PERCEPTRON SIMPLE Y MULTICAPA

Objetivos

Implementar sistemas de perceptrón simple y multicapa para distintos problemas planteados, evaluando cómo funcionan como solución, cuáles son sus ventajas y sus limitaciones.

Ejercicio 1

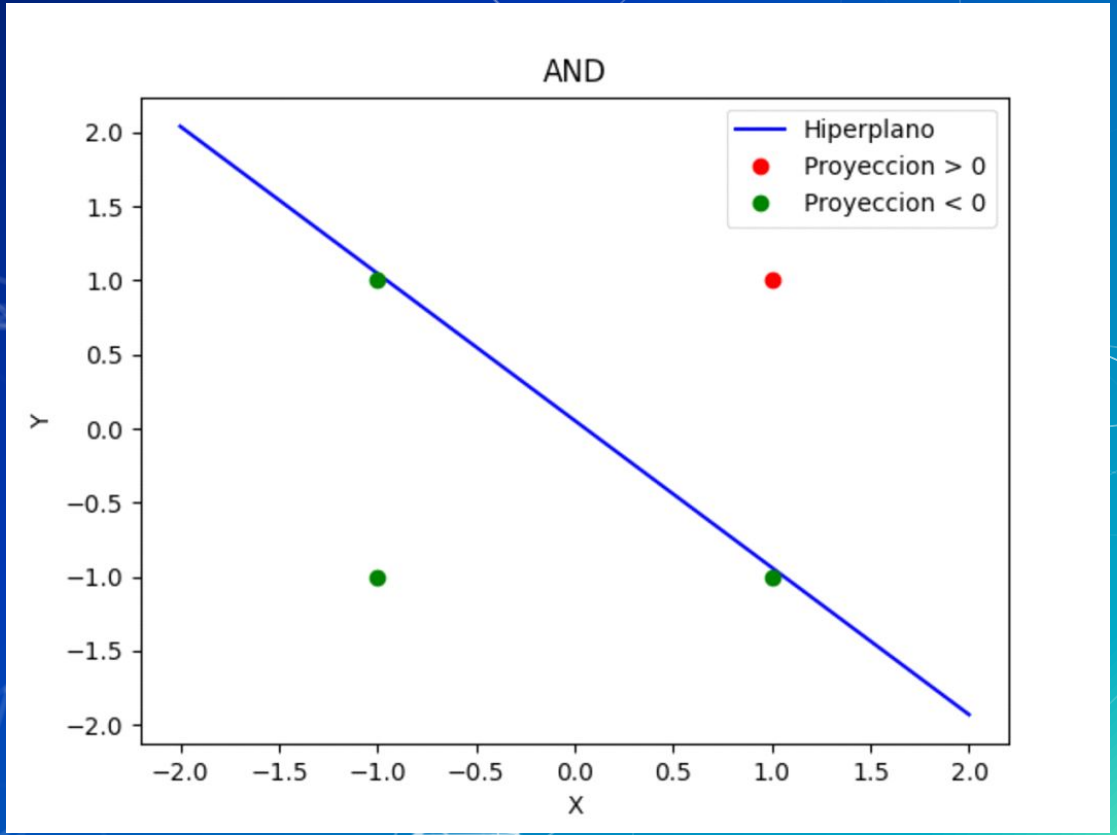
Perceptrón simple con función de activación de escalón

Funciones a implementar

- Operación lógica “y”
- Operación lógica “xor”

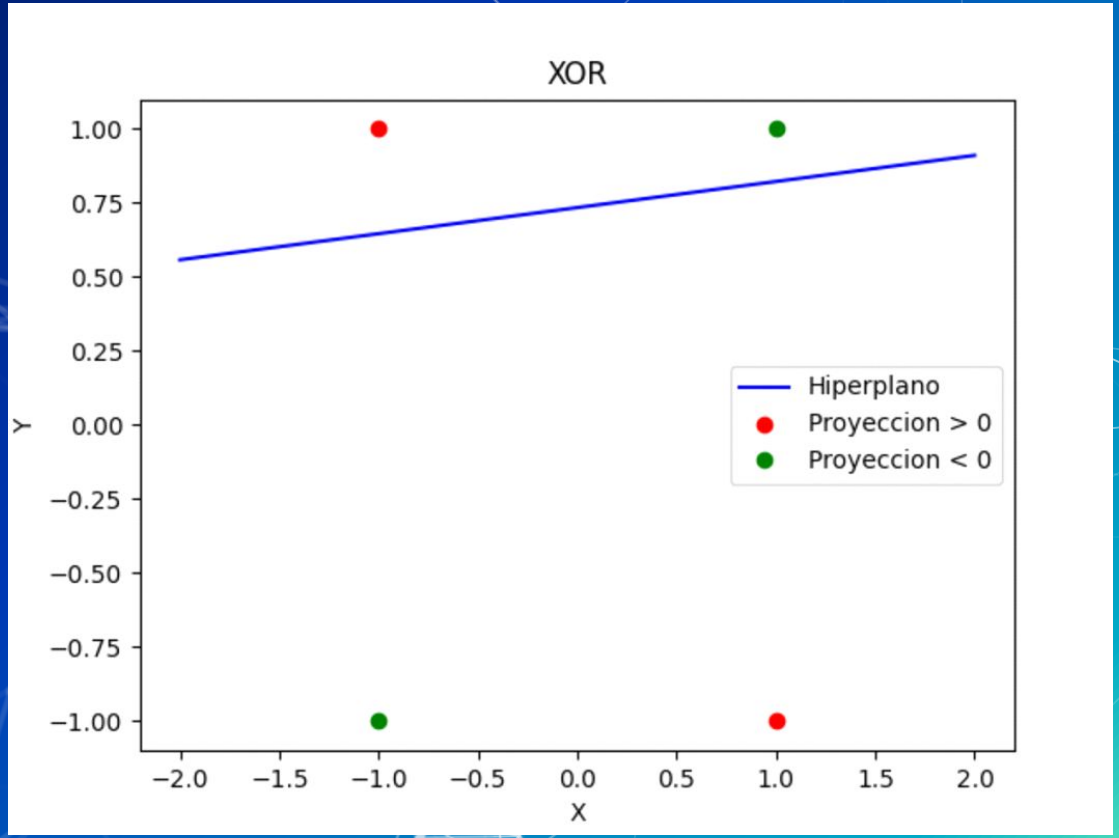
Operación “y”

- Cantidad de épocas: 500
- Learning Rate: 0.01



Operación “xor”

- Cantidad de épocas: 500
- Learning Rate: 0.01



¿Qué puede decir acerca de los problemas que puede resolver el perceptrón simple escalón?

- Como su función de activación es escalón, tiene salida acotada.
- En la operación “y” logra encontrar un hiperplano que separa las dos clases.
- Con la operación “xor” no logra encontrar un hiperplano que separe bien ambas clases. Más adelante vemos qué se puede hacer en este caso.

Ejercicio 2

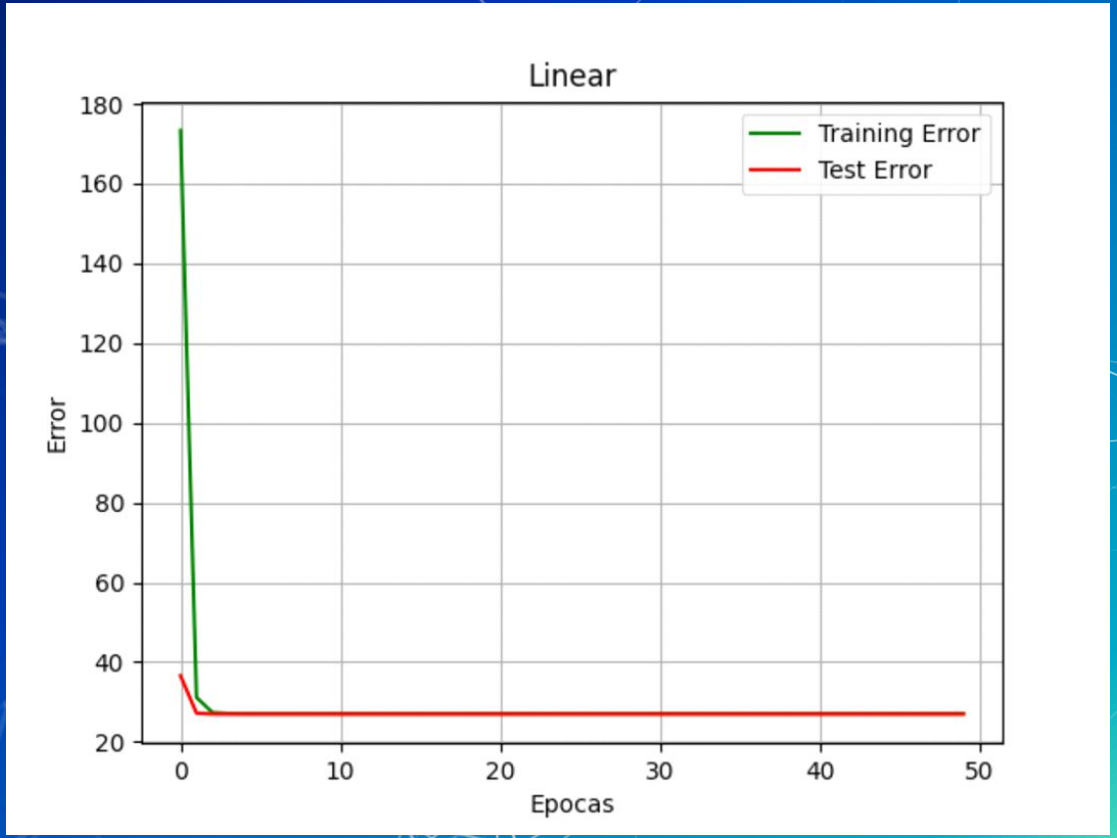
Perceptrón simple Lineal y No Lineal

Funciones a implementar

- Generar un perceptrón que aproxime una función con los datos de entrada propuestos y sus respectivos valores esperados.

