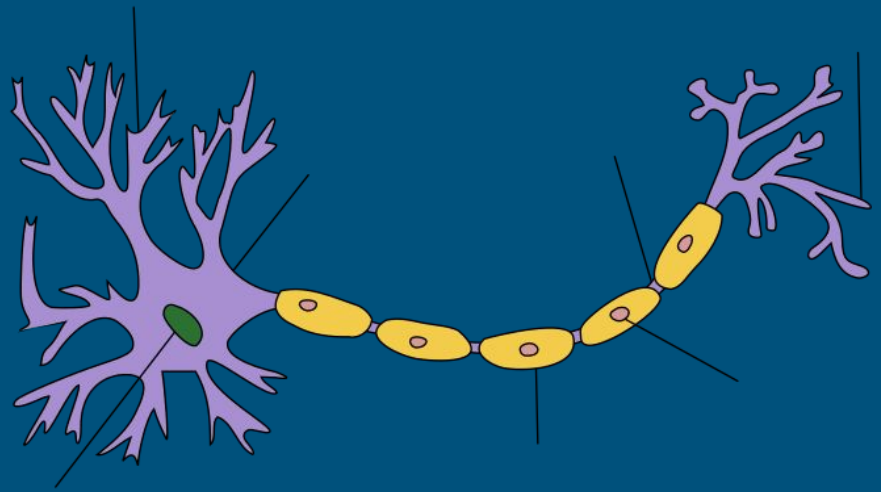
Redes neuronales



TP3: Perceptron simple y multicapa

Introducción

- Tipos de perceptrones simples:
  - Escalón
  - Lineal
  - No lineal (logística o tanh)
- Multicapa

Algoritmo Genérico de Red Neuronal

- Clase Network(structure, activation, seed, args={b})
  - feed
  - retropropagation
  - train
    - entrena hasta cumplir la condición dada

Algoritmo Genérico de Red Neuronal

Network
activation activation_der bias : int rng structure : list w : list
error(dataset) feedforward(input, with_values) randomize() retropropagation(expected, H, V, lr) train(dataset, batch_size, target_error, epochs, learning_rate, momentum, callback, epoch_callback)

Algoritmo Genérico de Red Neuronal

Cálculo del Error

$$Di = |Vi - Oi|$$

Distancia euclidiana

$$Ei = 0.5 * Di^2$$

$$E = \sum Ei$$

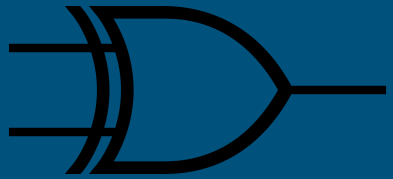
Problema 1

- Perceptrón simple con activación escalón para calcular:
  - and

Función lógica 'Y' con entradas
$x = \{\{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\}\}$,
y salida esperada
$y = \{-1, -1, -1, 1\}$.

  - xor

Función lógica 'O exclusivo' con entradas
$x = \{\{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\}\}$,
y salida esperada
$y = \{1, 1, -1, -1\}$.

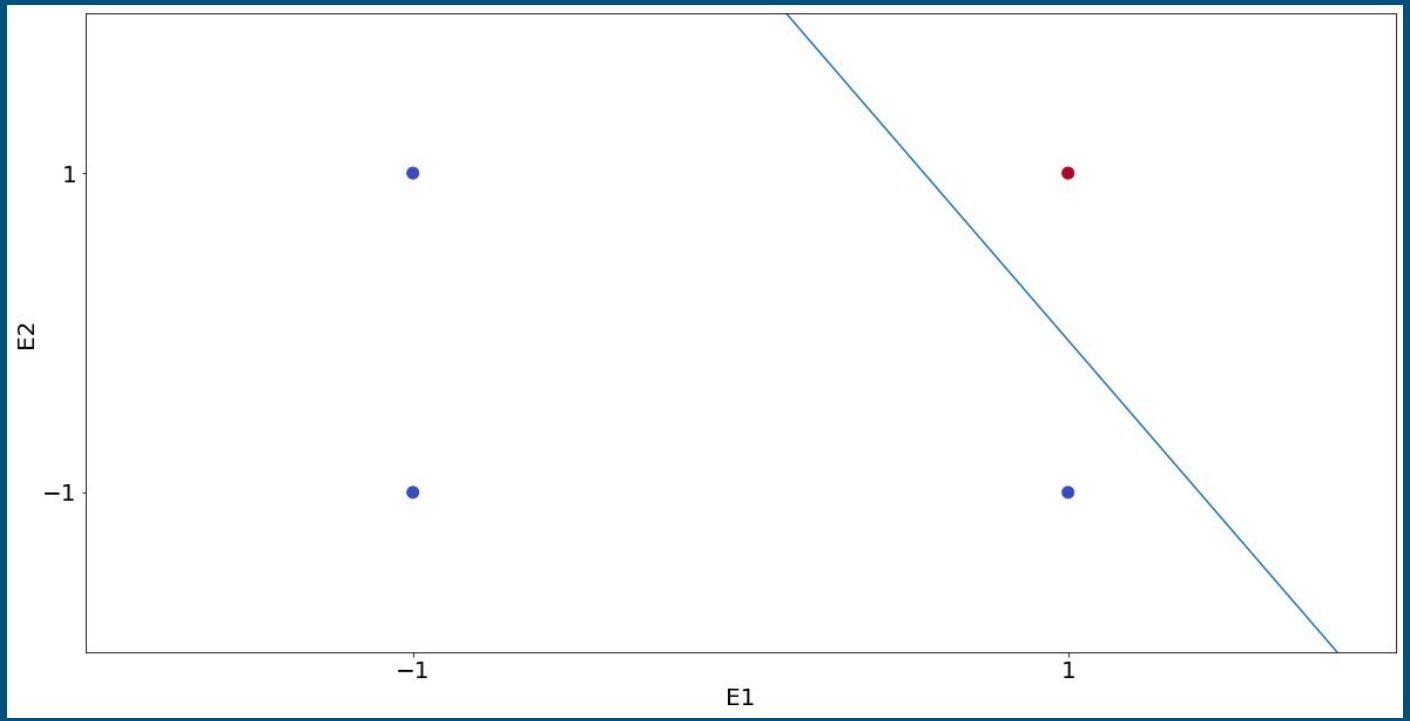


Resolución

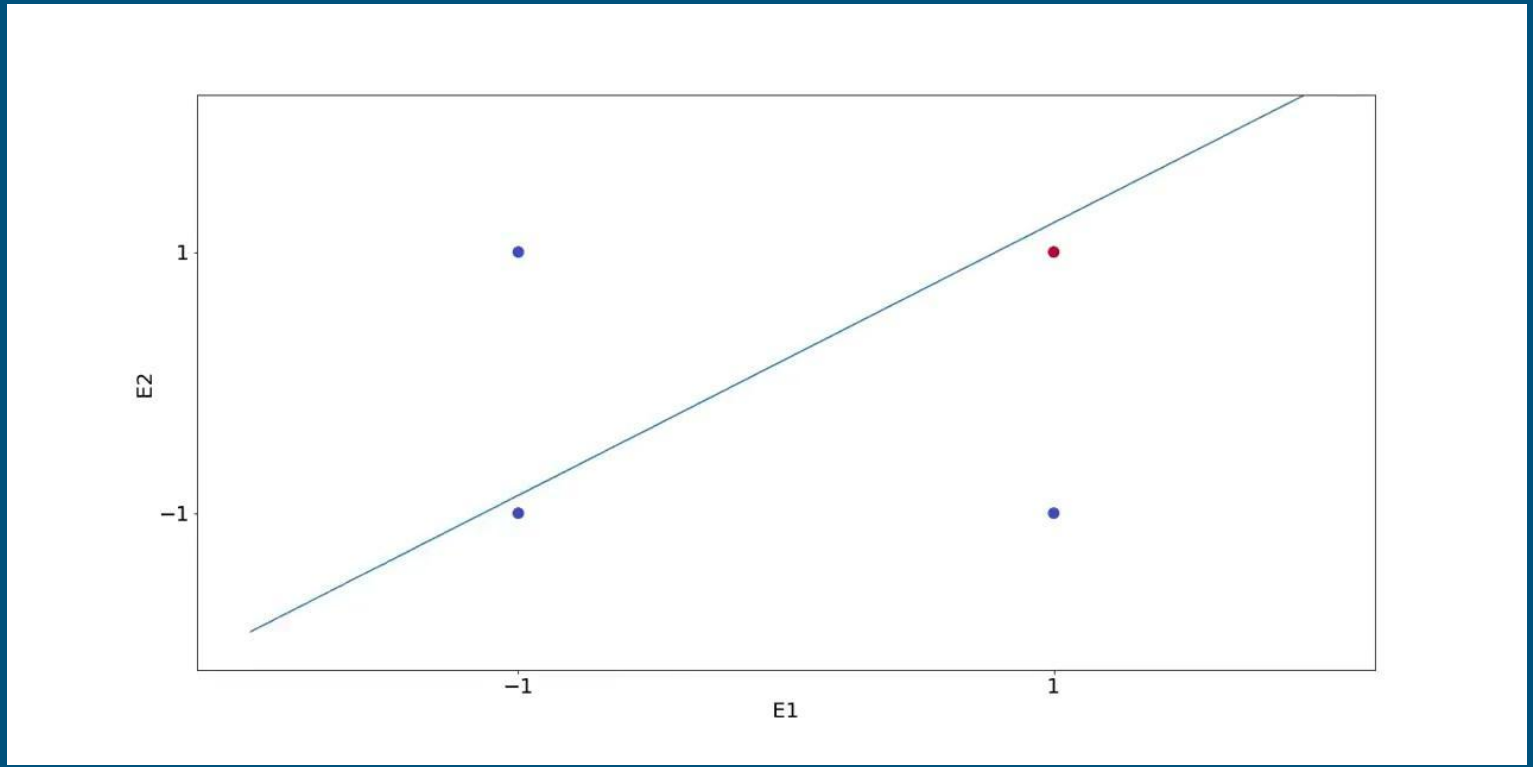
- 2 valores de entrada y 1 de salida,
- Forma de activación step

Resultados 1 a

Epochs: 3



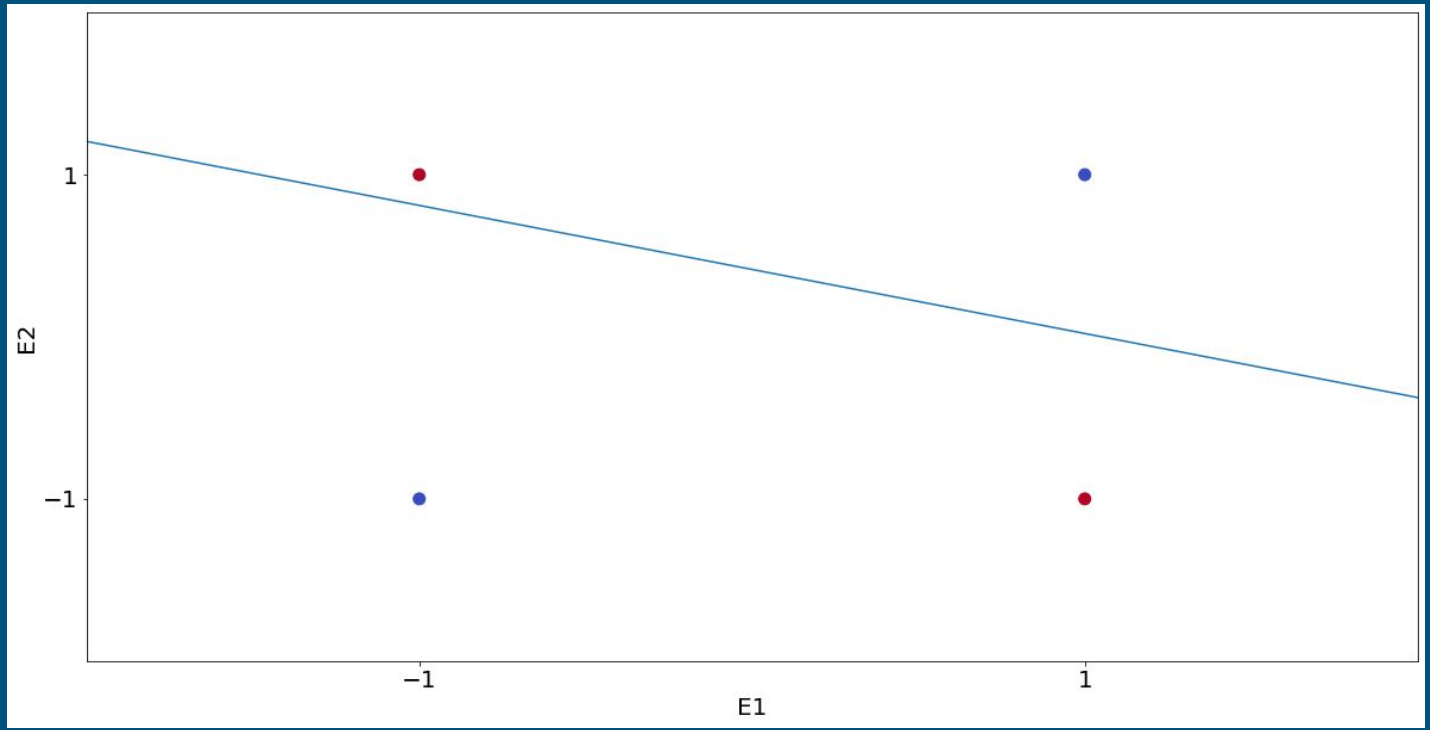
Resultados 1 a



Conclusiones 1 a

- Al ser un conjunto de datos linealmente separables, el perceptrón simple escalonado se adapta de buena manera para resolver el and.

