Otras arquitecturas de Redes Neuronales

Dr. Mauricio Toledo-Acosta

Diplomado Ciencia de Datos con Python

Table of Contents

Autoencoders

- Una de las principales características de un autoencoder es tener una capa de salida con la misma dimensión que la de entrada. La idea es tratar de reconstruir, de manera exacta, cada dimensión al pasarla por la red.

Autoencoders

- Una de las principales características de un autoencoder es tener una capa de salida con la misma dimensión que la de entrada. La idea es tratar de reconstruir, de manera exacta, cada dimensión al pasarla por la red.
- La reconstrucción de los datos podría parecer una cuestión trivial, sin embargo, esto no es posible cuando el número de unidades en cada capa intermedia suele ser menor que el de la entrada (y salida).

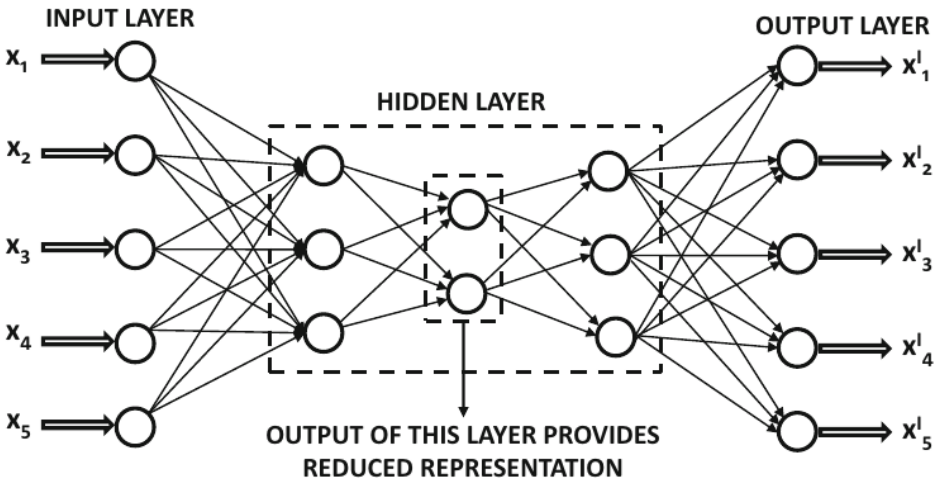
Autoencoders

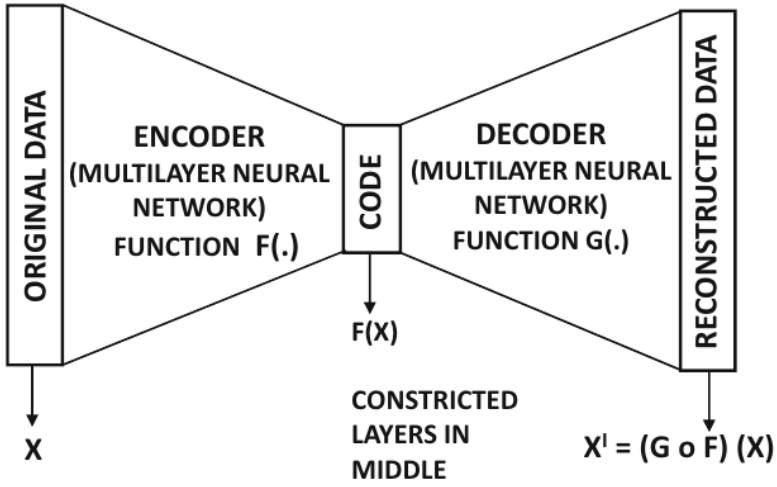
- Una de las principales características de un autoencoder es tener una capa de salida con la misma dimensión que la de entrada. La idea es tratar de reconstruir, de manera exacta, cada dimensión al pasarla por la red.
- La reconstrucción de los datos podría parecer una cuestión trivial, sin embargo, esto no es posible cuando el número de unidades en cada capa intermedia suele ser menor que el de la entrada (y salida).
- Las neuronas de las capas ocultas aprenden una representación reducida de los datos que les permita reconstruir la entrada.

Autoencoders

- Una de las principales características de un autoencoder es tener una capa de salida con la misma dimensión que la de entrada. La idea es tratar de reconstruir, de manera exacta, cada dimensión al pasarla por la red.
- La reconstrucción de los datos podría parecer una cuestión trivial, sin embargo, esto no es posible cuando el número de unidades en cada capa intermedia suele ser menor que el de la entrada (y salida).
- Las neuronas de las capas ocultas aprenden una representación reducida de los datos que les permita reconstruir la entrada.
- Esta reconstrucción, inherentemente, tiene pérdidas.