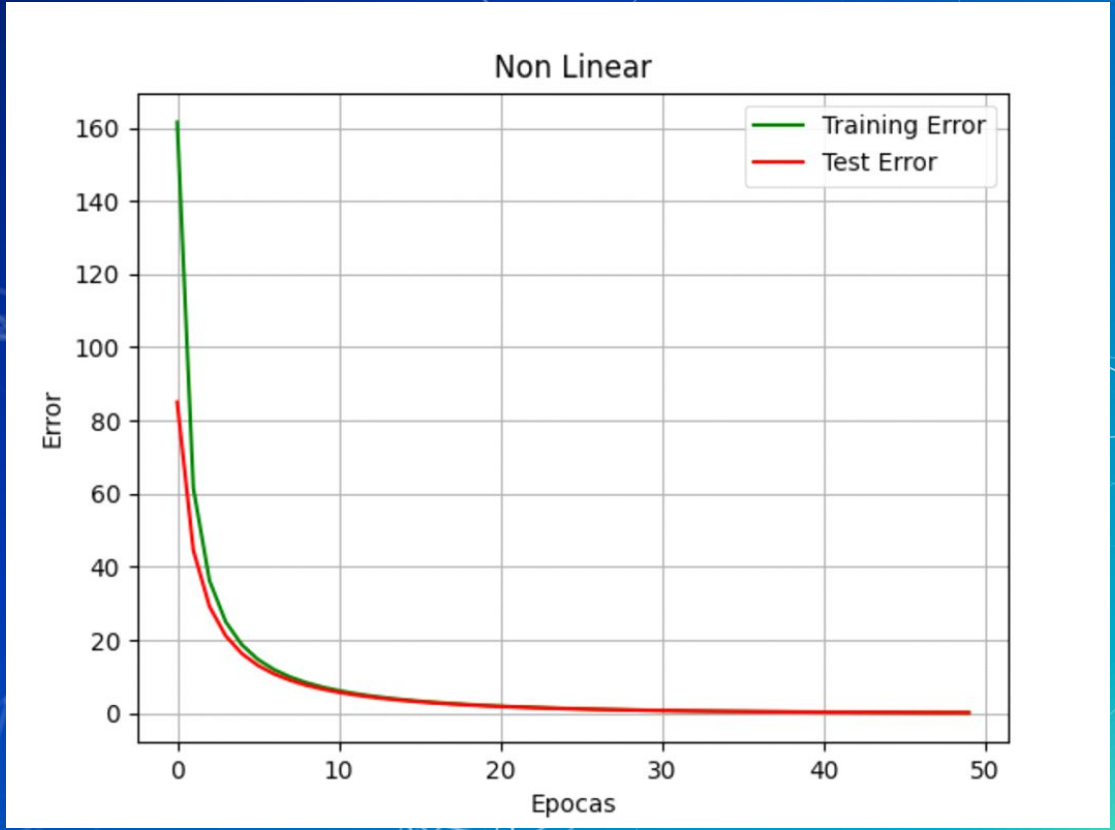
Perceptrón Lineal

- Cantidad de épocas: 50
- Learning Rate: 0.01
- Error final: 27.07



Perceptrón No Lineal

- Cantidad de épocas: 50
- Learning Rate: 0.01
- Error final: 0.156

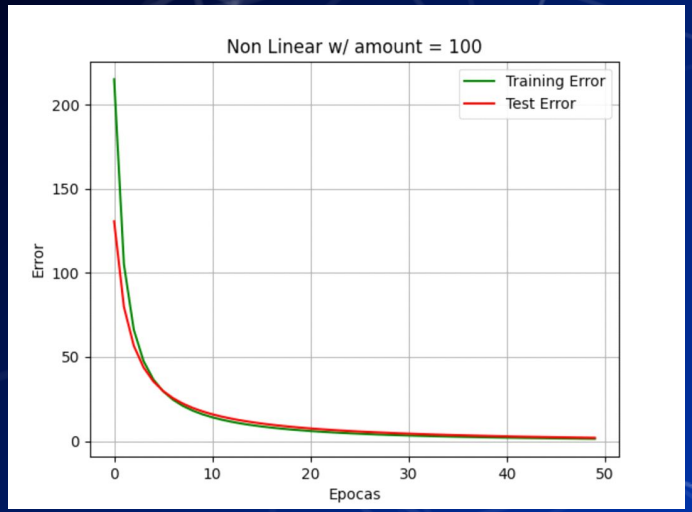


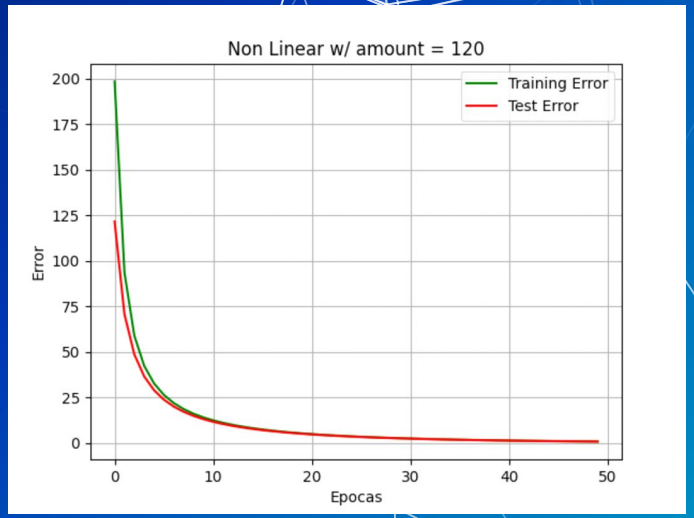
ANÁLISIS

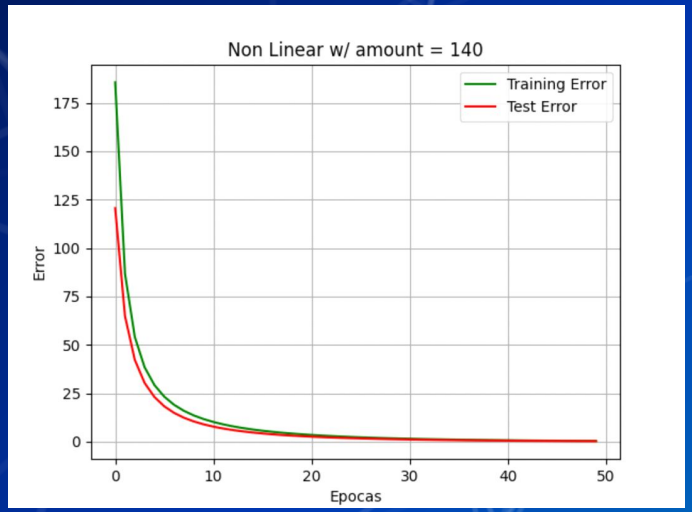
- Perceptrón Lineal: Luego de pocas épocas, el error de entrenamiento y testing es constante, no pudiendo bajar de 27.
- Perceptrón No Lineal: Luego de relativamente pocas épocas, ambos errores convergen a un valor muy cercano a 0.