Resultados 1 b



Epochs: 1000

Conclusiones 1 b

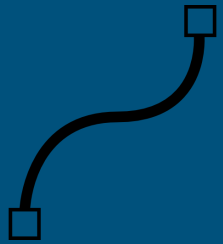
- Como el xor no es un problema linealmente separable el perceptrón simple no logra resolver el problema

Problema 2

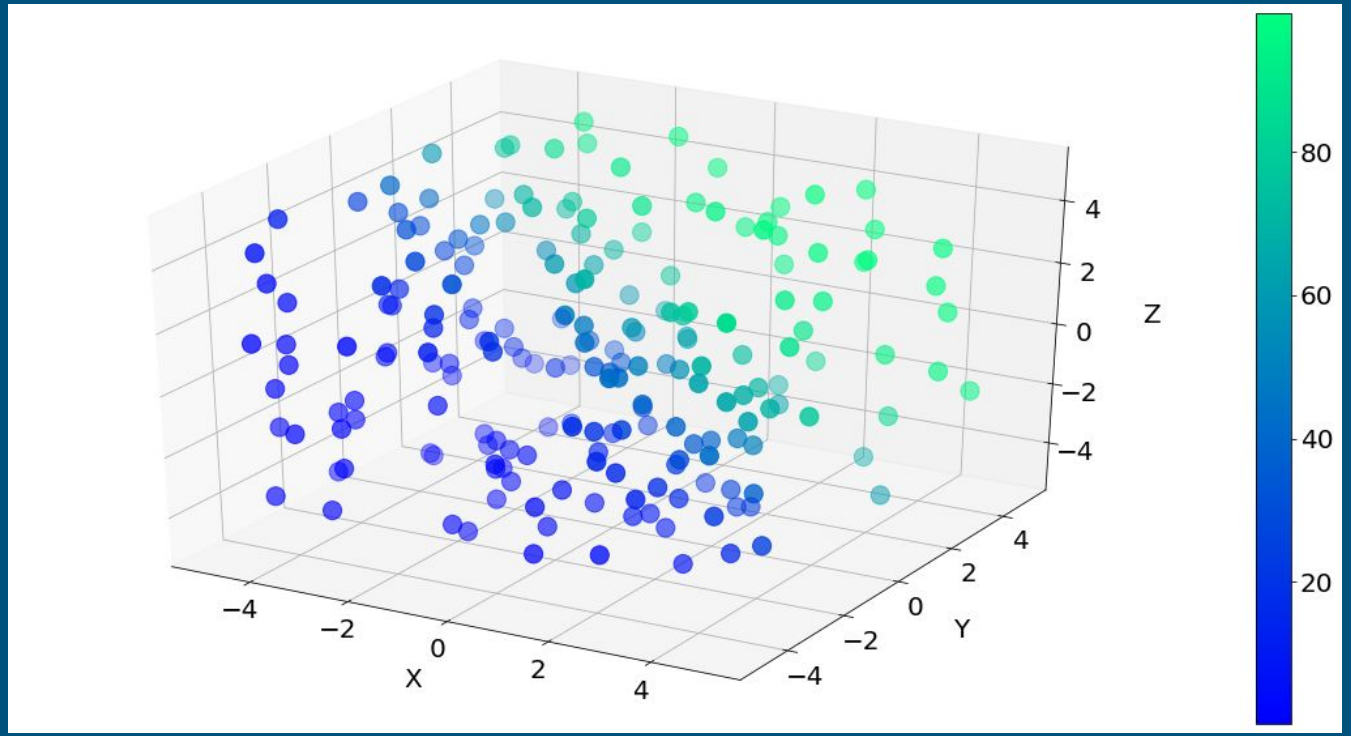
- Perceptrón simple lineal y no lineal
  - Evaluar su capacidad con el dataset dado
  - Evaluar capacidad de generalizar del no lineal (entrenamiento - testeo)
  - Escoger mejor conjunto de entrenamiento
  - Máxima capacidad de generalización

Parámetros

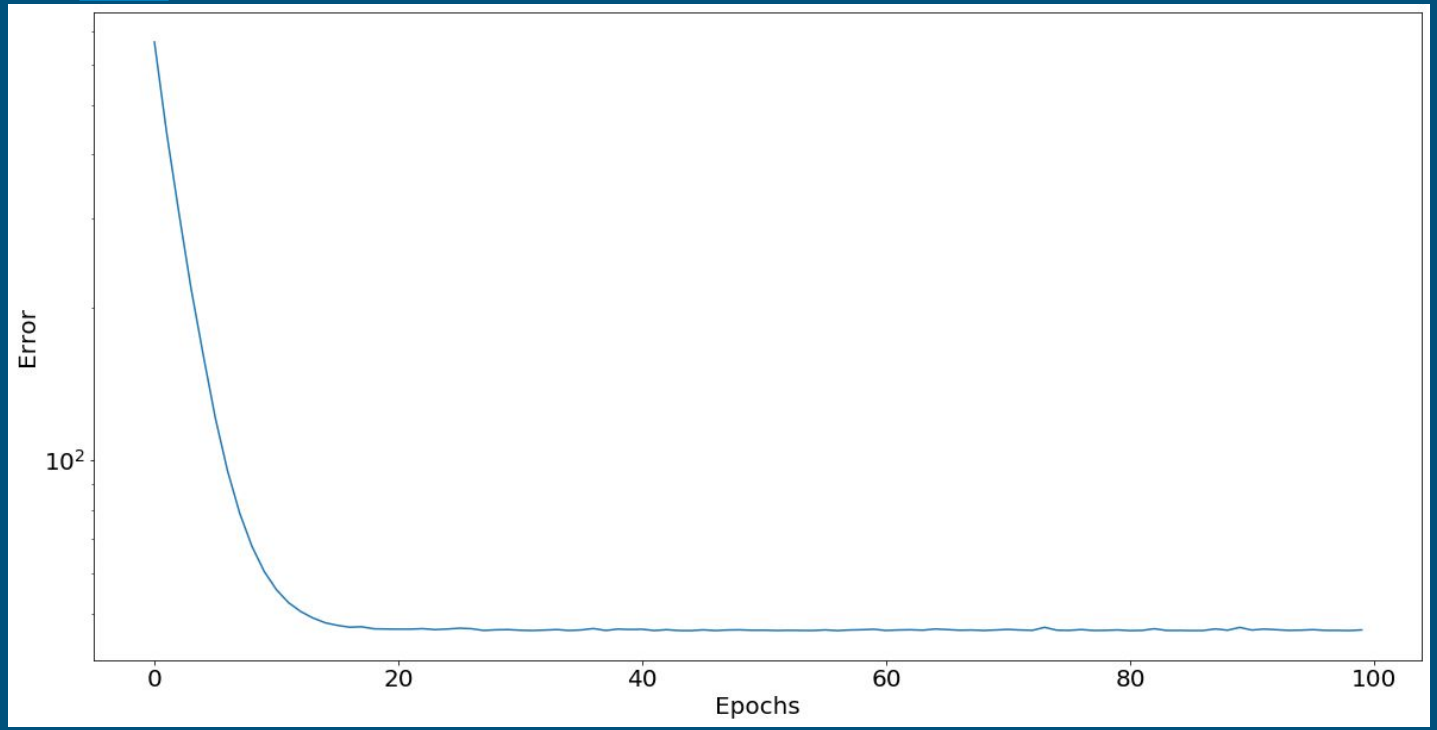
- Lineal
  - 3 valores de entrada, 1 de salida
  - b: 0.8 (coeficiente de x)
  - epochs: 100
  - learning rate: 0.001
  - momentum: 0.4
- No Lineal
  - 3 valores de entrada, 1 de salida
  - b: 0.8
  - epochs: 100
  - learning rate: 0.01
  - momentum: 0
  - activación: tanh



