Aprendizaje supervisado en redes multicapa

Avetta, Gastón
Redes neuronales

30 de octubre de 2023

1. Análisis de componente principal

Para este ejercicio, se utiliza la red neuronal representada en la figura 1.1, con cuatro neuronas en la capa de entrada y una de salida, donde la salida es la combinación lineal de las entradas,

$$V = \sum_{i=1}^{4} w_i \cdot x_i.$$

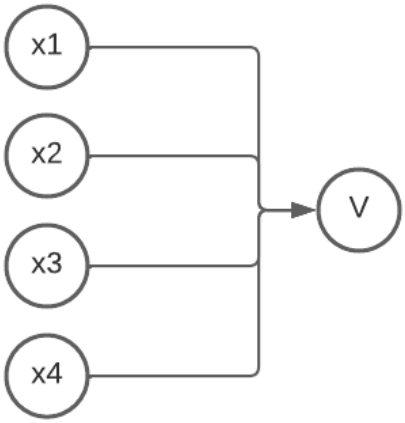


Figura 1.1: Red neuronal utilizada.

Además, las entradas provienen de una distribución de probabilidad gaussiana

$$P(\vec{x}) = \frac{e^{-\frac{1}{2}\vec{x}^T \Sigma^{-1} \vec{x}}}{(2\pi)^2 \sqrt{\det(\Sigma)}} \tag{1.1}$$

con matriz de correlación

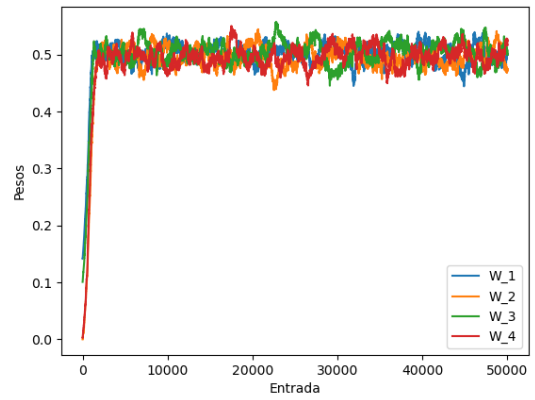
$$\Sigma = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix} \tag{1.2}$$

Por cada entrada $\vec{x}$ que pasa por la red neuronal en cuestión, obtenemos la salida V y luego modificamos las conexiones según la regla de Oja,

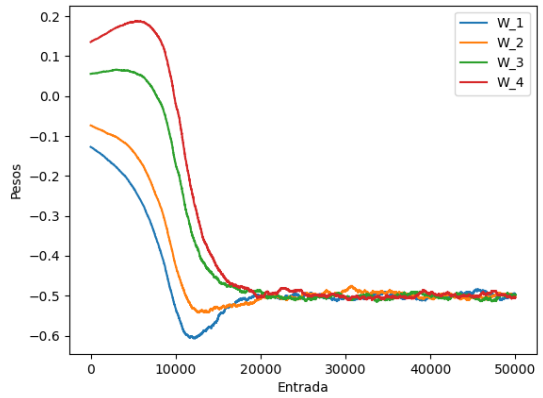
$$\Delta w_i = \eta V \cdot (x_i - V \cdot w_i), \tag{1.3}$$

donde η es el *learning rate* de la red neuronal.

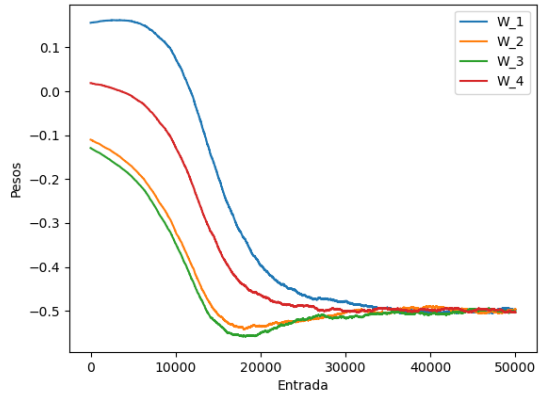
Entrenamos a la red con 50.000 entradas generadas con la distribución de probabilidad indicada. En la figura 2.2 vemos el valor de los pesos de cada una de las cuatro conexiones a medida que son modificados por las distintas salidas obtenidas.



