UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

Inteligencia Artificial

Ing. Juancarlos Santana Huamán
jsantana@ucss.edu.pe

¿Qué es el Perceptrón? | La red neuronal artificial más simple

- El **perceptrón** es una de las arquitecturas **de redes neuronales artificiales** más simples , introducida por Frank Rosenblatt en 1957. Se utiliza principalmente para **la clasificación binaria** .
- En aquella época, se utilizaban comúnmente métodos tradicionales como el aprendizaje automático estadístico y la programación convencional para realizar predicciones. A pesar de ser una de las formas más simples de redes neuronales artificiales, el modelo del perceptrón demostró ser muy eficaz para resolver problemas específicos de clasificación, sentando las bases para los avances en IA y aprendizaje automático.

¿Qué es el Perceptrón?

- **El perceptrón** es un tipo de red neuronal que realiza una clasificación binaria que asigna características de entrada a una decisión de salida, generalmente clasificando los datos en una de dos categorías, como 0 o 1.
- El perceptrón consta de una sola capa de nodos de entrada completamente conectados a una capa de nodos de salida. Es especialmente eficaz en el aprendizaje **de patrones linealmente separables** . Utiliza una variante de neuronas artificiales denominada **Unidades Lógicas de Umbral (TLU)** , introducidas por primera vez por McCulloch y Walter Pitts en la década de 1940. Este modelo fundamental ha desempeñado un papel crucial en el desarrollo de redes neuronales más avanzadas y algoritmos de aprendizaje automático.
- **Tipos de perceptrón**
  - **El perceptrón de una sola capa** es un tipo de perceptrón limitado al aprendizaje de patrones linealmente separables. Es eficaz en tareas donde los datos se pueden dividir en distintas categorías mediante una línea recta. Si bien su simplicidad lo hace potente, presenta dificultades en problemas más complejos donde la relación entre entradas y salidas no es lineal.
  - **Los perceptrones multicapa** poseen capacidades de procesamiento mejoradas ya que constan de dos o más capas, capaces de manejar patrones y relaciones más complejos dentro de los datos.

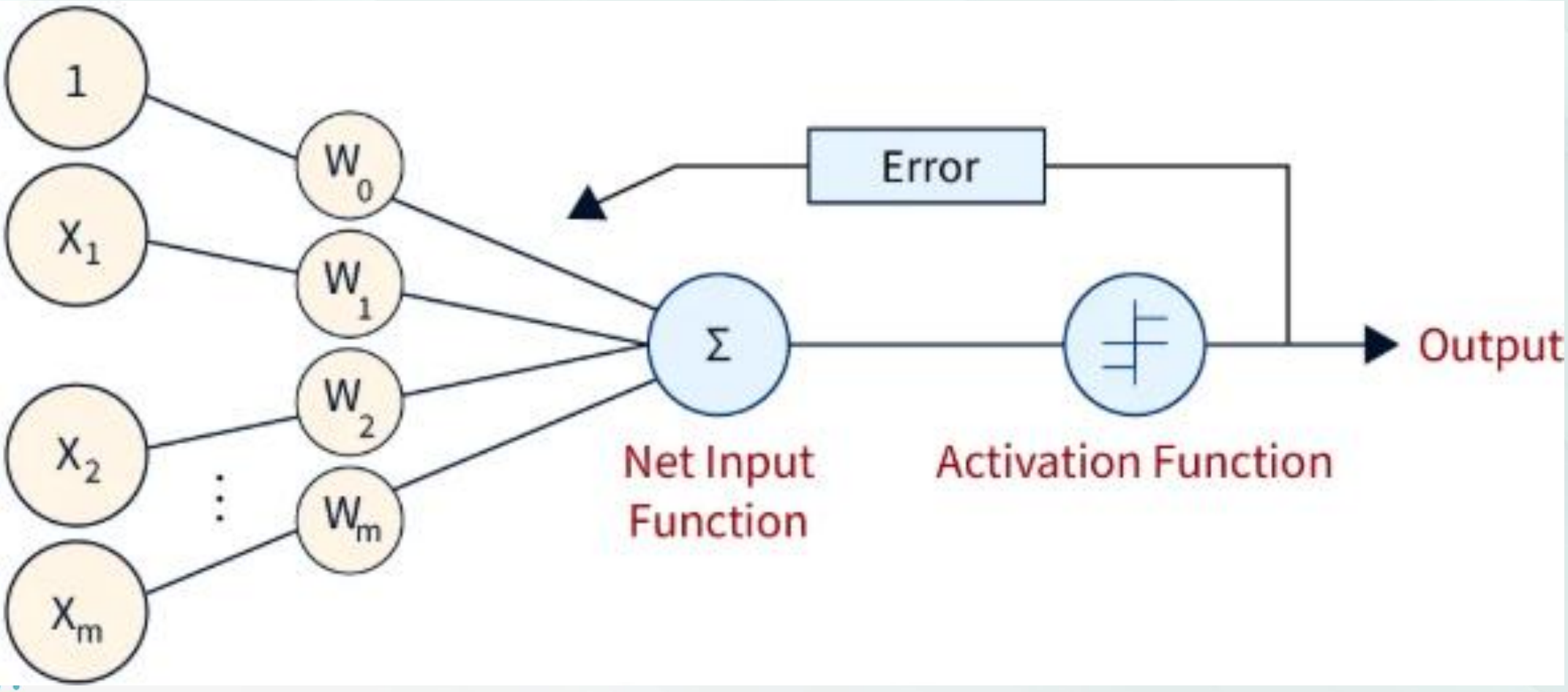
¿Qué es el algoritmo de aprendizaje del perceptrón?