Autoencoders



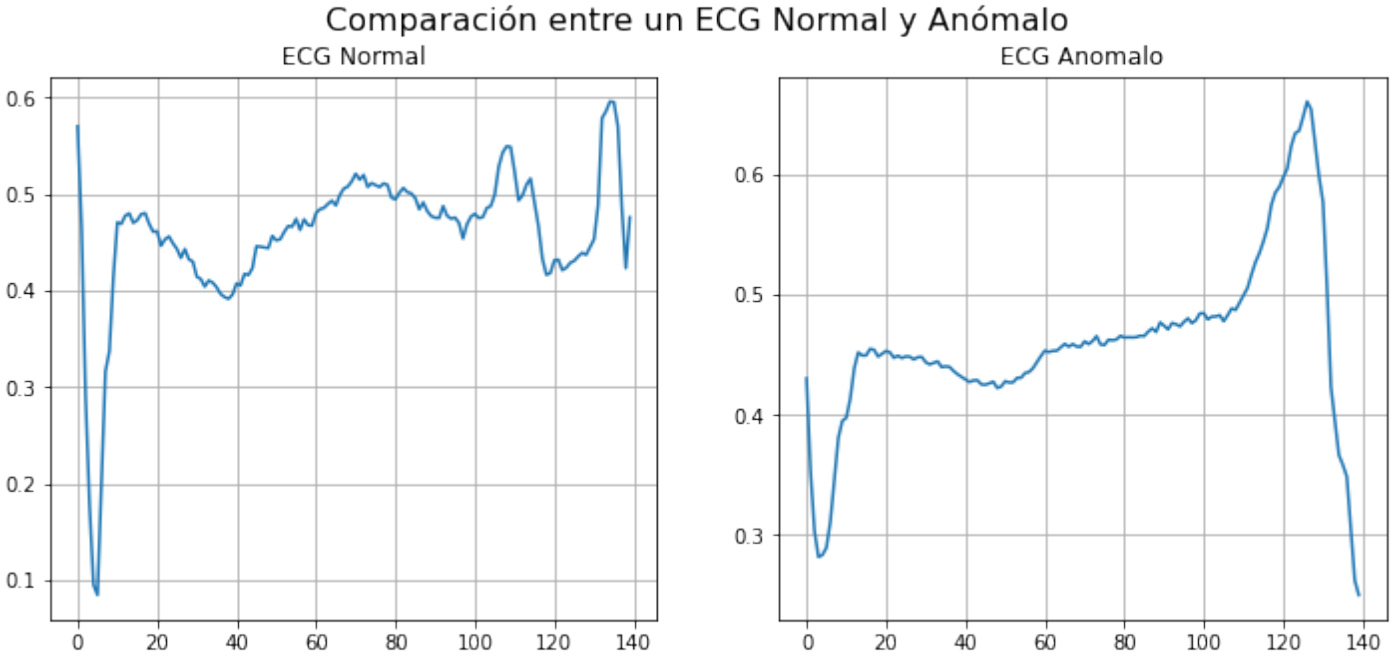


Aplicaciones

- Reducción de dimensionalidad.

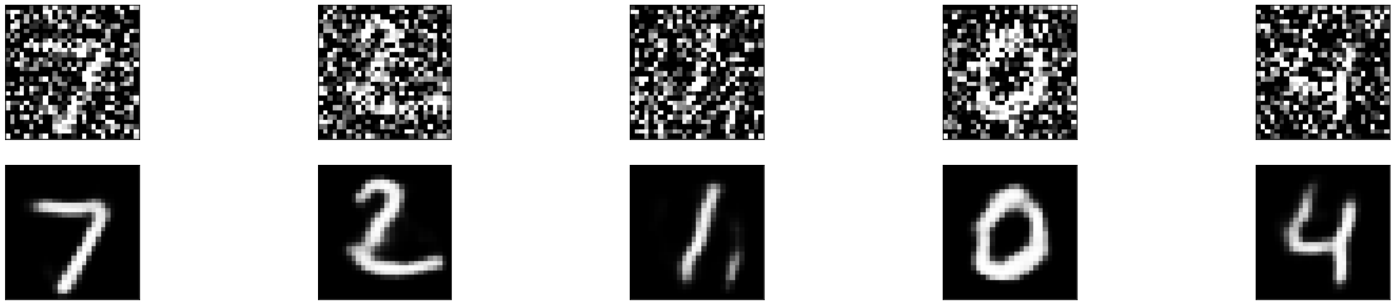
Aplicaciones

- Reducción de dimensionalidad.
- Detección de Anomalías (outliers)