Dataset



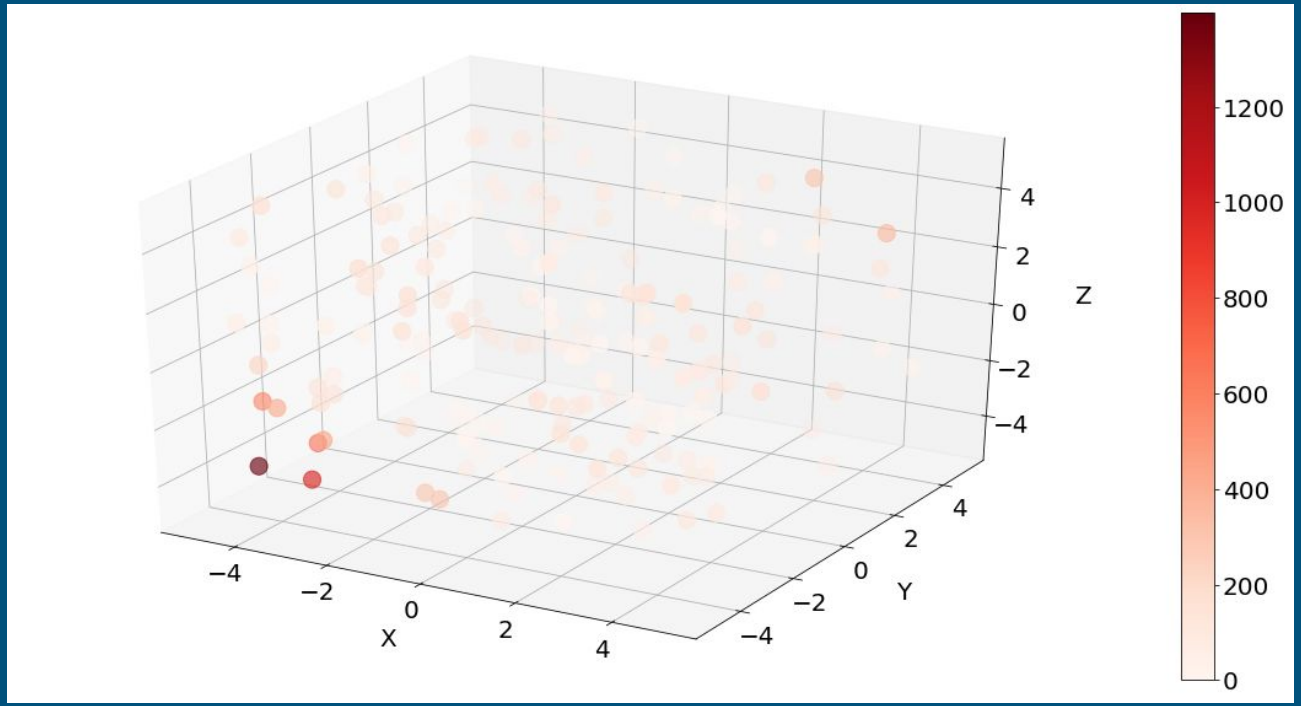
Error perceptrón lineal



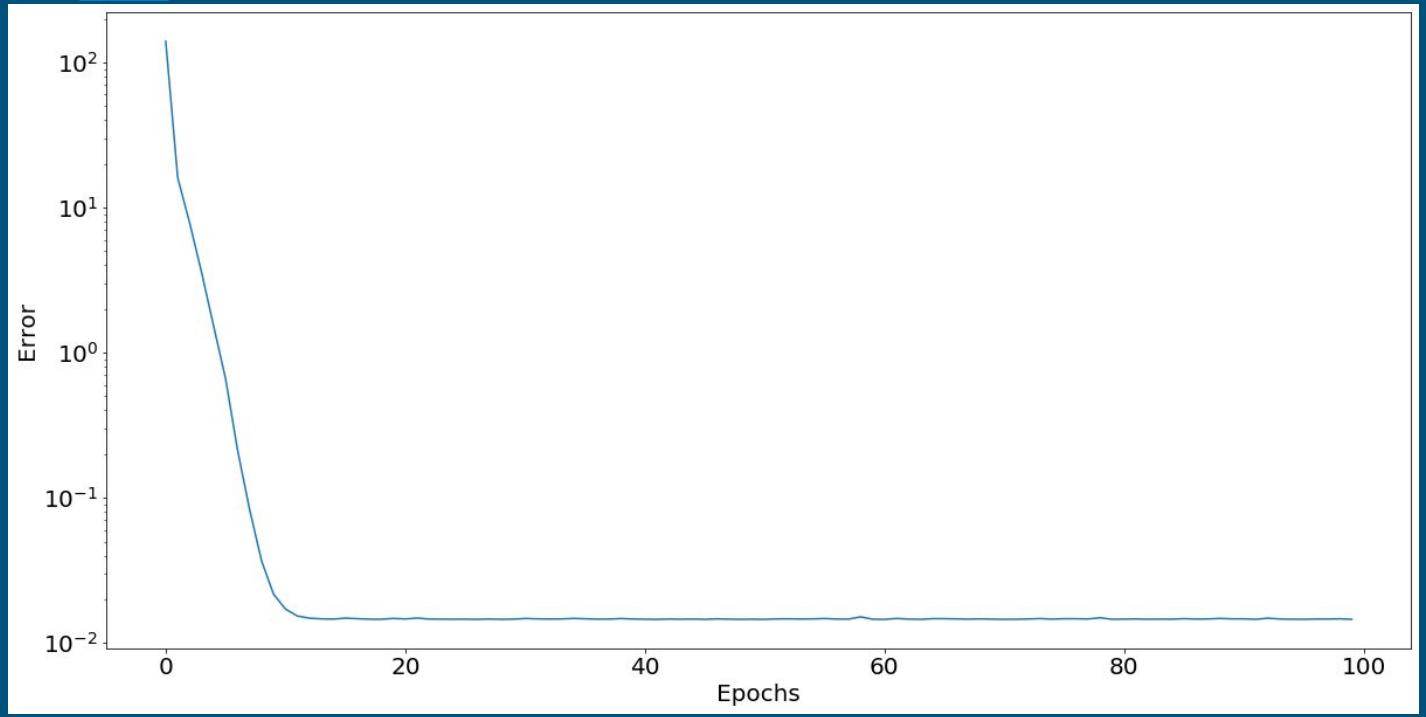
error promedio: 46.44

epochs: 100
learning rate: 0.01
momentum: 0

Perceptrón lineal Error por Entrada



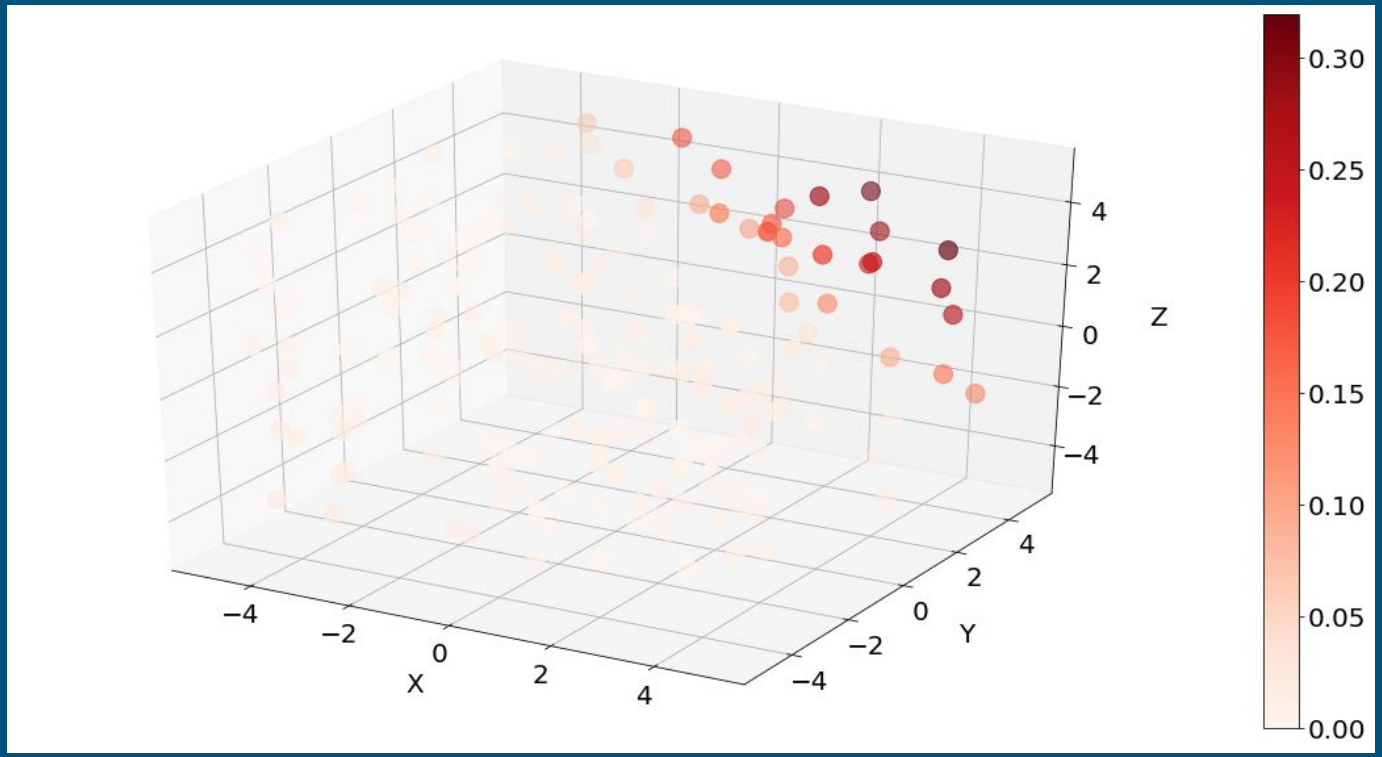
Error perceptrón no lineal



Error promedio: 0.015

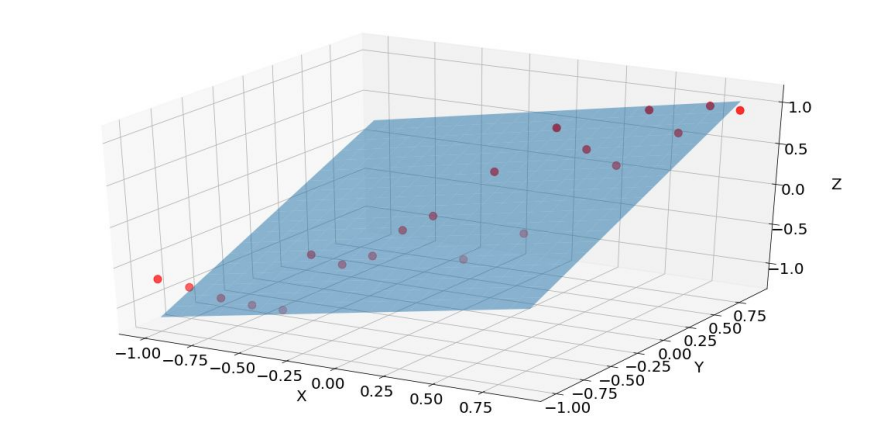
epochs: 100
learning rate: 0.01
momentum: 0

Perceptrón no lineal Error por Entrada

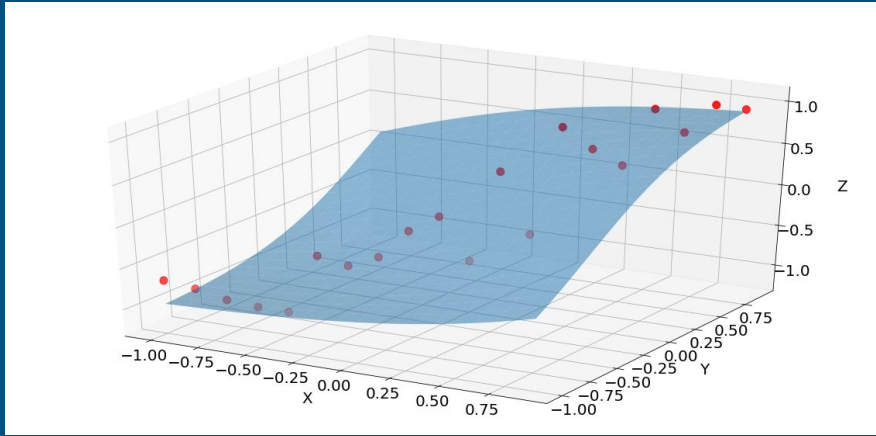


Lineal vs No Lineal

Error 1.04



Error 0.82

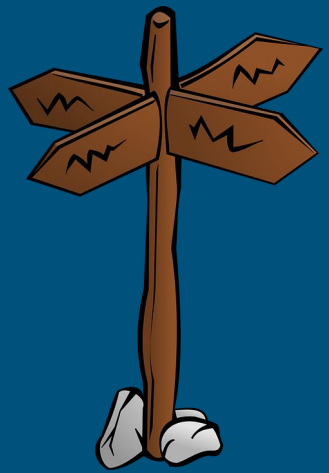


Conclusiones Lineal vs No Lineal

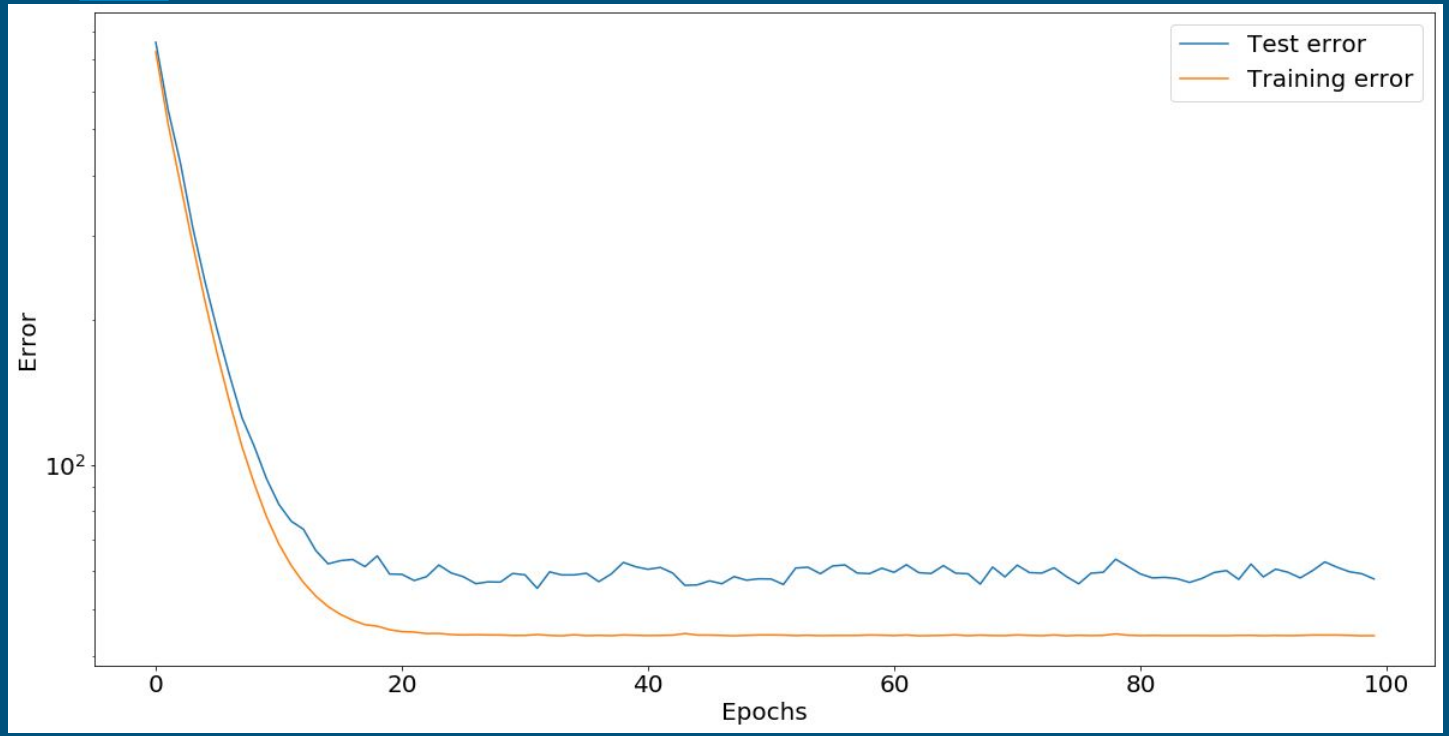
- Se puede ver que el dataset se adapta de una mejor manera a una transformación no lineal

Generalización y mejor conjunto de prueba

- Validación cruzada para probar la generalización
- Para elegir el mejor conjunto de prueba, al hacer la validación cruzada nos quedamos con el que mejor desempeño tiene en accuracy

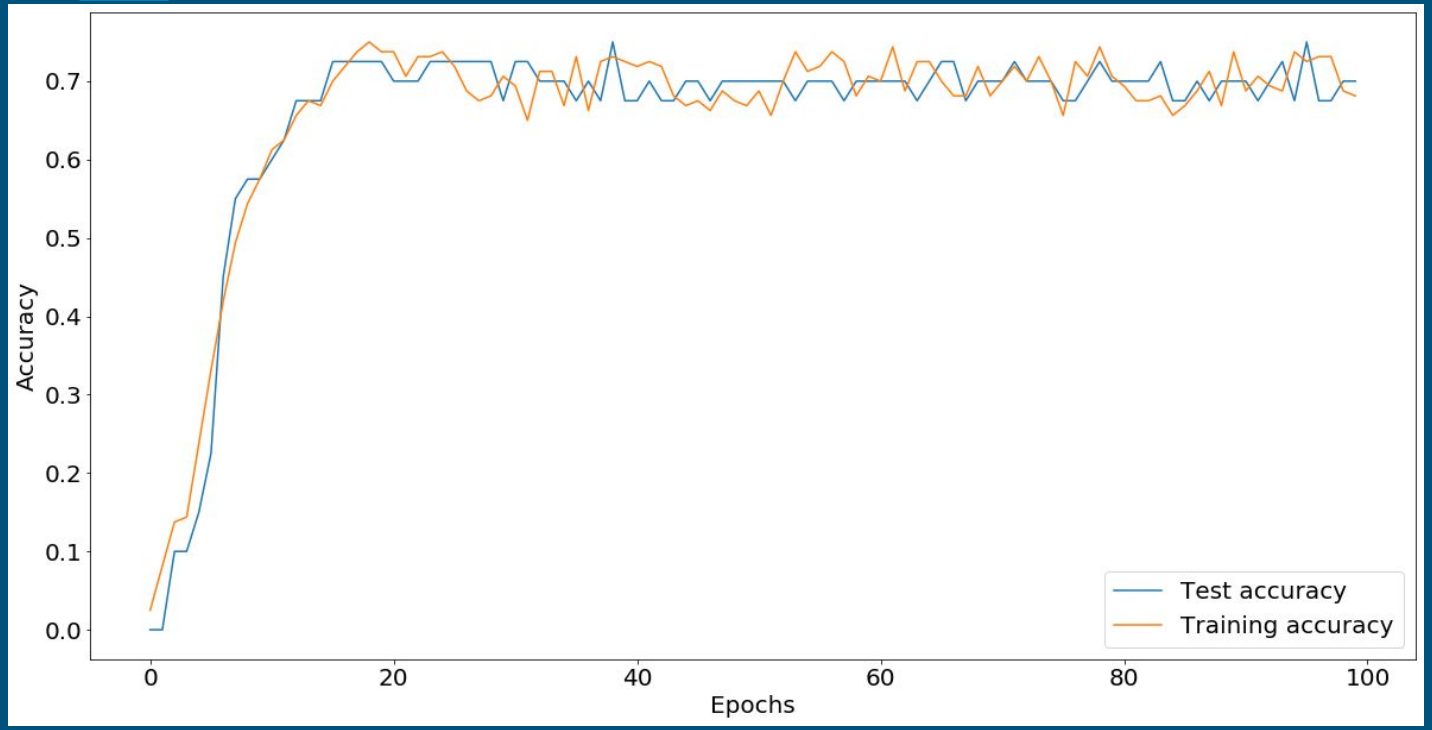


Generalización, error (Lineal)



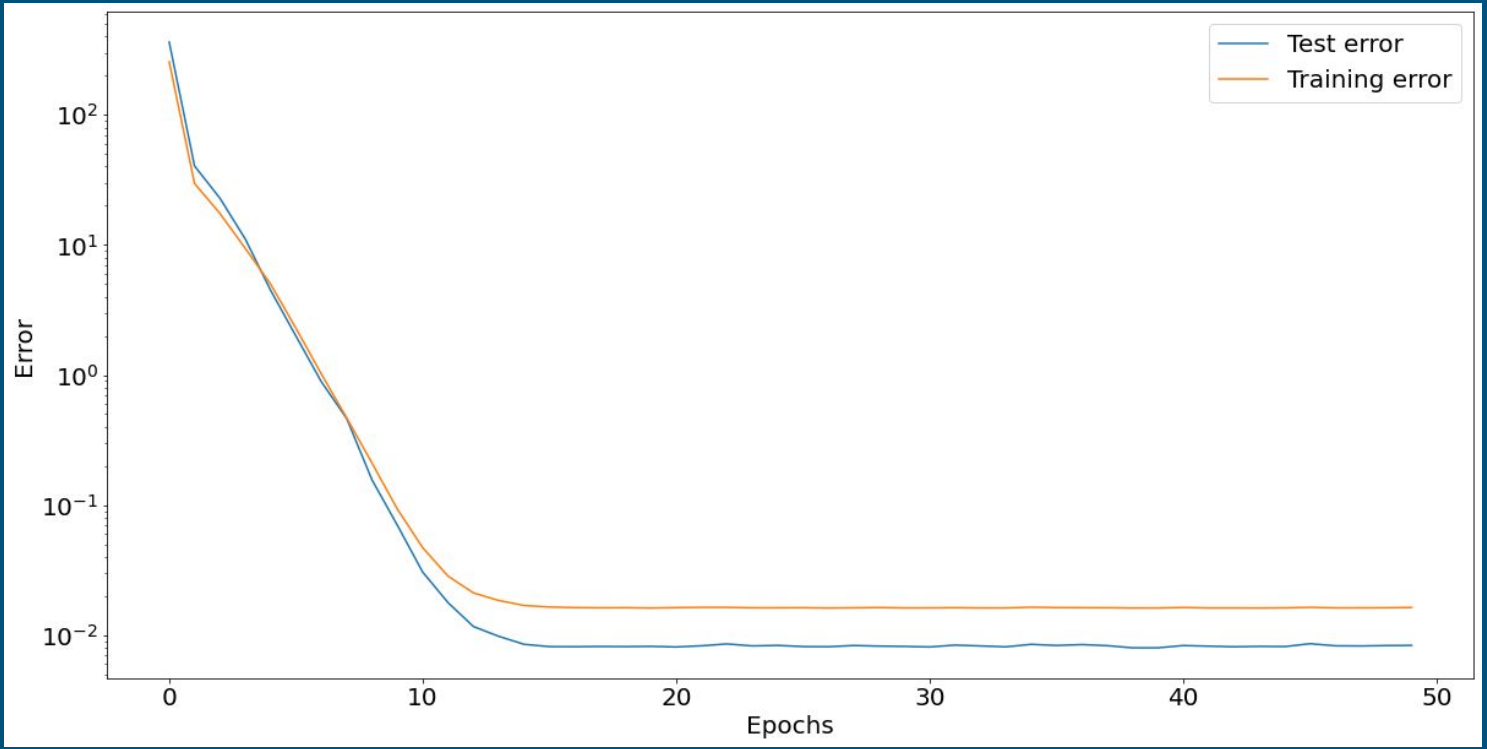
epochs: 100
learning rate: 0.001
b: 0.8
momentum: 0
K: 5