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

Hay cuatro pasos importantes en un algoritmo de aprendizaje de perceptrón:

1. Primero, multiplique todos los valores de entrada por sus ponderaciones correspondientes y luego súmelos para determinar la suma ponderada. Matemáticamente, podemos calcular la suma ponderada de la siguiente manera:$\sum w_i * \text{incógnita}_i = \text{incógnita}_1 * w_1 + \text{incógnita}_2 * w_2 + \cdots + w_{\text{norte}} * \text{incógnita}_{\text{norte}}$Agregue otro término esencial llamado sesgo 'b' a la suma ponderada para mejorar el rendimiento del modelo.$\sum w_i * \text{incógnita}_i + b$.
2. A continuación, se aplica una función de activación a esta suma ponderada, produciendo una salida binaria o de valor continuo.$Y = F(\sum w_i * \text{incógnita}_i + b)$
3. A continuación, se calcula la diferencia entre esta salida y el valor objetivo real para obtener el término de error, E , generalmente expresado como error cuadrático medio. Los pasos hasta este punto constituyen la parte de propagación hacia adelante del algoritmo.$\text{mi} = \left(Y - Y_{a\text{doeltú}ayo}\right)^2$
4. Optimizamos este error (función de pérdida) mediante un algoritmo de optimización. Generalmente, se utiliza algún tipo de algoritmo de descenso de gradiente para encontrar los valores óptimos de los hiperparámetros como la tasa de aprendizaje , el peso , el sesgo , etc. Este paso constituye la parte de propagación hacia atrás del algoritmo.

- En la siguiente figura se ilustra una descripción general de este algoritmo:



- En una notación más estandarizada, el algoritmo de aprendizaje del perceptrón es el siguiente:

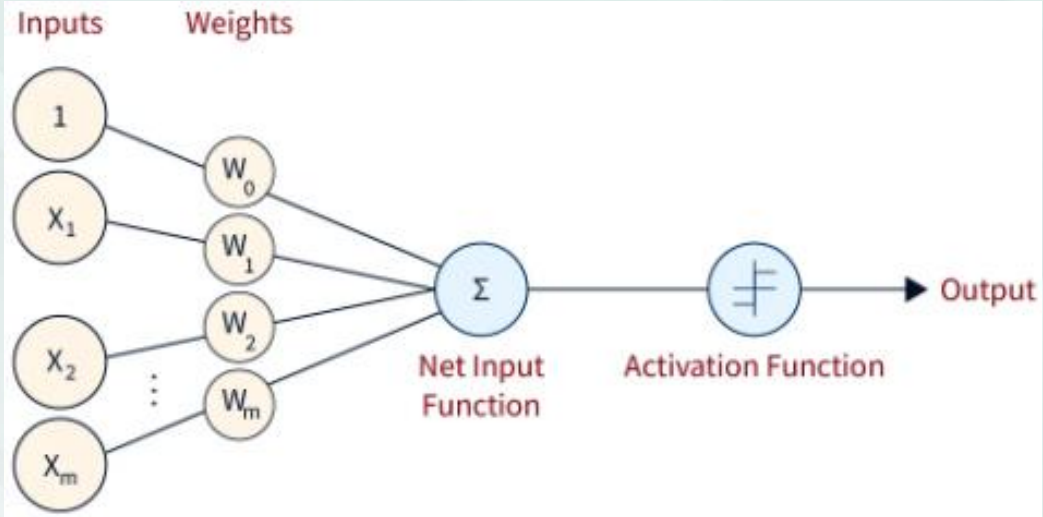
```
P <-- inputs with label 1
N <-- inputs with label 0
Initialise w randomly;
while !converge do:
$\hspace{2em}$ Pick random x $\in P \cup N;$
$\hspace{2em}$ if x $\in$ P and w.x $<$ 0 then
$\hspace{3em}$ w = w+x
$\hspace{2em}$ end
$\hspace{3em}$ if x $\in$ N and w.x $\ge$ 0 then
$\hspace{3em}$ w = w-x
$\hspace{2em}$ end
end
```

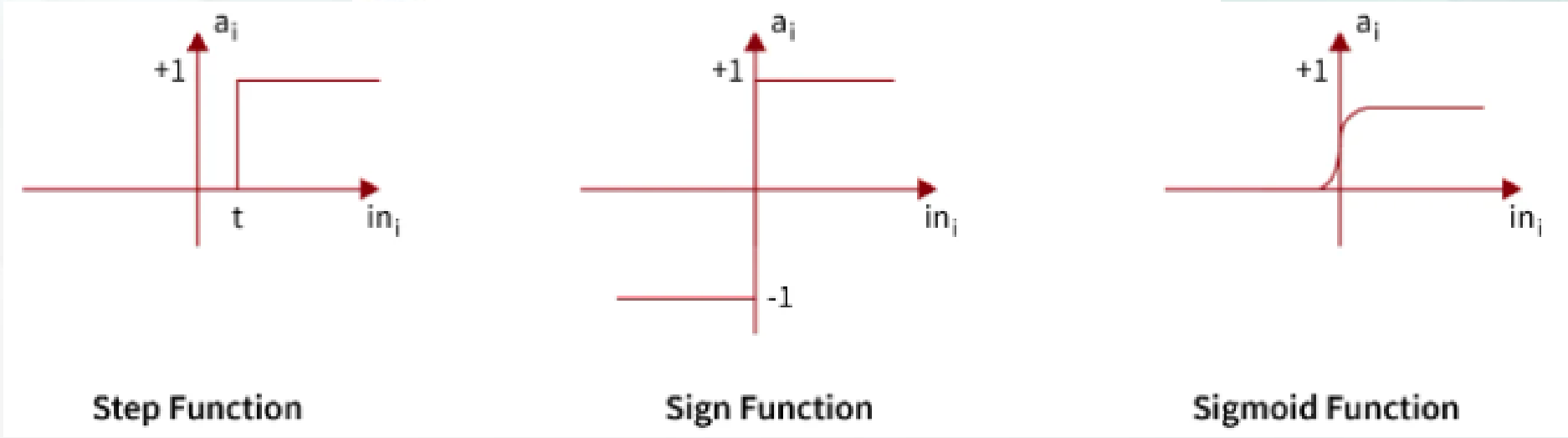
- Nuestro objetivo es encontrar el vector w que clasifique perfectamente las entradas positivas y negativas en un conjunto de datos. w se inicializa con un vector aleatorio. A continuación, iteramos las muestras positivas y negativas (PUN). Ahora bien, si una entrada x pertenece a P , wx debe ser mayor o igual a 0. Y si x pertenece a N , wx debe ser menor o igual a 0. Solo cuando no se cumplen estas condiciones, actualizamos las ponderaciones.

