

Aprendizaje no supervisado

Pablo Chehade

pablo.chehade@ib.edu.ar

Redes Neuronales, Instituto Balseiro, CNEA-UNCuyo, Bariloche, Argentina, 2023

EJERCICIO 1

Se entrenó de manera no supervisada una red neuronal lineal de una sola capa con cuatro entradas y una salida. Los datos de entrada presentan una distribución gaussiana con matriz de correlación Σ

$$\Sigma = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}.$$

El autovector asociado al mayor autovalor de Σ es $\vec{v} = \frac{1}{2}(1, 1, 1, 1)$.

Se partió de un vector de pesos $\vec{w} = (w_1, w_2, w_3, w_4)$, con cada componente dada por una distribución uniforme con máximo 0,01. Luego, se aplicó la regla de Oja

$$\Delta w_j = \eta V(\xi_j - V w_j).$$

Aquí, η representa la tasa de aprendizaje, V es la salida de la red y ξ_j es la componente j del dato de entrada $\vec{\xi}$. En base a la teoría desarrollada para la regla de Oja, se espera que el vector de pesos \vec{w} tienda al autovector \vec{v} .

En la figura 1a, se ilustra la evolución del módulo de los pesos $|w_j|$, las modificaciones Δw_j y la diferencia entre w_j y v_j , para $\eta = 0,001$. Tras un período transitorio, se observa que las componentes de \vec{w} tienden hacia el valor 0,5, indicando convergencia hacia \vec{v} . Dependiendo de las condiciones iniciales, el vector \vec{w} tiende a alinearse en la dirección $+\vec{v}$ o $-\vec{v}$. Debido a esto se graficó el módulo de cada componente. Posteriormente, en estado estacionario, los pesos continúan modificándose. Por ello, en la figura 1b, se procesan los datos anteriores realizando un promedio en cada paso de tiempo de los 200 pasos adyacentes. Esto resulta en un comportamiento más suave y una reducción notable en las variaciones de Δw_j de hasta un orden de magnitud. Luego de este procesamiento de los datos se confirma nuevamente que \vec{w} tiende a \vec{v} .

Por último, en la figura 1c, se evolucionó el sistema con $\eta = 0,01$. El sistema muestra una convergencia más rápida y variaciones Δw_j de mayor magnitud. Además, se determinó que existe un valor superior límite para η , más allá del cual el sistema tiende a diverger.

EJERCICIO 2

Se entrenó una red neuronal de Kohonen con dos neuronas de entrada y diez neuronas de salida alineadas en

una línea. La red se alimenta mediante una distribución de entrada definida como:

$$P(\xi) = P(r, \theta) = \begin{cases} \text{constante} & \text{si } r \in [0, 0.9, 1, 1], \theta \in [0, \pi] \\ 0 & \text{si no,} \end{cases}$$

donde r y θ son las coordenadas polares del vector $\vec{\xi}$.

Los pesos iniciales w_{ij} , donde i representa la neurona de salida y j la dimensión del espacio de entrada, se asignan aleatoriamente con una distribución uniforme con un valor máximo de 0,1. En cada iteración del entrenamiento, se selecciona un vector de entrada $\vec{\xi}$, y se identifica la neurona "ganadora" i^* como aquella que minimiza la distancia euclídea con el vector de entrada, es decir

$$i^* = \operatorname{argmin}_i |\vec{w}_i - \vec{\xi}|,$$

donde $\vec{w}_i = (w_{i1}, w_{i2})$. Posteriormente, se actualizan los pesos de todas las neuronas utilizando la regla

$$\Delta \vec{w}_i = \eta \Gamma(i, i^*) (\vec{\xi} - \vec{w}_i),$$

donde

$$\Gamma(i, i^*) = \exp\left(-\frac{(i - i^*)^2}{2\sigma^2}\right)$$

es la función de vecindad y η es la tasa de aprendizaje.

Se realizó un análisis sobre la evolución de los pesos de la red variando los parámetros η y σ , y se observó cómo afectan la convergencia, haciendo incapié cómo se modifican los pesos durante el entrenamiento.

