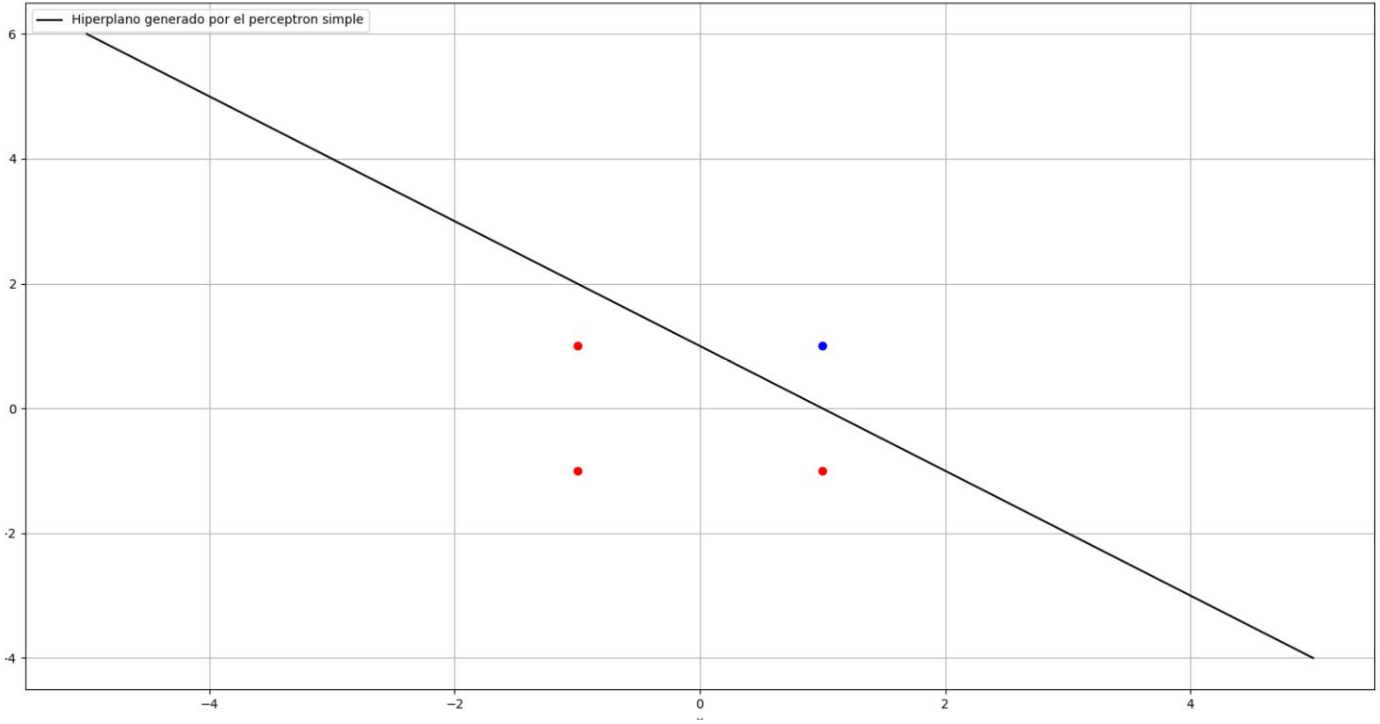
Redes Neuronales: Perceptron Simple y Multicapa