Componentes básicos del perceptrón

- Frank Rosenblatt inventó el algoritmo de aprendizaje del perceptrón.
- Es un clasificador binario y consta de tres componentes principales:
  - **Nodos de Entrada o Capa de Entrada:** Componente principal del algoritmo de aprendizaje del Perceptrón, que acepta los datos de entrada iniciales en el modelo. Cada nodo de entrada contiene un valor real.
  - **Peso y sesgo:** El parámetro de peso representa la fuerza de la conexión entre las unidades. El sesgo puede considerarse como la línea de intersección en una ecuación lineal.
  - **Función de activación:** Los componentes finales y esenciales ayudan a determinar si la neurona se activará. La función de activación puede considerarse principalmente una función escalonada. Existen varios tipos de funciones de activación utilizadas en un algoritmo de aprendizaje perceptrón. Algunas de ellas son la función signo, la función escalonada, la función sigmoidea, etc.

Componentes básicos del perceptrón

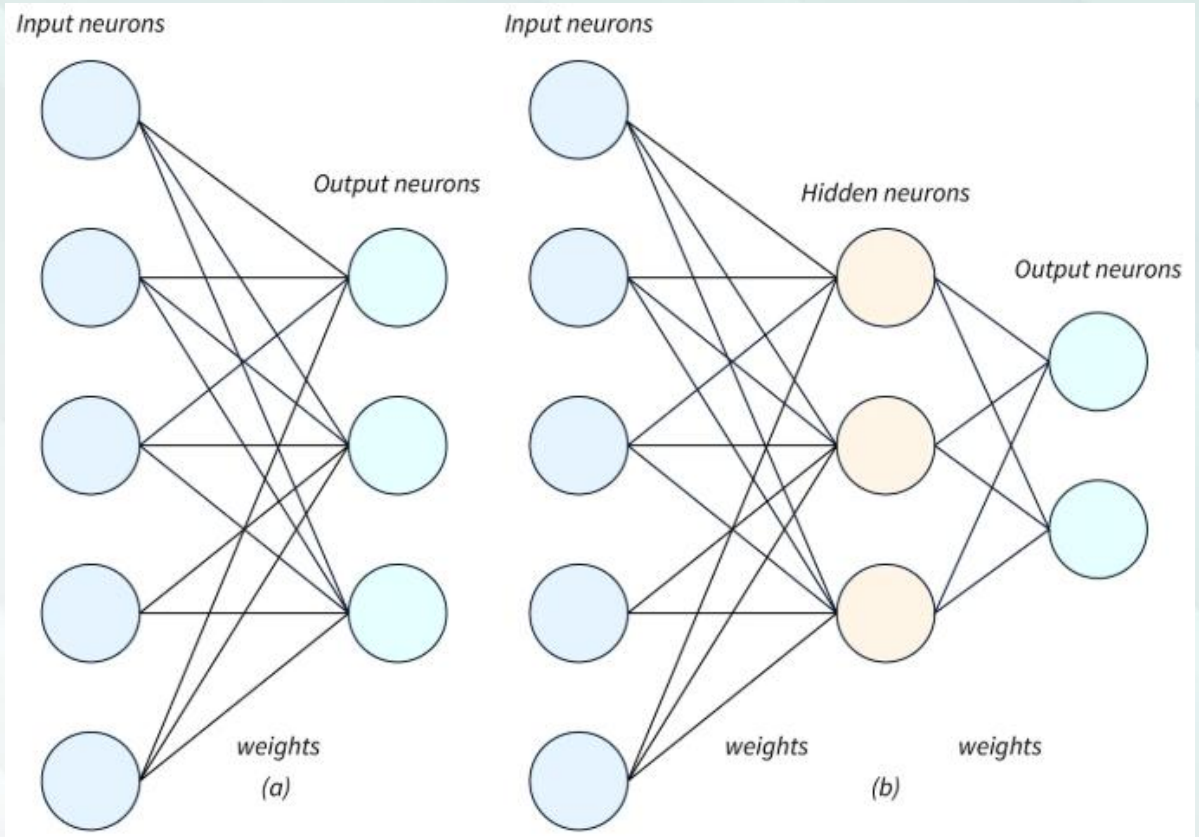




Tipos de modelos de perceptrón

- **Según el número de capas, los perceptrones se clasifican en dos categorías principales:**
- **Modelo de perceptrón de capa única:**
  Es el modelo de red neuronal artificial (RNA) más simple. Un modelo de perceptrón de capa única consiste en una red de propagación hacia adelante e incluye una función de transferencia de umbral para la umbralización de la salida. El objetivo principal del modelo de perceptrón de capa única es clasificar datos linealmente separables con etiquetas binarias.
- **Modelo de perceptrón multicapa:**
  El algoritmo de aprendizaje del perceptrón multicapa tiene la misma estructura que un perceptrón monocapa, pero consta de una o más capas ocultas adicionales, a diferencia de un perceptrón monocapa, que consta de una sola capa oculta. La distinción entre estos dos tipos de modelos de perceptrón se muestra en la figura a continuación.