Aplicaciones

- Reducción de dimensionalidad.
- Detección de Anomalías (outliers)
- Denoising

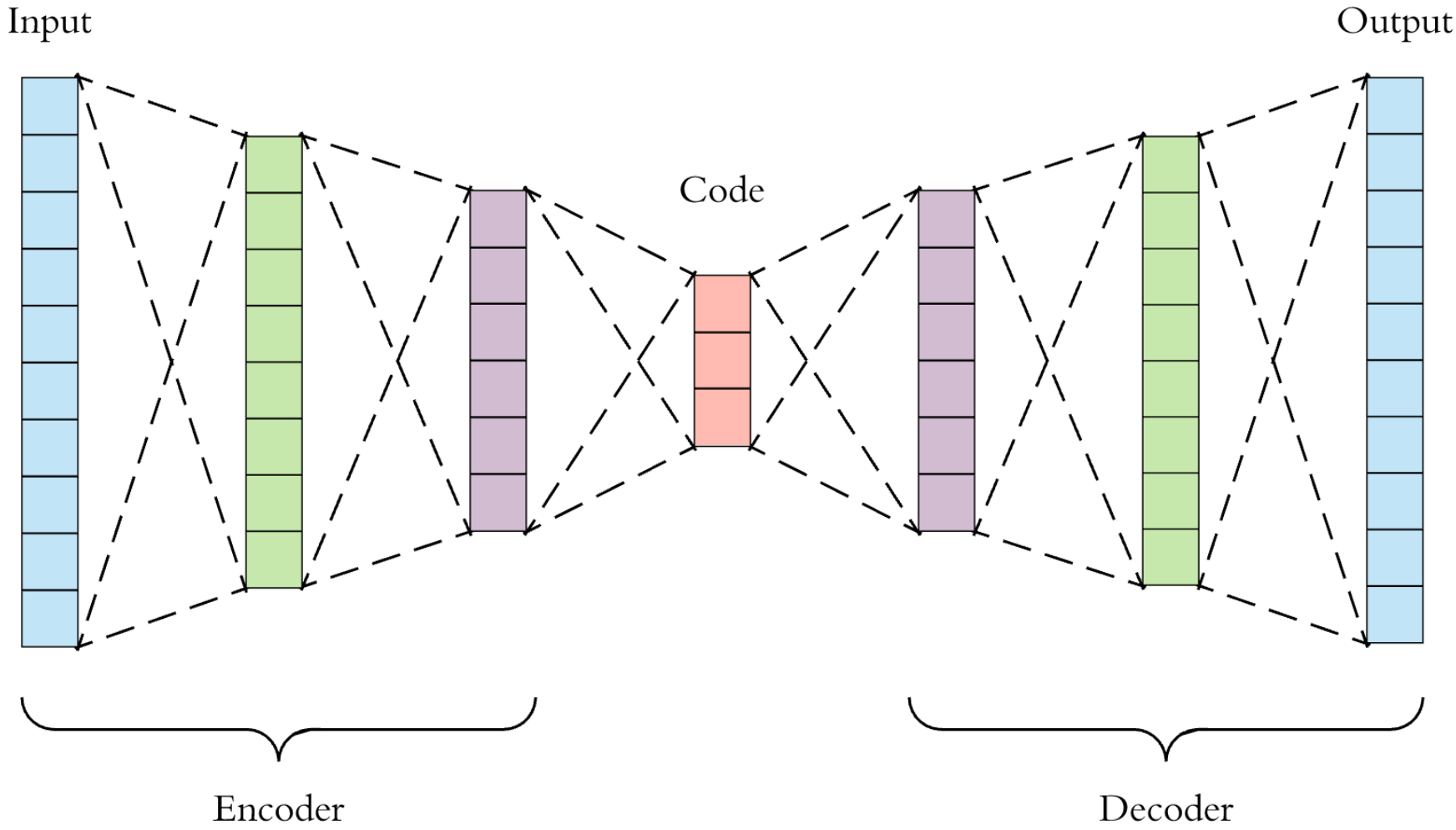


Aplicaciones

- Reducción de dimensionalidad.
- Detección de Anomalías (outliers)
- Denoising
- Compresión de imágenes.