Grupo 6: Katan, Paganini

Ejercicio 1
Perceptrón simple lineal

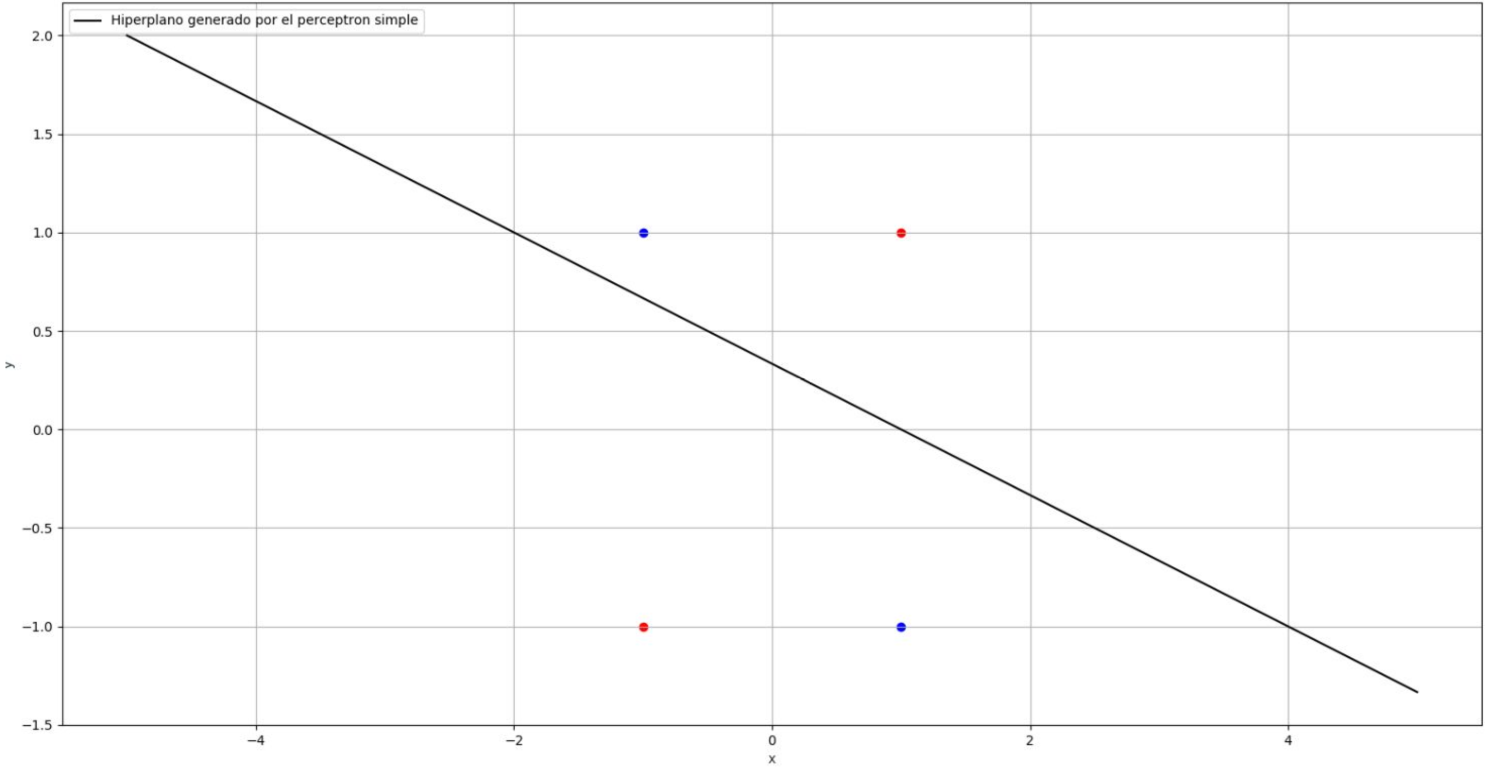
Aprendiendo la función AND

- ETA = 0.01
- Para esta prueba, tardó 12 iteraciones



Intentando aprender la función XOR

- ETA = 0.01
- No aprende



Conclusiones del perceptrón simple escalón

- Solo puede aprender problemas donde los datos son linealmente separables
- Como la salida está en {1,-1}, es sólo útil para problemas de clasificación binaria

Ejercicio 2
Perceptrón simple lineal y no lineal

Perceptrón simple lineal, aprendizaje

```
Linear test:
Weights obtained:
(0.068455316, 0.05594655, 0.06860652, 0.4003615)

Linear model evaluation, using only training set
(1/2) * Squared sum error = 0.7960861
Average error = 0.08890055
Total accumulated error: 12.446077
```

- Antes de entrenar, se normalizaron las salidas esperadas al rango [0,1], para interpretar más fácilmente el error

- Parámetros usados:
  - Eta = 0.01
  - Error mínimo = 0.9404
  - Optimización usando momentum con alpha = 0.8