(a) Learning rate $= 5 \cdot 10^{-4}$.



(b) Learning rate $= 10^{-4}$.



(c) Learning rate $= 5 \cdot 10^{-5}$.

Figura 1.2: Evolución de los pesos de las distintas conexiones de la red a partir de 50.000 entradas para distintos valores de learning rate.

En todos los casos analizados, los cuatro parámetros W_i convergen al mismo valor, ya sea $\frac{1}{2}$ o $-\frac{1}{2}$. Por otro lado, vemos que, a mayor *learning rate*, la convergencia al valor final se da más rápido (con menos entradas) pero resulta más ruidosa, oscilando con mayor amplitud alrededor del valor de convergencia en media.

Analizando la matriz de correlación de la distribución de probabilidad (1.2), vemos que sus autovalores son 5 y, con multiplicidad 3, 1. El autovector correspondiente al mayor autovalor es el vector $(1, 1, 1, 1)$. Sabemos que, modificando los pesos de la red según la regla de Oja (1.3), éstos convergen (en media) al autovector de norma unitaria correspondiente al mayor autovalor. Normalizando el vector $(1, 1, 1, 1)$ obtenemos el vector $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ o su opuesto, $(-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, lo cual explica los resultados observados en la figura 2.2.

2. Feature mapping: red de Kohonen

En este ejercicio, utilizamos una red neuronal que recibe como entrada un vector bidimensional y lo propaga a la capa de salida, compuesta por diez neuronas que representan puntos sobre una línea (unidimensional).

Las distintas entradas bidimensionales surgen de una distribución de probabilidad uniforme sobre un semicírculo entre los radios $r_1 = 0{,}9$ y $r_2 = 1{,}1$, tal como se ilustra en la figura 2.1, donde se grafican 10.000 entradas.

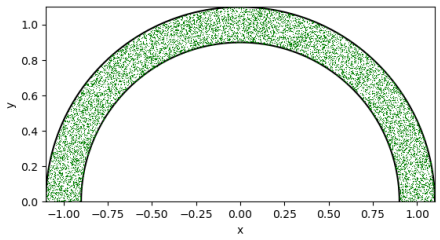


Figura 2.1: Distribución de probabilidad de las entradas bidimensionales de la red neuronal.

Aplicamos el **algoritmo de Kohonen** a esta red: inicializamos aleatoriamente los pesos de cada conexión entre las capas de entrada y salida y luego, por cada una de las 10.000 entradas de la figura 2.1, obtenemos

a la neurona ganadora de la capa de salida y aplicamos la regla de modificación, dada por la expresión

$$\Delta\vec{w}_i = \eta \cdot \Omega(i, i^*) \cdot (\vec{x}_\mu - \vec{w}_i)$$

donde i^* es la neurona ganadora, es decir, aquella cuyo vector de conexiones $\vec{w}_i$ es el más cercano a la entrada $\vec{x}_\mu$ y $\Omega(i, i^*)$ es la *función de vecindad* entre cada una de las neuronas de la capa de salida (i) y la neurona ganadora, dada por