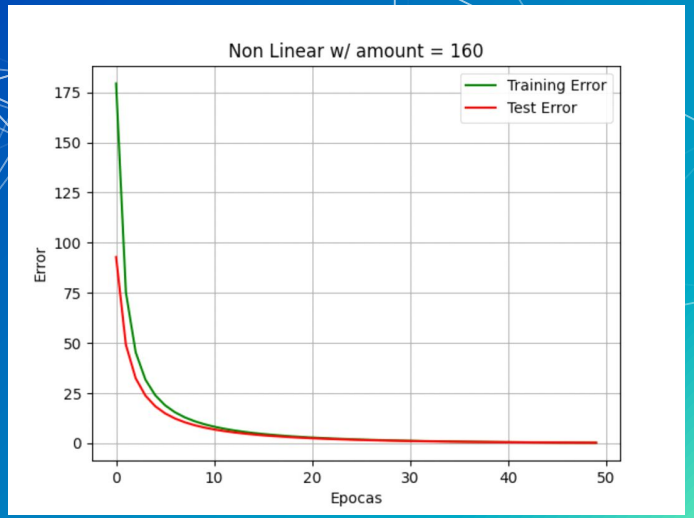
Capacidad de generalización del perceptrón simple no lineal

- Separamos el conjunto de datos en distintas cantidades para el set de testeo y el set de entrenamiento, obteniendo distintos resultados.
- En cada experimento nos referimos al “training amount” como la cantidad de datos en el set de training.


Non Linear w/ amount = 100
Training Error
Test Error
Error
Epocas
0
50
100
150
200
0
10
20
30
40
50


Non Linear w/ amount = 120
Training Error
Test Error
Error
Epocas
0
25
50
75
100
125
150
175
200
0
10
20
30
40
50


Non Linear w/ amount = 140
Training Error
Test Error
Error
Epocas
0
25
50
75
100
125
150
175
0
10
20
30
40
50


Non Linear w/ amount = 160
Training Error
Test Error
Error
Epocas
0
25
50
75
100
125
150
175
0
10
20
30
40
50

ANÁLISIS

- El error de testeo siempre converge junto con el error de entrenamiento.
- Distintos valores iniciales para los errores de testeo.

¿Cómo escoger el mejor conjunto de testeo?

- Distintas divisiones del conjunto de datos, evaluar y comparar cada una de ellas para elegir la mejor.
- Validación cruzada.

Máxima capacidad de generalización del perceptrón para este conjunto de datos

- Matriz de confusión.
- Calcular la precisión de cada división, promediarla y calcular su varianza.

Ejercicio 3

Perceptrón multicapa

Ejercicio 3.1

Perceptrón multicapa y operación “xor”

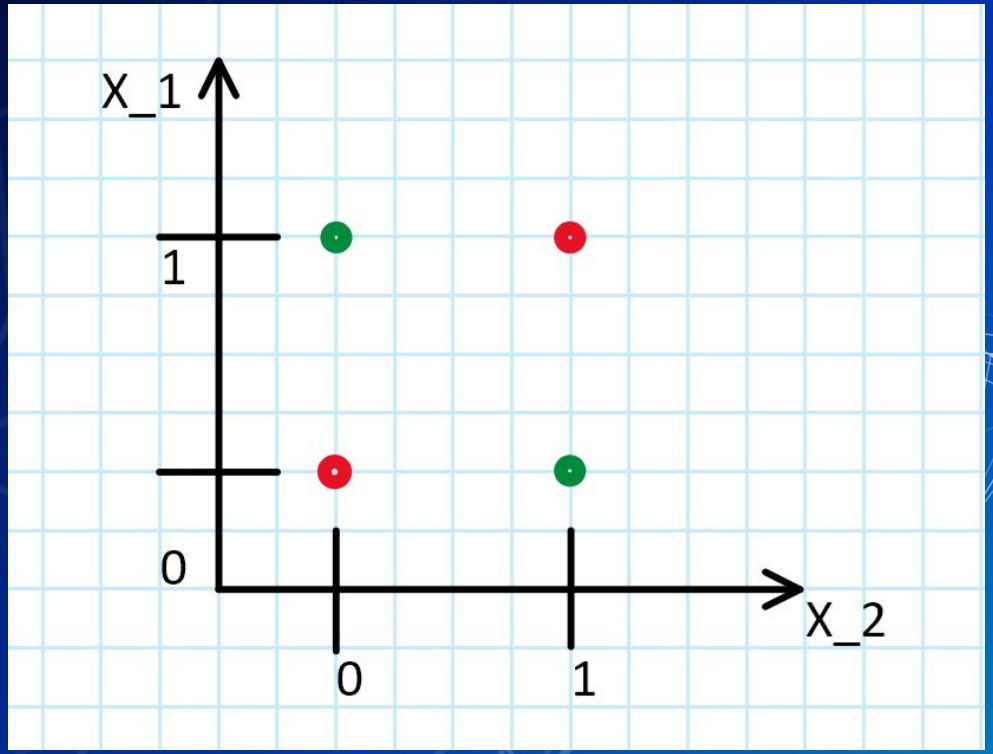
Función a implementar

- Operación lógica “xor”

Predicciones

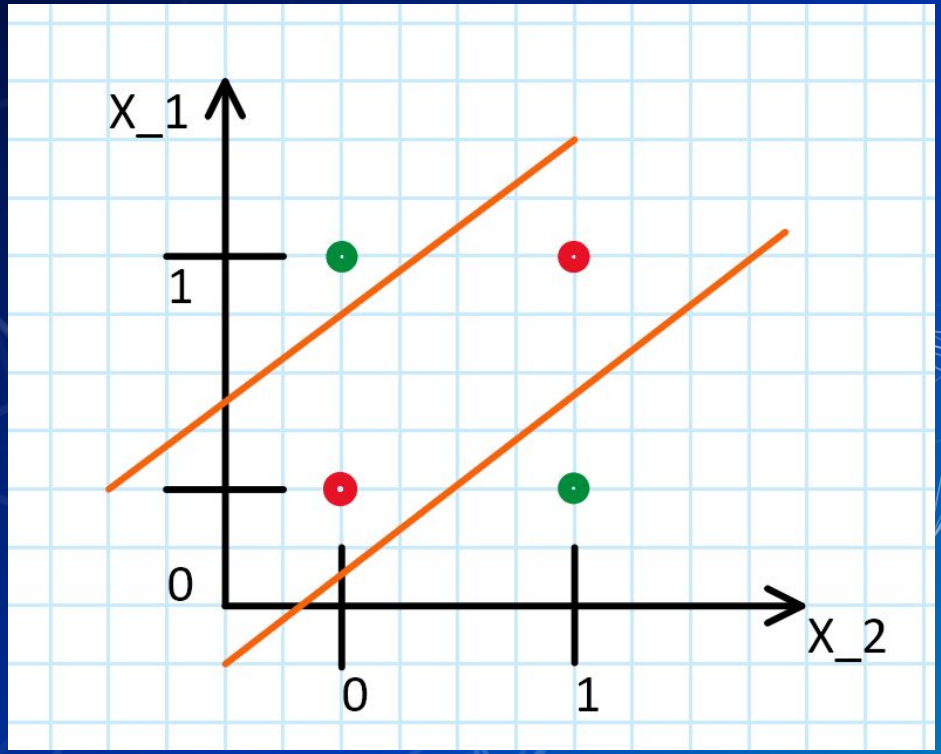
- En base a lo visto en los ejercicios anteriores, sabemos que un solo hiperplano no es suficiente para resolver la operación lógica xor.
- Como el conjunto de entrada es de 2 dimensiones y la salida de 1, podemos graficar los resultados.

Predicciones



- No podemos separar las clases con una recta pero...

