Trabajo Práctico N°3

Perceptron simple y multicapa

Grupo 14

Ian Mejalelaty
Justina Vacas Castro
Josefina Assaff Alvarez

Ej 1: Perceptrón simple escalón

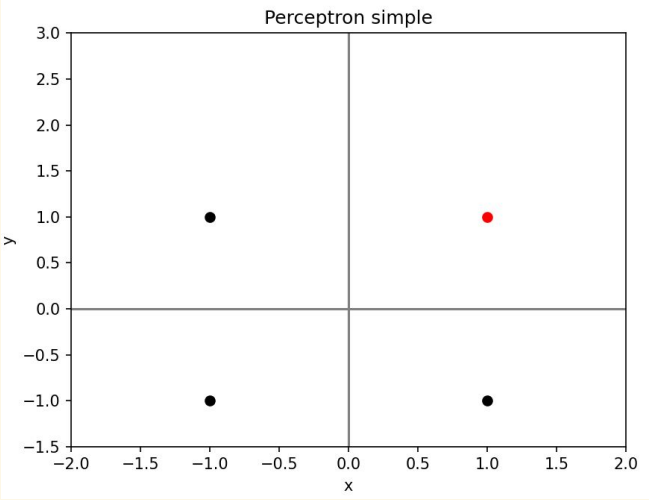
Ej 2: Perceptrón simple lineal y no lineal

Ej 3: Perceptrón multicapa

Ej 1: Perceptrón simple escalón

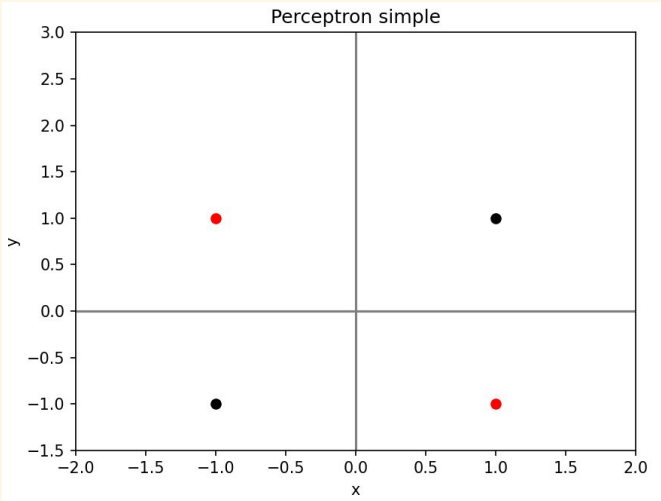
Problemas a analizar con entradas x = {{−1, 1}, {1, −1}, {−1, −1}, {1, 1}}