Tipos de modelos de perceptrón



(a) Architecture of a single layer perceptron. The architecture consists of a layer on input neurons fully connected to a single layer of output neurons.

(b) Extension to a multi-layer perceptron including more than one layer of trainable weights. In this example, the network includes 3 layers: input, hidden and output layer. Each connection between two neurons is given by a certain weight.

Clasificación de redes neuronales artificiales

Red neuronal Monocapa – Perceptrón simple

- La red neuronal monocapa se corresponde con la red neuronal más simple, está compuesta por una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan los diferentes cálculos.

Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

Los pasos para esta implementación son los siguientes:

- Importar todas las bibliotecas necesarias:

```
#import required library
import tensorflow as tf
```

- Definir variables vectoriales para entrada y salida:

```
#input1, input2 and bias
train_in = [
    [1., 1.,1],
    [1., 0,1],
    [0, 1.,1],
    [0, 0,1]]

#output
train_out = [
[1.],
[0],
[0],
[0]]
```

Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

- Defina la variable Peso:

```
#weight variable initialized with random values
using random_normal()
w = tf.Variable(tf.random_normal([3, 1], seed=12))
```

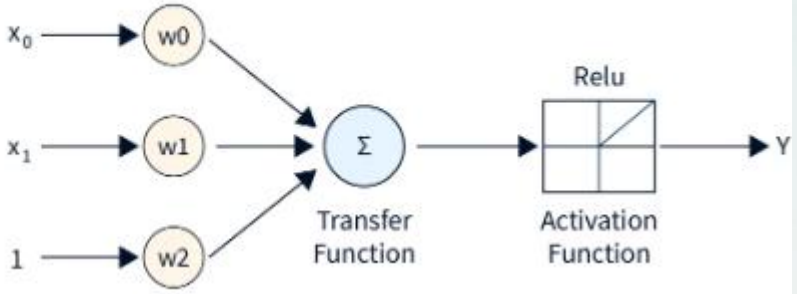
- Definir marcadores de posición para entrada y salida:

```
#Placeholder for input and Output
x = tf.placeholder(tf.float32,[None,3])
y = tf.placeholder(tf.float32,[None,1])
```

- Calcular la función de salida y activación:

AND Gate

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1




```
#calculate output
output = tf.nn.relu(tf.matmul(x, w))
```

Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- Calcular el coste o error:

```
#Mean Squared Loss or Error
loss = tf.reduce_sum(tf.square(output - y))
```