Predicciones

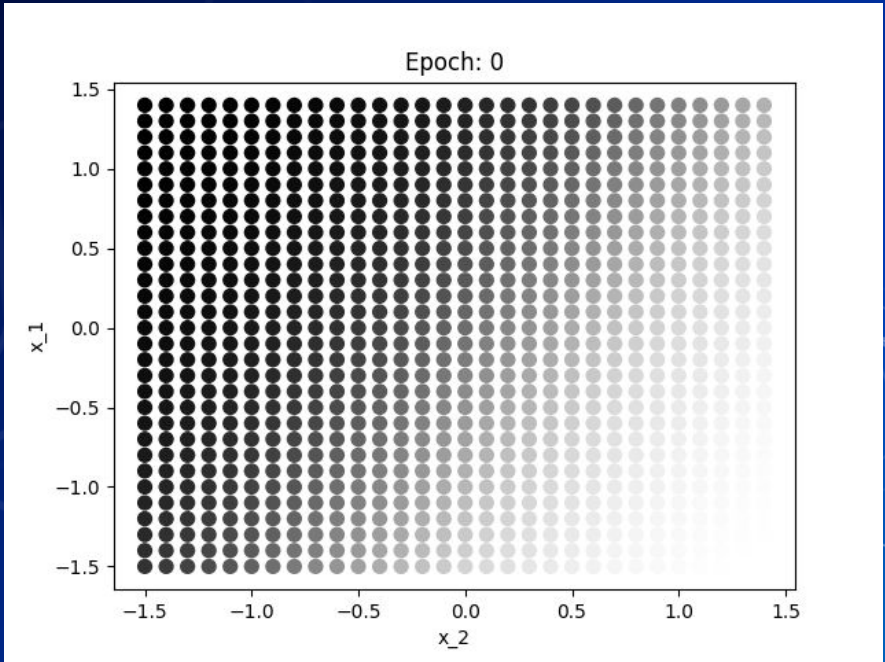


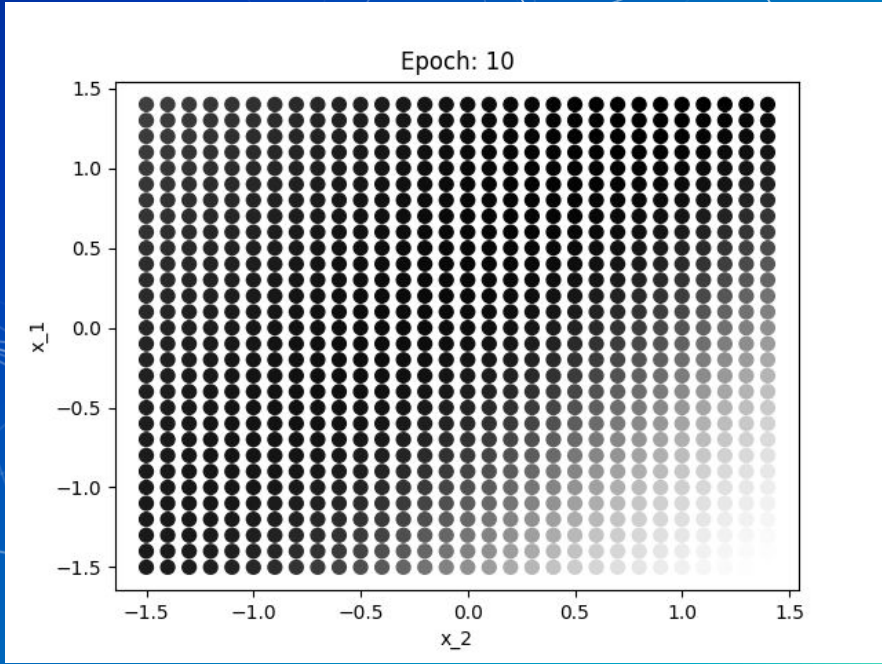
- Se resuelve fácilmente con dos rectas.
- Al menos necesitaremos dos perceptrones simples.

Configuración

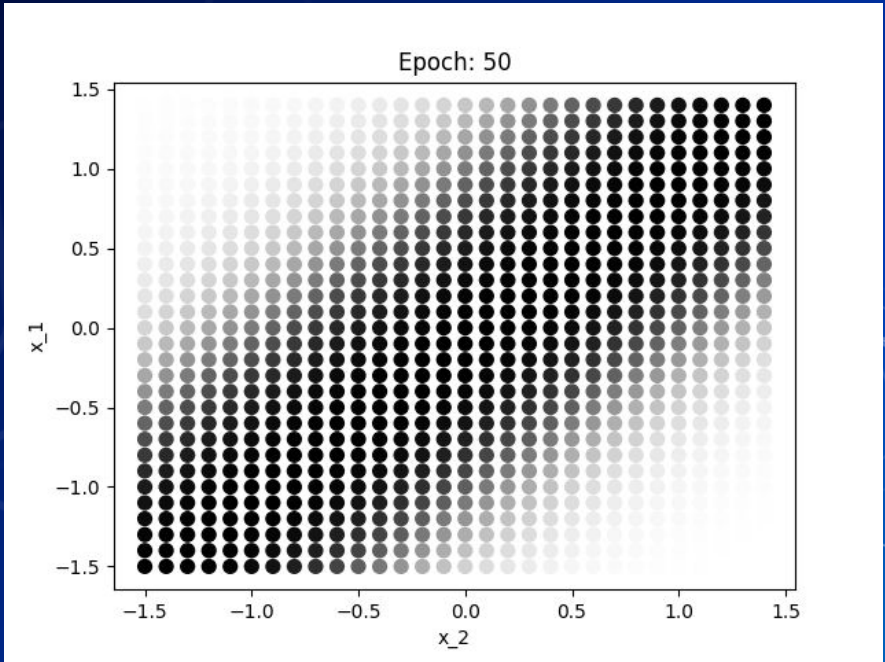
- Hidden Layers: 1
- Nodes per layer: 2
- Learning Rate: 0.01
- Epochs: 100
- Activation: tanh

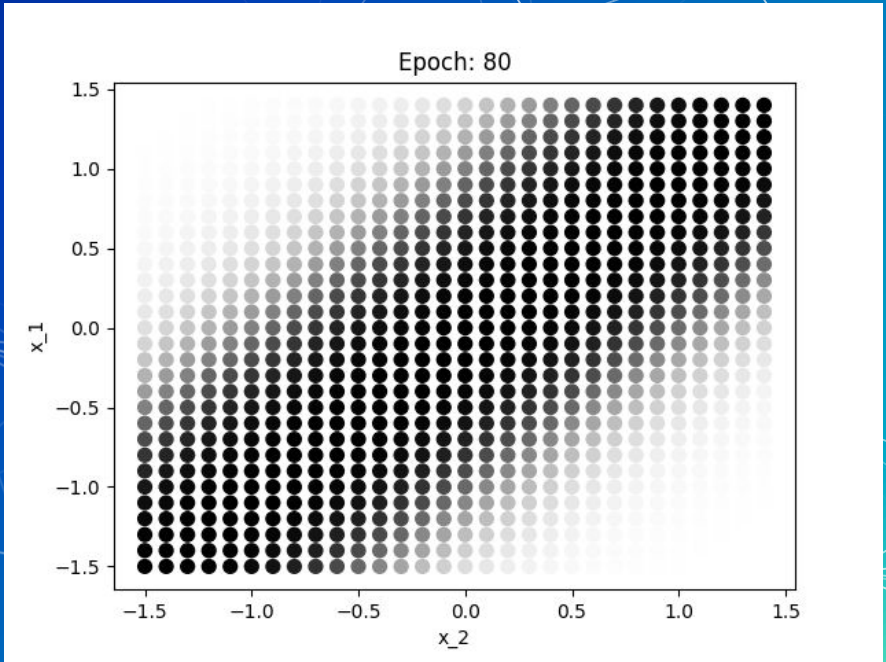
Resultados


Epoch: 0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x_2


Epoch: 10
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x_2

Resultados


Epoch: 50
x_1
x_2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5


Epoch: 80
x_1
x_2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

ANÁLISIS

- En las primeras épocas, aún no había aprendido a clasificar bien.
- Mientras avanza, se puede ver como separa el espacio con dos rectas, dando el resultado esperado.

Ejercicio 3.2

Perceptrón multicapa y números pares e impares

Función a implementar

- Dadas imágenes de números de 7x5 “pixeles”, crear un perceptrón multicapa para decidir si cada número es par o impar.

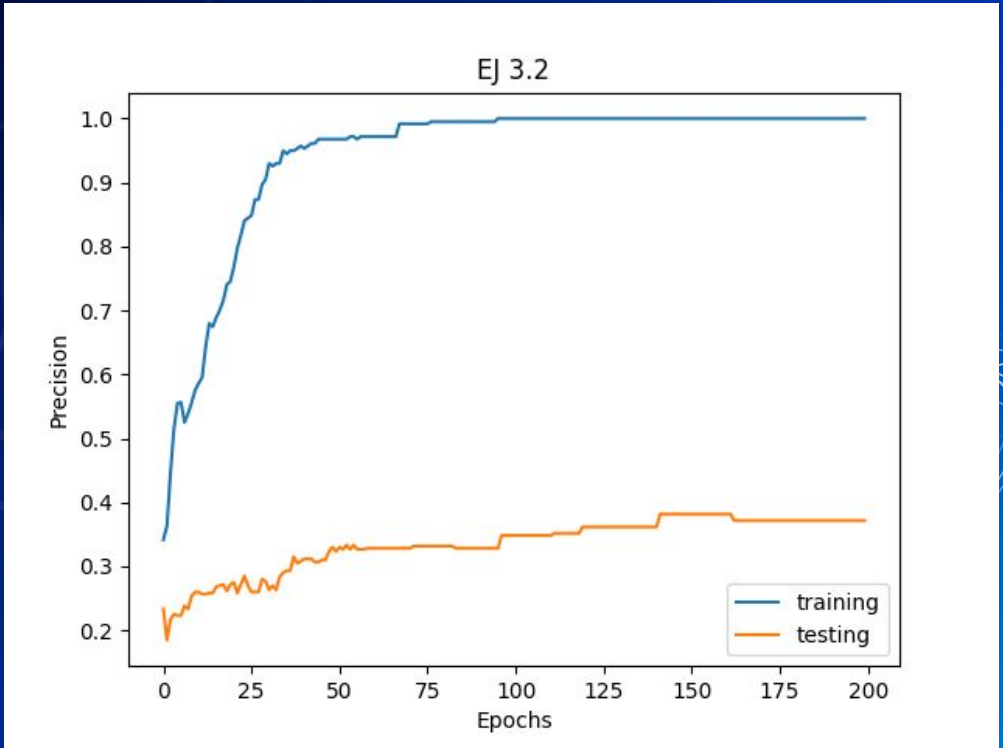
Procedimiento

- Se realizaron 50 runs distintas para 50 redes distintas.
- En cada run se seleccionó un subconjunto de 4 números para formar el set de testeo (Validación cruzada).
- Se evaluó la precisión de cada red y se promediaron sus resultados.