$$\Omega(i, i^*) = e^{\frac{-(i-i^*)^2}{2\sigma^2}}.$$

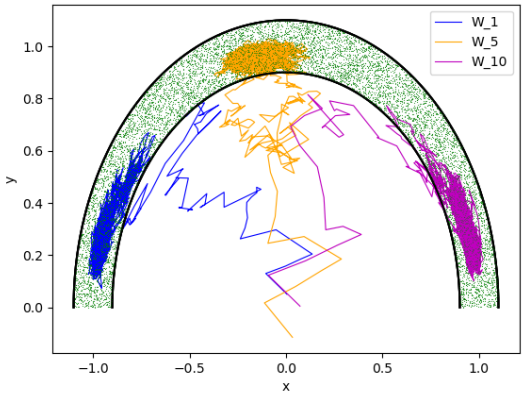
La función de vecindad es siempre positiva y menor o igual a 1, siendo siempre 1 para la neurona ganadora. A mayor σ, este factor afecta más a las demás neuronas de la capa de salida, desplazando los vectores $\vec{w}_i$ hacía la entrada $\vec{x}_\mu$ en cuestión, mientras que a medida que decrece σ, las neuronas de la capa de salida que no resultan ganadoras cada vez se modifican menos.

El objetivo de este algoritmo es obtener una disposición de los puntos representados por las diez neuronas de salida (cuyas coordenadas bidimensionales están dadas por su respectivo vector $\vec{w}_i$) para poder mapear las distintas entradas bidimensionales a puntos sobre una línea unidimensional conservando la noción de proximidad entre distintos puntos en ambos espacios.

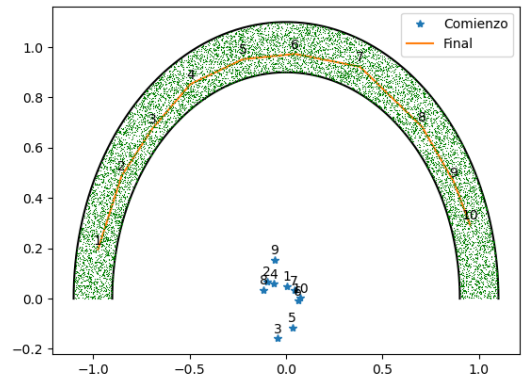
Con un *learning rate* (η) de 0,2 y un σ que decrece a medida que van pasando las entradas según la expresión

$$\sigma = \frac{10}{\sqrt[3]{\mu}}, \tag{2.1}$$

donde μ es el subíndice de las distintas entradas utilizadas en el entrenamiento (las 10.000 graficadas en la figura 2.1), se obtienen los resultados de la figura 2.2.



(a) Posición del vector $\vec{w}_i$ a medida que pasan las distintas entradas para i = 1, 5, 10.

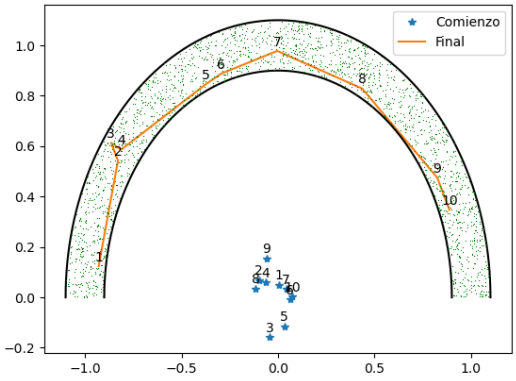


(b) Posición inicial y final de cada uno de los diez vectores $\vec{w}_i$ de las neuronas de la capa de salida.

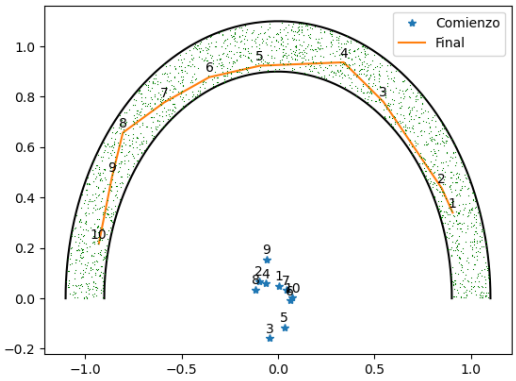
Figura 2.2: Resultados para 10.000 entradas, η = 0,2 y σ según la expresión 2.1.

