In [ ]:

Redes Neuronales - Aprendizaje no supervisado

Autor: Alexander Leaño Fecha: 29 de Noviembre de 2024

Introducción

El aprendizaje no supervisado es una rama del aprendizaje automático que se encarga de encontrar patrones en los datos sin la necesidad de tener etiquetas. En este trabajo se busca analizar el comportamiento de una red neuronal lineal de una capa y también red neuronal de Kohonen.

El codigo fuente del trabajo se puede encontrar en el siguiente notebook-github.

Problema 1: Red Neuronal Lineal de una capa

Se busca realizar el analisis de una red neuronal lineal con una sola capa de neuronas. La red neuronal tiene 4 entradas y una salida. La ecuación de la salida es:

$$V = \sum_{j=1}^{4} w_j \xi_j \tag{1}$$

La distribución de probabilidad de las entradas es una distribución Gaussiana con matriz de correlación Σ:

$$P(\bar{\xi}) = \frac{1}{(2\pi)^2 \sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2} \bar{\xi}^T \Sigma^{-1} \bar{\xi}\right) \tag{2}$$

Donde:

$$\Sigma = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix} \tag{3}$$

Para esto, se parte de pesos w_j aleatorios y pequeños para aplicar la regla de aprendizaje:

$$\Delta w_j = \eta V (\xi_j - V w_j) \tag{4}$$

Se busca comparar los valores asintóticos de los pesos con los autovectores de la matriz Σ. Para esto se busca hallar el autovector asociado al mayor de los autovales de la matriz de covarianza Σ para comparar con los pesos de la red. El autovector asociado al mayor autovector es:

$$\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

Nota: Como una ayuda para el cálculo va a ser útil utilizar la raíz cuadrada de la matriz de correlación:

$$\Sigma^{1/2} = \begin{pmatrix} 1.309 & 0.309 & 0.309 & 0.309 \\ 0.309 & 1.309 & 0.309 & 0.309 \\ 0.309 & 0.309 & 1.309 & 0.309 \\ 0.309 & 0.309 & 0.309 & 1.309 \end{pmatrix} \tag{5}$$

Resultados

En la Figura 1 se muestra la evolución del error cuadrático medio entre los pesos w_j y los componentes del mayor autovector de la matriz de correlación Σ en función de las iteraciones.

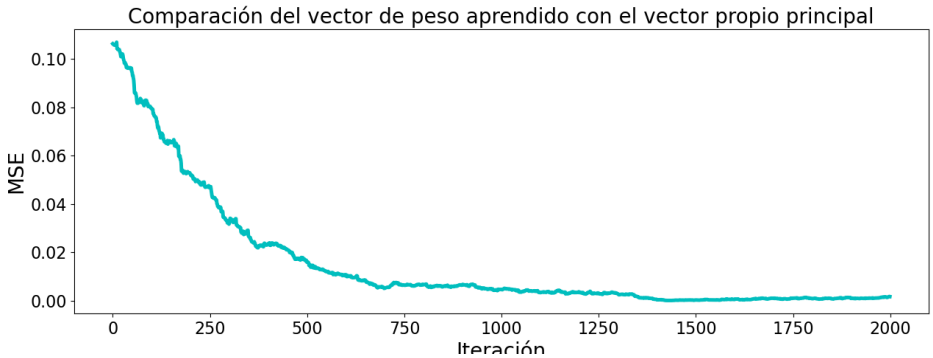

Figure 1: Error cuadrático medio entre el valor de los pesos de la red y el autovector asociado al autovalor más grande de la matriz de covarianza.

Analizando este resultado, se puede observar como el error disminuye rápidamente en las primeras iteraciones y luego se estabiliza en un valor cercano a cero. Esto indica que los pesos de la red convergen al autovector asociado al autovalor más grande de la matriz de covarianza Σ.

De esta manera podemos concluir que al utilizar este algoritmo de aprendizaje, de una sola salida, realizamos una reducción dimensional de los datos, ya que los pesos de la red se ajustan para que la salida de la red sea proporcional al autovector asociado al mayor autovalor de la matriz de covarianza Σ.