Configuración

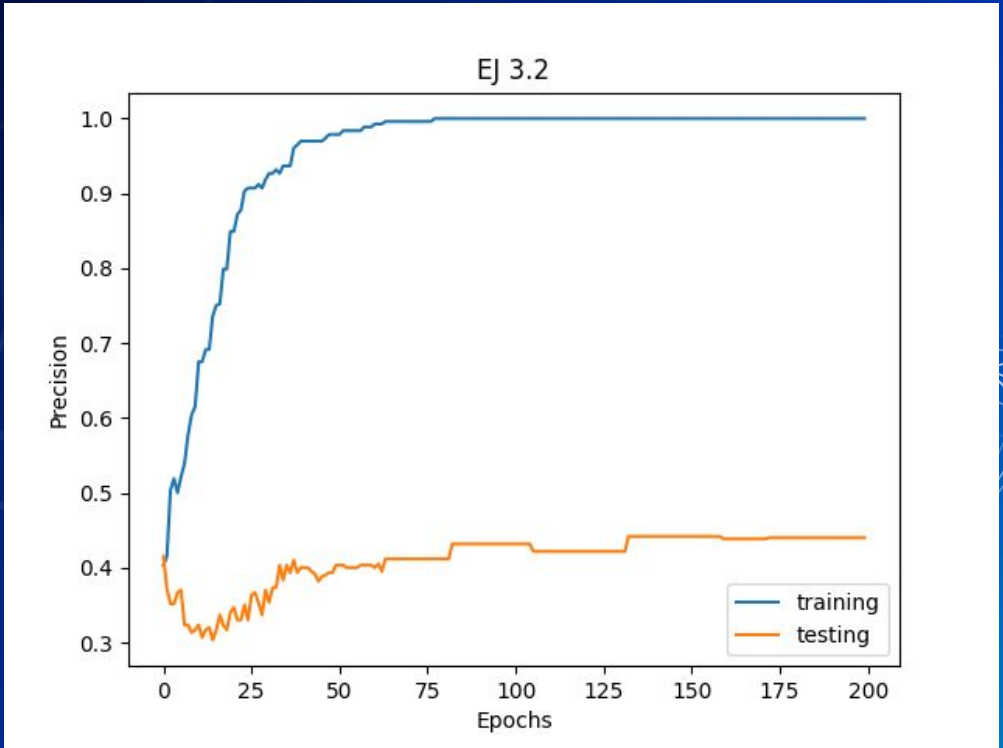
- Hidden Layers: 1
- Nodes per layer: 10
- Learning Rate: 0.2
- Epochs: 100
- Testing Division: 4
- Total Neural Networks: 50
- Activation: sigmoid

Resultados



- Wow! El training set converge en 1 pero el testing set no.
- Probemos otra vez...

Resultados



- Sucedió de nuevo!
- ¿Qué está pasando?

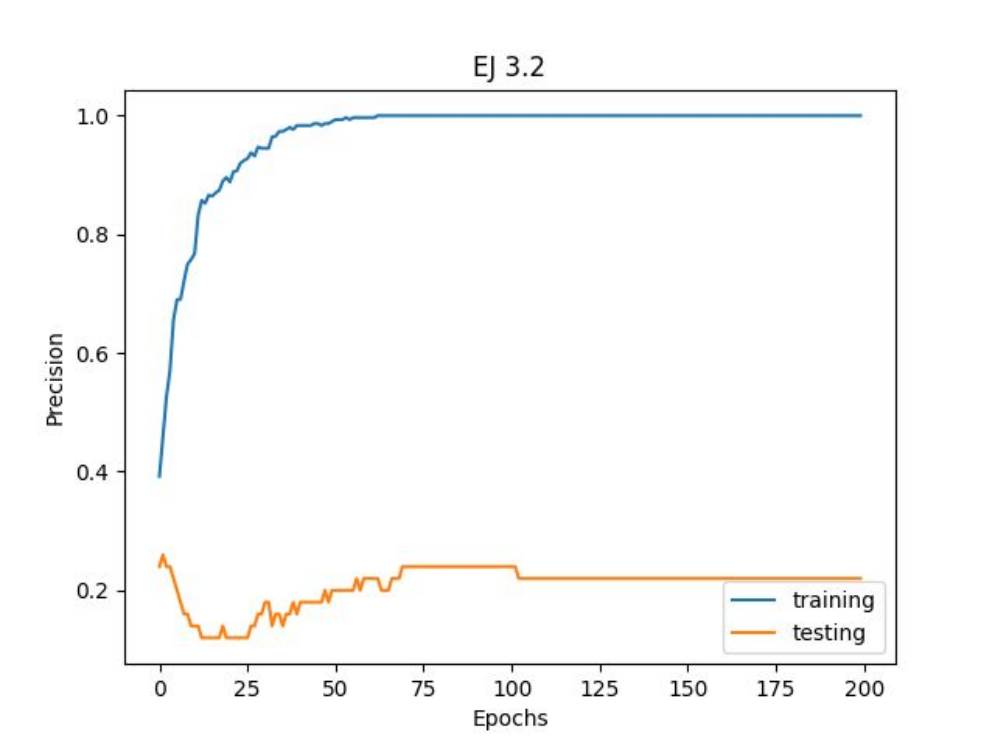
ANÁLISIS

- La precisión con el entrenamiento converge en 1, por lo que la red está aprendiendo.
- La precisión con el set de testeo decae.
- Esto puede deberse a que el conjunto de datos que tenemos es muy acotado, por lo que a la red le cuesta generalizar.
- Al elegir una porción de datos como testeo, estamos privando a la red de que aprenda de todos los números.

ANÁLISIS

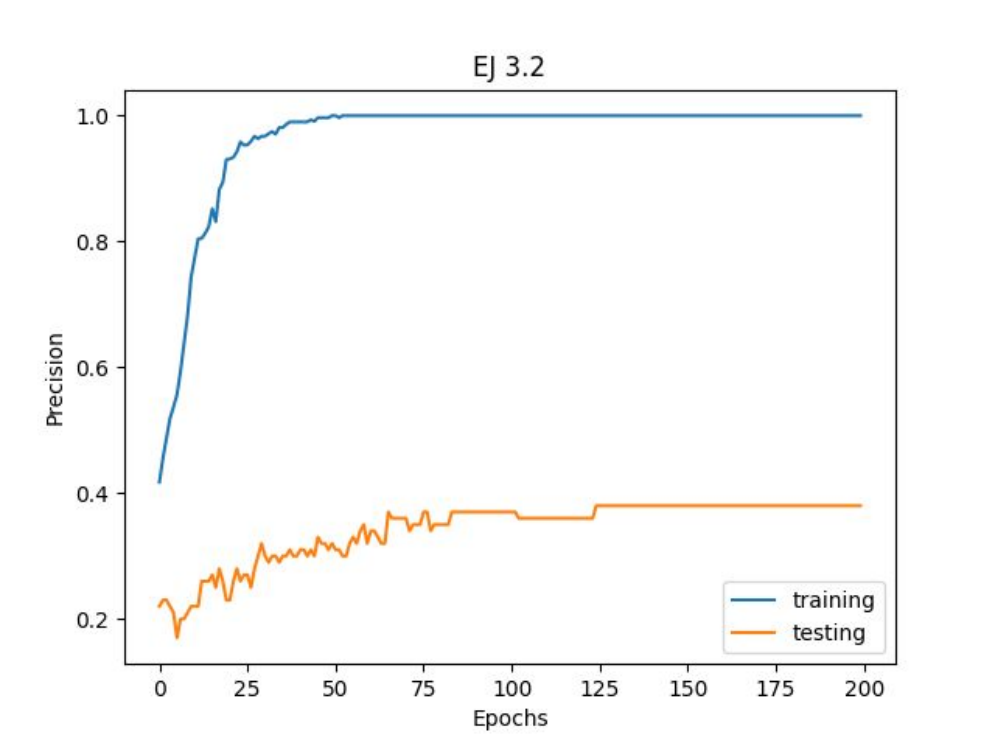
- Podemos probar cambiando la cantidad de datos que escogemos para el set de testeo.