Generalización, accuracy (Lineal)



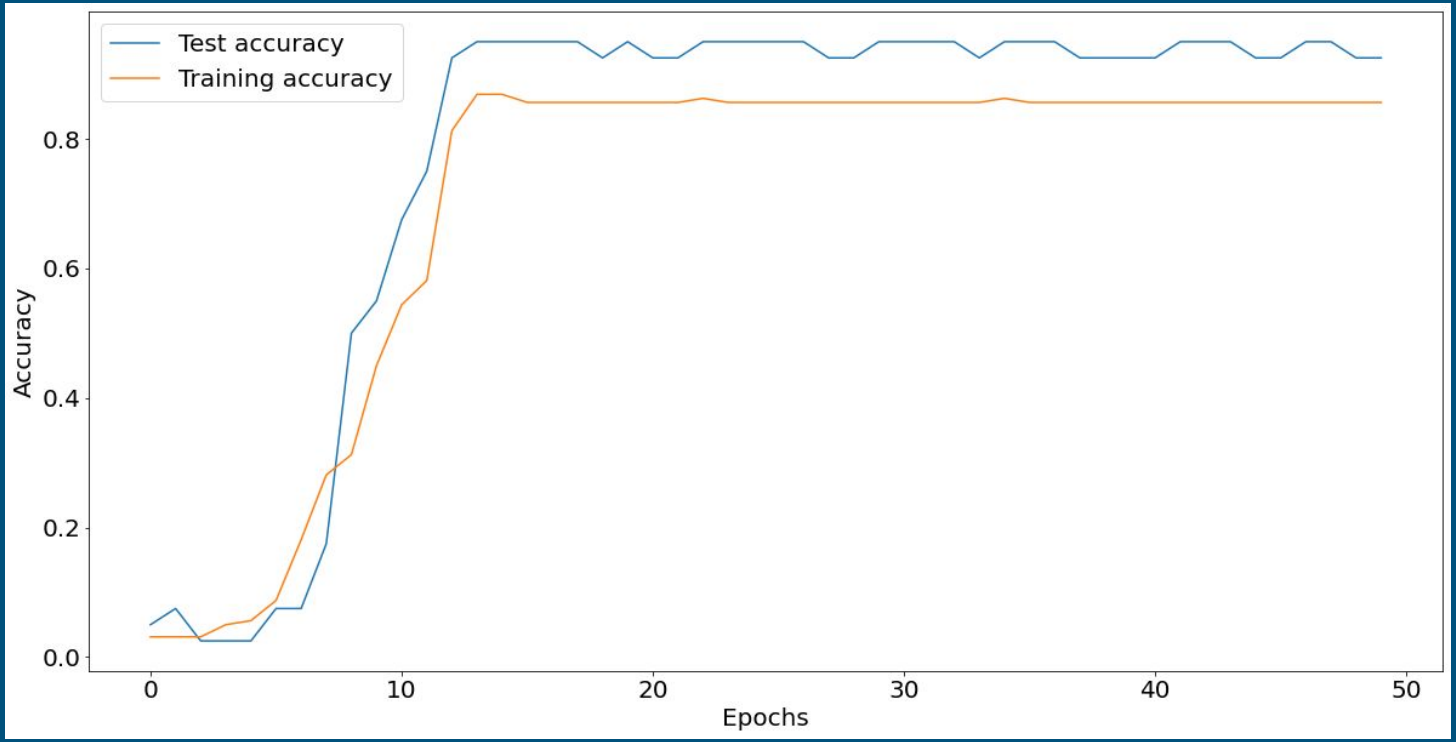
epochs: 100
learning rate: 0.001
b: 0.8
momentum: 0
K: 5
epsilon: 10

Generalización, error (No lineal)



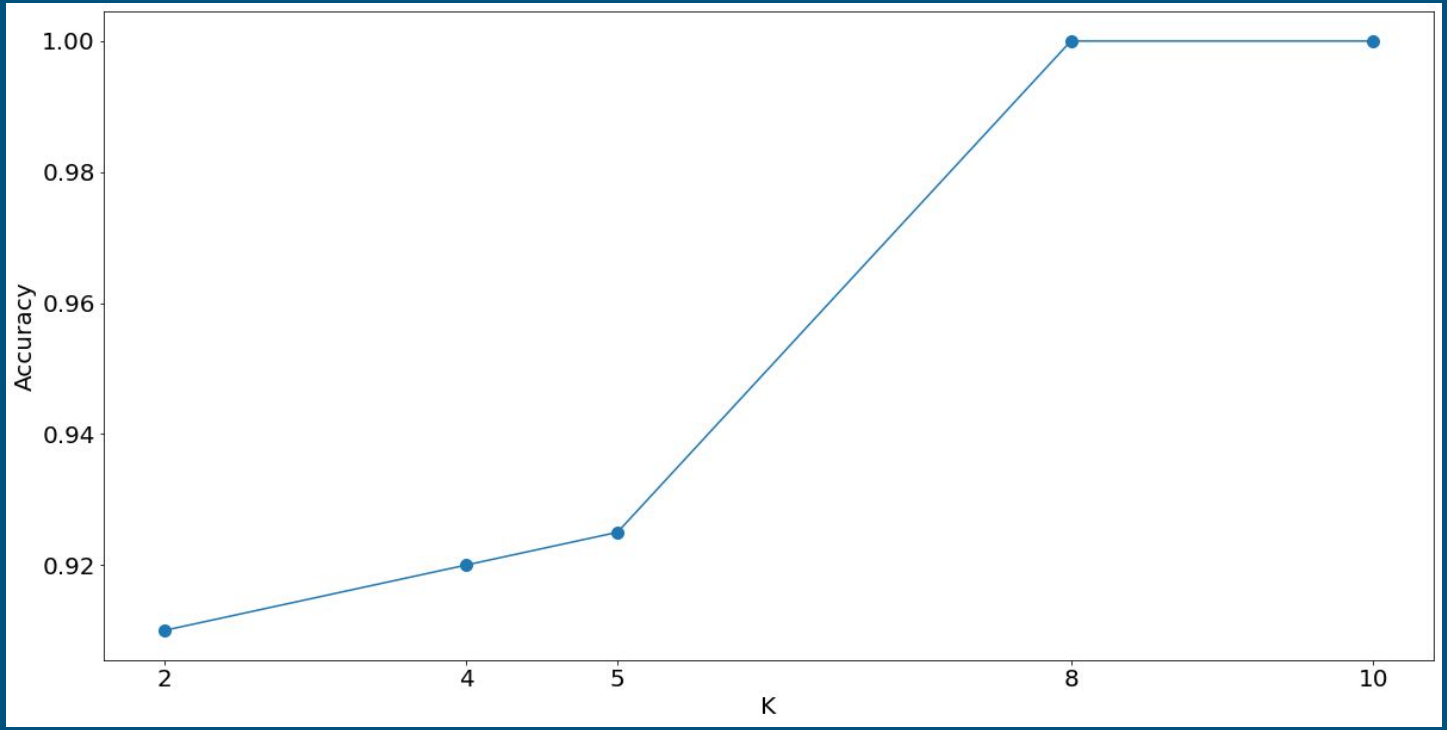
epochs: 50
learning rate: 0.01
momentum: 0
K: 5

Generalización, accuracy (No Lineal)



epochs: 50
learning rate: 0.01
momentum: 0
K: 5
epsilon: 0.005

Máxima capacidad de generalización



epochs: 50
learning rate: 0.01
momentum: 0
K: 5
epsilon: 0.005

Conclusiones generalización No Lineal

- El perceptrón no lineal logra generalizar muy bien
  - Esto se debe a que el dataset debe seguir una transformación no lineal
  - Por ende el perceptrón al entrenar con un conjunto de entrenamiento puede predecir bien el resultado para el conjunto de testeo
- Al aproximarse a una solución lineal, es complicado caer en un sobreajuste
- El conjunto de testeo puede llegar a ser un subconjunto de la aproximación

Problema 3

Perceptrón multicapa

Función lógica 'O exclusivo' con entradas
$x = \{\{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\}\}$,
y salida esperada
$y = \{1, 1, -1, -1\}$.

Número par o impar, por imágenes de 5*7 px

Entrene con un subconjunto de los dígitos y utilice el resto para testear a la red.

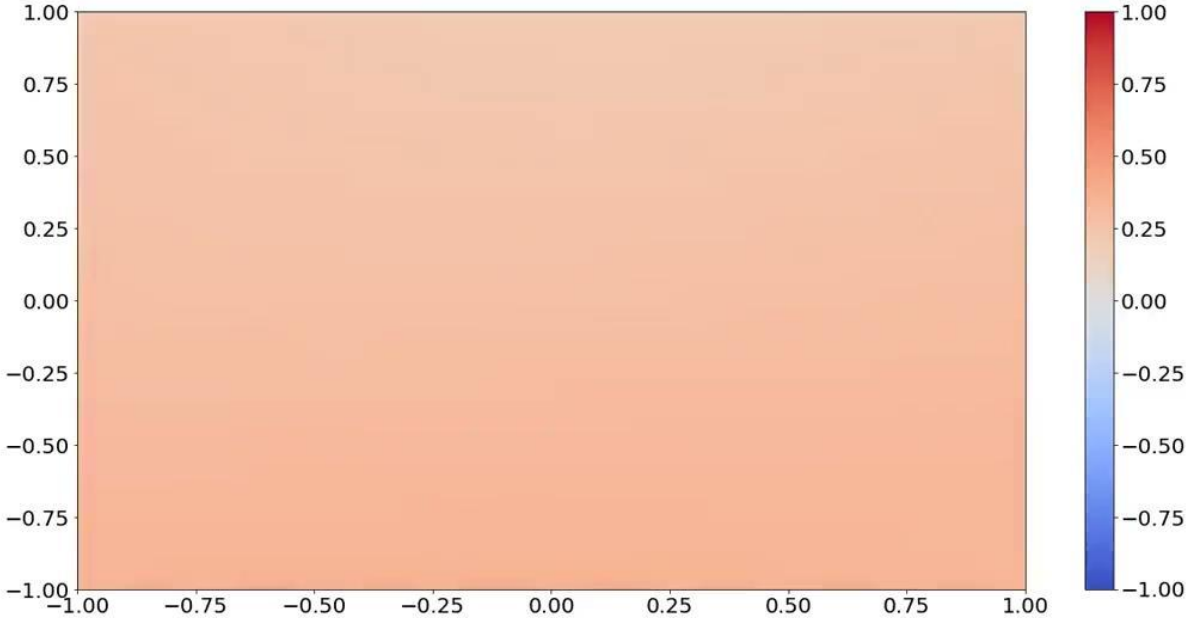
¿Qué podría decir acerca de la capacidad para generalizar de la red?

Distinguir cada número del 0-9, por imágenes de 5*7 px.

Después de que aprenda use patrones de entrada correspondientes a los dígitos de entrenamiento pero con sus píxeles afectados por ruido

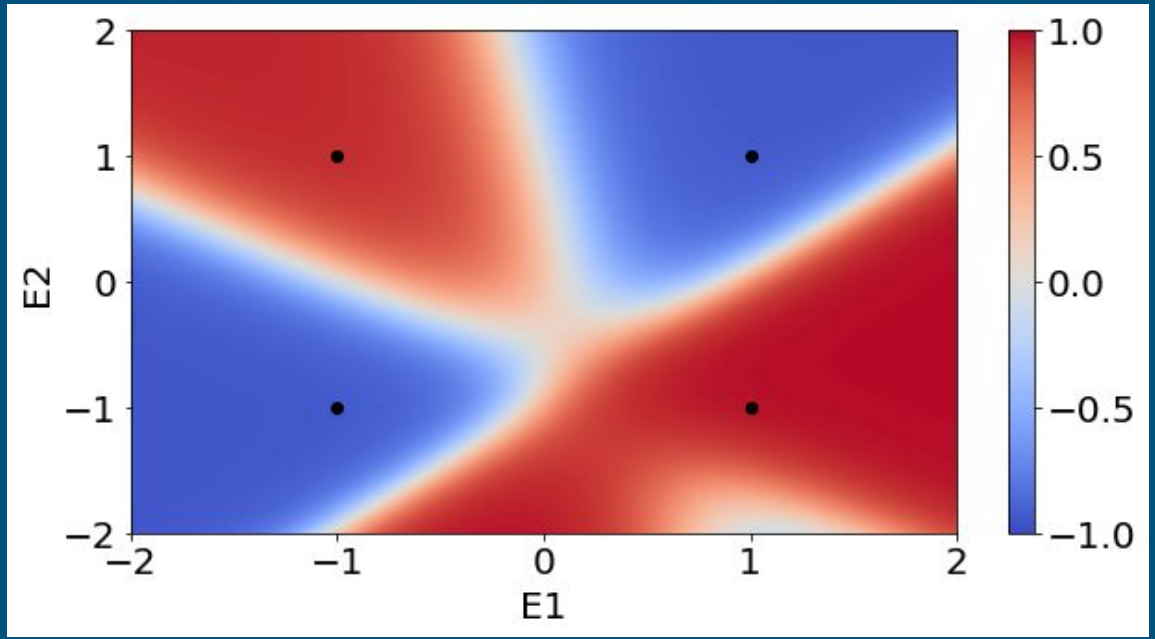
Resultados 3 a

Epochs: 44
Error: 9.63e-03
Structure: [2, 2, 1]



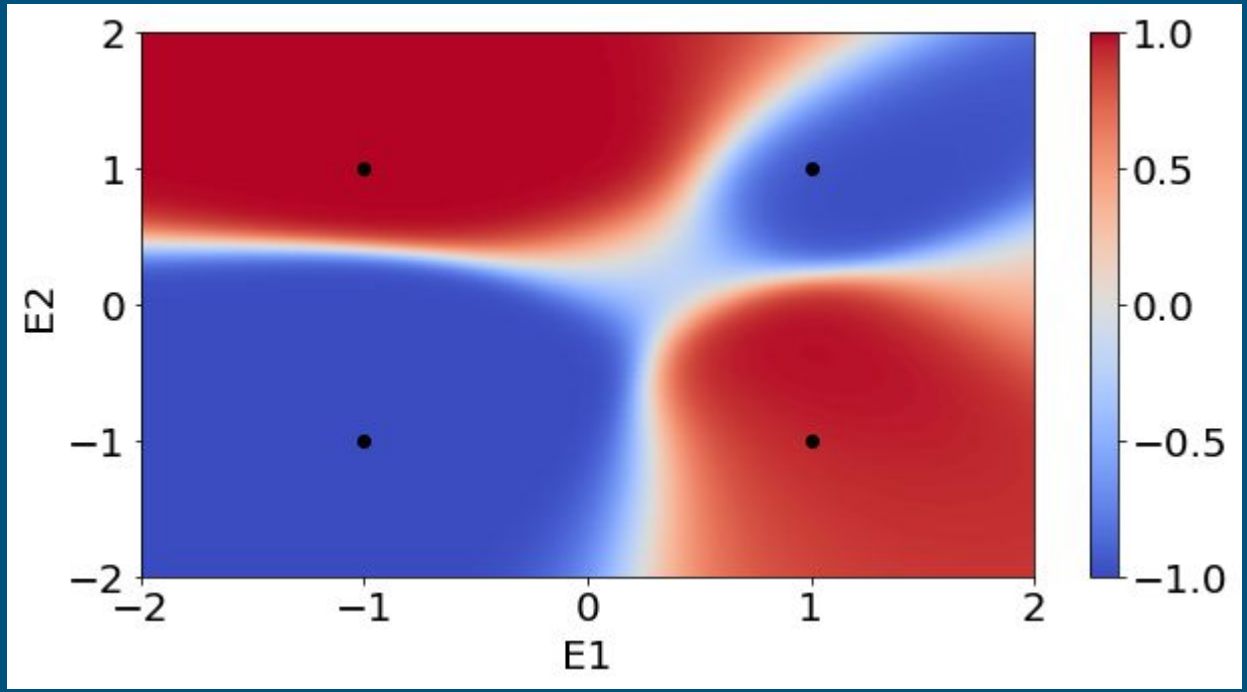
Resultados 3 a

Epochs: 32
Error: 9.73e-03
Structure: [2, 3, 1]