Problema 2: Red Neuronal de Kohonen con Entradas en Coordenadas Polares

Este problema investiga un mapa autoorganizado de Kohonen (SOM: *Self Organization Map*) con las siguientes especificaciones:

- **Neuronas de Entrada**: 2 (correspondientes a ξ_1 y ξ_2).
- **Neuronas de Salida**: 10, dispuestas linealmente.
- **Distribución de Entrada**:
  - $P(r, \theta) = \text{constante}$ si $r \in [0.9, 1.1]$ y $\theta \in [0, \pi]$,
  - $P(r, \theta) = 0$ en caso contrario.
  - r y θ son coordenadas polares derivadas como:
    - $r = \sqrt{\xi_1^2 + \xi_2^2}$,
    - $\theta = \arctan(\xi_2 / \xi_1)$.
- **Función de Vecindad**: Gaussiana:

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

Metodología

El SOM fue entrenado con:

- **Número de Neuronas**: 10,
- **Tasa de Aprendizaje**: $\eta = 0.01$,
- **Ancho de Vecindad (σ)**: $\sigma = \{2, 1, 0.5, 0.1\}$,
- **Número de Iteraciones**: $10,000$.
- **Pesos Iniciales**: ξ_1 se inicilizó con valores entre -1 y 1 equidistantes para completar el numero de neuronas. Mientras que ξ_2 se inicilizó con valores en 0.

Pasos de Simulación

1. **Muestreo de Entradas**:
   - Se generaron muestras aleatorias $\bar{\xi}$ de la distribución definida $P(r, \theta)$.
2. **Actualización de Pesos**:
   - Se actualizaron los pesos sinápticos de todas la $\mathbf{w}$ utilizando la regla de aprendizaje de Kohonen:

     $$\Delta w_i = \eta \cdot \Lambda(i, i^*) \cdot (\xi - w_i),$$

     donde i^* es el índice de la neurona ganadora. La neurona ganadora se determinón como la de menor distancia en norma 2 a la entrada.
3. **Función de Vecindad**:
   - Se ajustó el ancho gaussiano σ para analizar su impacto en la convergencia de los pesos.

Resultados

Los resultados para los distintos valores de σ se presentan en las figuras 2, 3, 4 y 5.

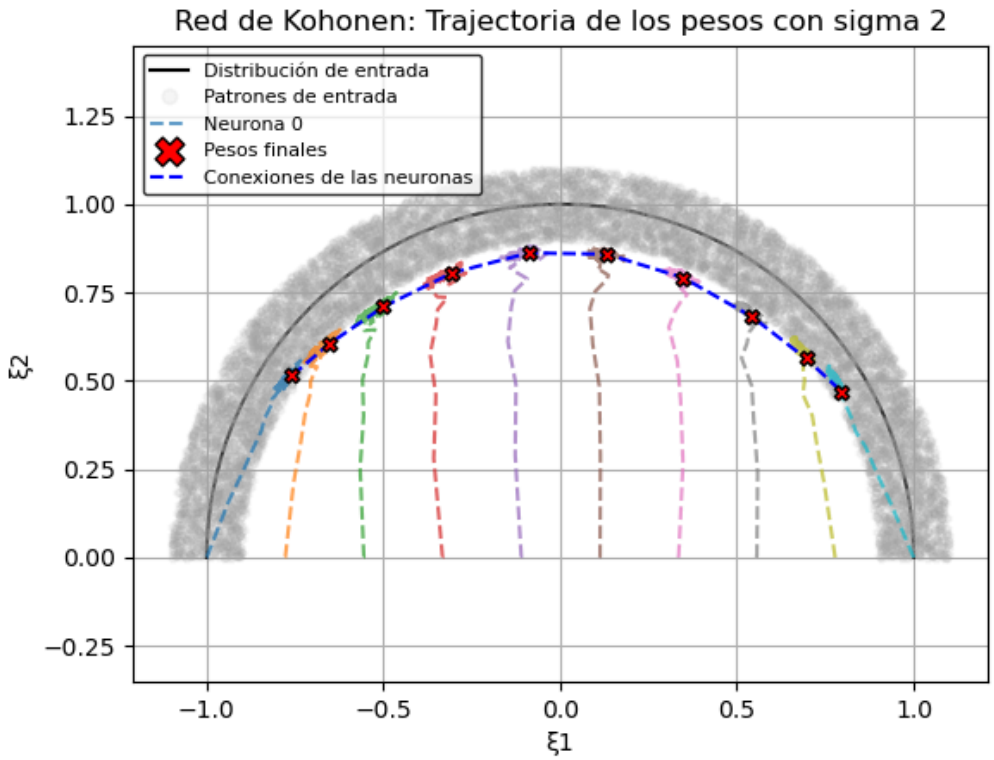

Figure 2: Aprendizaje con Red de Kohonen para coordenas polares con valor de sigma 2.

