**Clase 2 – Perceptron**

**La primera red neuronal – Perceptron**

🡪 1 unica neurona

🡪 Aprendizaje supervisado

🡪 Se adapta con el error entre la salida de la red y la esperada

🡪 Única función discriminante que separa linealmente en 2 clases

Diagrama

Descripción generada automáticamente

**Entrenamiento**

🡪 Adaptar los valores de las conexiones a medida que llegan datos de entrada

🡪 El estimulo de entrada = Producto de vectores X y W

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**Ajuste del vector de pesos**

**Salida del perceptrón** 🡪 Y = 1 si W.X >= θ

Y = 0 si W.X < θ

**Actualizacion de pesos** 🡪 Wnuevo = W + alfa\*(t – y)x

🡪 t es valor esperado

🡪 y es valor obtenido

**Entrenamiento**

🡪 Inicializacion de pesos con valores random 🡪 Vector W

Mientras no se clasifican todos los ejs correctamente:

* Ingresa ejemplo a red
* Si se clasifica incorrectamente:

🡪 Si se esperaba W\*X > θ y no se logro 🡪 Acerque el vector W al X

🡪 Si se esperaba W\*X < θ y no se logro 🡪 Aleje el vector W al X

🡪 **Actualizacion de pesos** 🡪 Wnuevo = W + alfa\*(t – y)\*x

🡪 **Actualizacion de bias 🡪** Bnuevo = b + alfa\*(t-y)

🡪 t es valor esperado

🡪 y es valor obtenido

**Parámetros de entrada de ClassPerceptron**

* Alpha 🡪 Velocidad de aprendizaje (entre 0 y 1)
* N\_iter 🡪 Max iteraciones
* Draw 🡪 Grafica o no
* Title 🡪 Nombres de ejes para el grafico
* Random\_state 🡪 None si los pesos se inician random

🡪 Valor entero para seed

**Parámetros de entrada para función Fit**

* X 🡪 Arreglo de NxM donde N es cant. ej. y M la cant. atributos
* T 🡪 Arreglo de N elementos donde N es cant. ej.

**Devuelve:**

* w\_ 🡪 Arreglo de M elementos siendo M la cant. atributos de entrada
* b\_ 🡪 Bias
* errors\_ 🡪 Errores cometidos en cada iteración

**Y = ppn.predict (X)**

**Parámetros de entrada**

**X** 🡪 Arreglo de NxM

**Devuelve**

🡪 Arreglo con el resultado de aplicar el perceptrón entrenado previamente con fit() a la matriz de ejemplos X

🡪 **Y** 🡪 Arreglo de N elementos