Resultados con #testing = 1



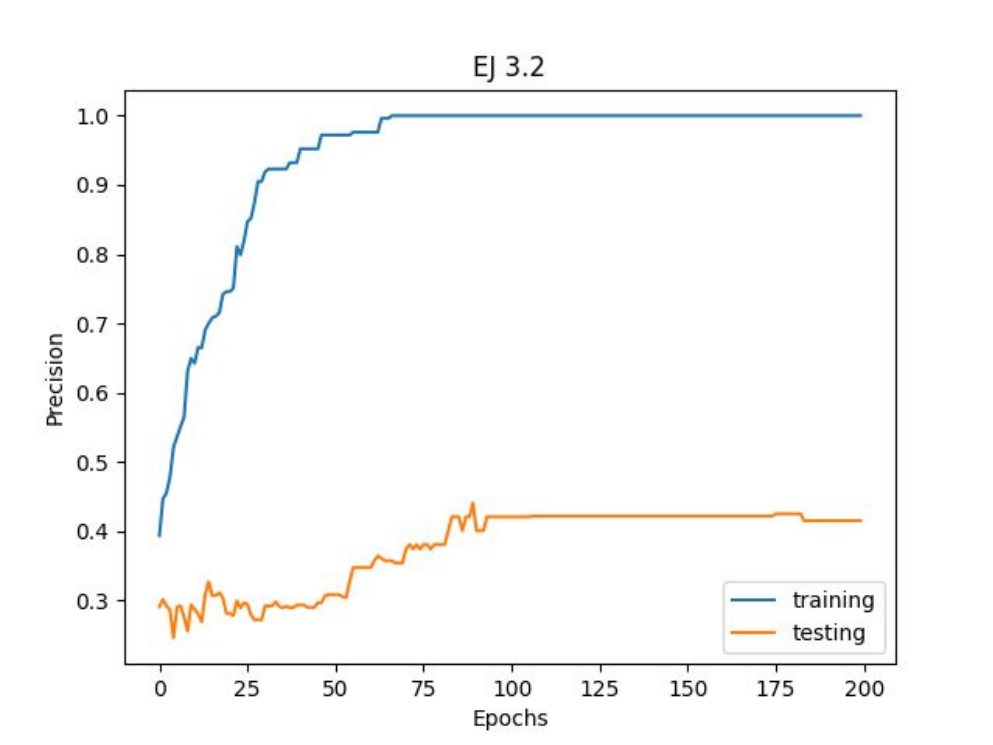
- Continua.

Resultados con #testing = 2



- Igual.

Resultados con #testing = 5



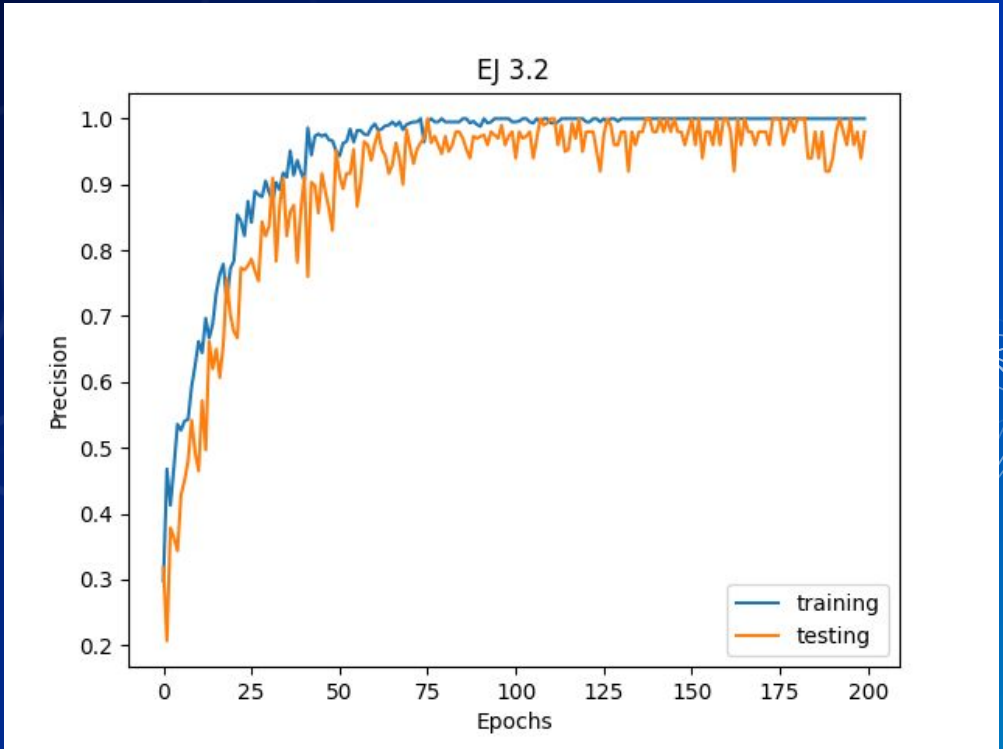
ANÁLISIS

- Como vemos, la cantidad no cambia el resultado.
- Elegir más de 5 números no tiene sentido, pues haría que la red se entrene con menos números que con los que se testea.

ANÁLISIS

- Podríamos ver que sucede con la red cuando entrena con todos los datos.
- Probamos implementar una estrategia de entrenamiento parecida a la validación cruzada, pero en donde cambiamos los sets por épocas.
- Así una red nunca entrena con un dato que usa para test en una misma época, pero a la larga termina entrenando con todos los datos.

Resultados



- **WOW**. Tanto el training set como el testing set convergen con precisión cercana a 1.
- Las redes funcionan pero... ¿Son útiles? **NO**.

CONCLUSIONES

- Un conjunto acotado de datos no permite que la red entrene bien, pues es privada de una gran porción de los datos.
- Sin embargo, no es imposible que pueda entrenarse para predecir bien, incluso con pocos datos. Es improbable.

Ejercicio 3.3

Perceptrón multicapa y reconocimiento de dígitos.

Función a implementar

- Similar al ejercicio anterior. Implementar un perceptrón multicapa que reconozca a qué dígito representa cada imagen.
- Una vez entrenado, agregar ruido a las imágenes y probar su efectividad.

PREDICCIONES

- Si elegimos un subconjunto de los datos para entrenar a las redes, la precisión de testeo será muy baja, haciendo a la red inútil.

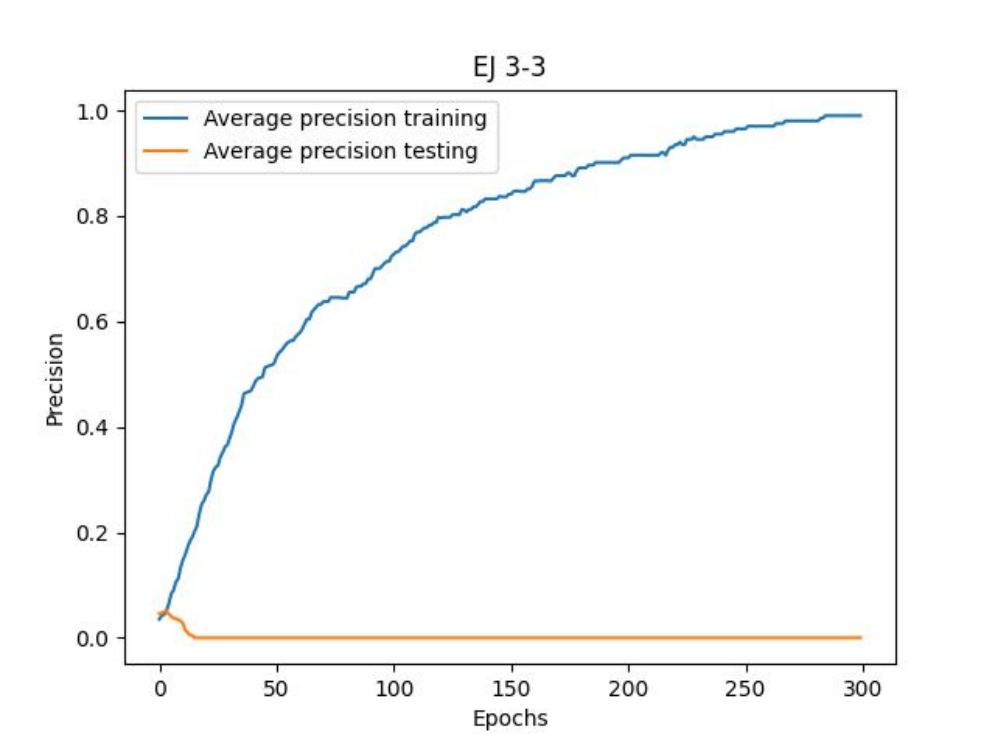
Procedimiento

- En este caso, como el perceptrón multicapa es multiclase, tenemos que calcular la precisión y accuracy de cada clase.
- Luego procedimos a promediar dichos valores por red.
- Finalmente, promediamos esos resultados entre todas las redes por época.