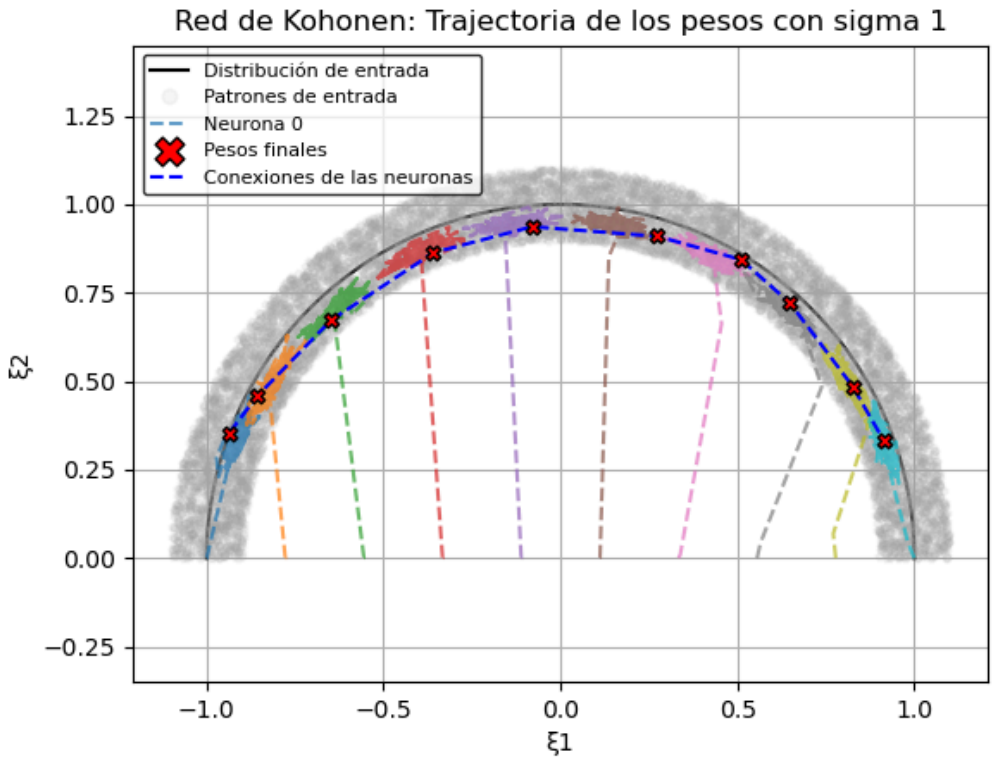

Figure 3: Aprendizaje con Red de Kohonen para coordenas polares con valor de sigma 1.

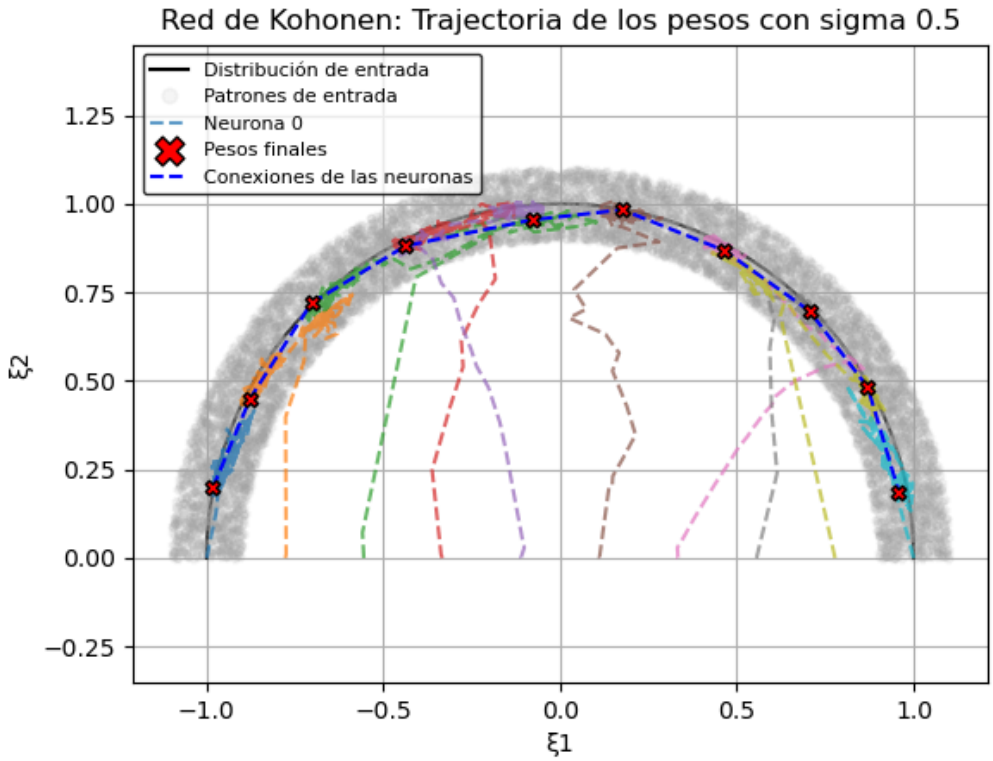

Figure 4: Aprendizaje con Red de Kohonen para coordenas polares con valor de sigma 0.5.