Perceptrón simple no lineal, aprendizaje

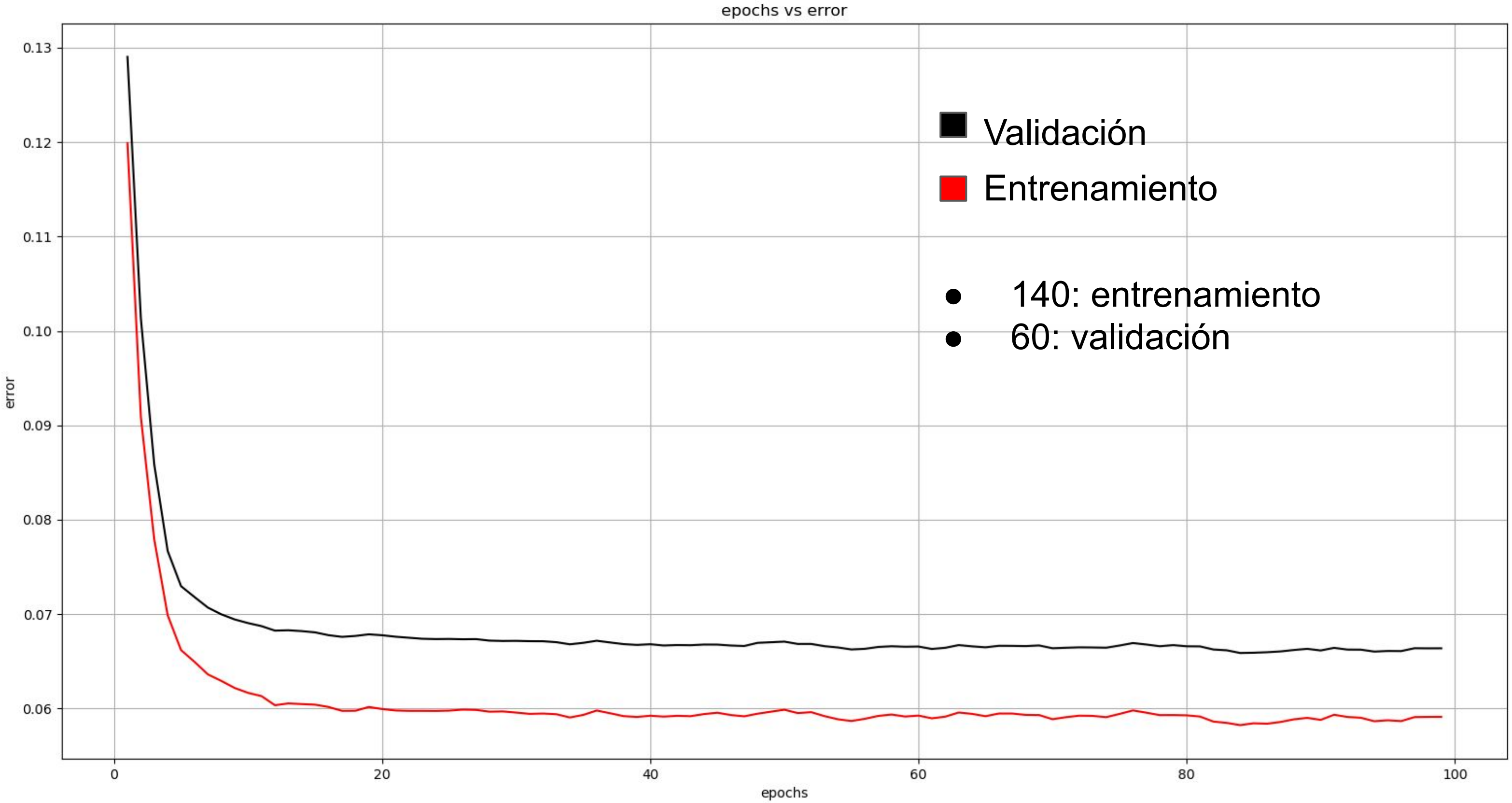
```
Non linear test:
Weights obtained:
(0.4873182, 0.53452605, 0.47007972, -4.4155845E-6)

Non linear model evaluation, using only testing set
(1/2) * Squared sum error = 0.35314763
Average error = 0.0567026
Total accumulated error: 7.938364
```

- Antes de entrenar, se normalizaron las salidas esperadas al rango [0,1], ya que se usó la función de activación sigmoide

- Parámetros usados:
  - Eta = 0.01
  - Error mínimo = 0.55

Perceptrón simple no lineal, capacidad de generalización



- 140: entrenamiento
- 60: validación

Eligiendo el mejor conjunto de entrenamiento

- Convendría realizar una validación cruzada en k-partes, utilizando como métrica de evaluación el error