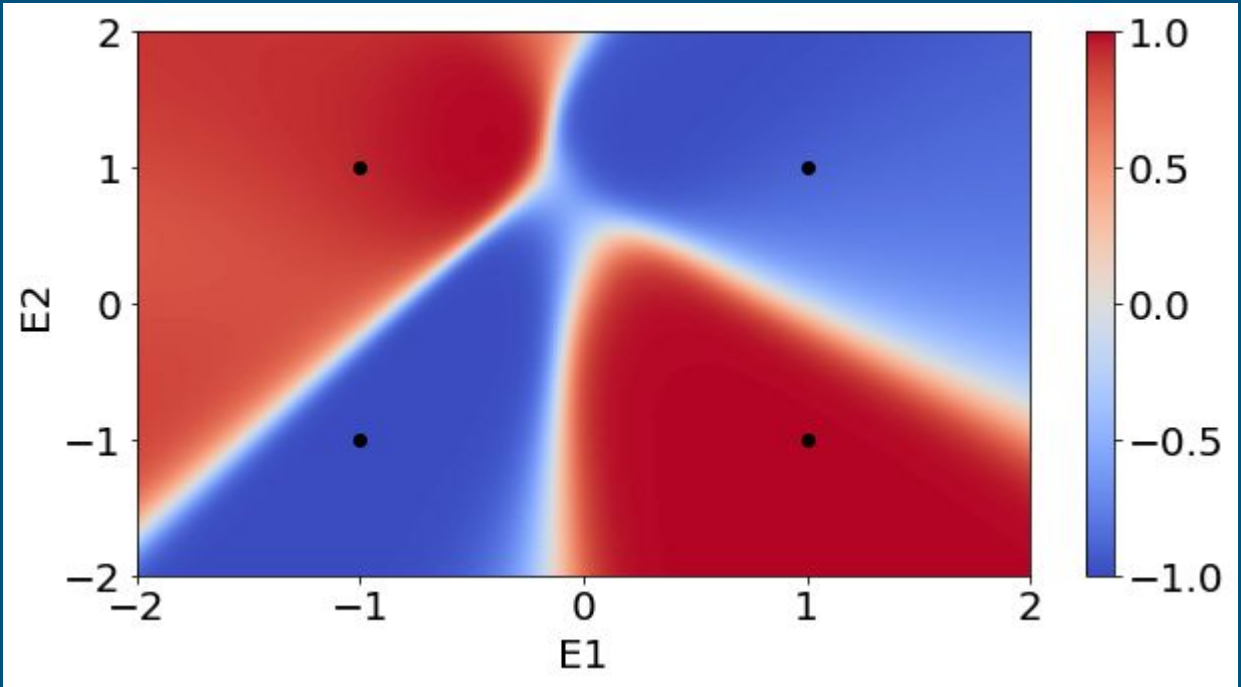
Resultados 3 a

Epochs: 65
Error: 3.60e-03
Structure: [2, 15, 10, 1]



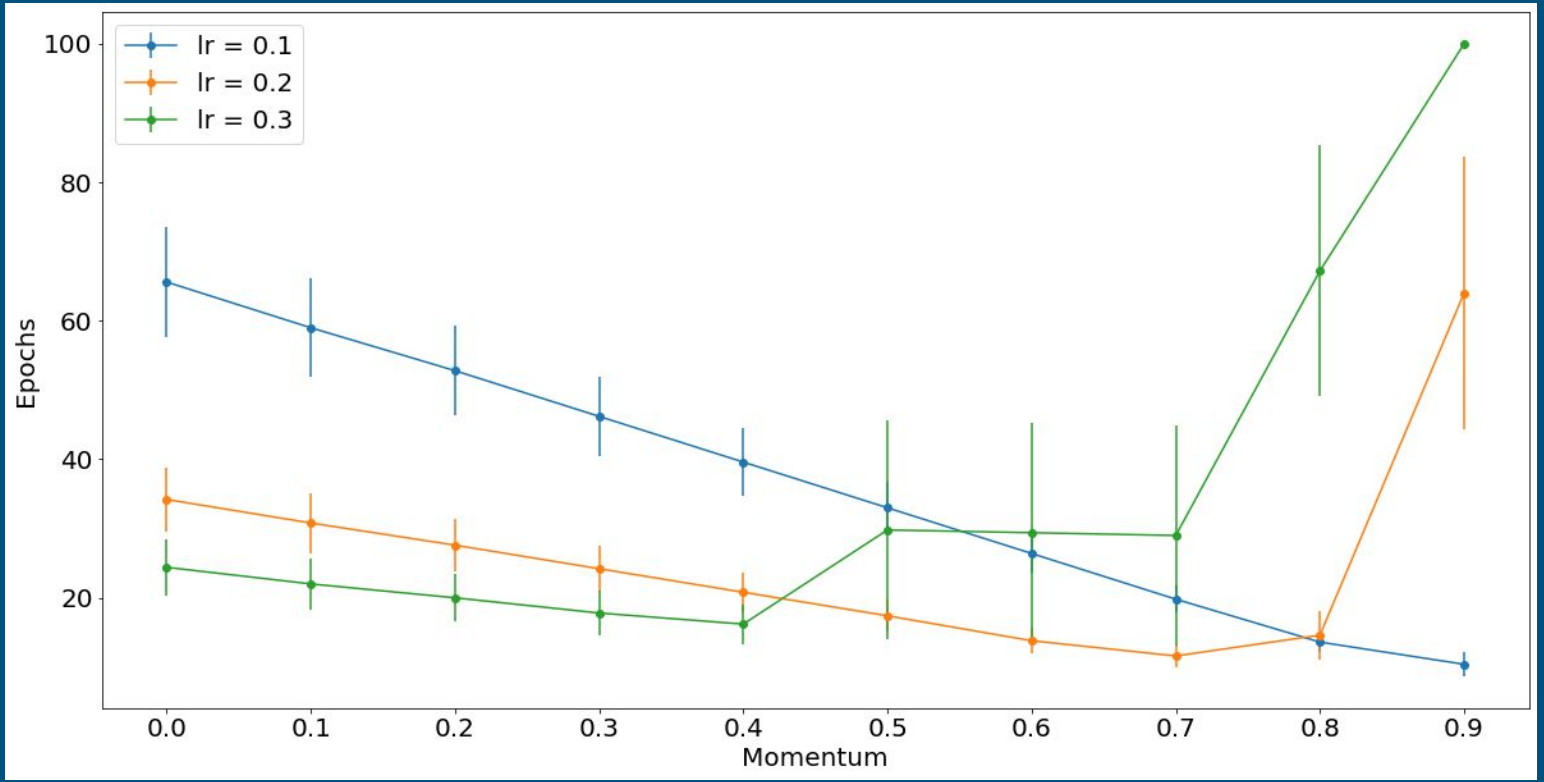
Resultados 3 a

Epochs: 18
Error: 8.23e-03
Structure: [2, 10, 15, 1]