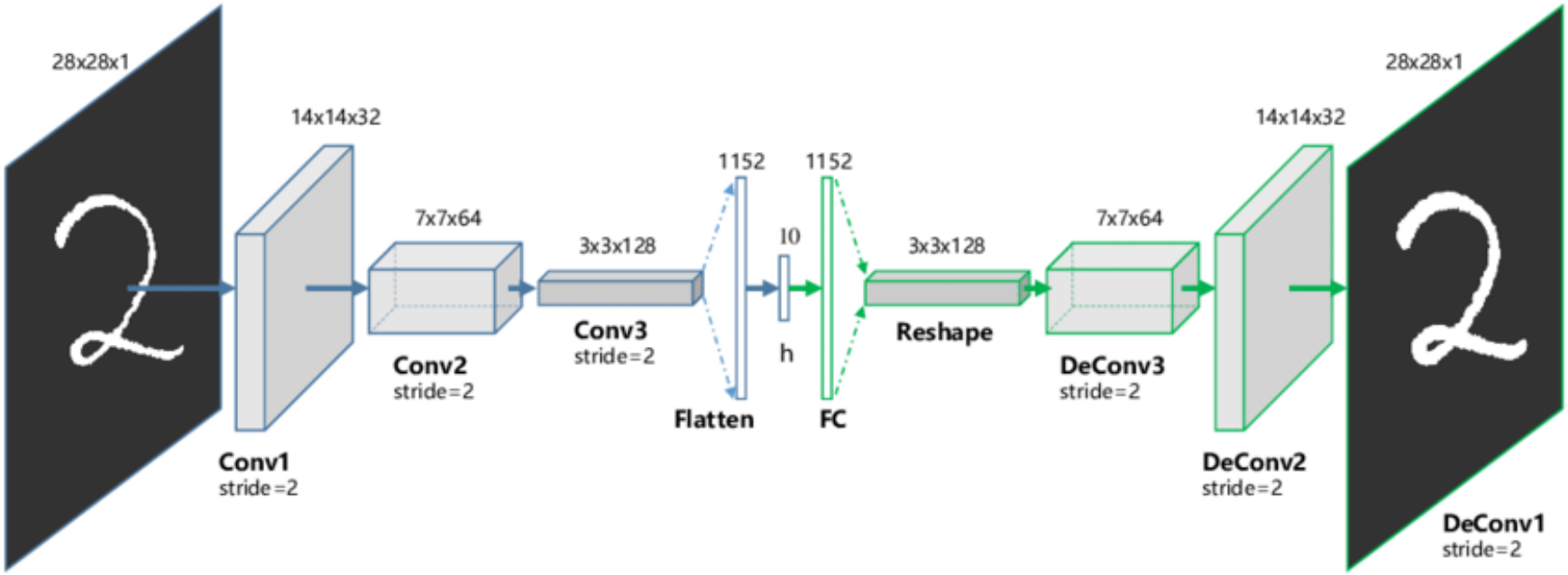
Autoencoders

- La arquitectura suele ser simétrica respecto a la capa oculta central.



Autoencoders

- La arquitectura suele ser simétrica respecto a la capa oculta central.



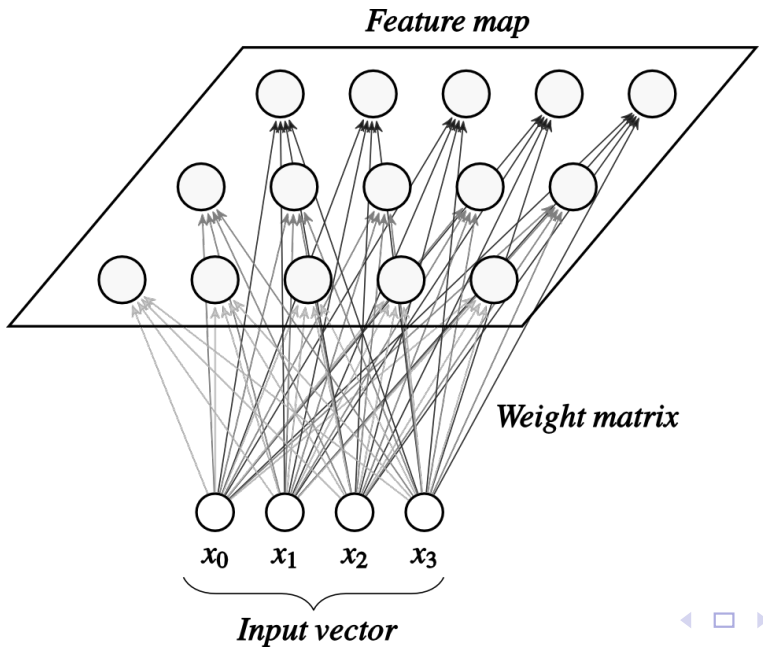
- De acuerdo a la tarea, puede ser que nos interese solamente la salida de la red, o la capa oculta latente.