Maximizando la capacidad de generalización

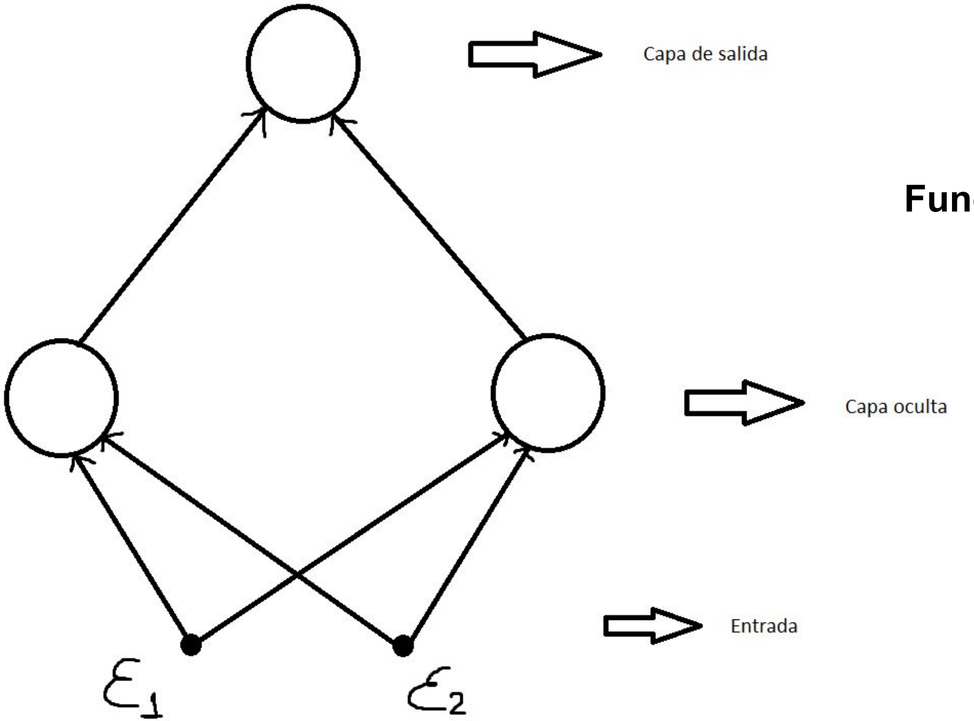
- Tenemos que buscar el conjunto de entrenamiento más chico que nos permita generalizar de la mejor manera el conjunto más grande de validación

- Podemos tomar distintos tamaños posibles del conjunto de entrenamiento, y para cada tamaño realizar una validación cruzada, obteniendo el mejor conjunto para ese tamaño

- Nos quedamos con el conjunto más chico de entrenamiento que permita la mejor generalización