AND



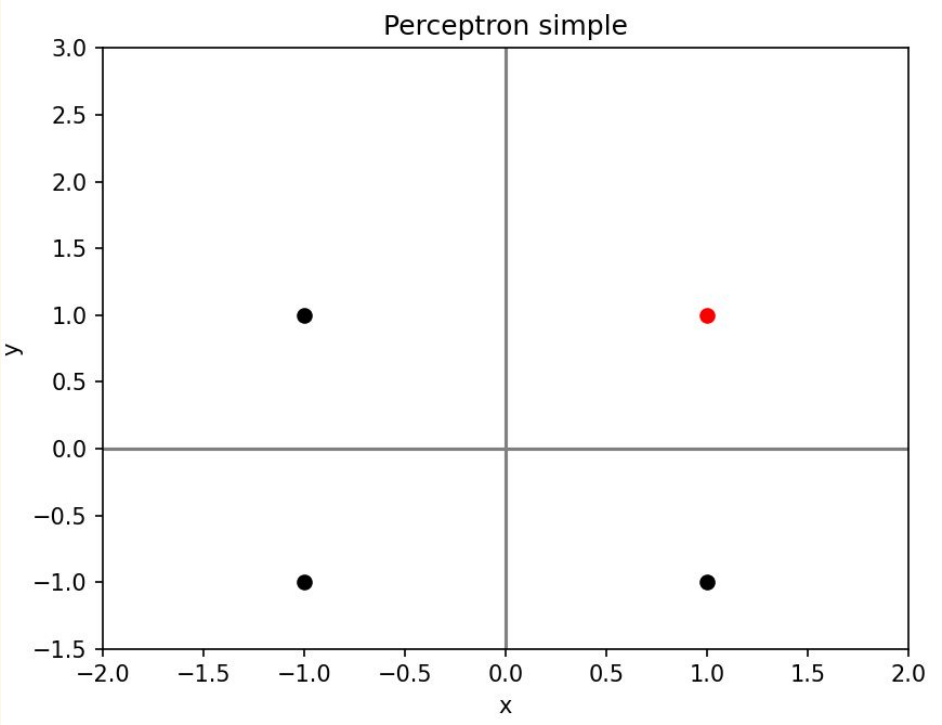
y = {−1, −1, −1, 1}

XOR



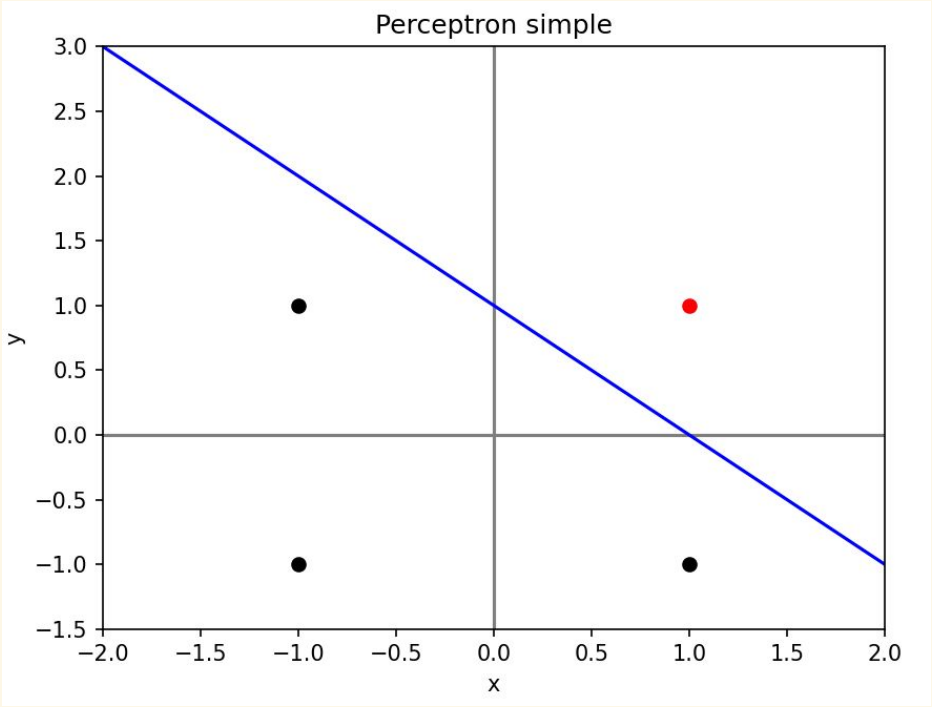
y = {1, 1, −1, −1}

Ej 1: AND



ξ_1	ξ_2	ζ	
-1	1	-1	●
1	-1	-1	●
-1	-1	-1	●
1	1	1	●

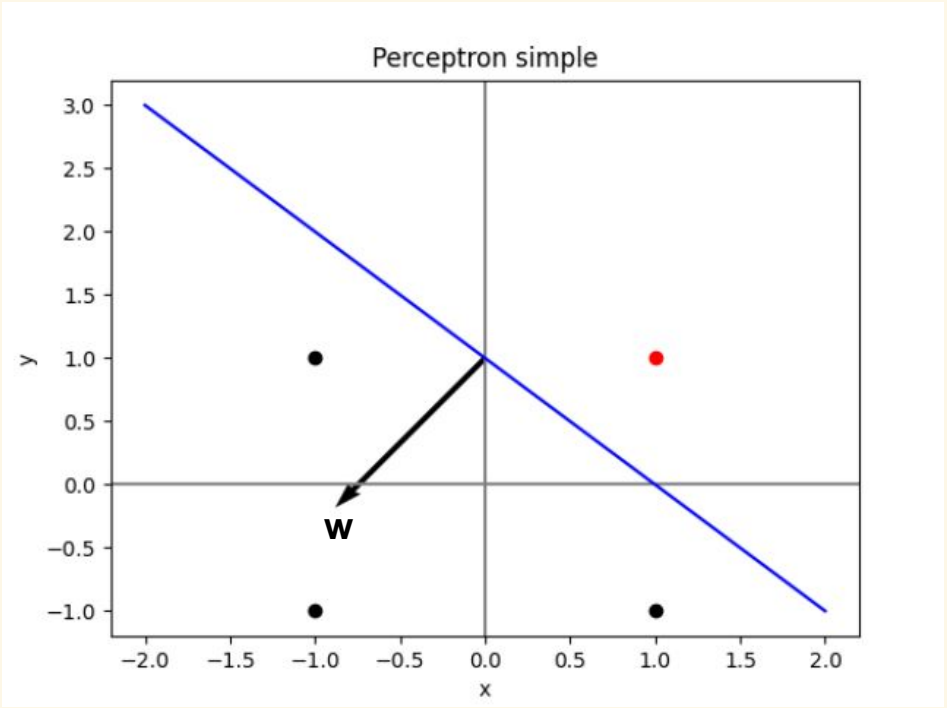
Ej 1: AND



$$y = -\frac{w_1}{w_2}x - \frac{w_0}{w_2}$$

η = 0.01
épocas = 1000

Ej 1: AND