Configuración

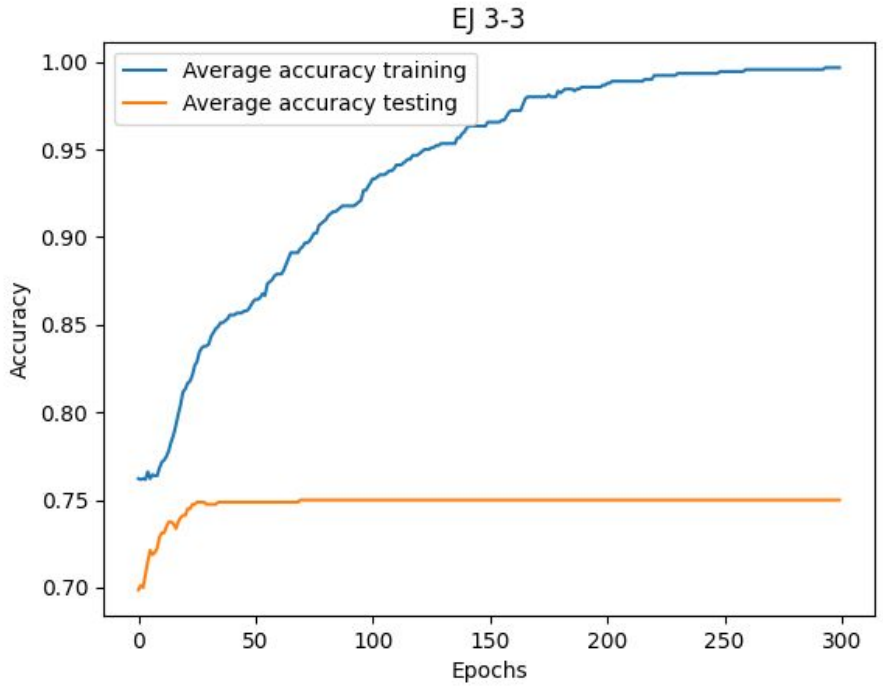
- Hidden Layers: 1
- Nodes per layer: 8
- Learning Rate: 0.1
- Epochs: 300
- Testing Division: 4
- Total Neural Networks: 50
- Activation: sigmoid

Resultados



- Como habíamos predicho, el set de testeo no da buenos resultados.

Resultados



- Podemos ver el gráfico de accuracy para ver que sucede el mismo efecto.

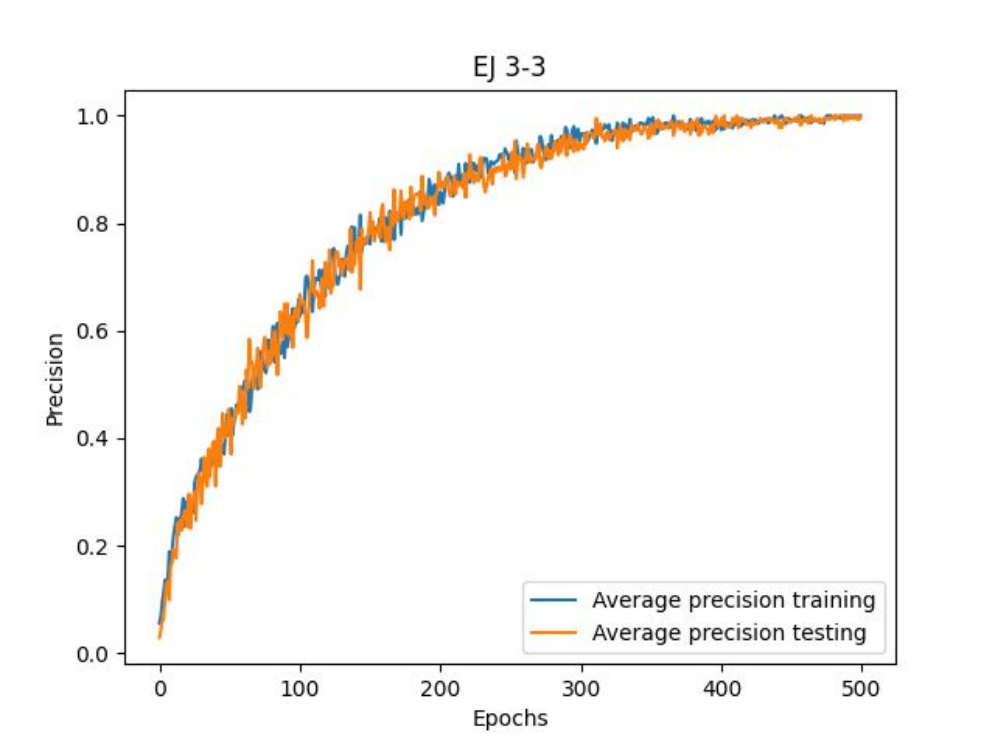
ANÁLISIS

- Podemos explicar este fenómeno de la misma manera que el ejercicio anterior.
- En este caso, al sacar datos para el entrenamiento, hacemos que la red ni siquiera pueda entrenar para algunas clases que deseamos que sea capaz de reconocer.
- Si volvemos a activar la selección del set de testeo por época, funcionará?
- Aprovechamos para agregar Momentum y learning rate dinámico

Configuración

- Hidden Layers: 1
- Nodes per layer: 10
- Learning Rate: 0.2
- Epochs: 100
- Testing Division: 4
- Total Neural Networks: 50
- Activation: sigmoid
- Momentum Alpha: 0.9
- Dynamic Learning Rate Values: A=0.02, B=0.01

Resultados



- Bien, las redes funcionan.

Luego...

- Para testear los dígitos con ruido entrenamos las redes con todo el conjunto de datos para testearlas luego.
- Para cada test, incluimos una probabilidad de ruido, la cual es la chance de que cada pixel se invierta.

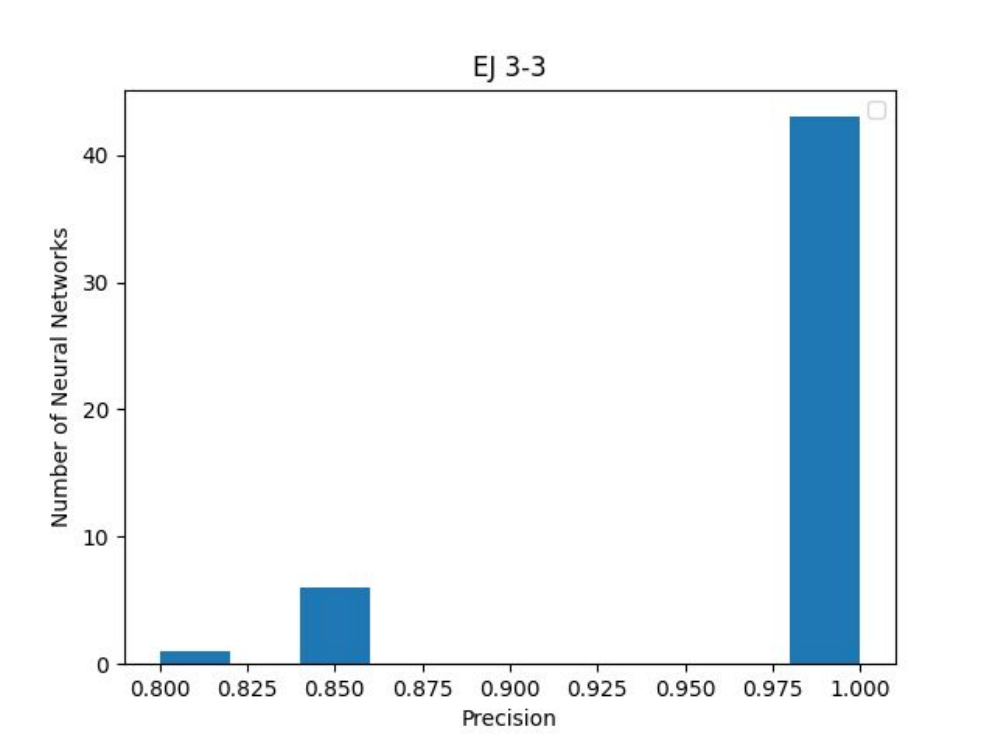
Procedimiento

- Creamos las redes con la misma configuración anterior.
- Como sabemos que al entrenar con todos los dígitos, las redes aprenden, no nos centramos en sus valores a través de las épocas sino en los valores de las redes ya entrenadas y testeadas con las imágenes con ruido.

