Técnicas de Inteligencia Artificial

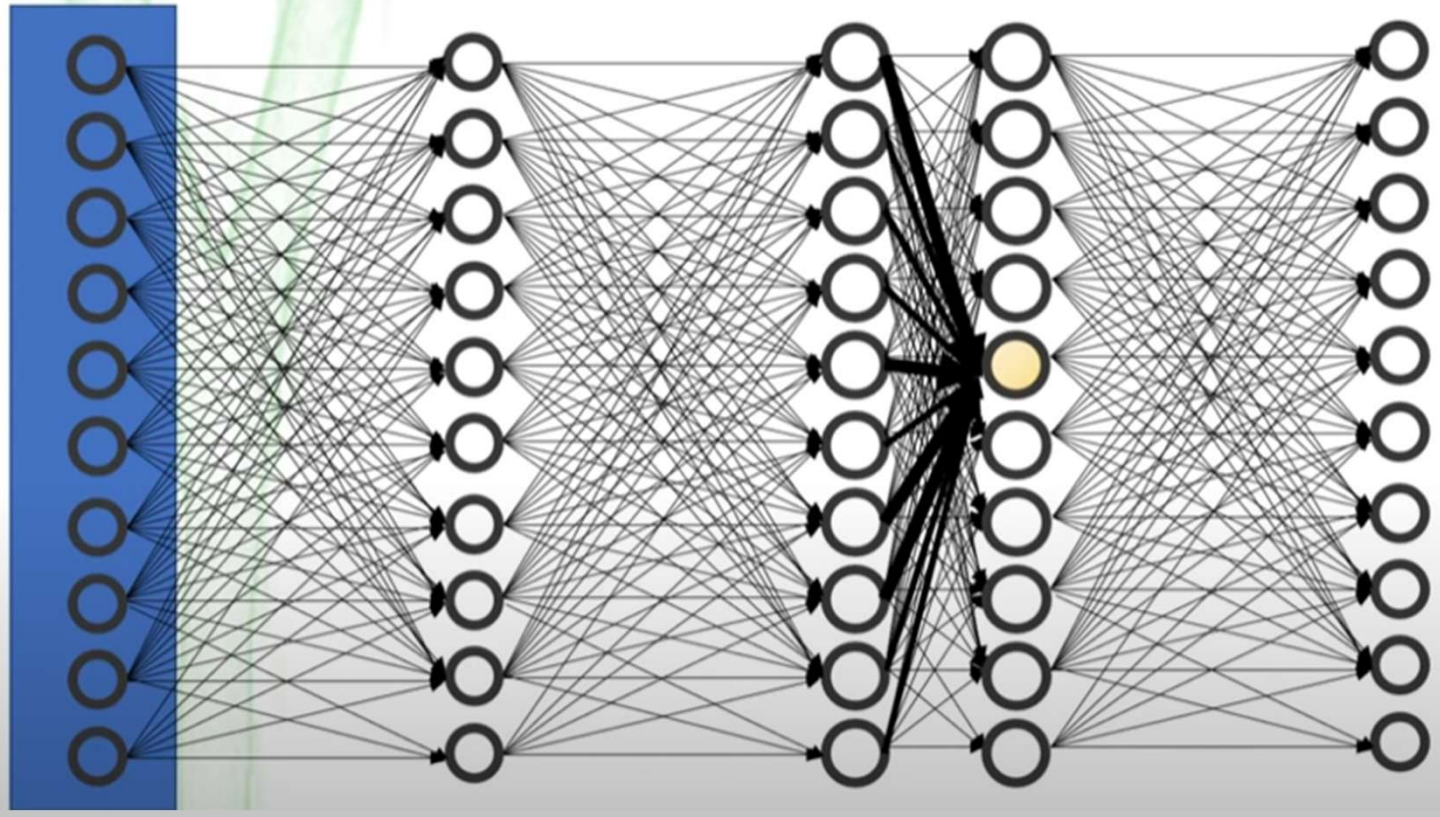
Semana 05. Redes neuronales

Contenido

- Perceptrón simple
- Perceptrón multicapa

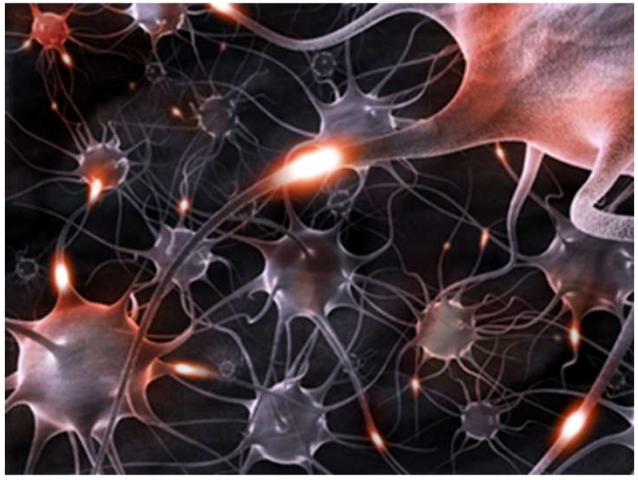
Red neuronal

https://www.youtube.com/watch?v=CU24iC3grq8&t=22s



Redes neuronales

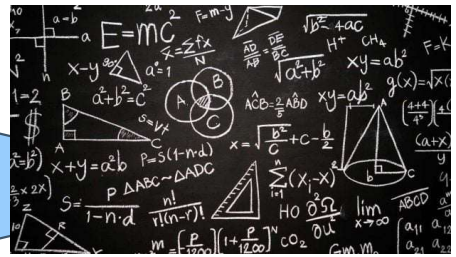
El modelo computacional de las redes neuronales se basa en el funcionamiento del cerebro humano.



Las neuronas reciben estímulos y transmiten los impulsos nerviosos conectándose con otras neuronas o con los músculos. A esta conexión se le denomina sinapsis.