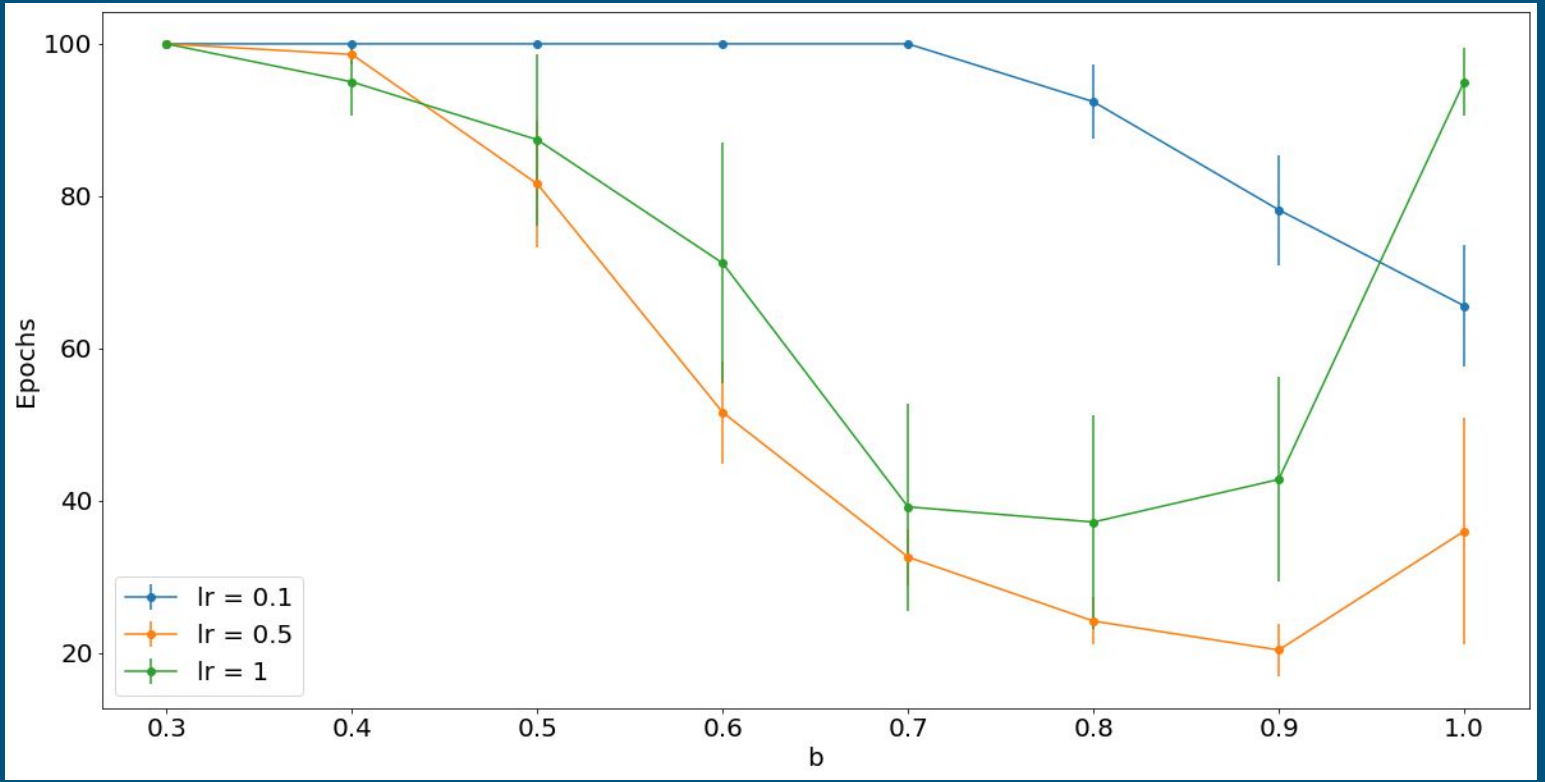
Resultados 3 a

target error: 0.1

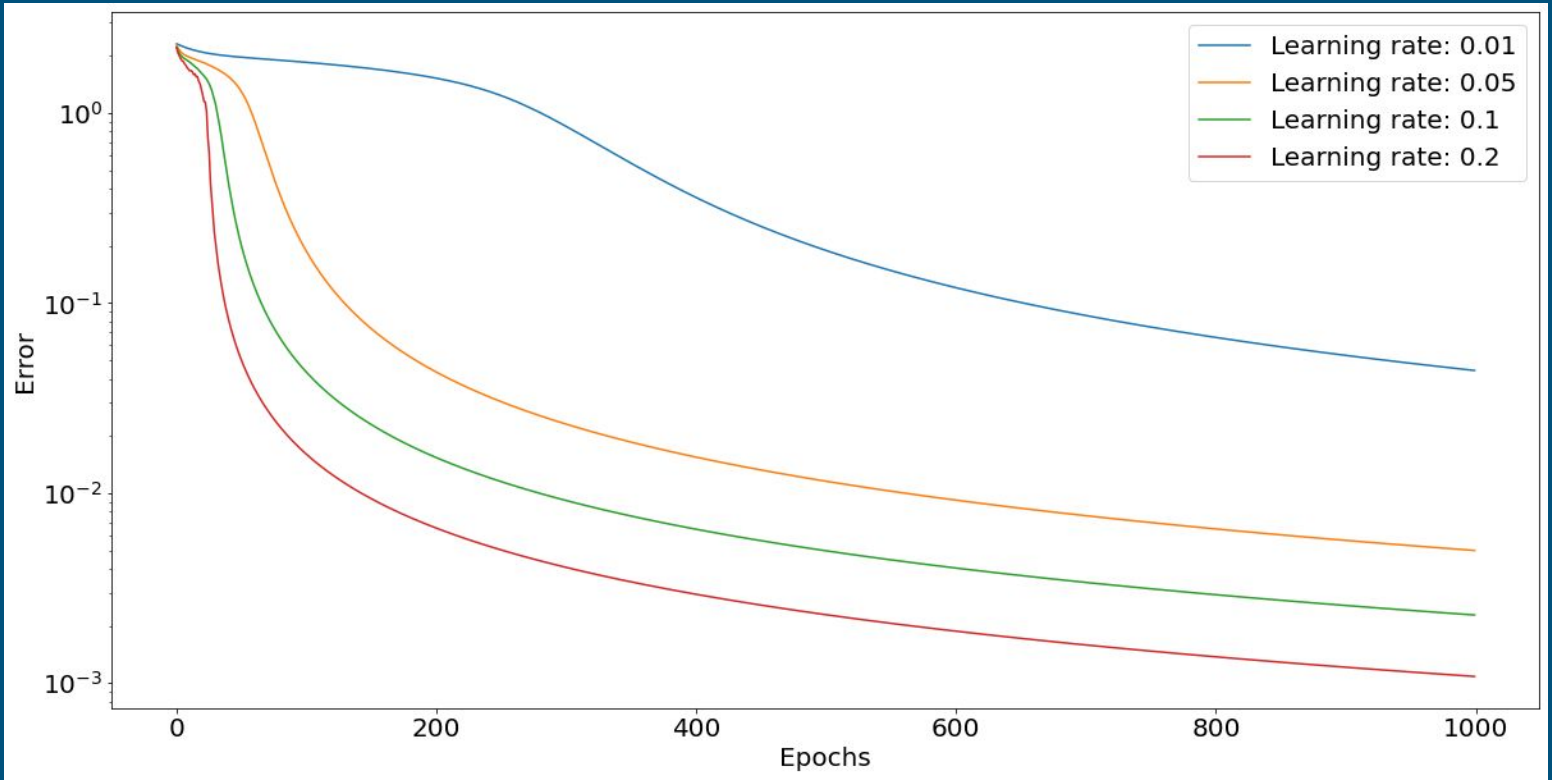


Resultados 3 a

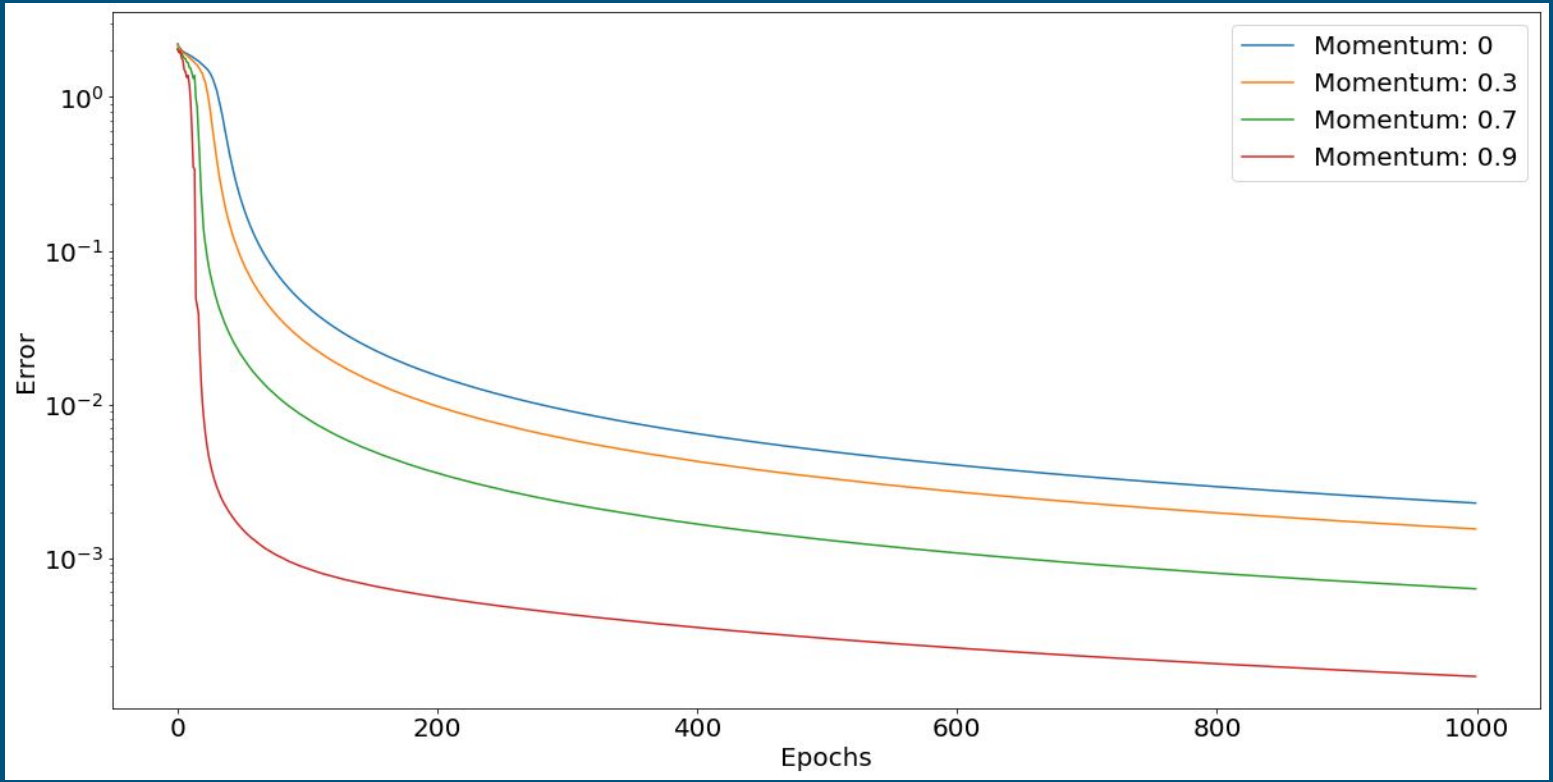
target error: 0.1



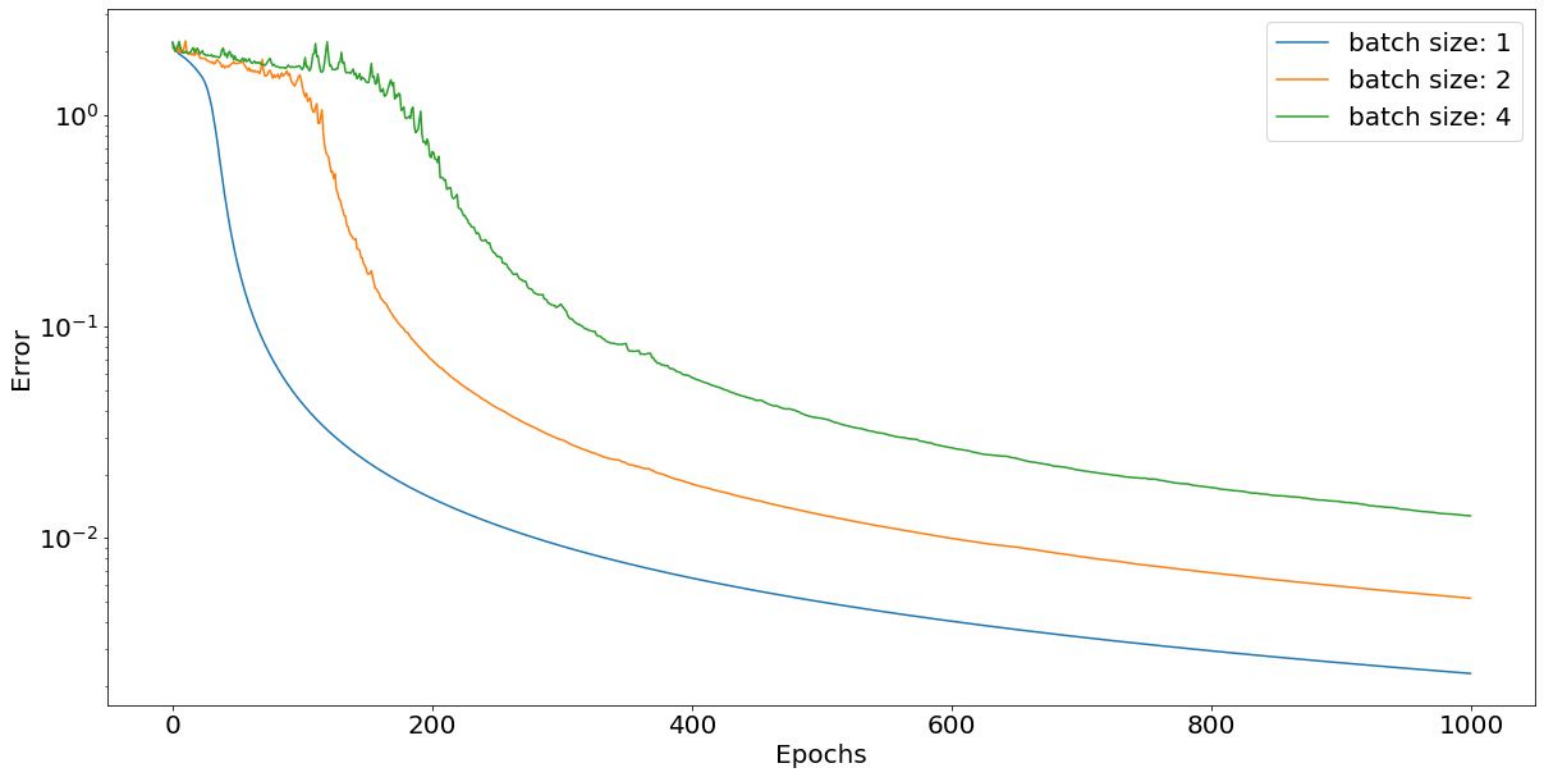
Resultados 3 a



Resultados 3 a

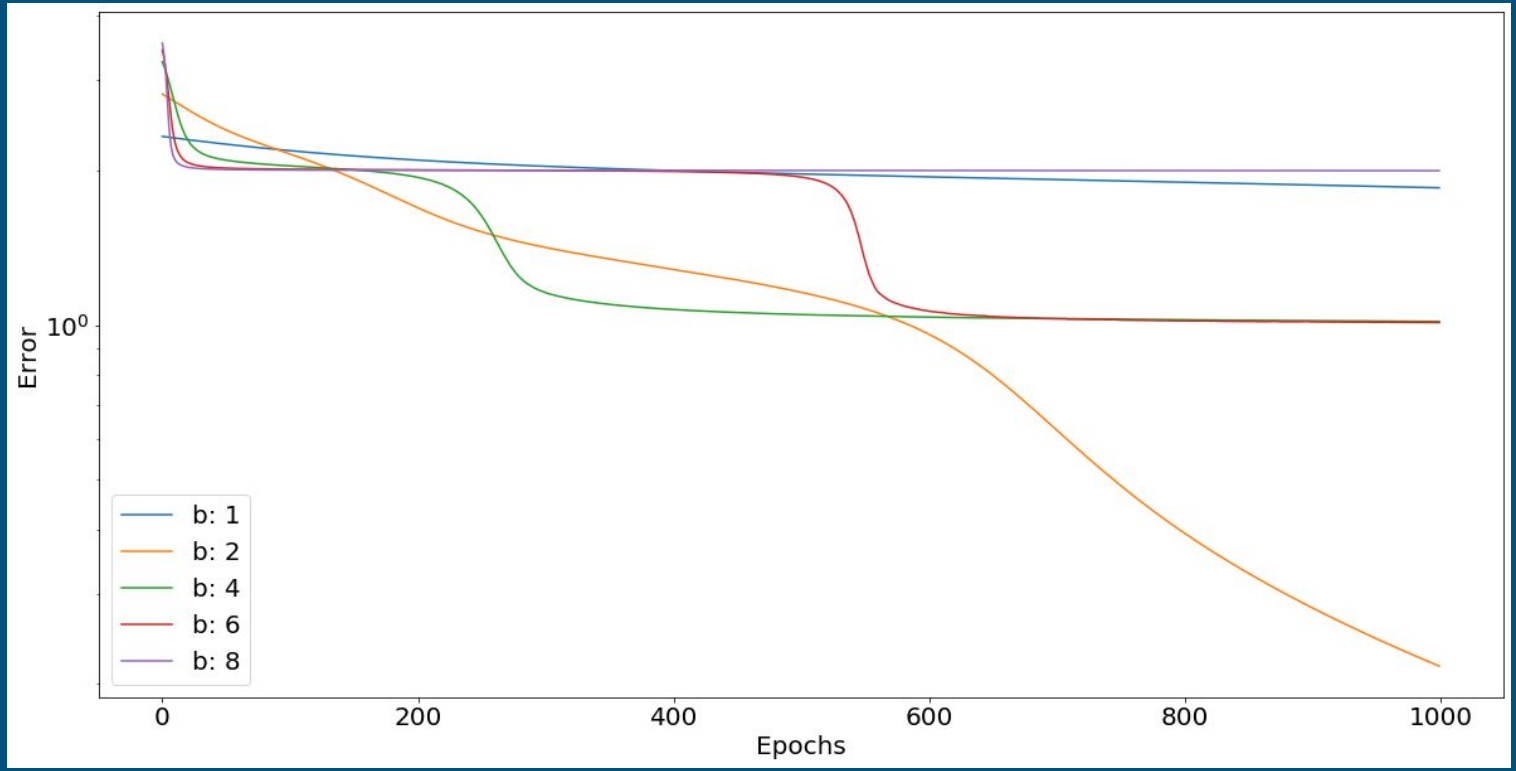


Resultados 3 a

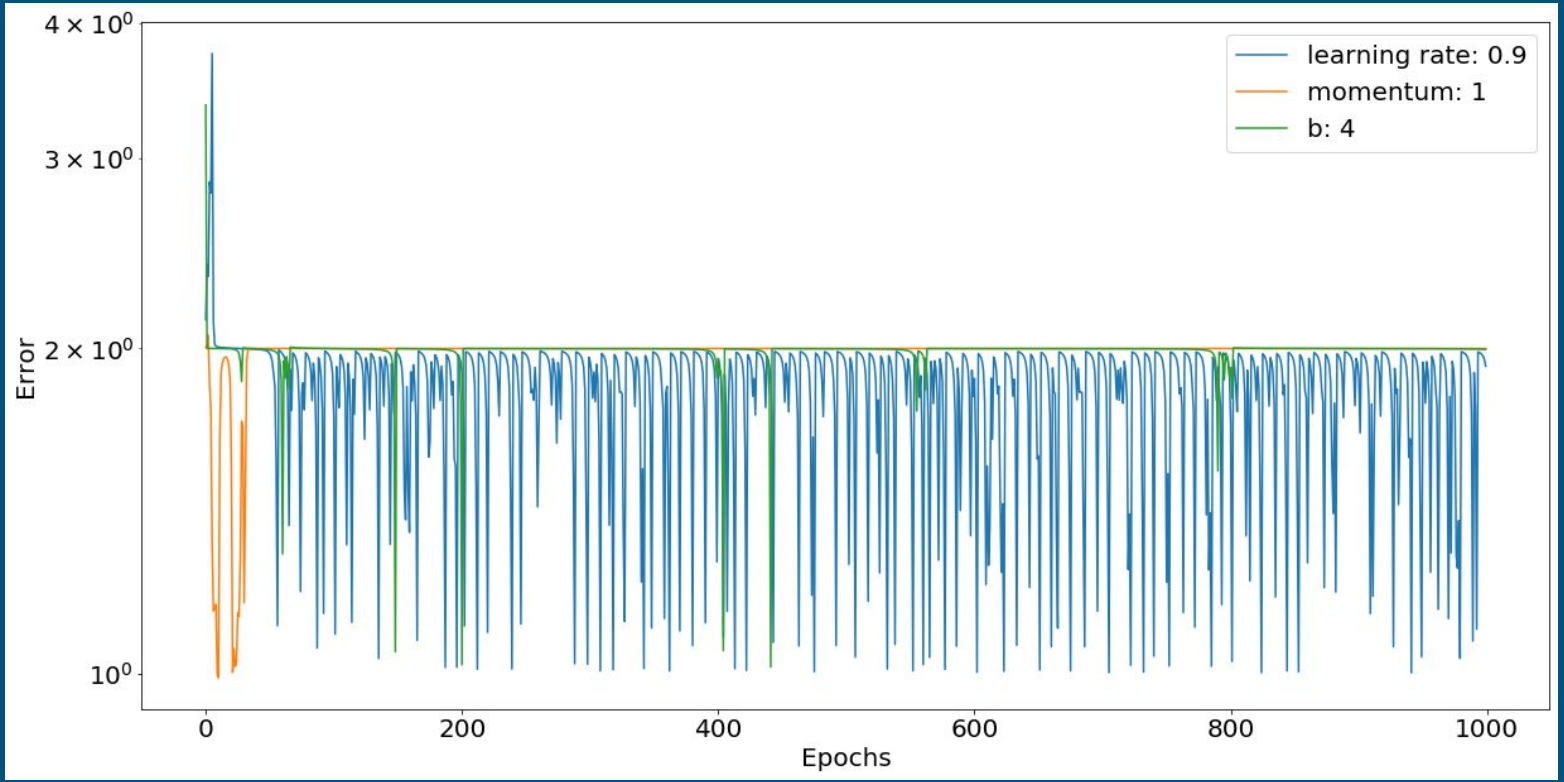


Resultados 3 a

Learning rate: 0.001

