Perceptrón Simple y Multicapa

Integrantes

Azul Kim

Felipe Cupitó

Juan Manuel De Luca

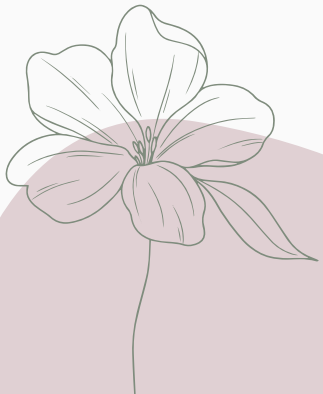
Hernán Finucci

Sol Konfederak

1

Perceptrón Simple Escalonado

AND y XOR



Datos de entrada y salida

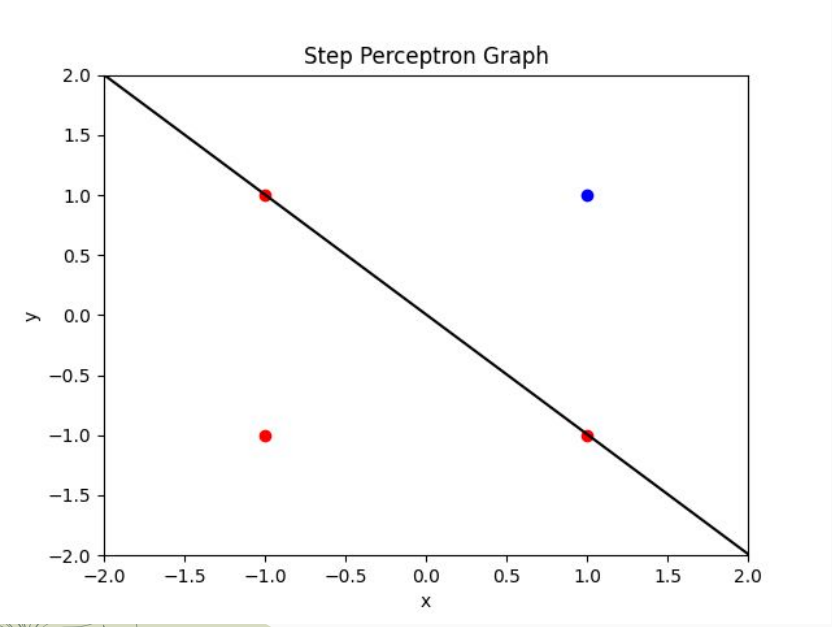
AND

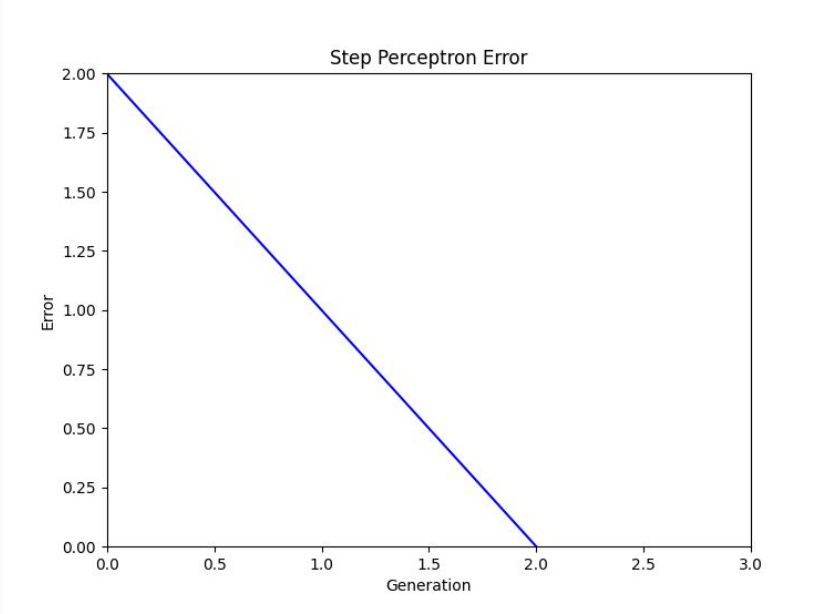
ξ^μ	ξ^μ	ζ^μ
-1	1	-1
1	-1	-1
-1	-1	-1
1	1	1