- Minimizar error:

```
#Minimize loss using GradientDescentOptimizer with a
learning rate of 0.01
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
```

- Inicializar todas las variables:

```
#Initialize all the global variables
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

Algoritmo de aprendizaje del perceptrón: implementación de la puerta AND

- Entrenamiento del algoritmo de aprendizaje del Perceptrón en Iteraciones:

```
training_epochs = 1000

#Compute cost w.r.t to input vector for 1000 epochs

for epoch in range(training_epochs):
    sess.run(train, {x:train_in,y:train_out})
    cost = sess.run(loss,feed_dict={x:train_in,y:train_out})
    if i > 990:
        print('Epoch--',epoch,'--loss--',cost)
```

Perceptrón con Scikit-Learn

- El algoritmo de aprendizaje del perceptrón está disponible en la biblioteca de aprendizaje automático de Python scikit-learn a través de la clase Perceptron. Algunos de los parámetros configurables importantes para esta clase son: la tasa de aprendizaje (eta0), cuyo valor predeterminado es 1.0 , y las épocas de entrenamiento (max_iter), cuyo valor predeterminado es 1000. La detención anticipada, cuyo valor predeterminado es "False" , y el tipo de regularización (penalización), cuyo valor predeterminado es "Ninguno" y cuyos valores pueden ser " l2 ", " l1 " y " elastic net ".
- **Un ejemplo de llamada de este algoritmo perceptrón es el siguiente:**

$$Modelo=Percepción\ (\ e\ t\ a\ 0=0\ .\ 1\ ,máximo=1\ 0\ 0\ 0\ 0\ )$$

- Ahora, demostraremos la implementación del algoritmo de aprendizaje del Perceptrón con un ejemplo práctico. Para ello, generaremos un conjunto de datos de clasificación sintética. Usaremos la función make_classification() para crear un conjunto de datos con 1000 ejemplos, cada uno con 20 variables de entrada.

Perceptrón con Scikit-Learn

```
# Evaluate a perceptron model on a synthetic dataset
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import Perceptron
# Define the dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=10,
n_redundant=0, random_state=1)
# Define the model
model = Perceptron()
# Define model evaluation method
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# Evaluate the model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# Summarize the results
print('Mean Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

Limitaciones del modelo de perceptrón

Un modelo de perceptrón tiene las siguientes limitaciones:

- La salida de un perceptrón solo puede ser un número binario (0 o 1) debido a la función de transferencia de límite rígido. Por lo tanto, es difícil de usar en problemas distintos a la clasificación binaria, como la regresión o la clasificación multiclase.
- El perceptrón solo puede utilizarse para clasificar conjuntos de vectores de entrada linealmente separables. Si los vectores de entrada no son lineales, no es fácil clasificarlos correctamente.

El futuro del perceptrón

- Los perceptrones tienen un futuro prometedor, ya que son un modelo muy intuitivo e interpretable que facilita la interpretación de los datos. Las neuronas artificiales constituyen la base de los perceptrones y representan el futuro de los modelos de redes neuronales de vanguardia y muy populares. Por lo tanto, con la creciente popularidad de la inteligencia artificial y las redes neuronales, los algoritmos de aprendizaje perceptrónico desempeñan un papel fundamental.

Ejercicios

- **Caso: Clasificación de flores Iris (Setosa vs. No Setosa)**
- Iris (150 muestras, 3 especies, 4 características por flor).
- **Objetivo:** Clasificar si una flor es *Setosa* (1) o *No Setosa* (0) usando solo dos características: **Longitud del pétalo (cm)** y **Ancho del pétalo (cm)**.

Ejercicios

- **Caso: Identificación del estado de ánimo de una persona.**
- **Objetivo:** Clasificar el estado de una persona si se encuentra:
  - Enojado
  - Disgustado
  - Miedoso
  - Feliz
  - Triste
  - Sorprendida
  - Neutral

UNIVERSIDAD
CATÓLICA
SEDES SAPIENTIAE