Se observaron diferentes comportamientos al variar los parámetros η y σ . En todos los casos se estudió la convergencia de los pesos \vec{w}_i mediante su evolución graficada en el espacio de los datos de entrada (figura 2)

1. $\eta = 0,1$ y $\sigma = 1$: Inicialmente los pesos \vec{w}_i son cercanos entre sí. Luego se desplazan hacia las regiones donde $P(\vec{\xi})$ es no nula. Finalmente, los pesos se organizan sobre la región, organizados de acuerdo a la disposición de las neuronas sobre la línea.
2. $\eta = 0,1$ y $\sigma = 2$: En este caso, los pesos se mueven más juntos en comparación con el caso anterior debido a un mayor valor de σ , que incrementa el efecto de la función de vecindad. Esto se puede entender más claramente en el caso extremo en el que $\sigma \rightarrow \infty$. En tal caso, $\Gamma \rightarrow 1$ y todas las neuronas se actualizan en la misma magnitud hacia la posición del dato de entrada $\vec{\xi}$. En cuanto al estado final, incluso con una

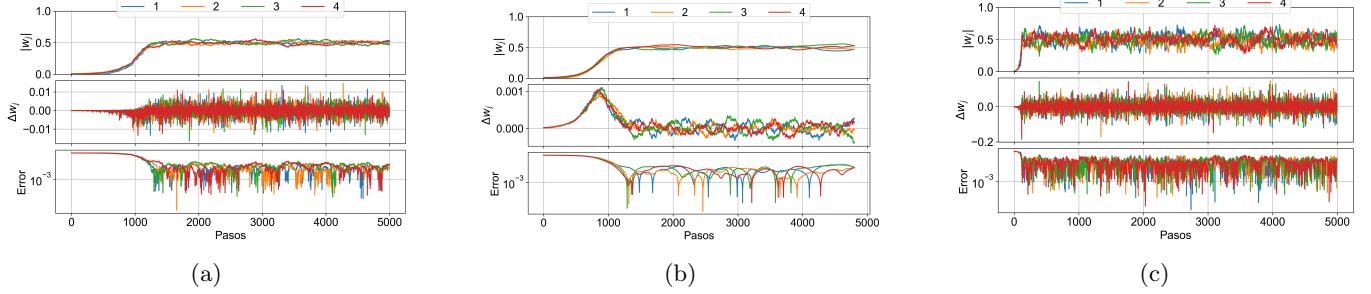


Figura 1: Evolución del módulo de los pesos w_j , las modificaciones Δw_j y la diferencia entre w_j y v_j durante el entrenamiento de la red neuronal. En **a** se empleó una tasa de aprendizaje de $\eta = 0,001$. En **b** se empleó la misma tasa de aprendizaje pero además se procesaron los datos promediando en cada paso de tiempo sobre los 200 pasos adyacentes. En **c** se empleó una tasa de aprendizaje de $\eta = 0,01$.

gran cantidad de pasos de entrenamiento no se llega al mismo estado que antes. Una forma de resolver este problema podría ser aprovechando el comportamiento visto en el caso anterior y disminuir progresivamente σ a medida que se realiza el entrenamiento.

3. $\eta = 0,01$ y $\sigma = 1$: Con una tasa de aprendizaje reducida, la evolución es más lenta. Además, los pesos

muestran variaciones menores durante el entrenamiento. A pesar de la lentitud, con suficientes iteraciones el sistema converge a la misma condición que el primer caso.

El estudio sugiere que la elección de η y σ es crucial. Un η más bajo lleva a una convergencia más lenta pero con menor ruido, mientras que un σ más alto promueve una adaptación global pero menos detallada de los pesos.