Redes neuronales

Problemas complejos



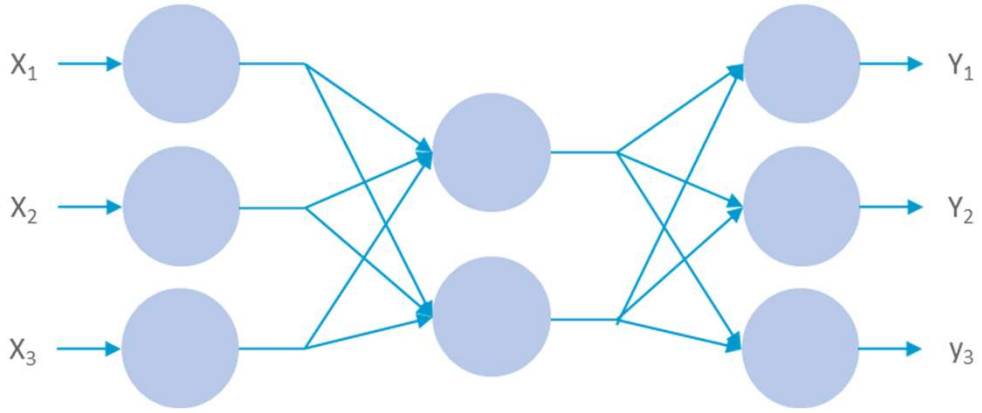
Tiempo de entrenamiento de las redes neuronales es largo,

Una vez aprendida la función objetivo, la evaluación de dicha función objetivo mediante el uso de instancias nuevas es rápida

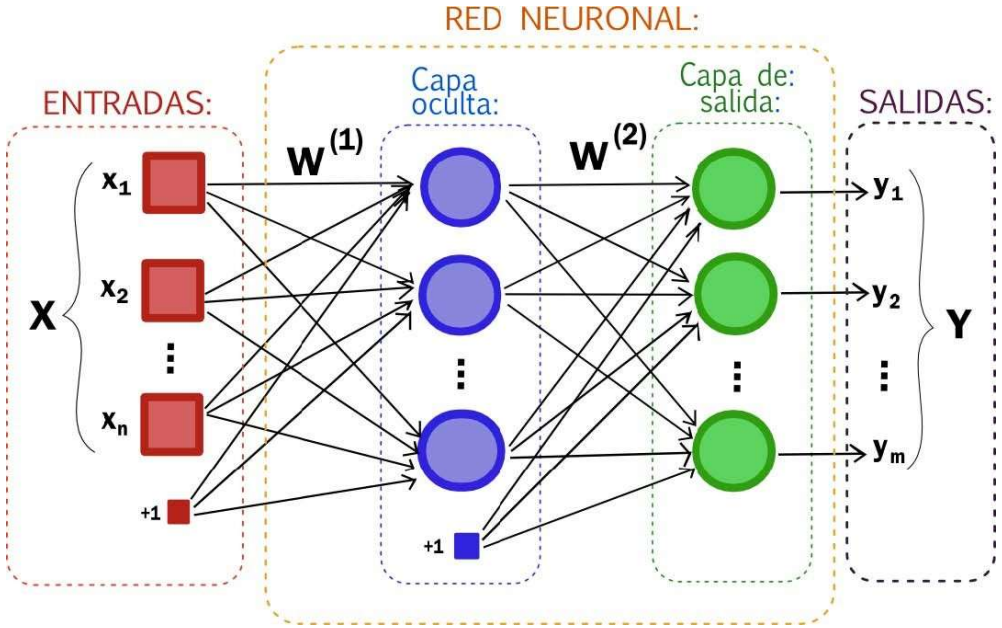
Se comportan de manera robusta frente al ruido, por lo que pueden ofrecer buenos resultados, aunque los ejemplos de entrenamiento contengan errores.