XOR

ξ^μ	ξ^μ	ζ^μ
-1	1	1
1	-1	1
-1	-1	-1
1	1	-1

AND - Gráfico





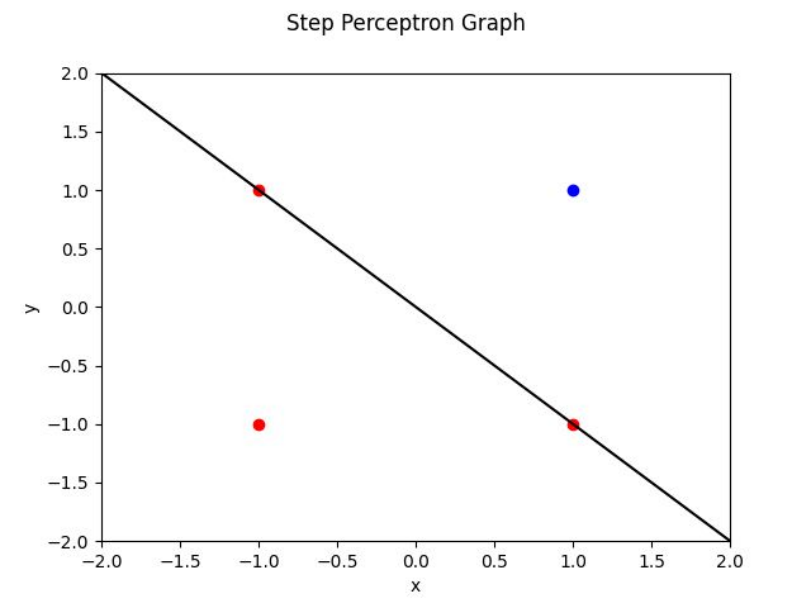
η = 0,001

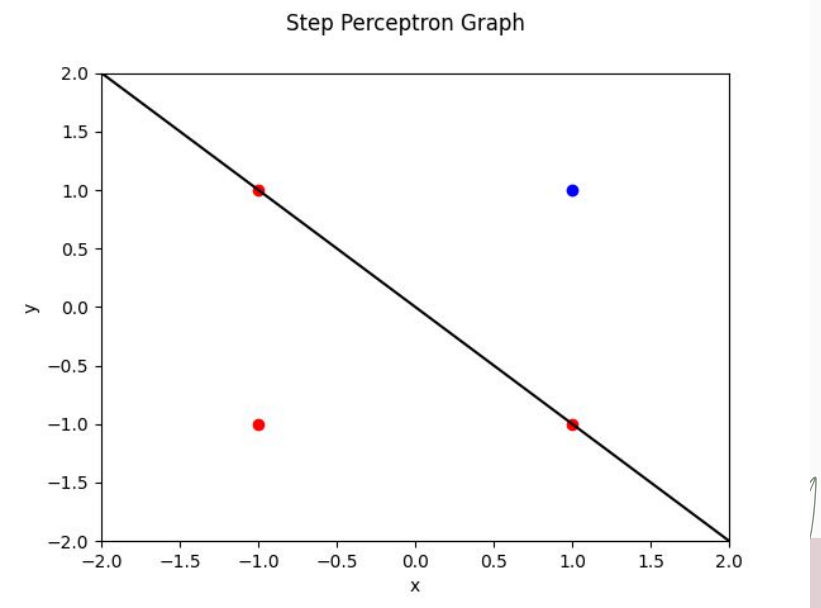
Epochs = 1000

Error mín = 0

w_min = [0,43 0,43 -0,004]

AND - Comparación de η





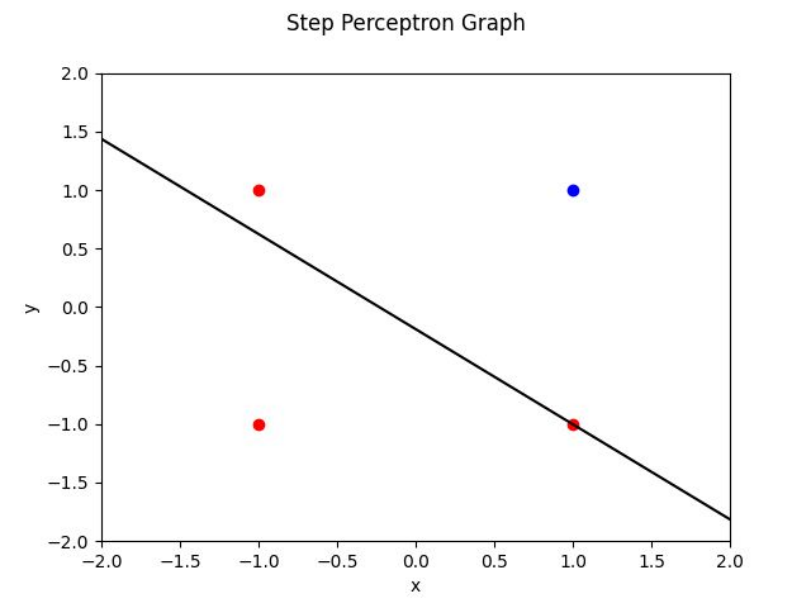
η = 0,001

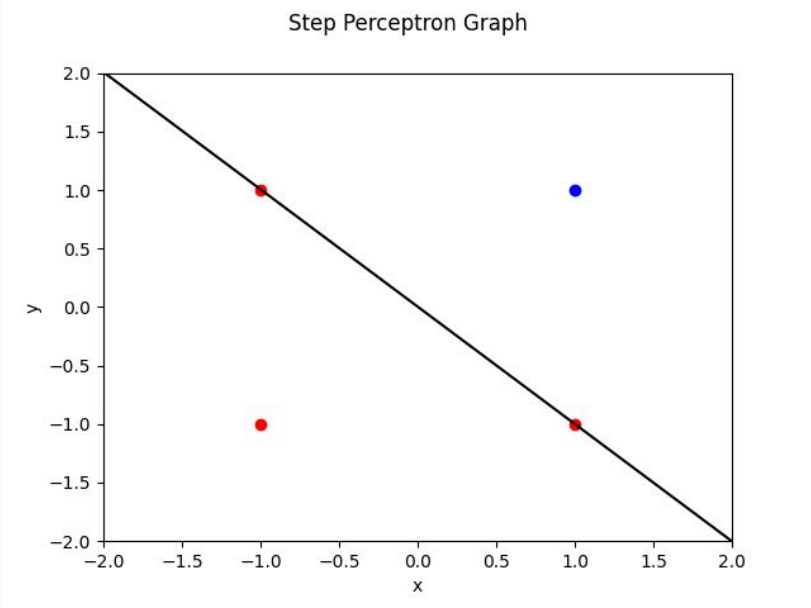
Epochs = 1000

η = 0,1

Epochs = 1000

AND - Comparación de epochs





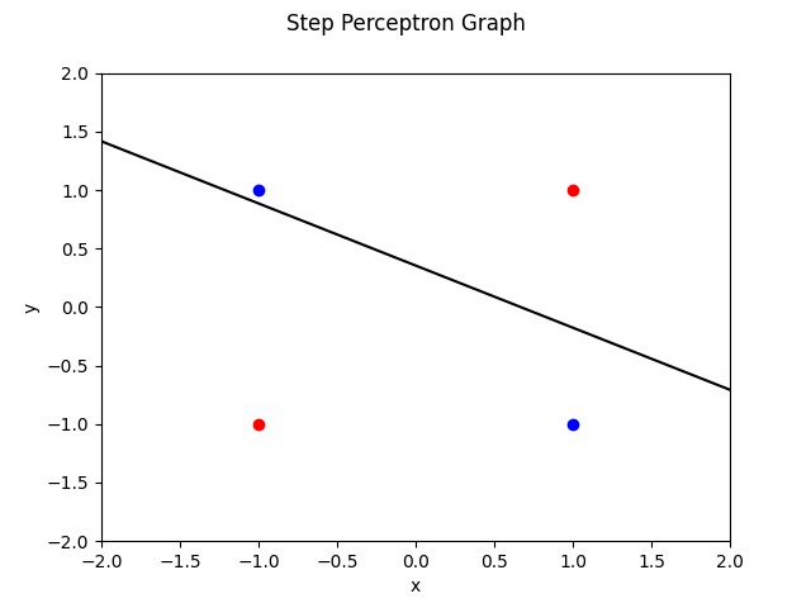
η = 0,001

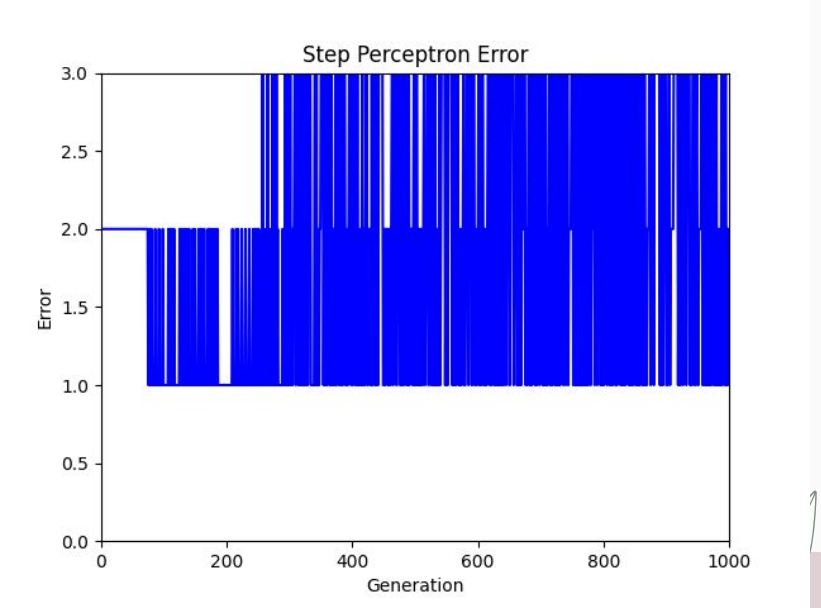
Epochs = 100

η = 0,001

Epochs = 1000

XOR - Gráfico





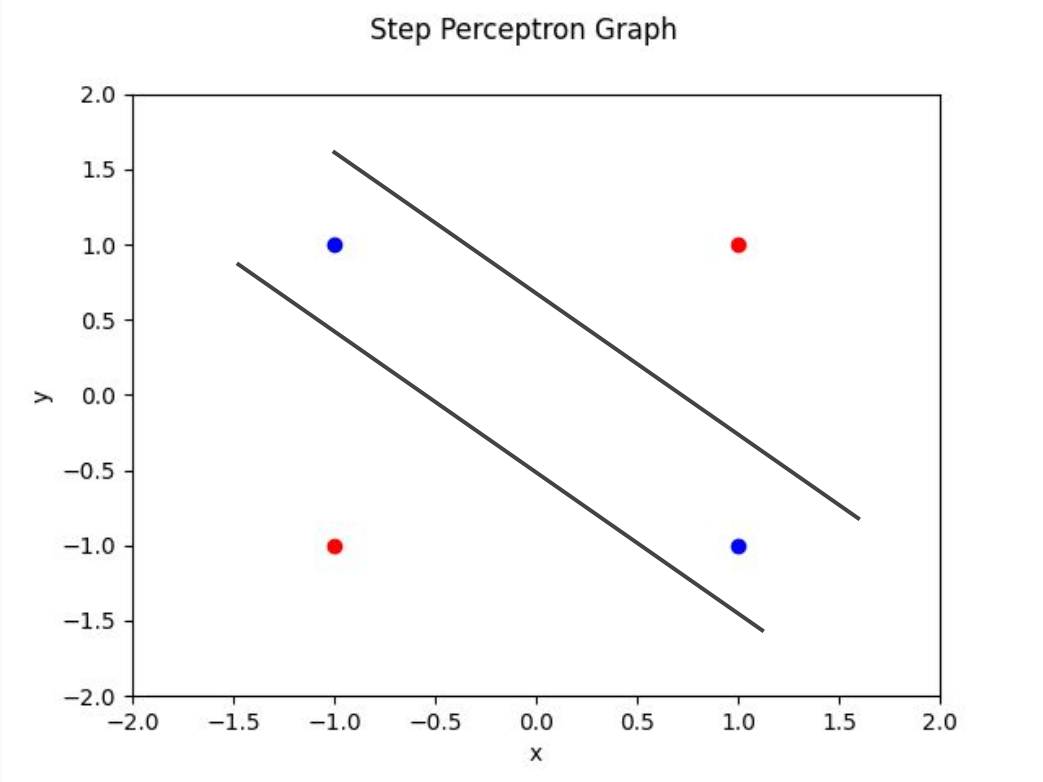
η = 0,001

Epochs = 1000

Error mín = 1

w_min = [-0,002 -0,003 0,001]

XOR - Resultado esperado



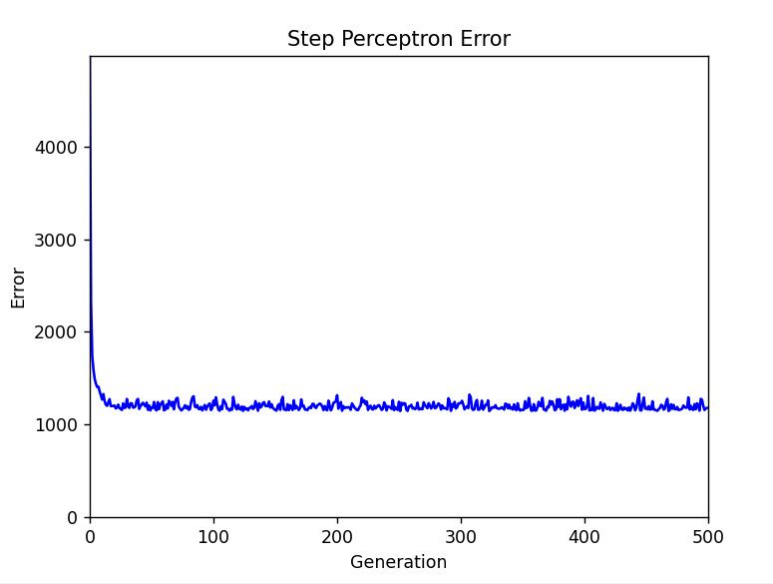
Conclusiones

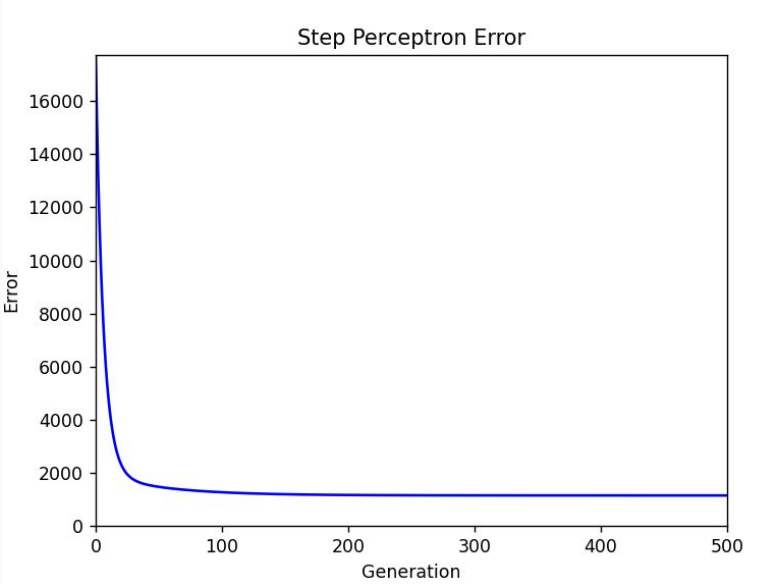
AND	XOR
Es un problema *linealmente* separable (se puede trazar una recta que separa las dos clases de datos).	No es un problema que se pueda resolver con este tipo de perceptrón, se requiere de dos rectas que separen las dos categorías.

2

Perceptron Simple Lineal y No Lineal

Perceptrón simple lineal (según η)





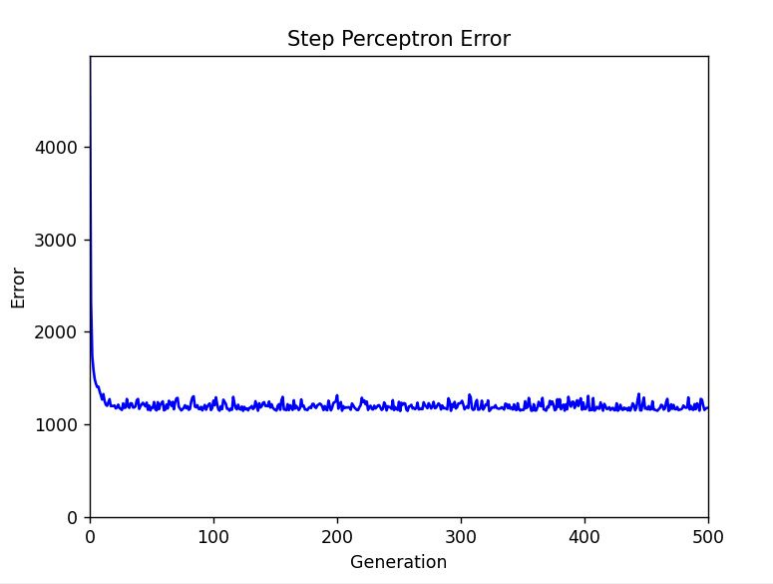
η = 0,01 β = 0,8

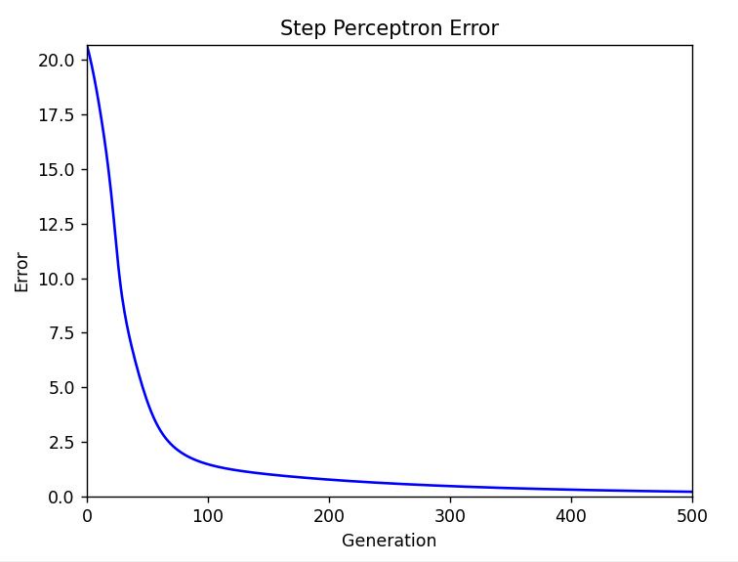
Epochs = 500

η = 0,001 β = 0,8

Epochs = 500

Perceptrón simple no lineal (según η)





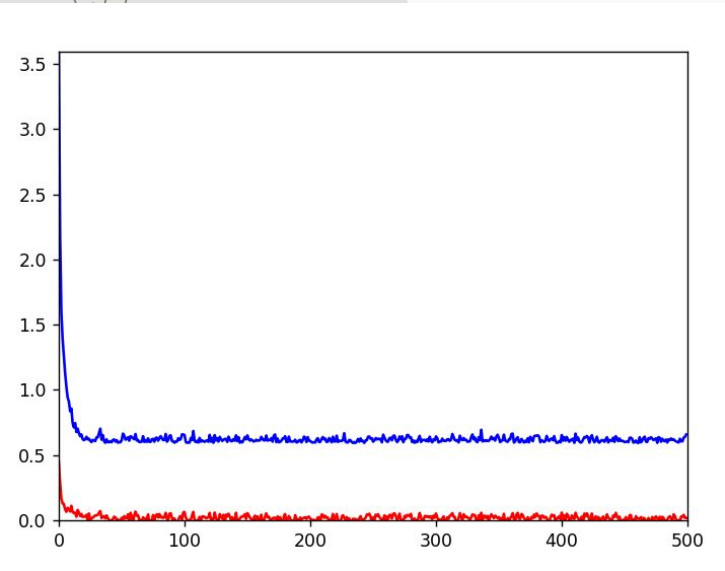
η = 0,01 β = 0,8

Epochs = 500

η = 0,001 β = 0,8

Epochs = 500

No lineal - Comparación de errores



η = 0,01

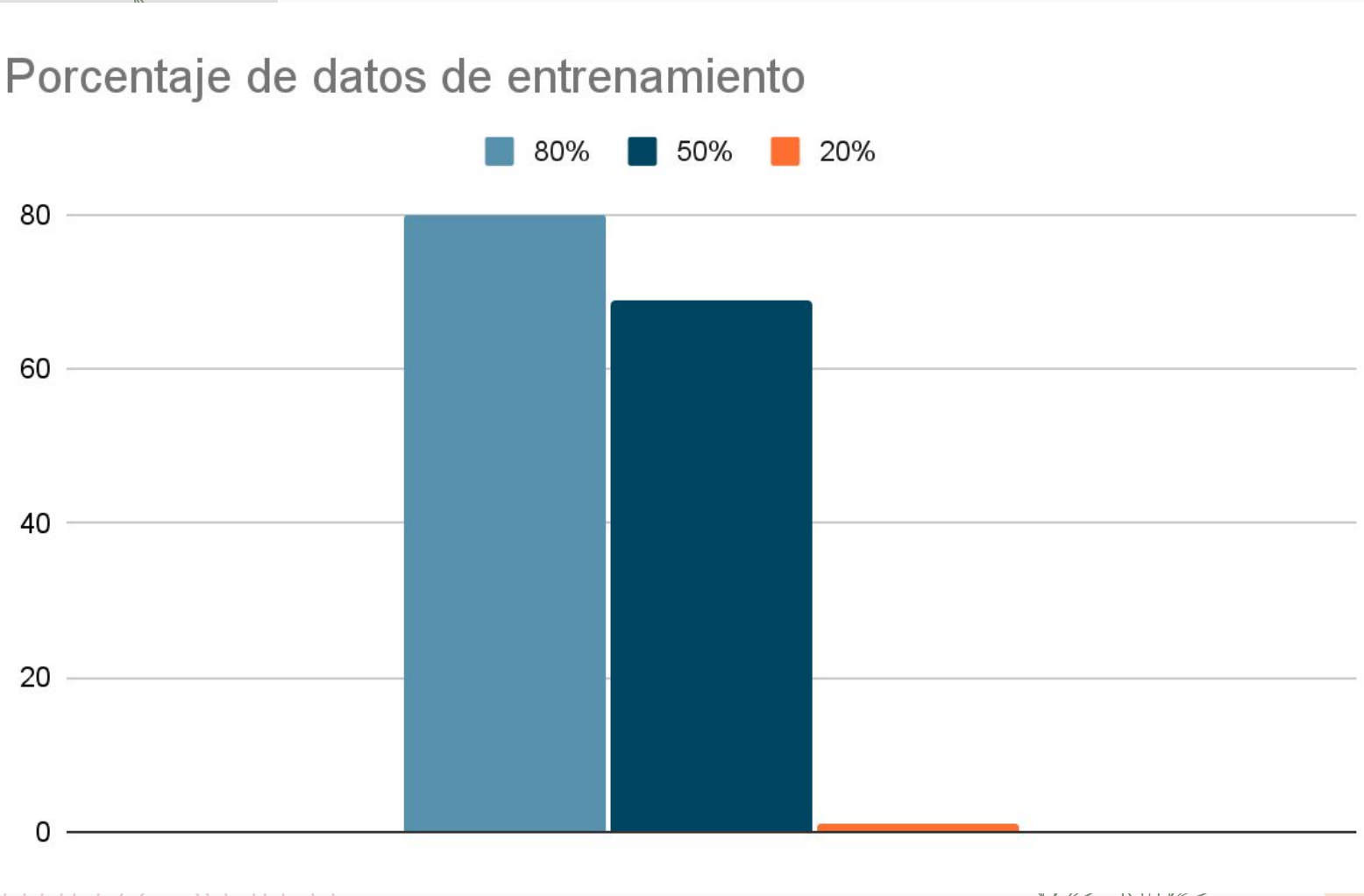
Epochs = 500

β = 0,8

Rojo -> error del test set

Azul -> error del training set

β = 0,8


Porcentaje de datos de entrenamiento
80%
50%
20%
80
60
40
20
0

Conclusiones

Sobre la capacidad de cada perceptrón para aprender la función:

- Si se observan los gráficos de error se puede ver que la capacidad de aprendizaje aumenta ya que el error disminuye.

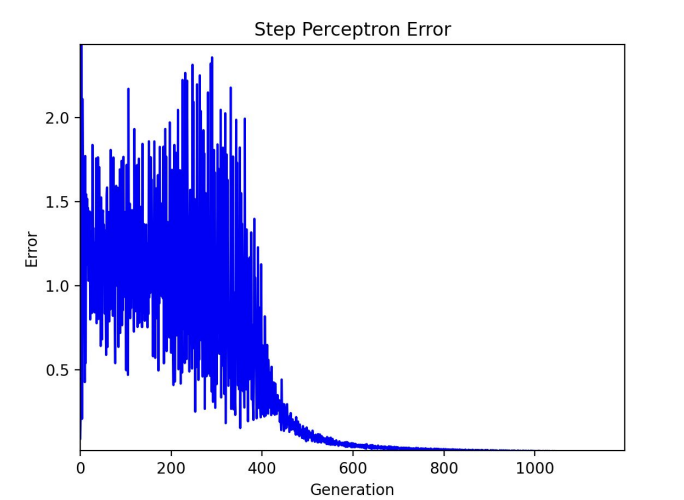
Sobre el conjunto de entrenamiento:

- La elección del mejor conjunto de entrenamiento dependerá de cuál es el que menor error presenta.

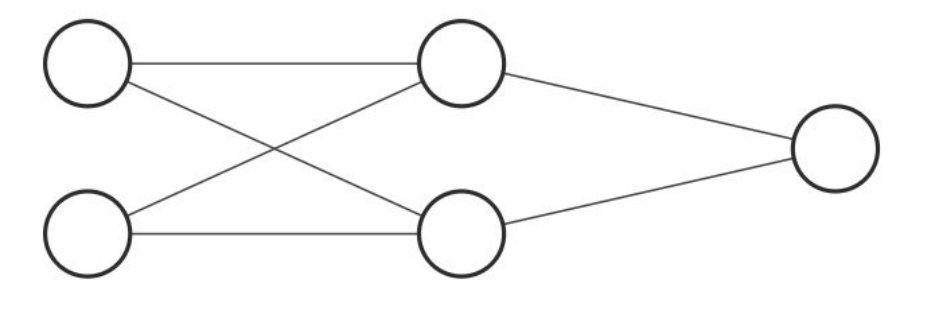
3

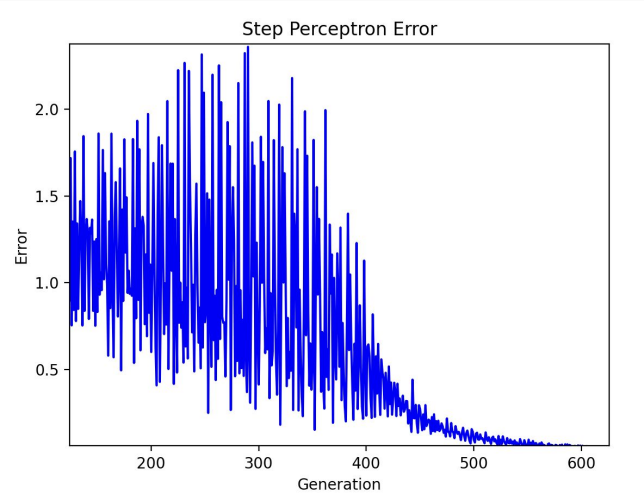
Perceptron Multicapa

Funcion XOR



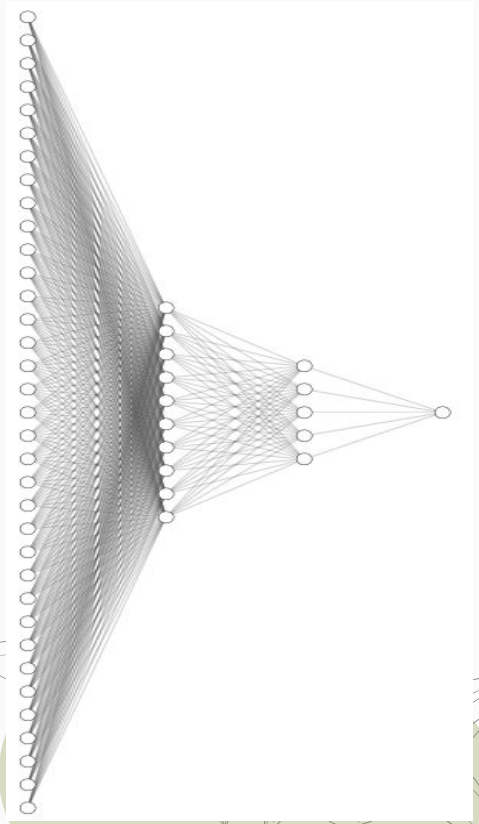
- Un nodo inicial por input
- Dos nodos intermedios
- Unico nodo final
- Tasa de aprendizaje de 0.1

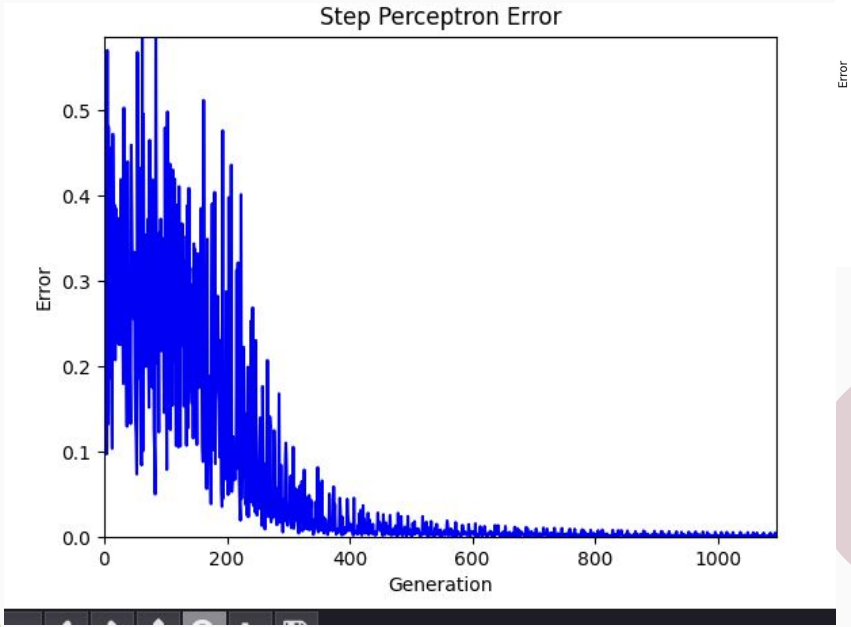


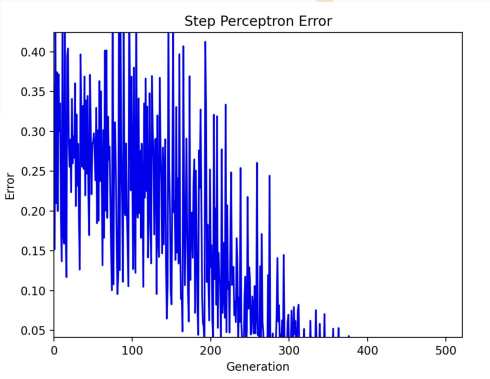


Funcion PAR

- Un nodo inicial por cada “pixel” (7x5)
- Dos capas intermedias de 10 y 5 nodos respectivamente + bias en cada capa
- Unico nodo final (par o impar)
- Tasa de aprendizaje de 0.1

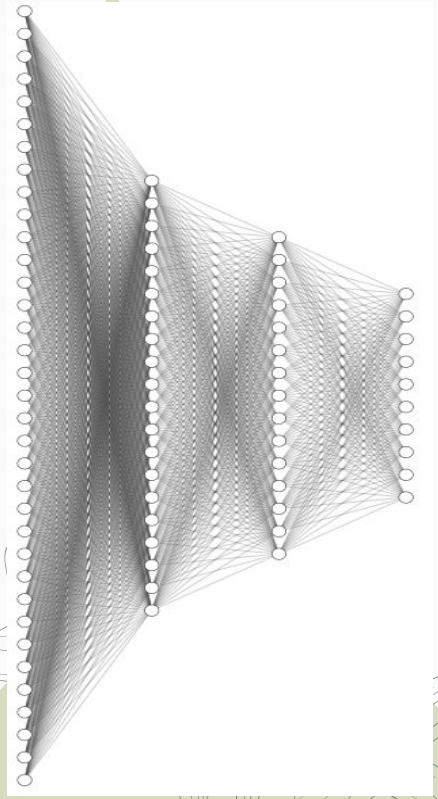


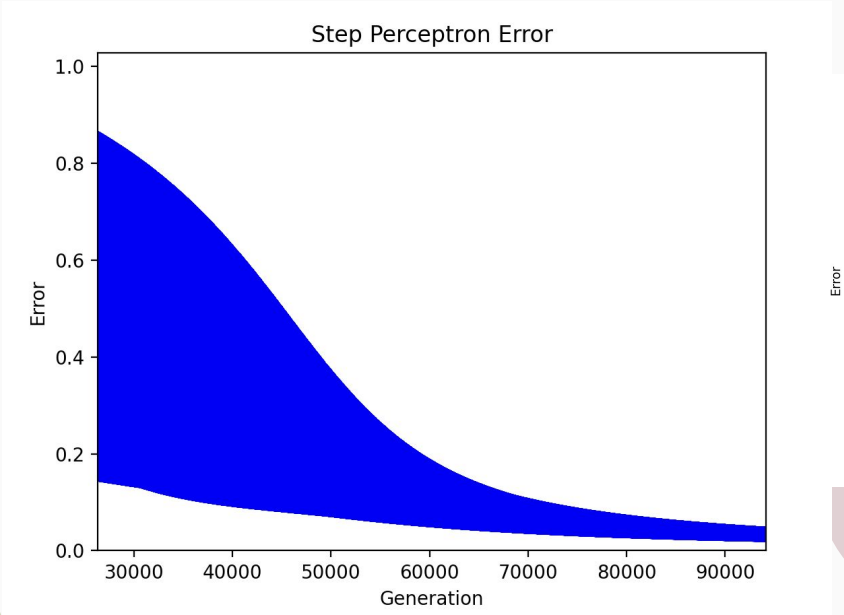


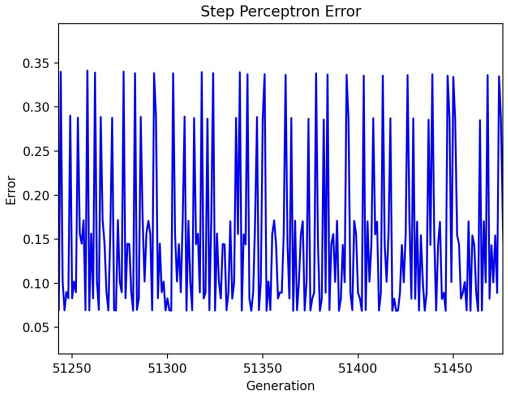


Funcion NUMBER

- Un nodo inicial por cada “pixel” (7x5)
- Dos capas intermedias de 20 y 15 nodos respectivamente + bias en cada capa
- 10 nodos finales (uno para cada número posible)
- Tasa de aprendizaje de 0.1