Table of Contents

SOM: Self-organizing map

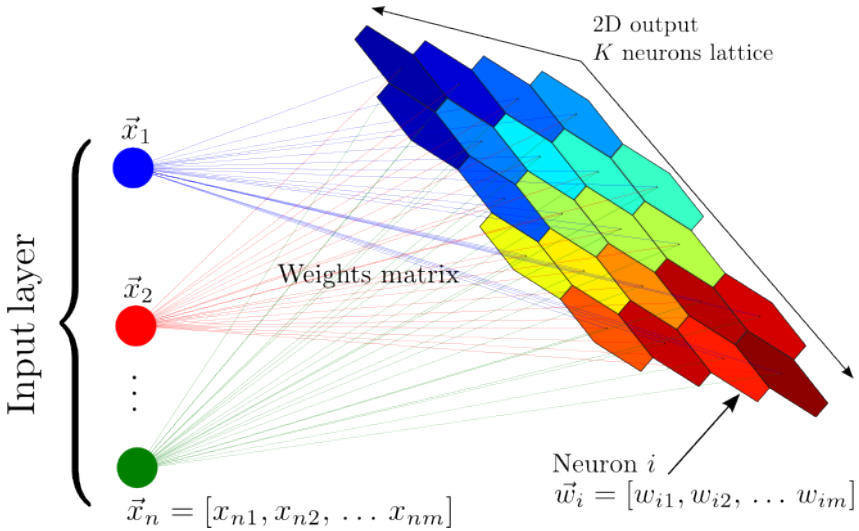
SOM

Las redes self-organizing map (SOM) son un algoritmo para mapear un espacio (usualmente de dimensión alta) a otro espacio (usualmente de dimensión baja) preservando la estructura topológica de los datos.



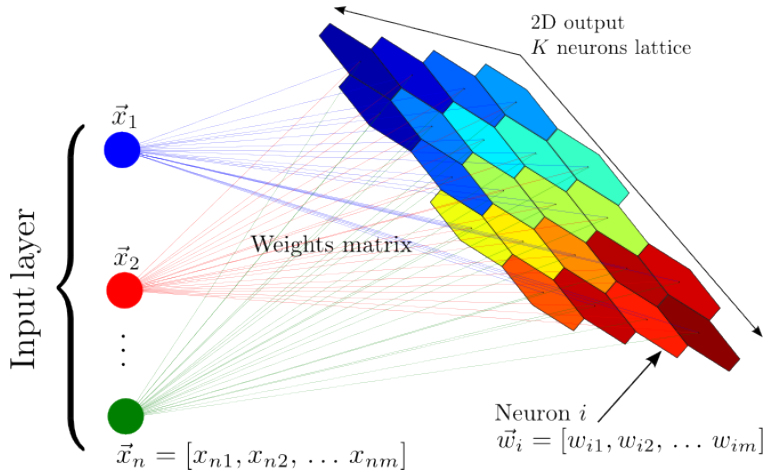
SOM

Un SOM es un tipo de red neuronal artificial, se entrena mediante aprendizaje competitivo en lugar del aprendizaje basado en descenso de gradiente. El SOM fue introducido por el Teuvo Kohonen en la década de 1980, por lo que a veces se denomina mapa de Kohonen o red de Kohonen.

