiento		CCR de test	
	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$
on-line	0.0007	0.0003	0.0007	0.0003	100	0	100	0
off-line	0.0027	0.0002	0.0026	0.0002	100	0	100	0

Se puede observar que obtenemos mejores resultados con la versión on-line. Para la obtención de estos datos se han utilizado siempre sesgo y los valores de factor de momento y tasa de aprendizaje por defecto.



### 3.3.2. Iris.

		Error de entrenamiento		Error de test		CCR de entrenamiento		CCR de test	
		Media	$\sigma$	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$
L = 1	n:5	4.e-6	3.e-6	4.e-6	3.e-6	35.36	12.33	35.26	13.79
	n:10	3.e-6	3.e-6	3.e-6	3.e-6	34.11	1.731	33.16	2.105
	n:50	6.e-6	2.e-6	6.e-6	2.e-6	33.93	0.565	32.63	1.289
	n:75	6.e-6	3.e-6	6.e-6	2.e-6	28.39	6.222	27.37	9.503
L = 2	n:5	4.e-6	1.e-6	4.e-6	1.e-6	33.04	0	34.21	0
	n:10	4.e-6	3.e-6	4.e-6	3.e-6	33.04	0	34.21	0
	n:50	4.e-6	2.e-6	4.e-6	2.e-6	33.21	0.357	33.68	1.053
	n:75	4.e-6	2.e-6	5.e-6	2.e-6	34.46	4.056	33.68	6.093

Podemos ver que la mejor configuración de topología es aquella con una capa oculta con 10 neuronas.

		Error de entrenamiento		Error de test		CCR de entrenamiento		CCR de test	
		Media	$\sigma$	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$
-f 0		0.0899	0.0029	0.0792	0.0029	91.429	0.9105	89.474	0
-f 0 -s		0.0399	0.0028	0.0287	0.0023	95	0.7143	97.895	1.9693
-f 1 -s		0.0315	0.0009	0.0247	0.0014	96.429	0	96.316	1.2892

Como podemos ver la mejor configuración es con entropía y función de activación softmax. Estos resultados han sido obtenidos con el algoritmo off-line. Ahora veremos si obtenemos mejores resultados con el algoritmo on-line o el off-line.



	Error de entrenamiento		Error de test		CCR de entrenamiento		CCR de test	
	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$	Media	$\sigma$
on-line	1.0417	0.4921	1.0412	0.5093	33.036	0	34.211	0
off-line	0.0315	0.0009	0.0247	0.0014	96.429	0	96.316	1.2892

Se puede observar que obtenemos mejores resultados con la versión off-line. Para la obtención de estos datos se han utilizado siempre sesgo y los valores de factor de momento y tasa de aprendizaje por defecto.

### 3.3.3. Digits.

Se ha podido ver que la mejor configuración de topología es aquella con una capa oculta con 50 neuronas. La mejor configuración es con entropía y función de activación softmax. Estos resultados han sido obtenidos con el algoritmo off-line. Ahora veremos si obtenemos mejores resultados con el algoritmo on-line o el off-line.

Se ha podido observar que obtenemos mejores resultados con la versión on-line. Para la obtención de estos datos se han utilizado siempre sesgo y los valores de factor de momento y tasa de aprendizaje por defecto.

No se ha realizado la combinación de la función de error entropía cruzada con la función de activación sigmoide en la capa de salida porque ésta no cumple el requisito de que las salidas de las neuronas sumen 1 y es necesario para la entropía cruzada.



### 3.4. Análisis de resultados.

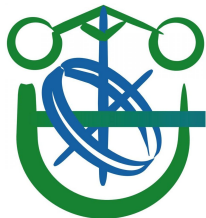
#### 3.4.1. Problema XOR.

Para esta base de datos el mejor modelo obtenido ha sido a través de una topología con dos capas ocultas de 100 neuronas cada una, utilizando como función de error la entropía cruzada y como función de activación la softmax. Además el algoritmo usado ha sido el on-line. A continuación mostraremos la matriz de confusión en test del mejor modelo de red neuronal obtenido para esta base de datos.

		Clase real	
		Verdadero	Falso
Clase estimada	Verdadero	2	0
	Falso	0	2

Comparando la salida obtenida y la esperada podemos ver el error de test cometido al usar este modelo con la topología especificada.

	Salida esperada	Salida obtenida
Patrón 1	1	0.998046
	0	0.00195413
Patrón 2	0	0.0025987
	1	0.997401
Patrón 3	1	0.997475
	0	0.00252533
Patrón 4	0	0.00205055
	1	0.997949
Error de test final		0.00073105



### 3.4.2. Iris.

Para esta base de datos el mejor modelo obtenido ha sido a través de una topología con una capa ocultas de 10 neuronas, utilizando como función de error la entropía cruzada y como función de activación la softmax. Además el algoritmo usado ha sido el off-line. A continuación mostraremos la matriz de confusión en test del mejor modelo de red neuronal obtenido para esta base.

		Clase real	
		Iris setosa	Iris virginica
Clase estimada	Iris setosa	13	0
	Iris virginica	0	10

		Clase real	
		Iris virginica	Iris versicolor
Clase estimada	Iris virginica	10	0
	Iris versicolor	2	13

		Clase real	
		Iris setosa	Iris versicolor
Clase estimada	Iris setosa	13	0
	Iris versicolor	0	13


Podemos observar como la mayor confusión que ha tenido nuestro modelo, ha sido diferenciando entre iris virginica e iris versicolor.



### 3.4.3. Digits.

Para esta base de datos el mejor modelo obtenido ha sido a través de una topología con una capa oculta de 50 neuronas, utilizando como función de error la entropía cruzada y como función de activación la softmax. Además el algoritmo usado ha sido el on-line. A continuación mostraremos la matriz de confusión en test del mejor modelo de red neuronal obtenido para esta base de datos.

		Clase real									
		0	1	2	3	4	5	6	7	8	9
Clase estimada	0	31						1	1		
	1		28								1
	2		1	32			3	1		1	1
	3				30		1			1	
	4		1			31					
	5	1			1		27				
	6							30			
	7					1			29		
	8		1						1	28	
	9	1	2		1				1	1	29

Podemos ver que el mayor error se ha producido en el 5 con el 2. Algún ejemplo de esto podría ser: .



#### 4. Referencias.

- [https://es.wikipedia.org/wiki/Perceptr%C3%B3n\\_multicapa](https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa)
- <http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/perceptron-multi-cap.html>
- [https://es.wikipedia.org/wiki/Red\\_neuronal\\_prealimentada](https://es.wikipedia.org/wiki/Red_neuronal_prealimentada)
- Auer, Peter; Harald Burgsteiner; Wolfgang Maass (2008). «A learning rule for very simple universal approximators consisting of a single layer of perceptrons». *Neural Networks* 21 (5): 786-795.
- Roman M. Balabin, Ravilya Z. Safieva, and Ekaterina I. Lomakina (2007). «Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction». *Chemometr Intell Lab* 88 (2): 183-188.
- <http://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit>