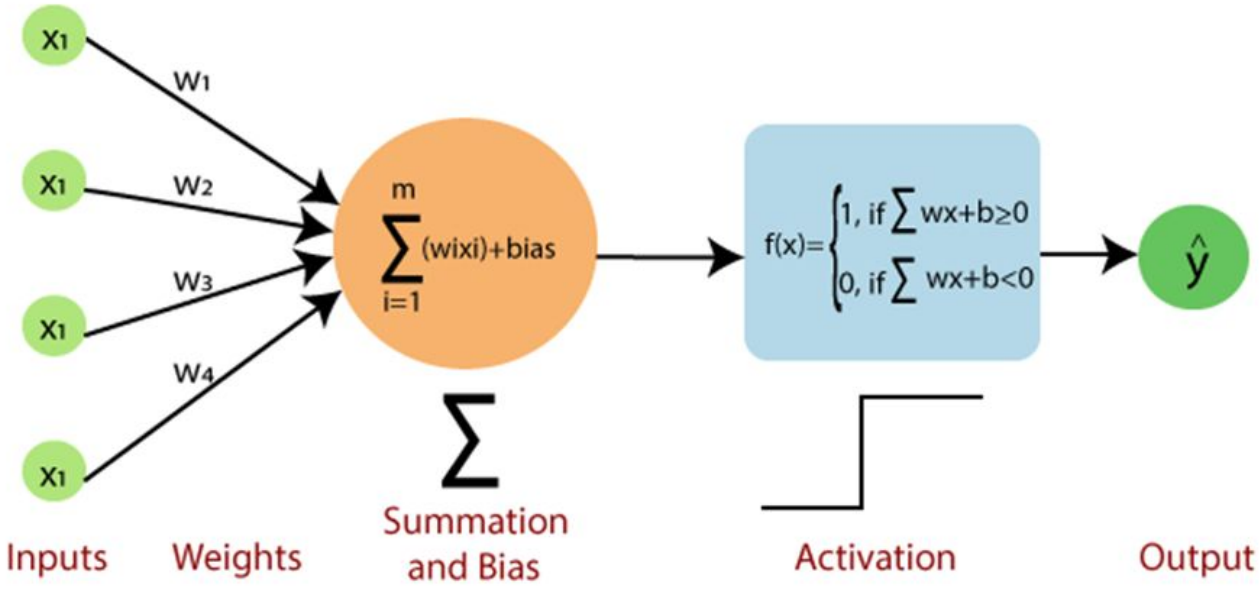
Practicando con los perceptrones

Profesor:
David Campoy Miñarro

El perceptrón

Un perceptrón es un modelo simple de una neurona artificial que puede realizar operaciones de **clasificación binaria**.



$$y = \theta\left(\sum_j w_{ij} x_j + \mu_i\right)$$

Actividad 1

Perceptrón para aprender la función lógica AND:

```python
import numpy as np

class Perceptron:
    def __init__(self, input_size, learning_rate=0.1, epochs=100):
        self.weights = np.zeros(input_size + 1)
        self.learning_rate = learning_rate
        self.epochs = epochs

    def predict(self, inputs):
        summation = np.dot(inputs, self.weights[1:]) + self.weights[0]
        return 1 if summation > 0 else 0

    def train(self, training_inputs, labels):
        for _ in range(self.epochs):
            for inputs, label in zip(training_inputs, labels):
                prediction = self.predict(inputs)
                self.weights[1:] += self.learning_rate * (label - prediction) * inputs
                self.weights[0] += self.learning_rate * (label - prediction)

training_inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
labels = np.array([0, 0, 0, 1])

perceptron = Perceptron(input_size=2)
perceptron.train(training_inputs, labels)

test_inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
for inputs in test_inputs:
    print(f"Entrada: {inputs}, Predicción: {perceptron.predict(inputs)}")
```