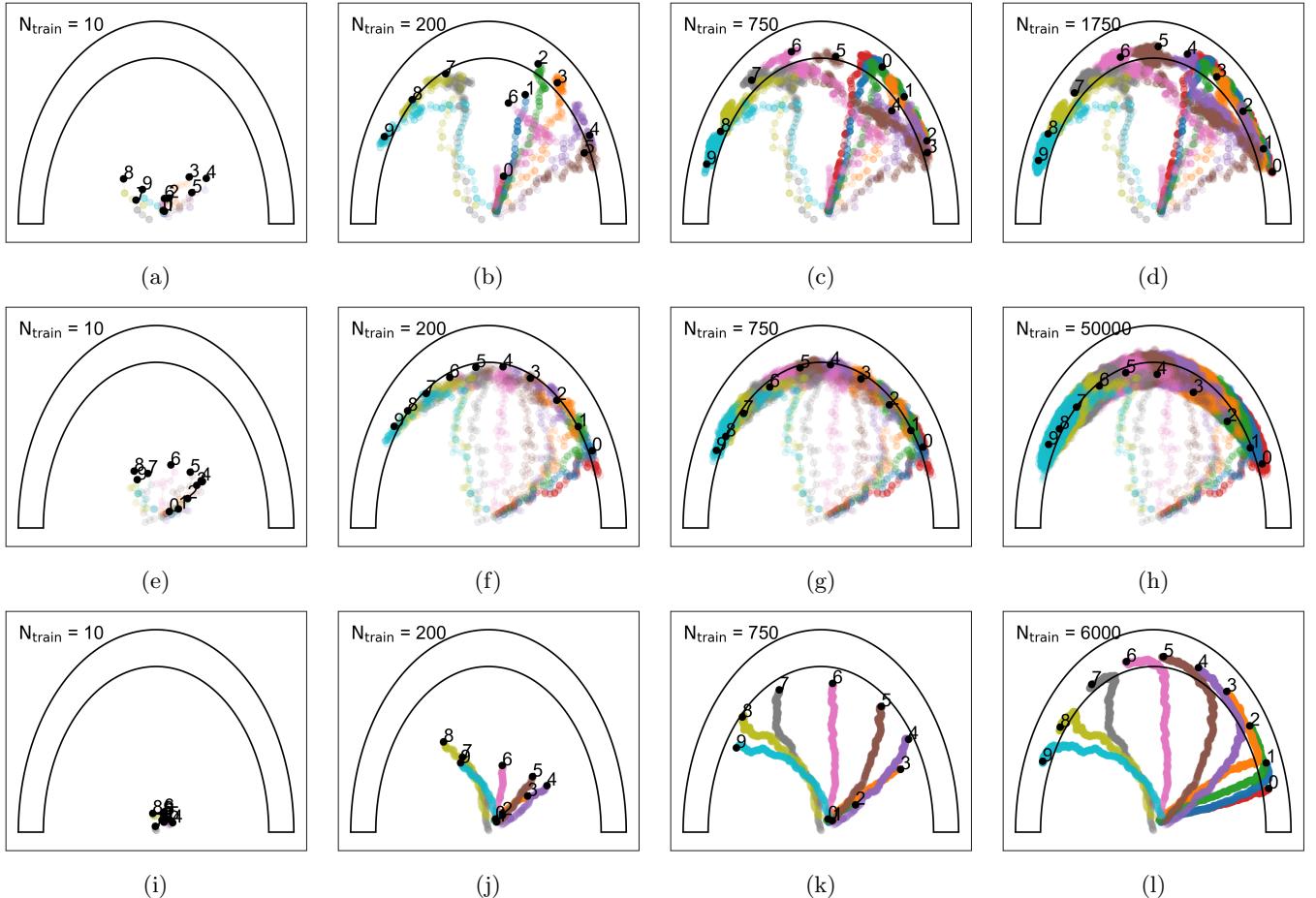


Figura 2: Evolución de los peso \vec{w}_i con $i = 1, 2, 3, \dots, 10$ en el espacio de los datos de entrada para distintos pasos de entrenamiento N_{train} . Se empleó a-d $\eta = 0,1$, $\sigma = 1$, e-h $\eta = 0,1$, $\sigma = 2$, i-l $\eta = 0,01$, $\sigma = 1$. La región semicircular corresponde a la región donde se encuentran los datos de entrada. En negro se grafica la posición de los pesos finales para cada valor de N_{train} , junto al número correspondiente de neurona. La numeración de las neuronas se realiza en base a que se encuentran dispuestas en una línea. En color se grafican las posiciones previas. No se grafican los ejes por simplicidad.

APÉNDICE

A continuación se desarrolla el código empleado durante este trabajo implementado en Python.

```

1 #Import libraries
2 import numpy as np
3 import matplotlib
4 import matplotlib.pyplot as plt
5 import tensorflow as tf

```