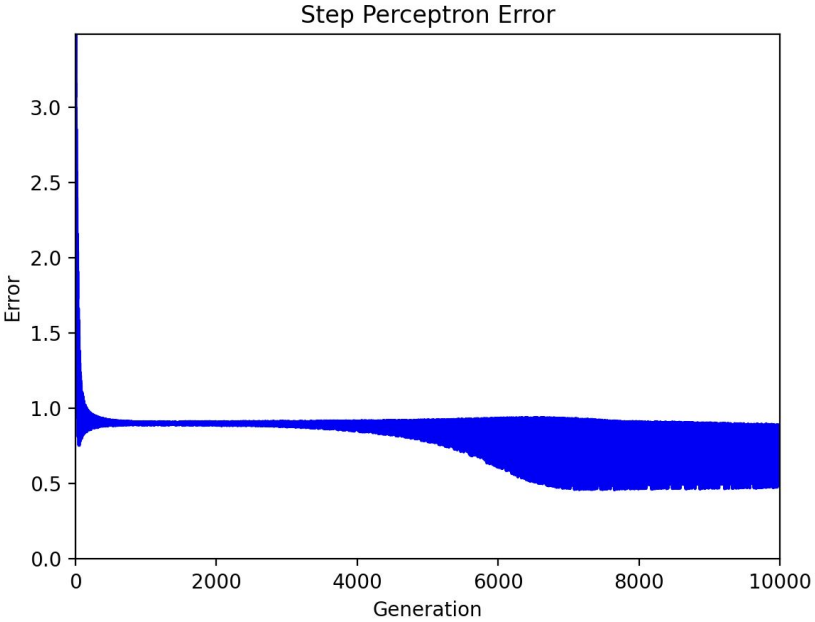


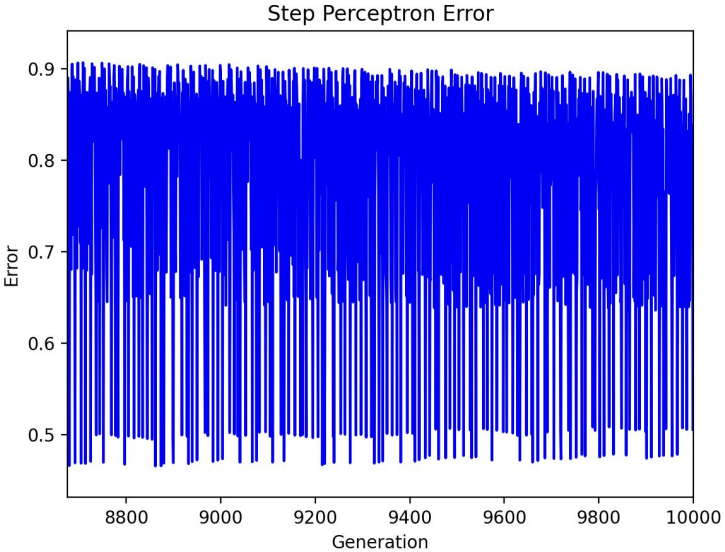




Función NUMBER con más neuronas

- Un nodo inicial por cada “pixel” (7x5)
- 4 capas intermedias de 20, 15, 10 y 10 neuronas respectivamente
- Unico nodo final (par o impar)
- Tasa de aprendizaje de 0.1





Funcion NUMBER modificando la entrada de testeo (η=0.01, max_gen=10000)

Output: 0.989

Output: 0.951

[0.02456512784112388,

0.9511763216951983,

0.05193819166990036,

0.09183692027195284,

0.0003393033148881079 4,

0.04888153528028594,

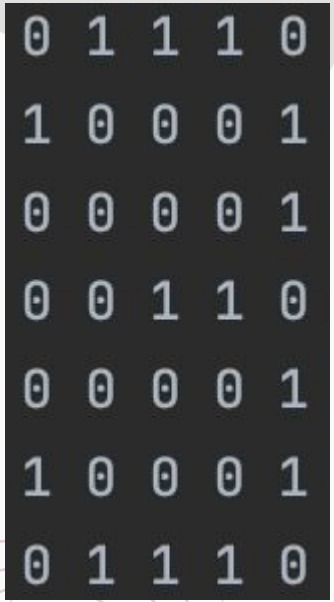
0.02239636495311867 6,

0.05578903393024555 4,

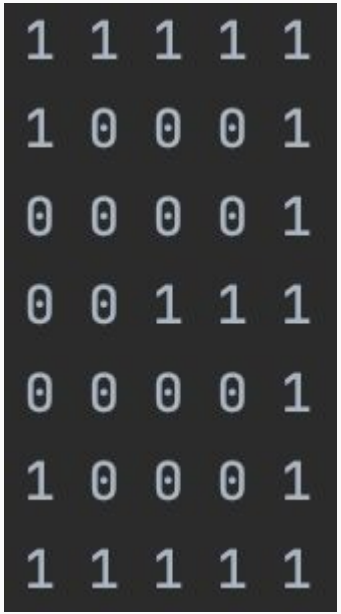
0.0019042670547446453 2,

0.09671389055833823]

Funcion NUMBER modificando la entrada de testeo (η=0.01, max_gen=10000)



Output: 0.982



Output: 0.922

[0.03157905214412388,
0.036708911613961436,
0.04344992027195284,
0.9218252214835786,
0.0001253039680127 34,
0.02375153528028594,
0.01874636495311867 6,
0.04784671560424555 4,
0.0001732670547446 4532,
0.04407789055833823]

Conclusiones

- Se puede representar la operación XOR usando dos neuronas de entrada que eventualmente convergen en una neurona de salida
- Modificar un poco el bitmap que se usó para entrenar el modelo no modifica la salida. Sin embargo, al tratarse de poca cantidad de “pixeles”, un par de cambios puede alterar mucho la salida esperada
- Incrementar la cantidad de nodos/layers no necesariamente lleva a un error menor