Aprendizaje con redes neuronales

El objetivo del aprendizaje se reduce a, dado un **conjunto de datos de entrenamiento** que consiste en una serie de entradas a esa red y las salidas conocidas correspondientes, escoger los pesos que se ajusten mejor a esas entradas y salidas definidas a priori.



Al formar la red neuronal se desconocerá el **valor de los pesos**, por lo que estos se asignarán inicialmente de **forma aleatoria**. De este modo, la salida que se obtiene al aplicar los pesos en cada iteración será diferente a la salida conocida o esperada. Esta diferencia entre la salida obtenida y la esperada es lo que se conoce **como el error (también llamado pérdida – *loss* – o costo – *cost*), valor que se utiliza para ajustar los pesos.**



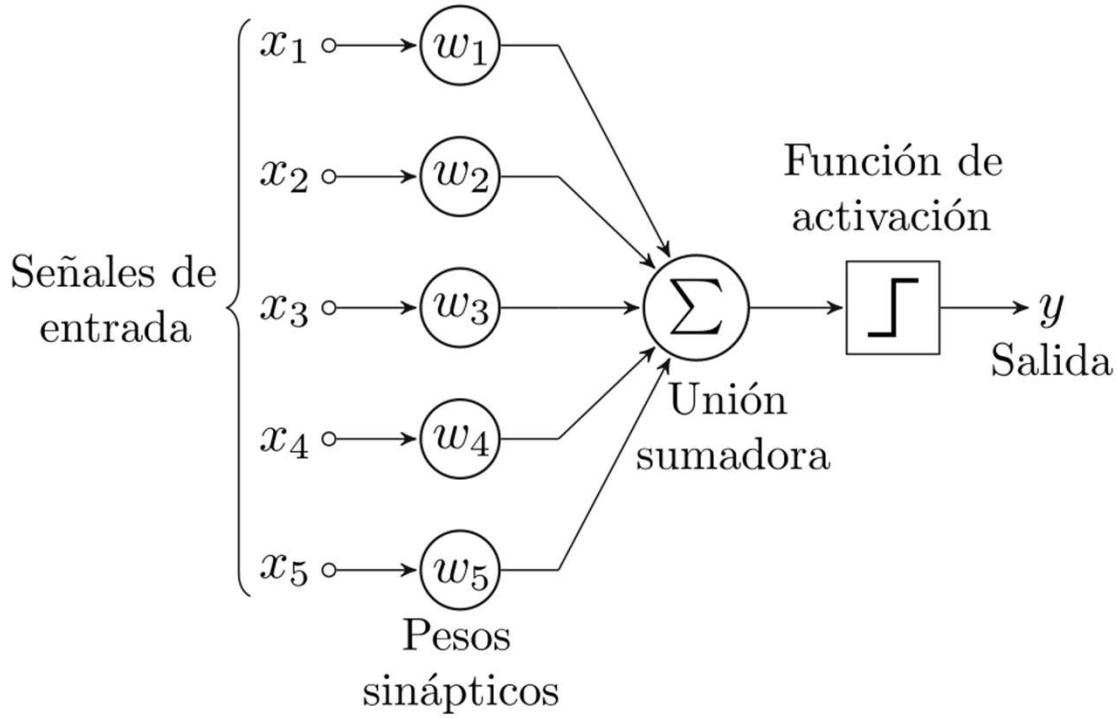
El funcionamiento de la red neuronal vendrá determinado por los siguientes parámetros:

•**Arquitectura de la red**, esto es, el número de capas, número de neuronas por capa y las conexiones entre neuronas entre diferentes capas.

•**Función de activación**: función signo, función escalón, etc.