Procedimiento

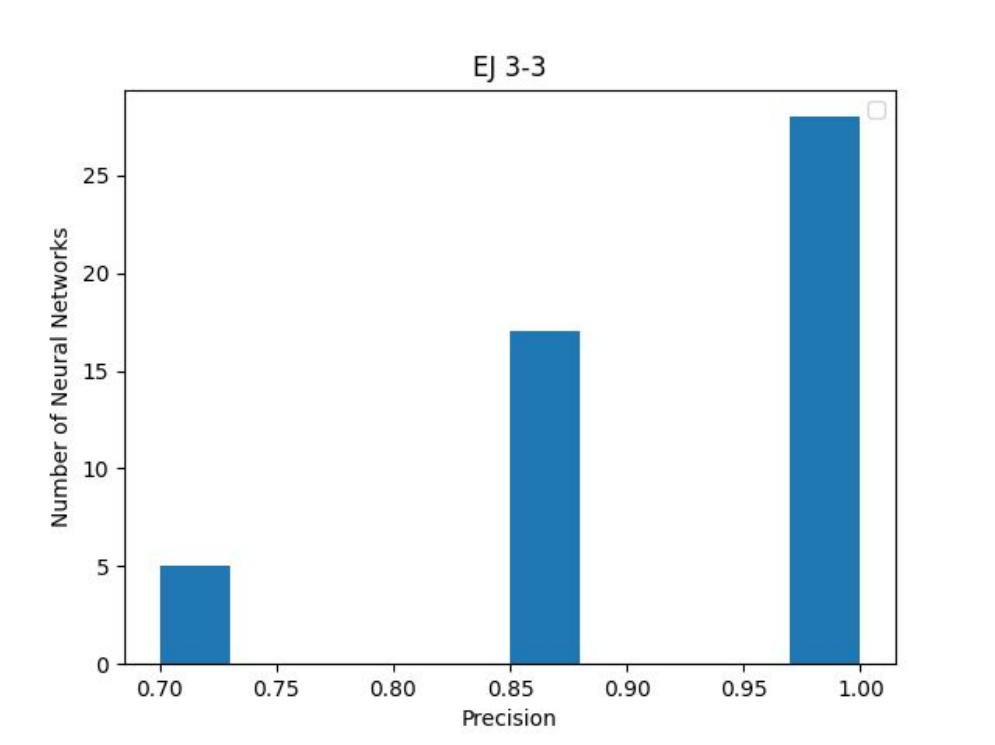
- Por esta razón, graficamos un histograma mostrando la precisión de las redes ya entrenadas para diferentes probabilidades de ruido.

Resultados con probabilidad de 5%



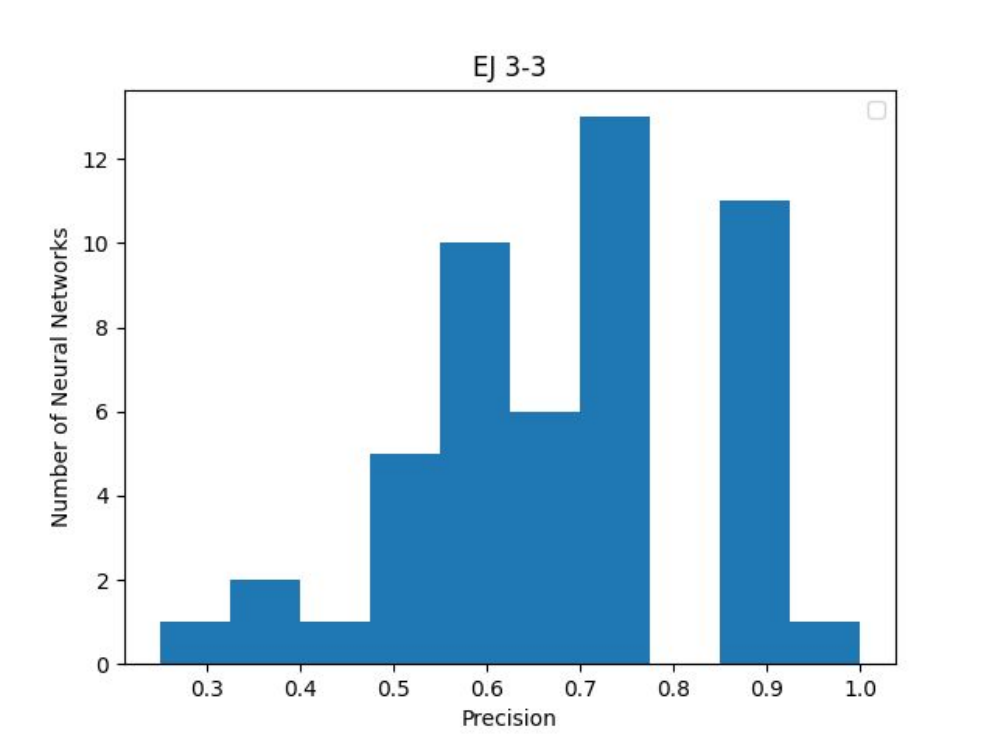
- En promedio cambian entre uno y dos píxeles por imagen.
- La mayoría de las redes obtuvieron una precisión cercana a 1!

Resultados con probabilidad de 10%



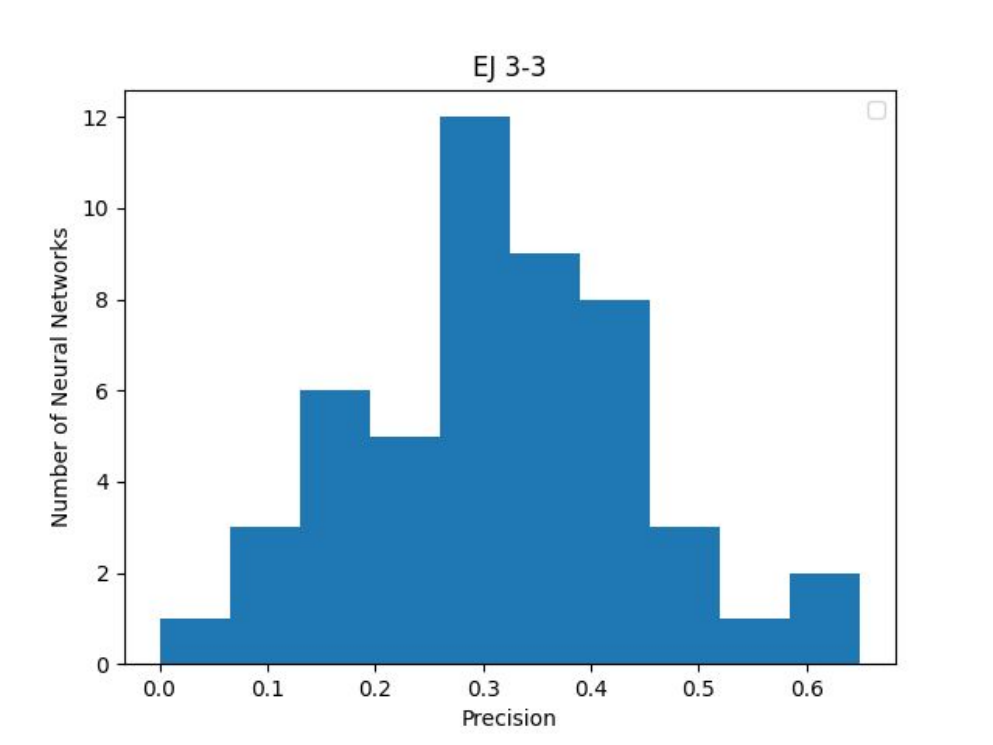
- En promedio cambian 3 o 4 píxeles por imagen.
- Más de la mitad de las redes tiene una precisión muy cercana a 1, increible!

Resultados con probabilidad de 20%



- En promedio cambian 7 píxeles por imagen.
- Aquí las redes comienzan a flaquear, pero hay que tener en cuenta que ⅕ de la imagen esta distorsionada.

Resultados con probabilidad de 30%



- Cambian entre 10 y 11 píxeles por imagen.
- Aquí las redes no son confiables, pero con éste nivel de distorsión, a la mayoría de nosotros nos costaría distinguir números.

CONCLUSIONES

- Con un poco de ruido, la red fue capaz de utilizar todos los datos como entrenamiento y testearse con datos nuevos.
- Esto nos da la idea de que, en caso de tener un conjunto de datos acotado, podemos transformarlo un poco para hacer más grande el conjunto disponible.

CRÉDITOS Y BIBLIOGRAFÍA

- Apuntes disponibles de la materia.
- Make Your Own Neural Network - Tariq Rashid
- The Nature of Code - Daniel Shiffman
- Confusion Matrix for Multi-Class Classification

GRACIAS!

Alumnos

- Pedro Jeremías López Guzmán
- Martín Craviotto