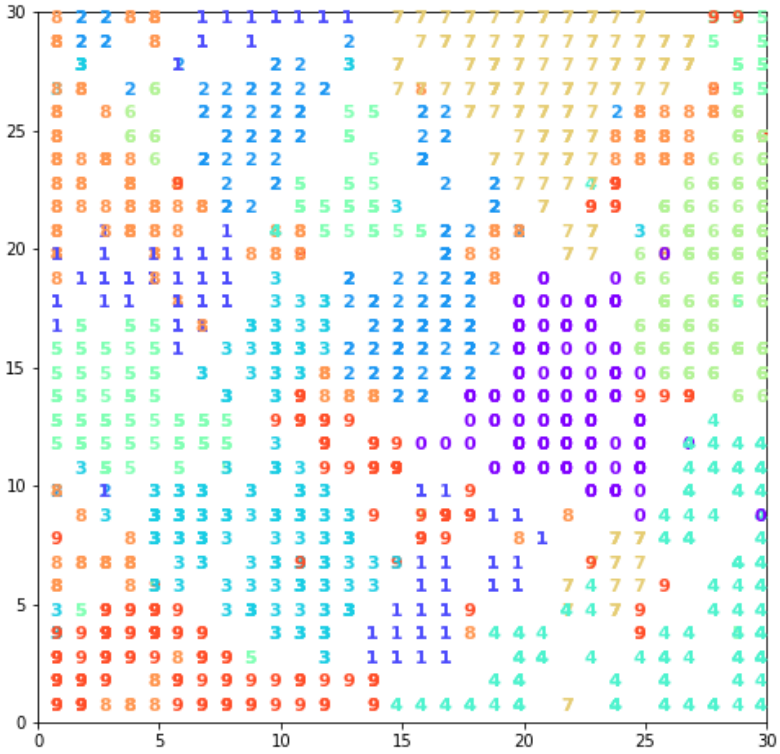


Objetivo

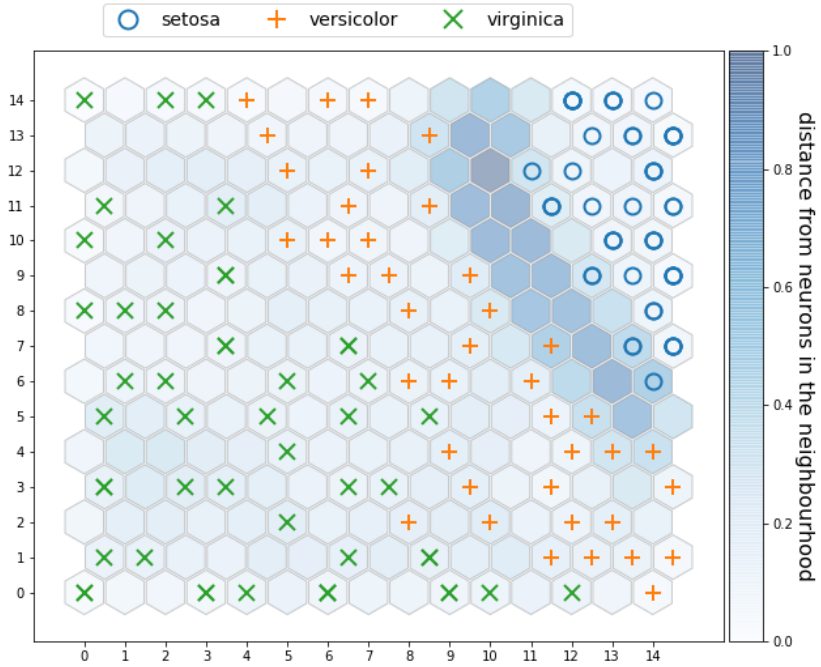
En general, el objetivo del entrenamiento es representar un espacio de entrada con D dimensiones como un espacio de 2 dimensiones. El espacio de salida (feature map) está formado por componentes denominados nodos (neuronas), que se disponen como una rejilla hexagonal o rectangular de dos dimensiones. El número de nodos y su disposición son hiperparámetros.