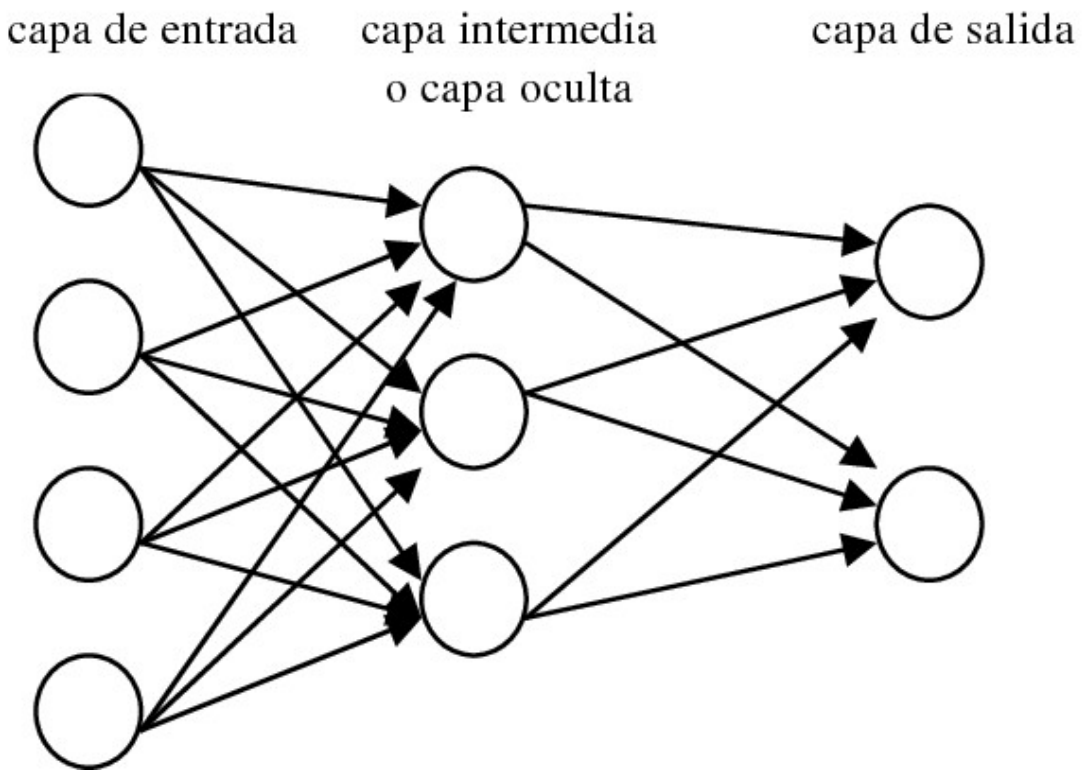
•**Algoritmo de aprendizaje** determinado principalmente por la regla de aprendizaje para ajustar pesos

Perceptrón simple



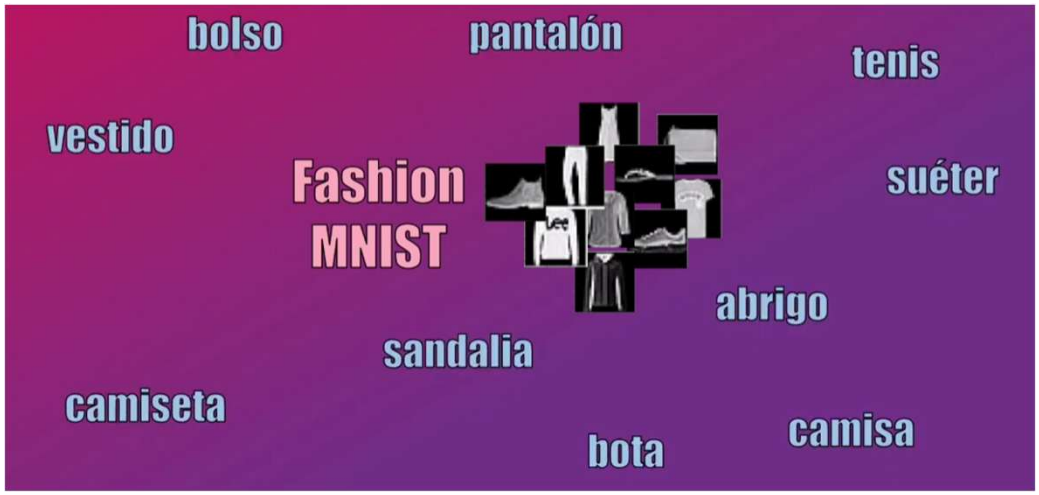
Fuente:https://es.wikipedia.org/wiki/Perceptr%C3%B3n