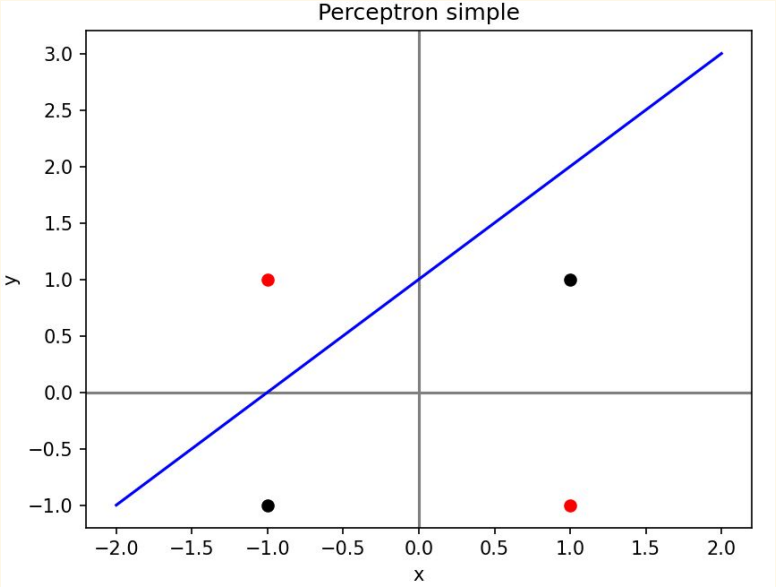


$$y = -\frac{w_1}{w_2}x - \frac{w_0}{w_2}$$

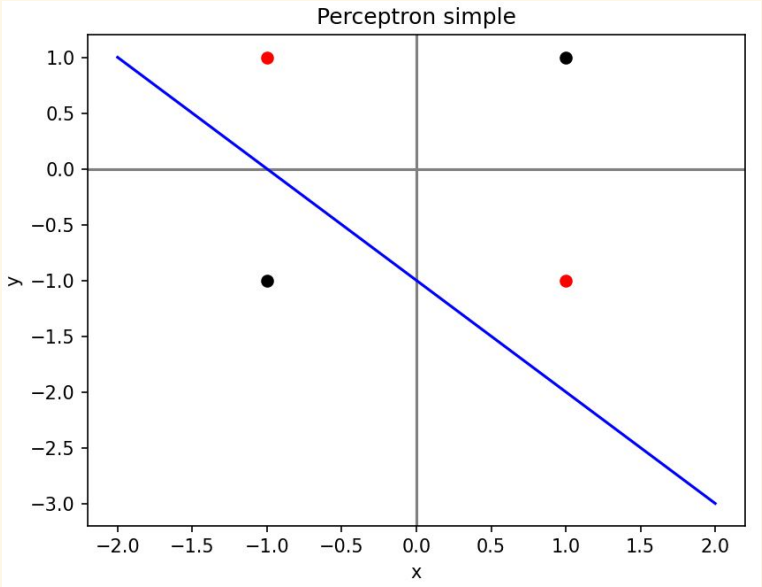
$$w = (-\frac{w_1}{w_2}, -1)$$

Ej 1: XOR

$$y = -\frac{w_1}{w_2}x - \frac{w_0}{w_2}$$



η = 0.01
épocas = 1000



η = 0.1
épocas = 1000

Ej 1: Weights

AND:
η = 0.1
épocas = 1000

Error mínimo = 0
W: [-0.2 0.2 0.2]