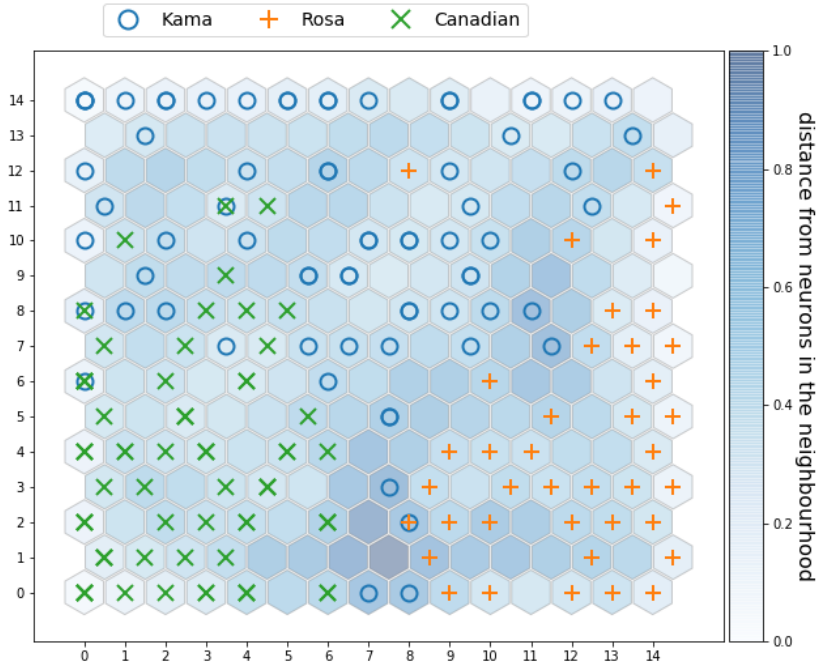
Examples



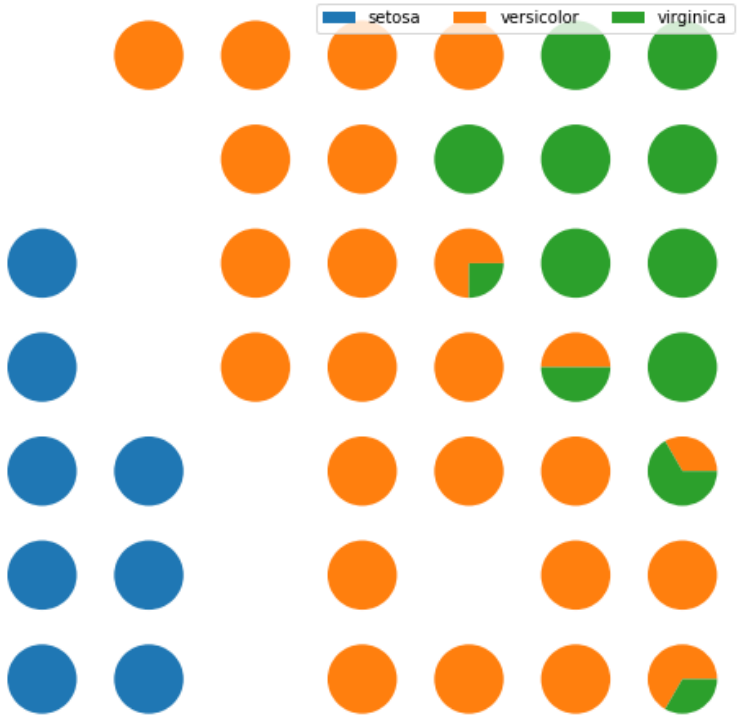
Examples


setosa
versicolor
virginica
distance from neurons in the neighbourhood
0.0
0.2
0.4
0.6
0.8
1.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Examples


Kama
Rosa
Canadian
distance from neurons in the neighbourhood
0.0
0.2
0.4
0.6
0.8
1.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Examples



Aplicaciones

El objetivo de las SOM es el análisis y la exploración de los datos.

- Clasificación
- Clustering
- Visualización
- Detección de outliers

El algoritmo

1. Inicializar aleatoriamente los vectores de pesos de las neuronas (celdas).
2. Seleccionar aleatoriamente un dato de entrada X_j.
3. Recorrer cada neurona k en el espacio de salida.
4. Usando la distancia Euclidiana, calculamos cada distancia