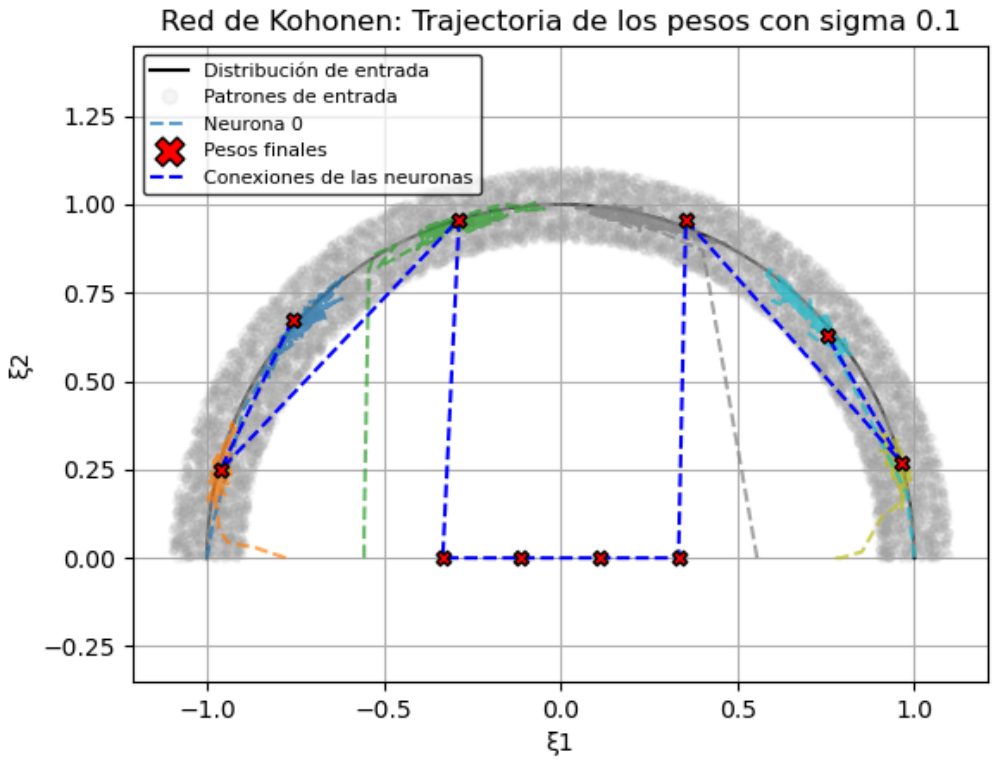

Figure 5: Aprendizaje con Red de Kohonen para coordenas polares con valor de sigma 0.1.

De los resultados obtenidos se puede observar que para σ 0.5 se obtiene suficiente flexibilidad para que los pesos de la red se ajusten a la distribución de probabilidad de las entradas. Sin embargo, al aumentar o disminuir el valor de σ se observa que la red no logra converger a la distribución de probabilidad de las entradas. Para σ 1 y 2 observamos que la influencia de la neurona ganadora en la actualización de los pesos es muy grande, lo que impide que la red se ajuste a la distribución de probabilidad de las entradas. Por otro lado para σ 0.1 tampoco se logra ajustar la red a la distribución de probabilidad de las entradas, pero en este caso la influencia de la neurona ganadora es muy pequeña. Para valor pequeños algunos pesos quedan sin actualizar ya que pocas veces fueron la neurona ganadora.

En conclución, el valor de σ es un hiperparámetro importante en el entrenamiento de una red de Kohonen, ya que determina la influencia de la neurona ganadora en la actualización de los pesos. Un valor adecuado de σ permite que la red se ajuste a la distribución de probabilidad de las entradas. Valores muy grandes no brindan la suficiente flexibilidad para que los pesos de la red se ajusten a la distribución de probabilidad de las entradas. Por otro lado, valores muy pequeños no permite la actualización de todos los pesos de la red, ya que pocas veces son la neurona ganadora y la actualización indirecta es debil por la función de vecindad.