XOR
η = 0.1
épocas = 1000

Error mínimo = 2
W: [-0.2 0.2 -0.2]

Obs: la mejor aproximación al XOR es el resultado de un AND

¿Qué puede decir acerca del perceptrón simple escalón en relación a los problemas AND y XOR?

Ej 2: Perceptrón Simple Lineal y No Lineal

Ej 2: Perceptrón Simple Lineal

```
def calculate_error(x, y, w, p):
    error = 0
    for i in range(p):
        output = np.dot(x[i], w)
        error += (y[i] - output) ** 2
    return (1 / p) * error
```

Ej 2: Perceptrón Simple Lineal

`Épocas = 1000`

η	Error
0.001	1883.26
0.005	1885.40
0.01	**1884.65**
0.05	1947.95
0.1	6396.68

Obs: a mayor eta, más grande el error

Ej 2: Perceptrón Simple Lineal

$\eta = 0.01$

Límite	Error	Tiempo
100	1895.10	0.03125
500	1885.22	0.171875
1000	1883.20	0.3125
5000	1883.46	1.53125
10000	1883.00	3.171875

Ej 2: Perceptrón Simple Lineal

Conclusiones:

- Error muy elevado
- $+ \eta \longrightarrow +$ Error

$\longrightarrow$ No es linealmente separable

Ej 2: Perceptrón Simple No Lineal

- **Función de activación**: tangente hiperbólica
- **Conjunto de entrenamiento**: 150 elementos

- Los valores de salida del conjunto de entrenamiento fueron escalados al intervalo [-1, 1] con la fórmula:

$$2\,\frac{(X - t_{min})}{t_{max} - t_{min}}$$

t_min = valor mínimo del conjunto
t_max = valor máximo del conjunto

Ej 2: Perceptrón Simple No Lineal

error_range = 10

```
------ Not Linear Training... --------
Iterations =  1000
Time: 0.23292800000000002 s
Error_min = 37.72991612415149
Error_min (without scale) = 0.016160755100481795


------ Testing... --------
Test error =  2818.461447665695
Hits =  43
Success k=1:  86.0 %
```

Ejemplo de consola

Ej 2: Perceptrón No Lineal

Épocas = 1000 β = 0.7

η	Error	Error (sin escala)
0.001	71.32	0.0289
0.005	53.98	0.0219
0.01	**53.95**	**0.0219**
0.05	54.81	0.0223
0.1	55.92	0.0227

Ej 2: Perceptrón No Lineal

$\eta = 0.01$ $\beta = 0.7$

Límite	Error	Error (sin escala)	Tiempo
100	72.68	0.0295	0.047
500	**53.96**	**0.0219**	**0.25**
1000	53.97	0.0219	0.5
5000	53.92	0.0219	2.5
10000	53.91	0.0219	4.95

Ej 2: Perceptrón No Lineal

$\eta = 0.01$ Épocas = 1000

β	Error	Error (sin escala)
0.6	54.67	0.0222
0.7	**53.96**	**0.0219**
0.8	54.0148	0.0219
1	**53.93**	**0.0219**

Ej 2: Validación cruzada

k = 4

k1 = [0:50]
k2 = [50:100]
k3 = [100:150]
k4 = [150:200] (test)

$\eta = 0.01$ Épocas = 1000 $\beta = 0.7$

k	Error (sin escala)	Success
1	0.01798	80.0 %
2	**0.01621**	**86.0 %**
3	0.02496	78.0 %

Obs: el conjunto de entrenamiento k2 es el mejor

¿Como podría escoger el mejor conjunto de entrenamiento?