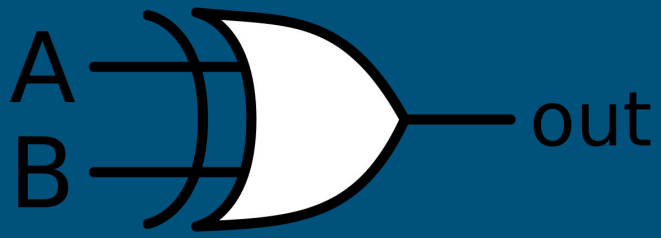


Resultados 3 a



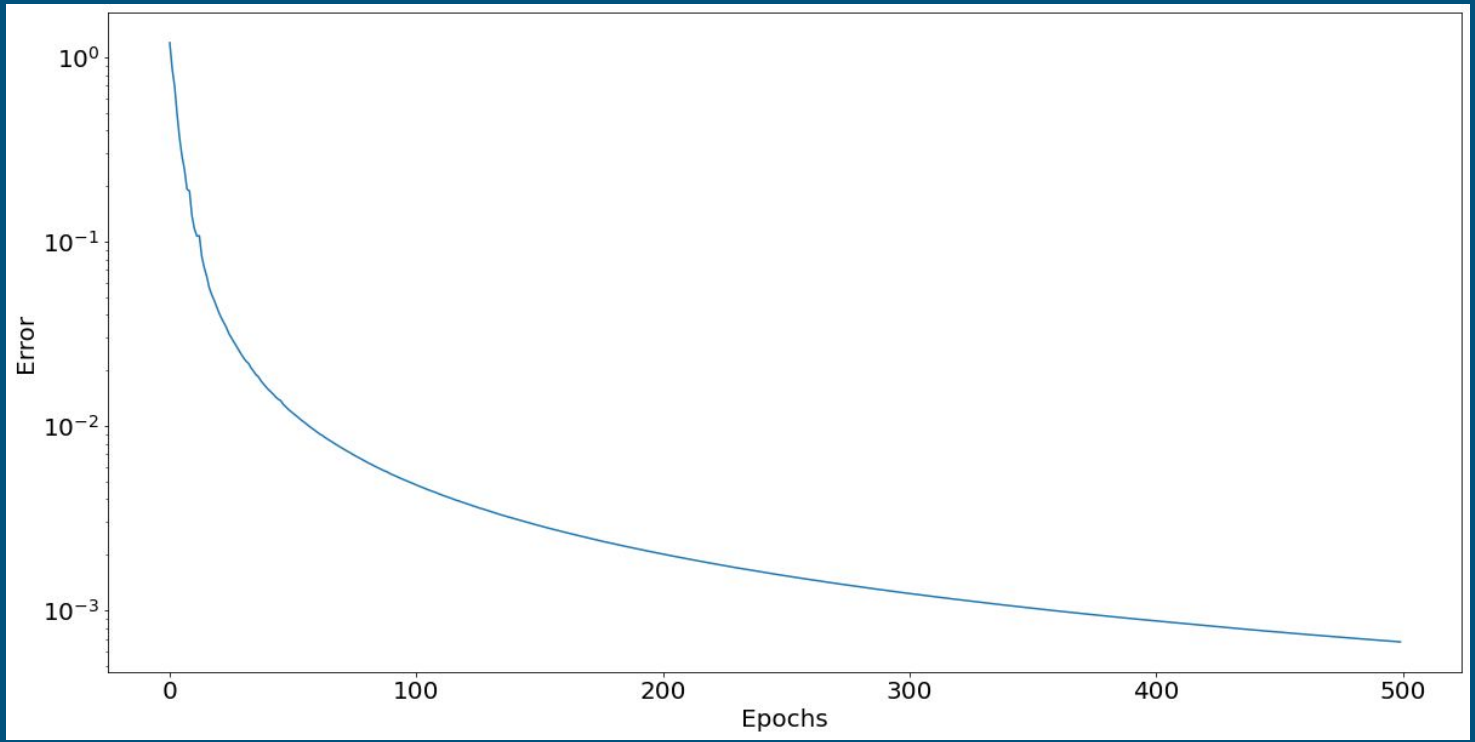
Conclusiones 3 a

- learning grandes, b grandes y momentum grande hace mas rapido la convergencia pero puede pasarse de largo
- EL xor se puede resolver fácilmente con multicapa

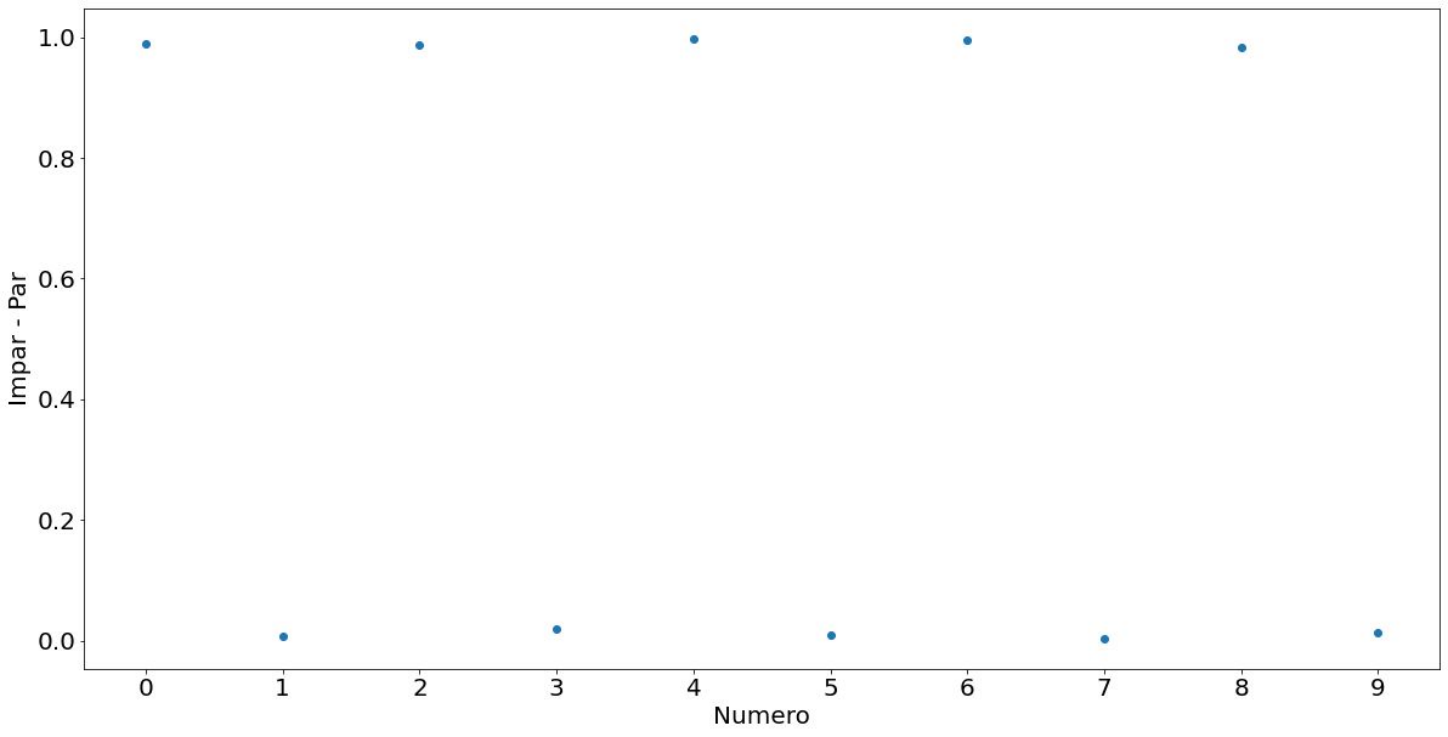


Resultados 3 b

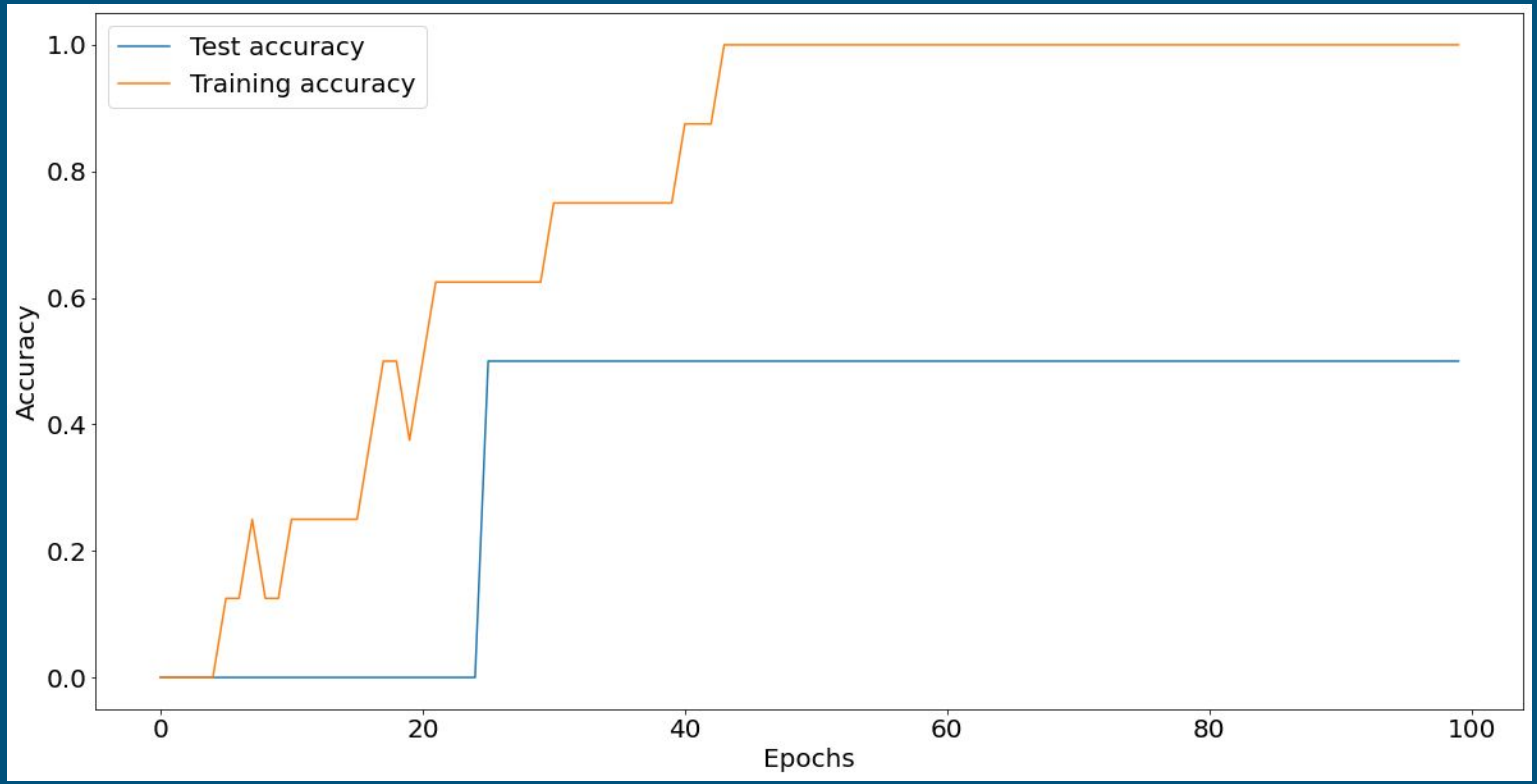
Epochs: 500
Error: 6.73e-04
[35, 20, 10]