Perceptrón multicapa



Como crear una red neuronal con TensorFlow y keras en python

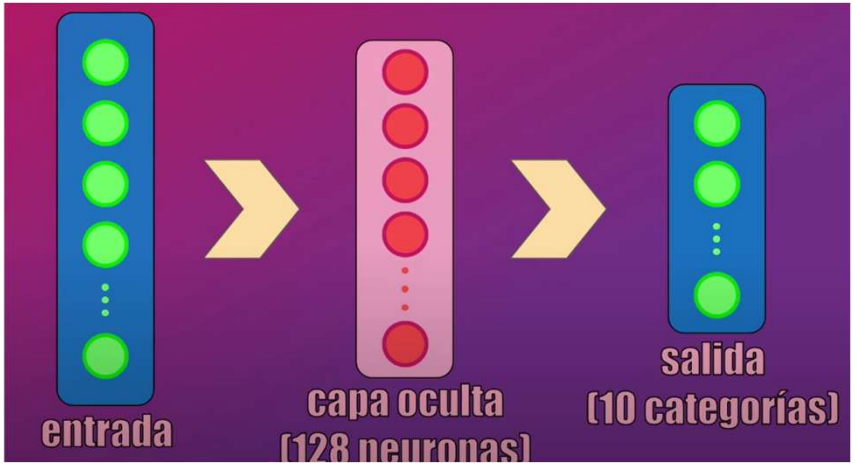
1. **Creación**
2. **Compilación**
3. **Entrenamiento**
4. **Predicción**



Base de datos Fashion Mnist

- 60,000 imágenes de entrenamiento de 28X28 pixeles
- 10,000 imágenes de validación
- 10 categorías(Bolso, vestido, camiseta, sandalia, bota, abrigo, camisa, suéter, tenis, pantalón,)

Red neuronal

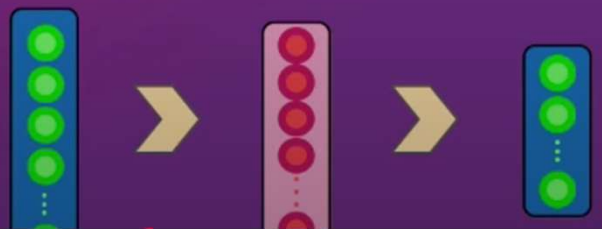


RED NEURONAL

```
from tensorflow.keras import datasets, Sequential
from tensorflow.keras.layers import Flatten, Dense
```

1 - creación

```
modelo = Sequential()
modelo.add( Flatten(input_shape=(28,28)) )
modelo.add( Dense(128, activation = 'relu') )
modelo.add( Dense(10, activation = 'softmax') )
```



2 - compilación

```
modelo.compile(optimizer='adam',
    loss='categorical_crossentropy', metrics=['accuracy'])
```

3 - entrenamiento

```
modelo.fit(x_train, y_train, epochs=10, verbose=1)
```

4 - predicción

```
predicciones = modelo.predict(x_test)
```

Ejemplo de una red neuronal con keras

```
[8]  import keras
     import numpy as np

     from tensorflow.keras import datasets, Sequential
     from tensorflow.keras.layers import Flatten, Dense
```

```
[12] num_classes = 10
     input_shape = (28, 28, 1)
```

```
     (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load
     x_train = x_train.astype("float32") / 255
     x_test = x_test.astype("float32") / 255
     x_train = np.expand_dims(x_train, -1)
     x_test = np.expand_dims(x_test, -1)
     print("x_train shape:", x_train.shape)
     print(x_train.shape[0], "train samples")
     print(x_test.shape[0], "test samples")
     y_train = keras.utils.to_categorical(y_train, num_classes)
     y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
[14] modelo = Sequential()
     modelo.add(Flatten(input_shape=(28,28)))
     modelo.add(Dense(128,activation='relu'))
     modelo.add(Dense(10,activation='softmax'))
```

```
[16] batch_size = 128
     epochs = 15
     modelo.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
     modelo.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)
```

```
score = modelo.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
#Y_pred = modelo.predict(x_test)
#Y_pred[0]
imagen = x_test[0];
imagen = np.array([imagen])
prediccion = modelo.predict(imagen)
print("Prediccion: " + nombres_clases[np.argmax(prediccion[0])])
```

Referencias

https://www.youtube.com/watch?v=CU24iC3grq8&t=22s
https://www.youtube.com/watch?v=iX_on3VxZzk&t=250s
https://www.youtube.com/watch?v=CU24iC3grq8
https://www.youtube.com/watch?v=uM4u7P2xkO8
https://www.youtube.com/watch?v=ITH4mUcjDVk
https://colab.research.google.com/drive/1EjxAt_OFrdpNgCNmV15XMCE8RI3PF3sD?usp=sharing#scrollTo=hzkGVPqnSvRE
https://colab.research.google.com/drive/1GA2GnhpAlVO7jLV88iW1NgvAhflskh8q?hl=es
https://keras-io.translate.goog/examples/vision/mnist_convnet/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Gracias por tu atención