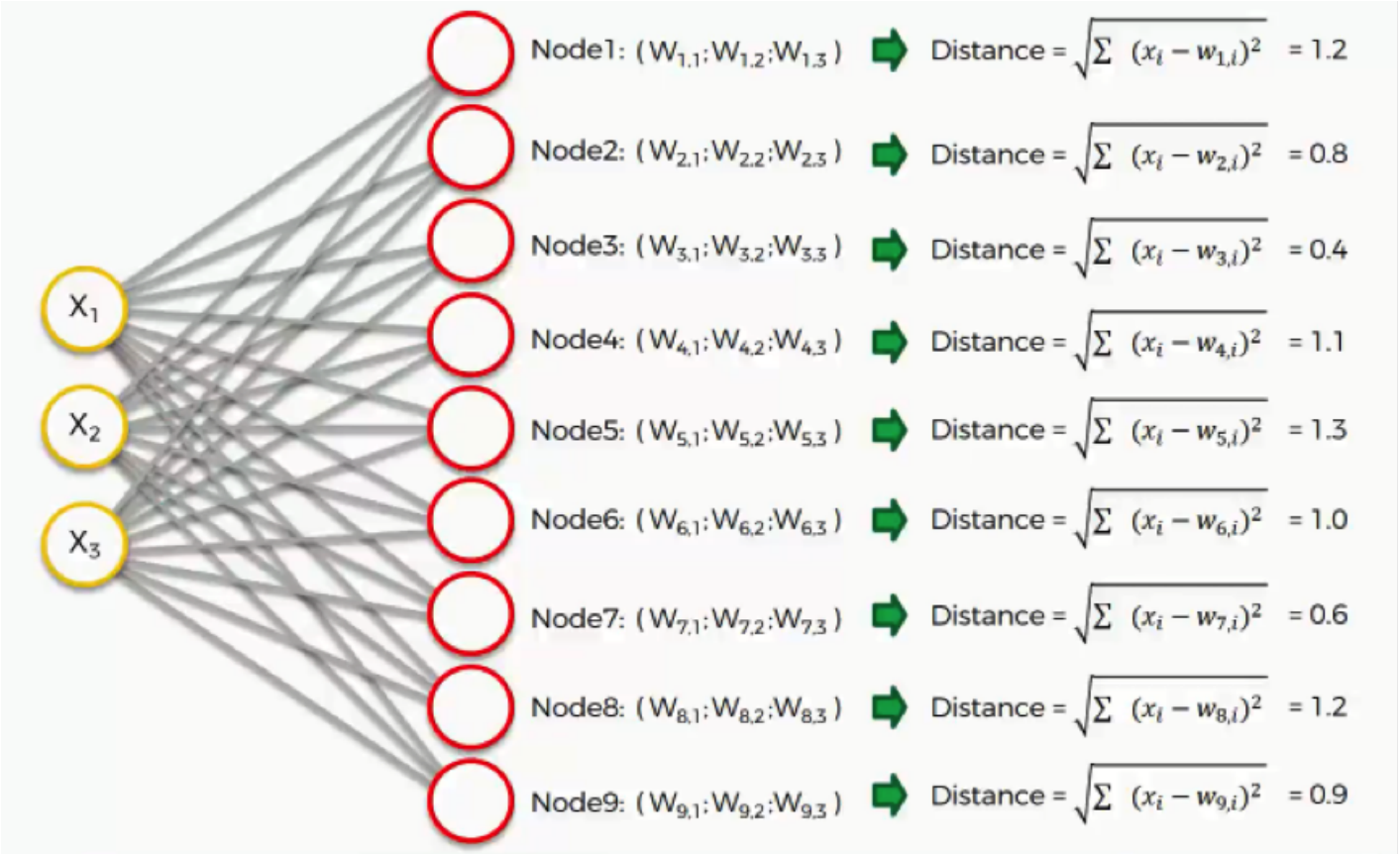
$$d(X_j, W_k)$$

5. Seleccionamos la que produce la distancia más pequeña (best matching unit, BMU).
6. Actualizamos los pesos de las neuronas cercanas a la BMU (incluyendo la BMU) acercándolas al vector X_j

$$W_k(s+1) = W_k(s) + \theta(k, s) \cdot \alpha(s) \cdot (X_j - W_k(s)).$$

7. Continuamos con la siguiente época s y repetimos hasta alcanzar el límite de iteraciones.

Ejemplo


X1
X2
X3
Node1: (W1,1;W1,2;W1,3) Distance = $\sqrt{\sum (x_i - w_{1,i})^2}$ = 1.2
Node2: (W2,1;W2,2;W2,3) Distance = $\sqrt{\sum (x_i - w_{2,i})^2}$ = 0.8
Node3: (W3,1;W3,2;W3,3) Distance = $\sqrt{\sum (x_i - w_{3,i})^2}$ = 0.4
Node4: (W4,1;W4,2;W4,3) Distance = $\sqrt{\sum (x_i - w_{4,i})^2}$ = 1.1
Node5: (W5,1;W5,2;W5,3) Distance = $\sqrt{\sum (x_i - w_{5,i})^2}$ = 1.3
Node6: (W6,1;W6,2;W6,3) Distance = $\sqrt{\sum (x_i - w_{6,i})^2}$ = 1.0
Node7: (W7,1;W7,2;W7,3) Distance = $\sqrt{\sum (x_i - w_{7,i})^2}$ = 0.6
Node8: (W8,1;W8,2;W8,3) Distance = $\sqrt{\sum (x_i - w_{8,i})^2}$ = 1.2
Node9: (W9,1;W9,2;W9,3) Distance = $\sqrt{\sum (x_i - w_{9,i})^2}$ = 0.9