Ej 2: Comparamos distintos k

$\eta = 0.01$ Épocas = 1000 $\beta = 0.7$

	size	Success (avg)
k=40	5	40.0%
k=20	10	70.0%
k=8	25	76.0 %
k=4	50	80.0 %
k=2	100	80.0 %

Ej 3: Perceptrón Multicapa

Ej 3.1: XOR

Problemas a analizar XOR

Entrada x = {{−1, 1}, {1, −1}, {−1, −1}, {1, 1}}

Salida esperada y = {1, 1, -1, -1}

n = 0.01
Épocas = 1000
batch = true
momentum = true

Esperado	Obtenido capas : [10]	Obtenido capas: [10,10,10]	Obtenido capas: [2]	Obtenido capas: [2,2,2,2]
1	0.9468	0.9556	0.9117	-0.3298
1	0.9491	0.9600	0.9098	0.9305
-1	-0.9492	-0.9628	-0.9337	-0.3277
-1	-0.9510	-0.9448	-0.9336	-0.327
Total error	0.0509	**0.0441**	0.0777	0.6859

obs: es mejor aumentar la cantidad de nodos en cada capa que la cantidad de capas

```
n = 0.01
Épocas = 1000
```

Esperado	Obtenido capas : [10] batch: true momentum: true	Obtenido capas : [10] batch: true momentum: false	Obtenido capas : [10] batch: false momentum: true	Obtenido capas : [10] batch false momentum false
1	0.9468	0.8506	0.9309	0.8451
1	0.9491	0.8279	0.9349	0.8806
-1	-0.9492	-0.8561	-0.9388	-0.8471
-1	-0.9510	-0.8321	-0.9526	-0.8449
Total error	0.0509	0.1582	0.0606	0.1455

```
batch = true
momentum = true
Épocas = 1000
```

Esperado	Obtenido capas : [10] n = 0.001	Obtenido capas : [10] n = 0.01	Obtenido capas : [10] n = 0.1
1	0.7435	0.9468	0.9529
1	0.72425	0.9491	0.9613
-1	-0.72204	-0.9492	-0.9541
-1	-0.733228	-0.9510	-0.9533
Total error	0.2693	0.0509	0.0445

Ej 3.2: Conjunto de números del 0 al 9

Discriminar si un número es par, con entradas dadas por el conjunto de números decimales del 0 al 9.

K = 2
Épocas = 1000

Data	Valor esperado	Predicción
0	1	0.9551
1	-1	-0.9554
2	1	0.9540
3	-1	-0.9566
4	1	0.9589
5	-1	-0.9520
6	1	0.9550
7	-1	-0.9555
Testing		
8	1	0.8690
9	-1	0.0146

Entrenando solo pares

Testing	Esperado	Obtenido
1	-1	0.801
3	-1	0.995
5	-1	0.994
7	-1	0.983
9	-1	0.987

Entrenando solo impares

Testing	Esperado	Obtenido
0	1	-0.9845
2	1	-0.8977
4	1	-0.9218
6	1	-0.9628
8	1	-0.9755

¿Que podrıa decir acerca de la capacidad para generalizar de la red?

Ej 3.3: Conjunto de números del 0 al 9 con 10 unidades de salida

Construya un perceptrón multicapa, con entradas dadas por el conjunto de números decimales del 0 al 9. Con 10 unidades de salidas de modo que cada salida representa a un dıgito.

Error total = 1
Sin ruido

Ejemplo: 0
Resultado = [1 0 0 0 0 0 0 0 0 0]
Obtenido = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10]

error = sum(errores_bit_a_bit)/cant_de_bits
cant_de_bits = 10

Resultado	Error total
0	0.017
1	0.00028
2	0.03798
3	0.023712
4	0.013655
5	0.041551
6	0.0849
7	0.020145
8	0.02108
9	0.00925

```
Error total = 4
p = 0.02
```

Resultado	Error total
0	0.00214
1	0.00088
2	0.0286
3	0.0898
4	0.1061
5	0.00065
6	0.0443
7	0.0025
8	0.29122
9	0.0551

```
Error total = 6
p = 0.05
```

Resultado	Error total
0	0.0332
1	0.04546
2	0.04561
3	0.3208
4	0.0917
5	0.2700
6	0.03576
7	0.31667
8	0.33099
9	0.11019

Ej 3.3: Conjunto de números del 0 al 9 con 10 unidades de salida

Conclusiones:

- a mayor ruido, mas error
- sin ruido el resultado es bueno

¡Gracias!