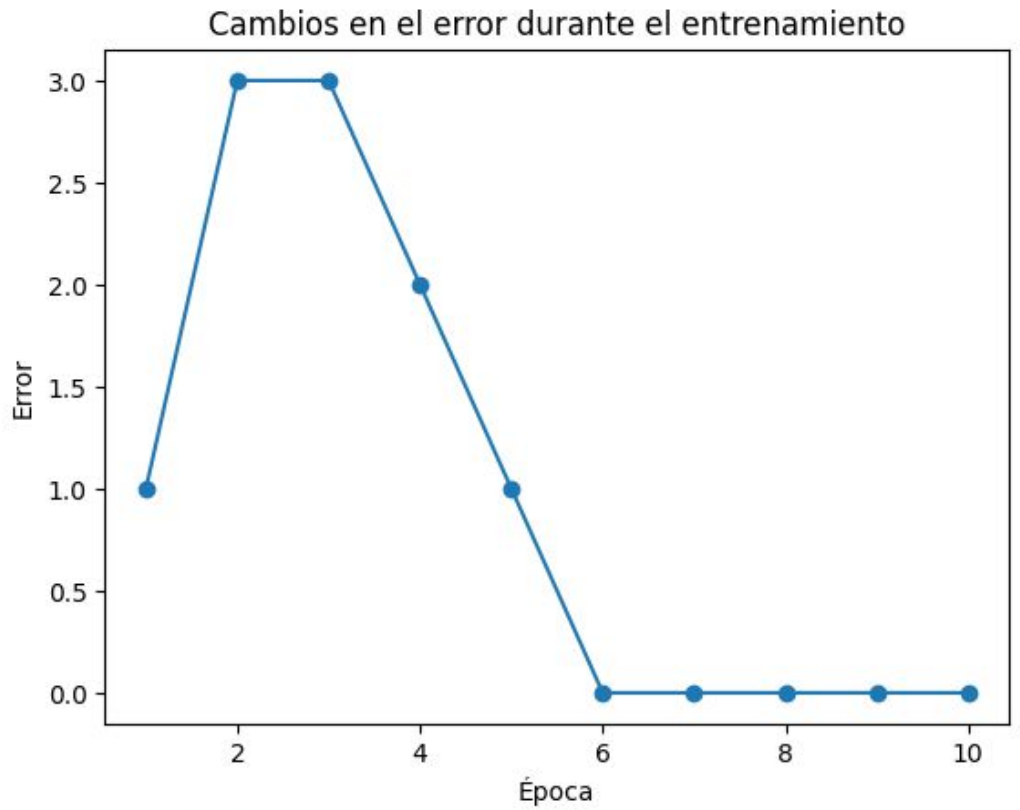
INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Modifica los hiperparámetros: `learning_rate` y `epochs`.

¿Cuántas épocas han sido necesarias?
¿Serías capaz de dibujar la red neuronal con los pesos obtenidos?

Actividad 1

¿Puedes modificar el código para mostrar el error en función de las épocas?

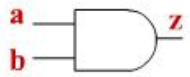
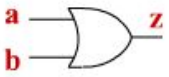
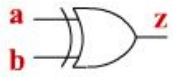
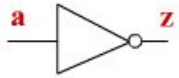
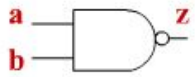
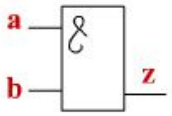
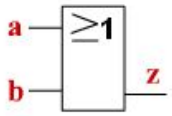
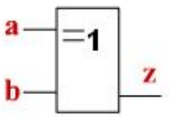
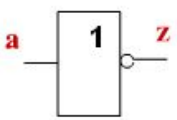
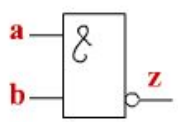
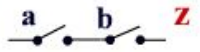
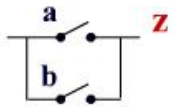
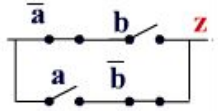
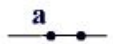
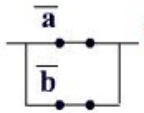



```
Entrada: [0 0], Predicción: 0
Entrada: [0 1], Predicción: 0
Entrada: [1 0], Predicción: 0
Entrada: [1 1], Predicción: 1
```

Actividad 1

Ahora, prueba a entrenar el perceptrón con las siguientes tablas:

FUNCIONES LÓGICAS BÁSICAS

NOMRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR
SÍMBOLO						
SÍMBOLO						
TABLA DE VERDAD	a b z 0 0 0 0 1 0 1 0 0 1 1 1	a b z 0 0 0 0 1 1 1 0 1 1 1 1	a b z 0 0 0 0 1 1 1 0 1 1 1 0	a z 0 1 1 0	a b z 0 0 1 0 1 1 1 0 1 1 1 0	a b z 0 0 1 0 1 0 1 0 0 1 1 0
EQUIVALENTE EN CONTACTOS						
AXIOMA	$z = a \cdot b$	$z = a + b$	$z = \bar{a} \cdot b + a \cdot \bar{b}$	$z = \bar{a}$	$z = \overline{a \cdot b}$	$z = \overline{a + b}$

¿El perceptrón ha sido capaz de aprender todas las funciones lógicas?

¿Por qué?

¿Se te ocurre alguna solución?

Actividad 2

Ahora, vamos a utilizar el conjunto de datos Iris, que contiene información sobre tres especies de iris y sus características. Se ha convertido en un ejemplo comúnmente utilizado para la demostración de técnicas de clasificación y reconocimiento de patrones

El conjunto de datos Iris proporciona una muestra de mediciones de diferentes características de tres especies de iris: setosa, versicolor y virginica. Estas mediciones incluyen:

- Longitud del sépalo: longitud de la parte verde externa de la flor.
- Anchura del sépalo: ancho de la parte verde externa de la flor.
- Longitud del pétalo: longitud de la parte interior de la flor.
- Anchura del pétalo: ancho de la parte interior de la flor.

https://es.wikipedia.org/wiki/Conjunto_de_datos_flor_iris

Actividad 2

```python
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

# Cargar el dataset Iris
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Dividir el dataset en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Escalar las características para un mejor rendimiento
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

# Crear un perceptrón y entrenarlo
ppn = Perceptron(max_iter=100, eta0=0.1, random_state=42)
ppn.fit(X_train_std, y_train)

# Predecir con el conjunto de prueba
y_pred = ppn.predict(X_test_std)

# Calcular la precisión del perceptrón
accuracy = accuracy_score(y_test, y_pred)
print(f'Precisión: {accuracy}')
```

¿Qué precisión ha conseguido el perceptrón?

¿Te parece aceptable la precisión?

Modifica el código para mostrar los pesos y dibuja el perceptrón con sus valores.

Lo increíble es que esos 3 valores numéricos representan el conocimiento de una tabla de 150 muestras.

Actividad 2

```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.datasets import load_iris
import numpy as np

# Función para ingresar los valores de las características de la flor
def ingresar_valores():
    sepal_length = float(input("Introduce la longitud del sépalo (cm): "))
    sepal_width = float(input("Introduce el ancho del sépalo (cm): "))
    petal_length = float(input("Introduce la longitud del pétalo (cm): "))
    petal_width = float(input("Introduce el ancho del pétalo (cm): "))
    return [sepal_length, sepal_width, petal_length, petal_width]

# Cargar el dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Dividir el dataset en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Escalar las características para un mejor rendimiento
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

# Crear un perceptrón y entrenarlo
ppn = Perceptron(max_iter=100, eta0=0.1, random_state=42)
ppn.fit(X_train_std, y_train)

# Obtener los valores del usuario
nuevos_valores = ingresar_valores()

# Escalar los valores ingresados
nuevos_valores_std = sc.transform([nuevos_valores])

# Predecir el tipo de flor con los valores ingresados
prediccion = ppn.predict(nuevos_valores_std)
especies = iris.target_names
print(f"El tipo de flor predicho es: {especies[prediccion[0]]}")
```

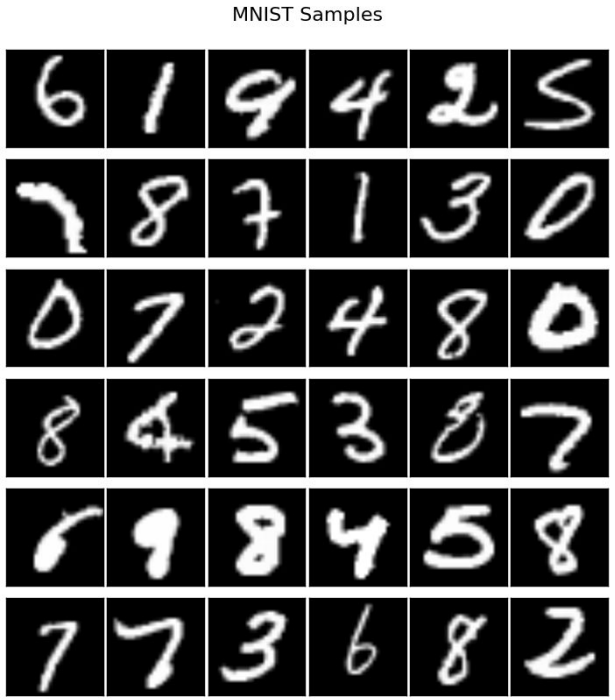
Pon a prueba con el aprendizaje del perceptrón introducido valores de la tabla e incluso valores inventados por ti.

Siempre dará una respuesta, si no la sabe, será capaz de generalizar y responder.

Profesor: David Campoy Miñarro

Actividad 3

Clasificación de imágenes con un perceptrón.



Vamos a averiguar la capacidad que tiene perceptrón de aprender números manuscritos.

Vamos a utilizar el dataset: `mnist_784`

Consiste en un conjunto de imágenes en escala de grises de dígitos escritos a mano, del 0 al 9. Cada imagen tiene una resolución de 28x28 píxeles, lo que da un total de 784 píxeles por imagen. MNIST que contiene 70.000 imágenes de dígitos escritos a mano.

https://www.openml.org/search?type=data&sort=runs&id=554

```python
from sklearn.datasets import fetch_openml
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np

# Cargar el dataset MNIST
mnist = fetch_openml('mnist_784', version=1, parser='auto', as_frame=False)

# Obtener características (X) y etiquetas (y)
X = mnist.data
y = mnist.target.astype(int)

# Dividir el dataset en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Escalar las características para un mejor rendimiento
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

# Crear y entrenar un perceptrón
ppn = Perceptron(max_iter=100, eta0=0.1, random_state=42)
ppn.fit(X_train_std, y_train)

# Predecir con el conjunto de prueba
y_pred = ppn.predict(X_test_std)

# Calcular la precisión del perceptrón
accuracy = accuracy_score(y_test, y_pred)
print(f'Precisión: {accuracy}')
```

```python
# Mostrar 10 imágenes con sus etiquetas y predicciones
num_images = 10
indices = np.random.choice(X_test.shape[0], num_images, replace=False)

for idx in indices:
    plt.imshow(X_test[idx].reshape(28, 28), cmap='gray')
    plt.title(f"Valor real: {y_test[idx]}, Predicción: {y_pred[idx]}")
    plt.axis('off')
    plt.show()
```

¿Qué precisión ha obtenido el perceptrón?

¿Cuántos pesos ha utilizado el perceptrón para aprender a diferenciar las 70.000 muestras de entrada?

Modifica los hiperparámetros.

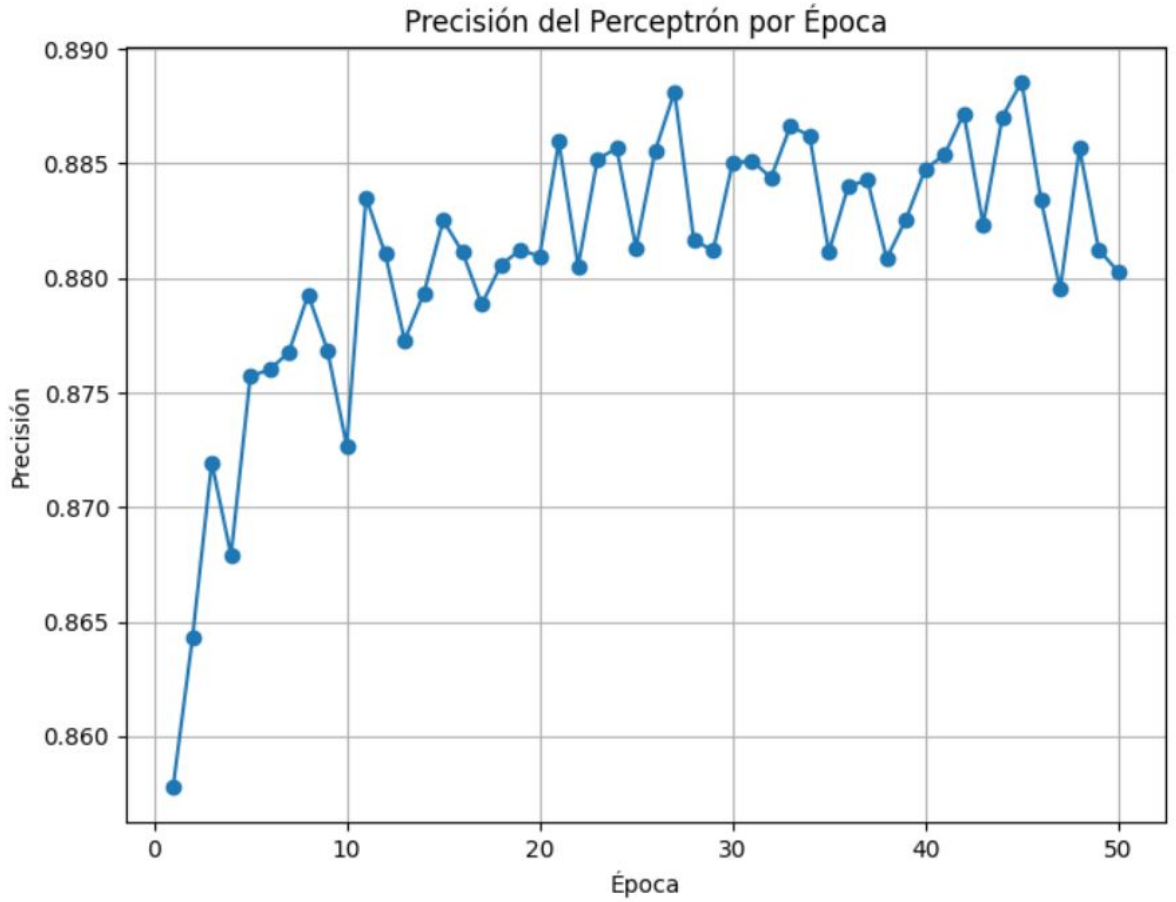
Actividad 3

Para ver los pesos del perceptrón, añade el siguiente código:

```
# Obtener los pesos y el sesgo del perceptrón
weights = ppn.coef_
bias = ppn.intercept_

# Mostrar los pesos de las características
print("Pesos de las características:")
for i, weight in enumerate(weights[0]):
    print(f"Feature {i}: {weight}")

# Mostrar el peso del sesgo
print(f"Sesgo (Bias): {bias[0]}")
```

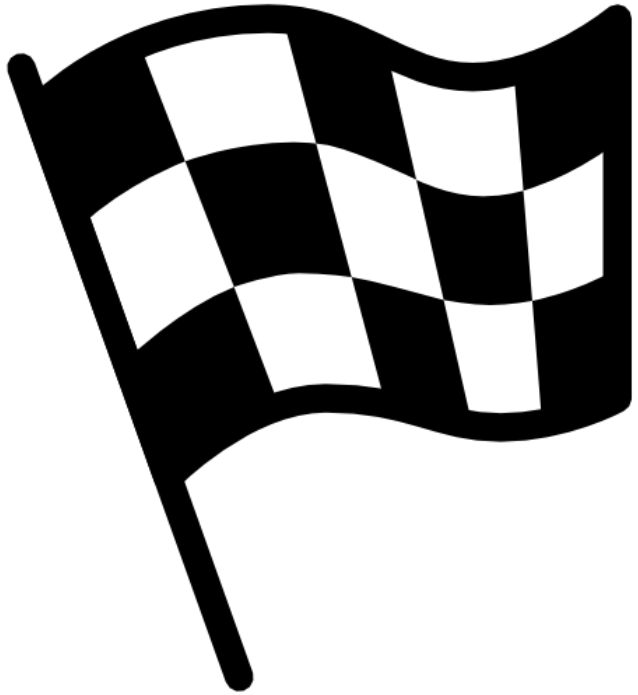


Modifica el código para que te muestre una gráfica de cómo evoluciona su aprendizaje en función de las épocas. Documenta los resultados.

¿Qué has aprendido?

- A utilizar de manera sencilla el Perceptrón.
- Interpretar el funcionamiento del Perceptrón.
- Optimizar hiperparámetros.
- Utilizar el perceptrón para el aprendizaje supervisado.
- Aprendizaje con imágenes con una sola neurona.
- Utilizar el perceptrón para categorizar.

y comprender las características fundamentales de la neurona virtual más simple y a la vez poderosa.



"*Tan importante es educar a las nuevas generaciones que vienen como también creo que es importante enseñar a la fuerza laboral existente, para que puedan entender cómo hacer que la Inteligencia Artificial les sirva a ellos y a sus roles*"

Sarah Aerni, ingenier y Directora de Machine Learning en Salesforce