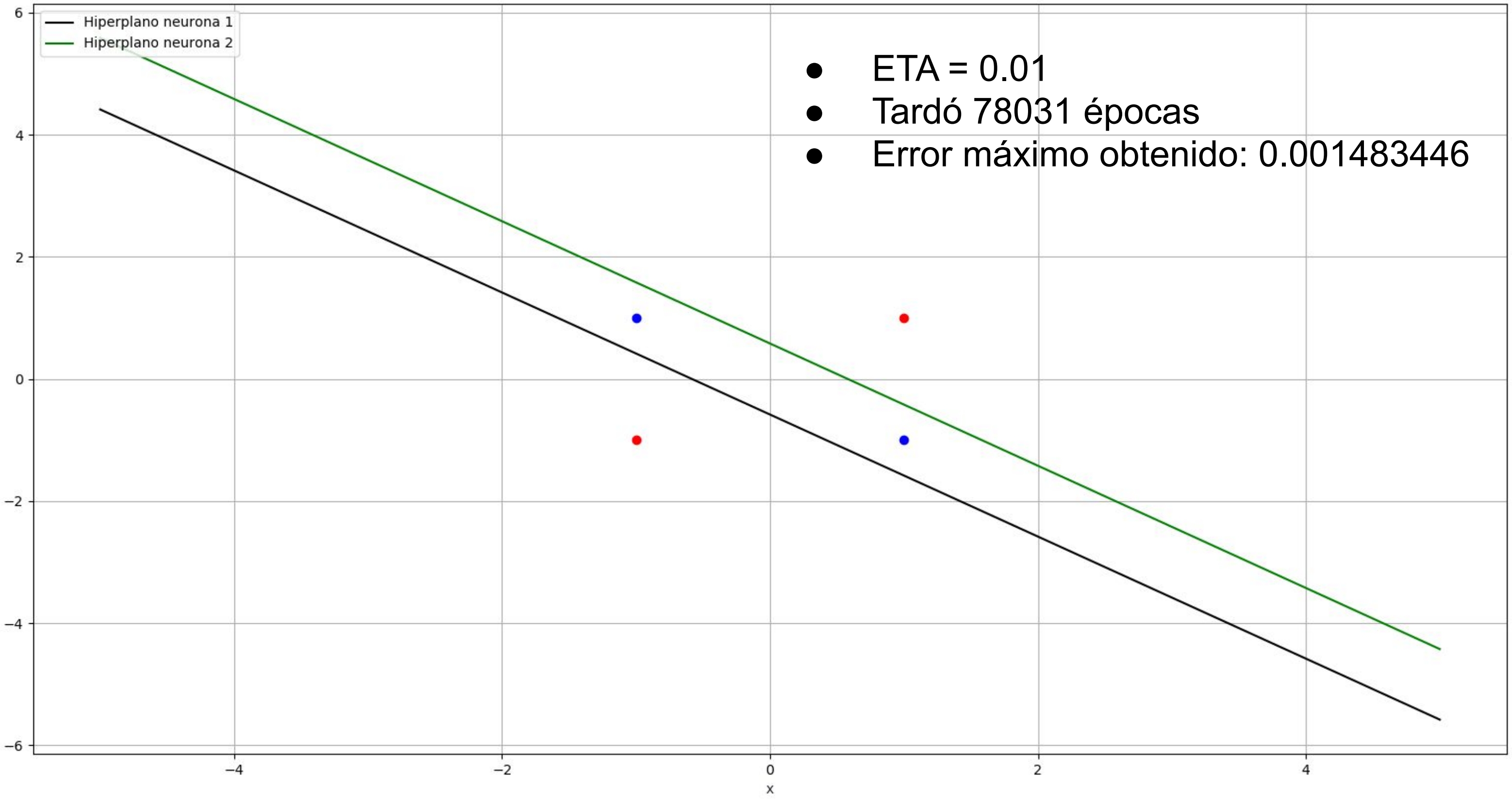
Ejercicio 3
Perceptrón Multicapa

Arquitectura para XOR



Función de activación: tanh

Aprendiendo la función XOR

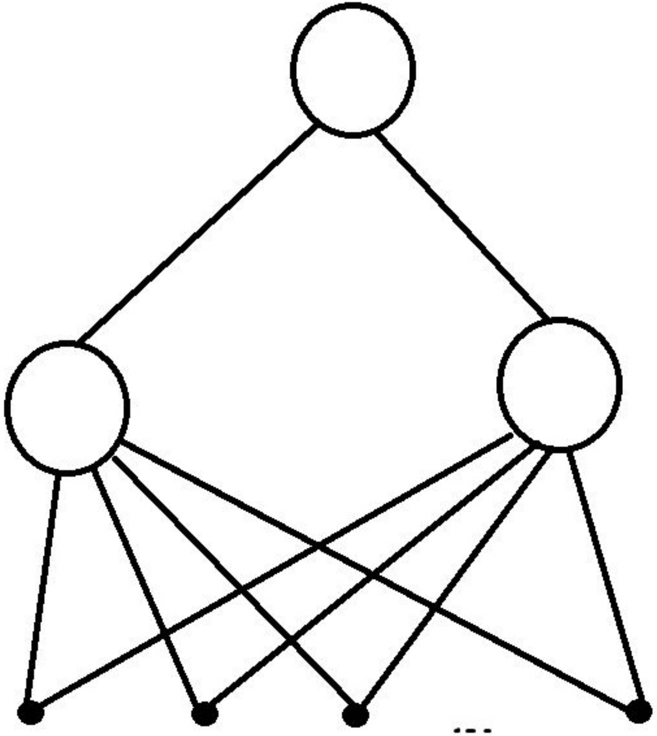


- ETA = 0.01
- Tardó 78031 épocas
- Error máximo obtenido: 0.001483446

Aprendiendo la función XOR

```
XOR test results
Results:
(Inputs: [-1.0, 1.0]) -> (Outputs: [0.998506612948233])
Expected outputs: [1.0]
(Inputs: [1.0, -1.0]) -> (Outputs: [0.9984439547733555])
Expected outputs: [1.0]
(Inputs: [-1.0, -1.0]) -> (Outputs: [-0.9976948665518068])
Expected outputs: [-1.0]
(Inputs: [1.0, 1.0]) -> (Outputs: [-0.997716611593947])
Expected outputs: [-1.0]
```

Arquitectura para Imágenes



- **Función de activación: tanh**
- **5x7 entradas**

Intentando aprender “paridad” de las “imágenes”

```
Test results using training data:
Results:
Correct predictions: 6
Incorrect predictions: 0

Tests results using testing data:
Results:
Correct predictions: 1
Incorrect predictions: 3
```

- ETA = 0.001
- Tardó 26600 épocas
- Error máximo = 0.00981
- Elegidos al azar, 6 elementos para entrenar, y 4 para validar
- Aprende bien el conjunto de entrenamiento
- No generaliza bien
- Pero el problema a resolver, ¿es generalizable?