Resultados 3 b



Resultados 3 b - Accuracy con cross validation

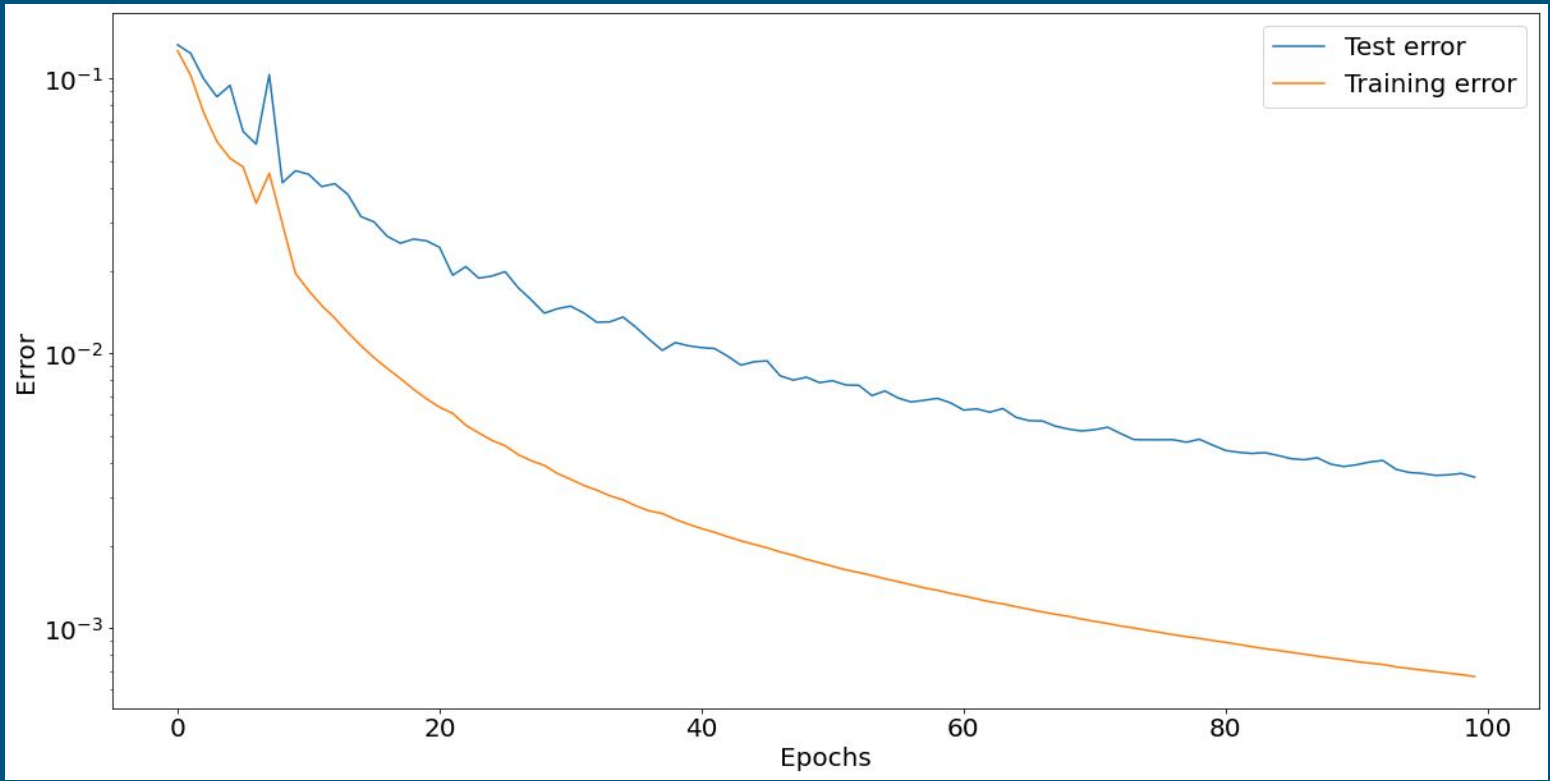


Precision test set: 0.5

Precision for training set: 1.0

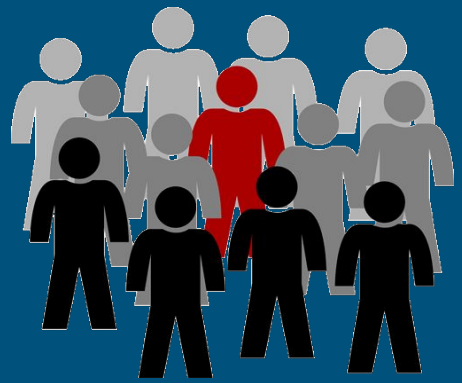
k: 5

Resultados 3 b - Accuracy con cross validation

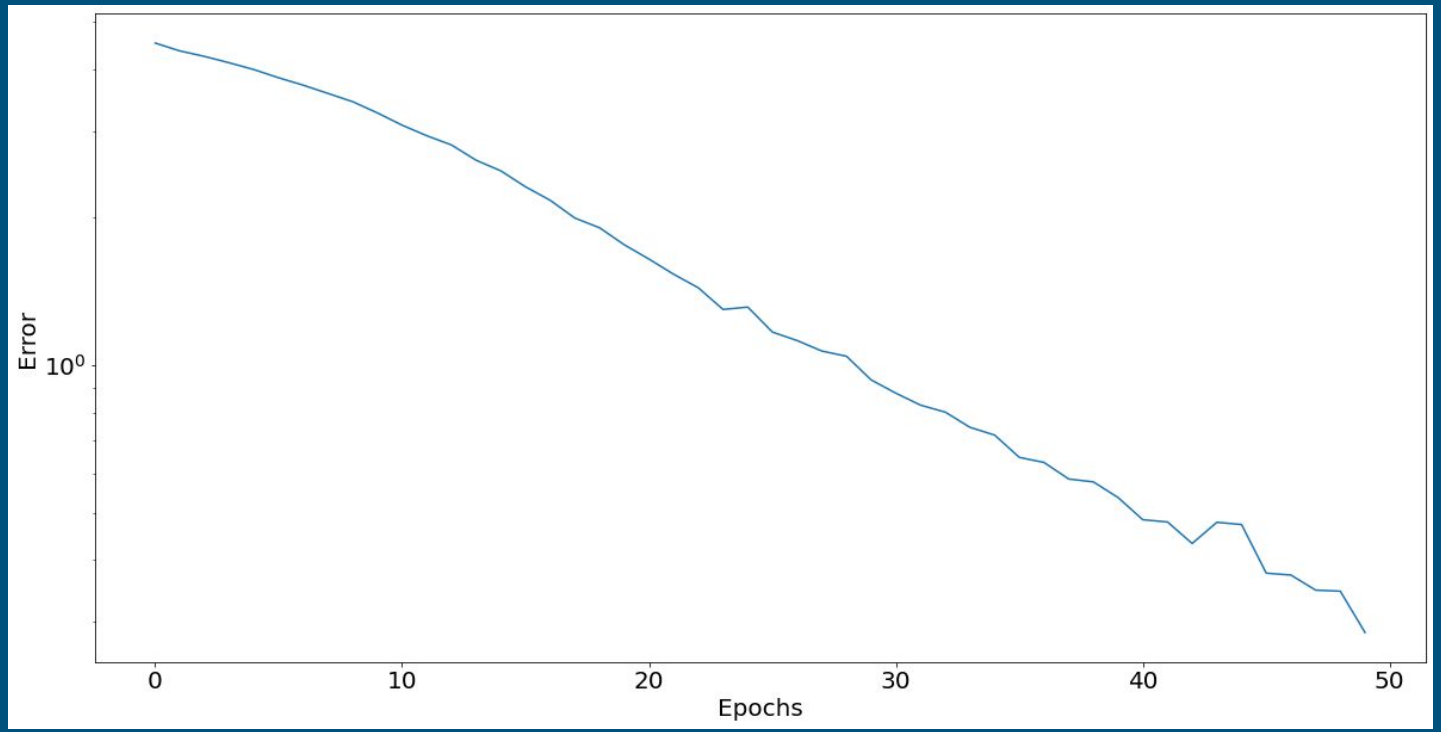


Conclusiones 3 b

- Queda claro que al no tener seguir una ecuación fácilmente definida (como en el ejercicio 2), la capacidad de generalización es pobre
- Generalizar sin el conjunto completo de elementos no sirve ya que el testeo va a tener números muy diferentes al entrenamiento y no podrá generalizar

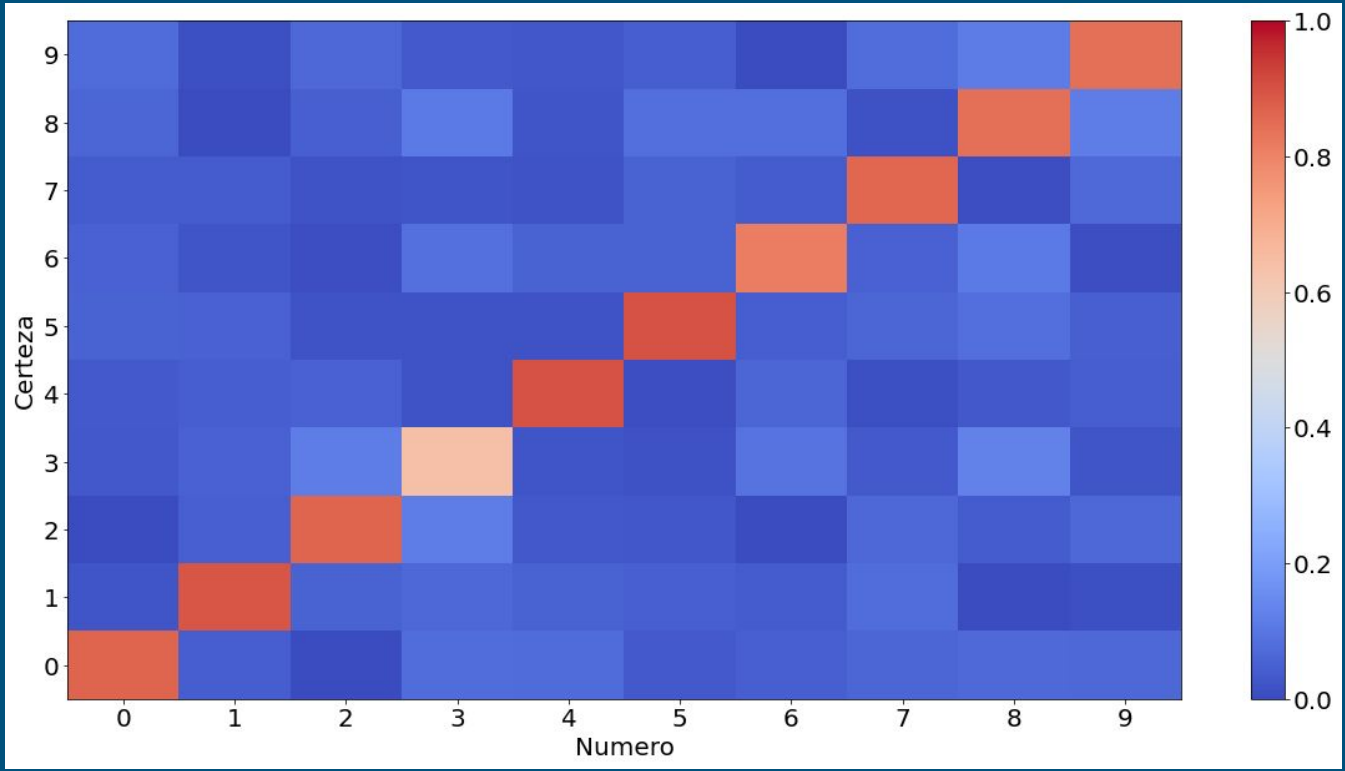


Resultados 3 c sin ruido



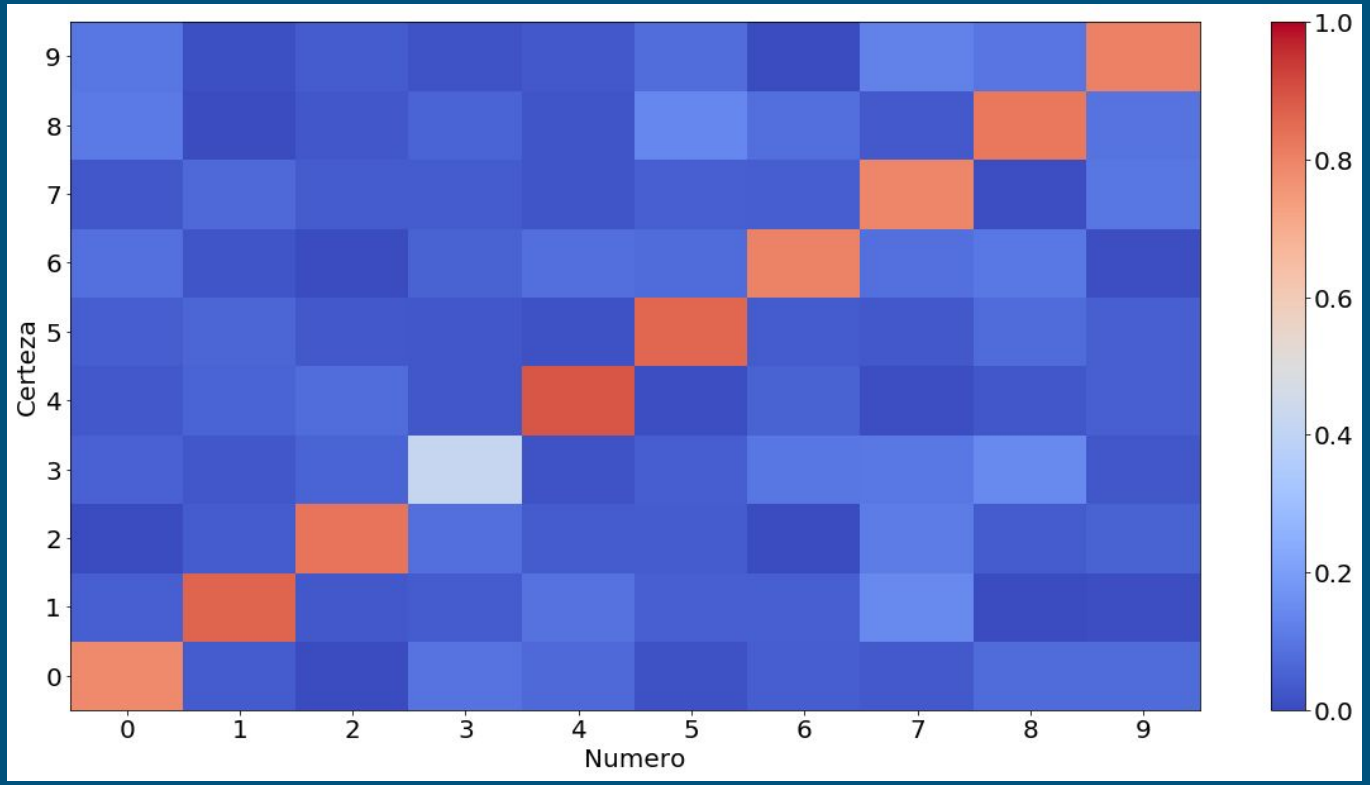
[35, 30, 20, 10]
learning rate: 1
epochs: 50
momentum: 0

Resultados 3 c sin ruido



Error: 0.28472

Resultados 3 c - Noise



Error: 0.46838
Noise: 0.02

Resultados 3 c - Noise

Accuracy: 0.9968

	0	1	2	3	4	5	6	7	8	9	R
0	1000	0	0	0	0	0	0	0	0	0	100
1	0	1000	0	0	0	0	0	0	0	0	100
2	0	0	1000	0	0	0	0	0	0	0	100
3	0	0	5	973	0	0	2	0	19	1	97
4	0	0	0	0	1000	0	0	0	0	0	100
5	0	0	0	0	0	1000	0	0	0	0	100
6	0	0	0	0	0	0	999	0	1	0	99
7	0	0	0	0	0	0	0	1000	0	0	100
8	2	0	0	2	0	0	0	0	996	0	99
9	0	0	0	0	0	0	0	0	0	1000	100
P	99	100	99	99	100	100	99	100	98	99	0

Conclusiones 3 c

- El ruido no afecta mucho al error
- El entrenamiento con el conjunto de números bien definidos es suficiente para generalizar bien en casos de ruido

Opcional Softmax

Usa el softmax como función de activación para la excitación de la última capa. Se pueden observar las probabilidades de que la imagen pertenezca a tal número.

	0	1	2	3	4	5	6	7	8	9
0	92.9%	0.63%	0.02%	1.17%	1.14%	0.46%	0.7%	0.91%	1.04%	0.99%
1	0.26%	95.68%	0.64%	0.78%	0.61%	0.52%	0.41%	0.93%	0.01%	0.09%
2	0.01%	0.69%	94.03%	1.86%	0.44%	0.4%	0.03%	0.97%	0.54%	0.99%
3	1.26%	2.16%	5.55%	76.18%	0.96%	0.68%	4.46%	1.43%	6.24%	1.03%
4	0.38%	0.46%	0.57%	0.21%	96.71%	0.05%	0.7%	0.11%	0.32%	0.44%
5	0.57%	0.55%	0.2%	0.2%	0.19%	95.73%	0.42%	0.69%	0.89%	0.5%
6	1.09%	0.46%	0.1%	1.91%	1.18%	1.12%	90.41%	1.06%	2.5%	0.13%
7	0.57%	0.57%	0.3%	0.32%	0.27%	0.86%	0.61%	95.2%	0.09%	1.15%
8	1.08%	0.03%	0.77%	2.03%	0.37%	1.49%	1.48%	0.25%	90.26%	2.19%
9	1.36%	0.19%	1.16%	0.57%	0.46%	0.76%	0.03%	1.37%	2.12%	91.92%


THANKS!