En la subfigura 2.2a se puede ver que, en un comienzo, para σ grande, los distintos vectores son modificados de forma similar, más allá de estar distanciados sobre la línea representada por las neuronas de la capa de salida; luego, a medida que decrece σ, cada uno de ellos se desplaza por separado hacia su posición final. En la subfigura 2.2b podemos ver como el resultado final parece cumplir con el objetivo planteado: los distintos puntos representados por las neuronas de salida se distribuyen razonablemente en el espacio bidimensional de donde se obtienen las entradas.

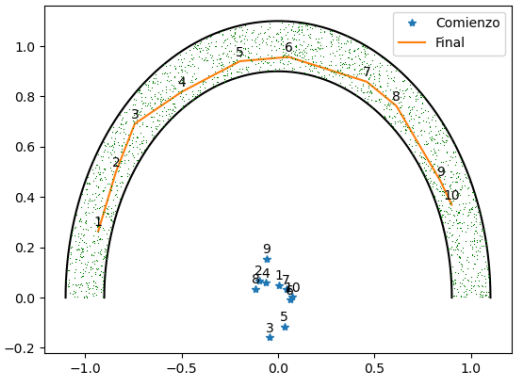
Veamos ahora qué ocurre al contar con 2.000 entradas y con distintos valores del learning rate



(a) $\eta = 0{,}8$.



(b) $\eta = 0{,}4$.

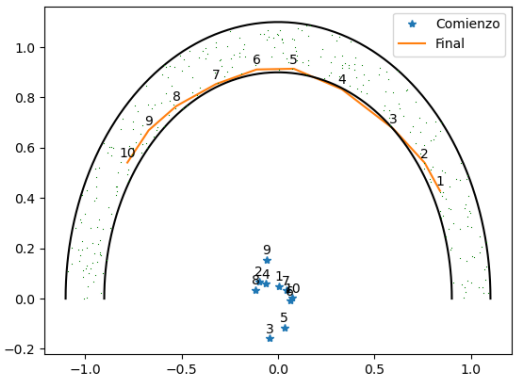


(c) $\eta = 0{,}2$.

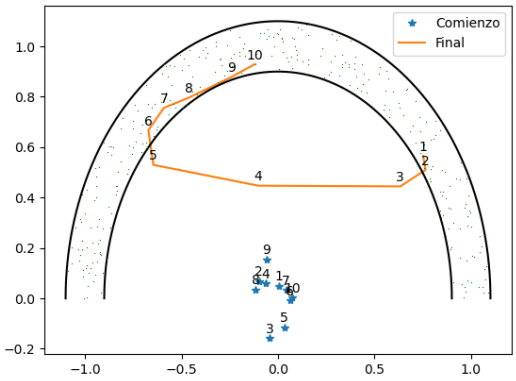
Figura 2.3: Resultados obtenidos con distintos valores de learning rate para 2.000 entradas y σ según la expresión 2.1.

Uno podría pensar que, al tener menos entradas, debería aumentar el *learning rate* para obtener una solución aceptable. De todas formas, de todos los resultados observados en la figura 2.3, el de menor η parece ser el más prolijo mientras que, para $\eta = 0{,}8$, la solución comienza a fallar, como se puede ver con las neuronas 2, 3 y 4.

Por último, en un caso con solo 300 entradas y con un *learning rate* razonable, también se puede obtener una solución aceptable, tal como se ve en la figura 2.4.



(a) $\eta = 0{,}4$.



(b) $\eta = 0{,}5$.

Figura 2.4: Resultados obtenidos con distintos valores de learning rate para 300 entradas y σ según la expresión 2.1.

En la subfigura 2.4a vemos que, aún teniendo tan pocas entradas, obtenemos una solución razonable, aunque no tan buena como en los casos anteriores. Luego, en la figura 2.4b vemos que, si intentamos mejorar dicha solución aumentando el *learning rate*, esto lleva a un resultado final completamente incorrecto, por lo cual hay que ser cuidadosos a la hora de elegir este parámetro.

Códigos utilizados

A continuación, se adjunta una carpeta con los códigos utilizados para resolver ambos ejercicios: https://drive.google.com/drive/folders/16xur35s0H4hDK_XYSQRVDxFSRtDSN6Pv?usp=sharing.