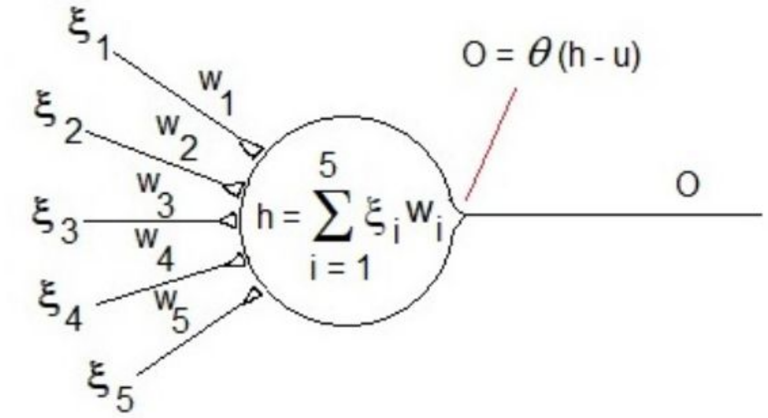
Trabajo Práctico N°3 - Correcciones

- De Simone, Franco - 61100
- Dizenhaus, Manuel - 61101
- Cornidez, Milagros - 61432

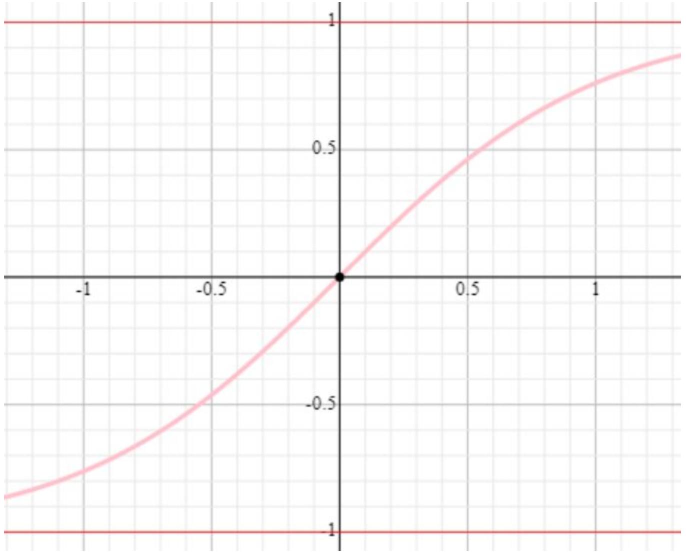
Ejercicio 2: Perceptrón Simple Lineal y No Lineal

¿Cuál fue el error?



$$\Theta : \mathbb{R} \to (-1; 1)$$

Tanh(βx) (con β=1)



¿Cuál fue el error?

Algunas funciones de **activación (las sigmoideas)** estaban acotadas entre *(-1;1)* o *(0;1)* pero los valores de salida esperados vivían en el dominio de los ℝ...

$$E(w) = \frac{1}{2}\sum_{\mu=1}^{p}(\zeta^{\mu} - O^{\mu})^2$$

61.2837
88.3402
34.1122
23.7819
11.4246
42.6881
95.9901
71.3692
4.7821
20.3834

A izquierda, vemos un muestreo de 10 valores de los 200 extraídos de el conjunto de salida.

Tomando que a estos números se les restará **+-1,** elevarlos al cuadrado, y dividirlos por 2, la **suma de los mismos dará, como mínimo, ≅ 14544.70** (asumiendo que a todos los valores les restamos 0.99)

Imaginemos que este es el error de nada más que 10 números, cuando sumemos los **200...**

Entonces, ¿qué solución proponen?

Normalizar la salida

$$x' = \frac{2(x_i - min(x))}{max(x) - min(x)} - 1$$

Resultados

Perceptrón Lineal:

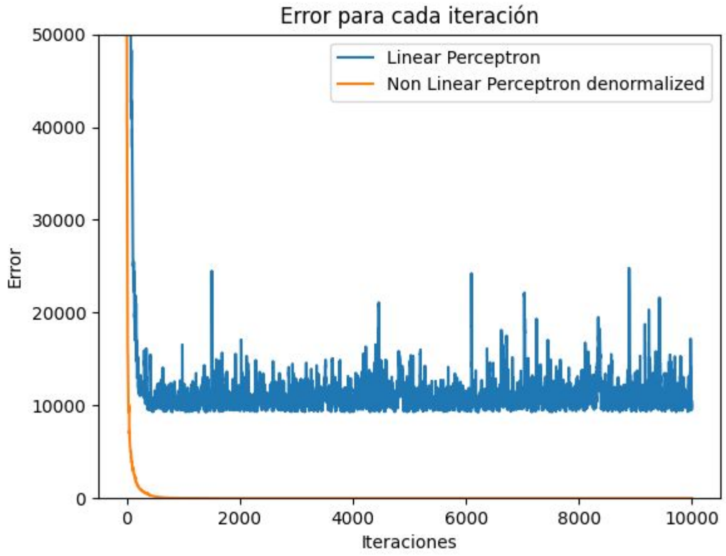
w mínimo = [5,966817 6,766640 7,506589 42,541762], error mínimo = 9267,3938

Perceptrón No Lineal:

w mínimo = [0,253289 0,253281 0,253326 -0,247009], error mínimo = 0,0011805,

error mínimo denormalizado = 2,9111777

Resultados



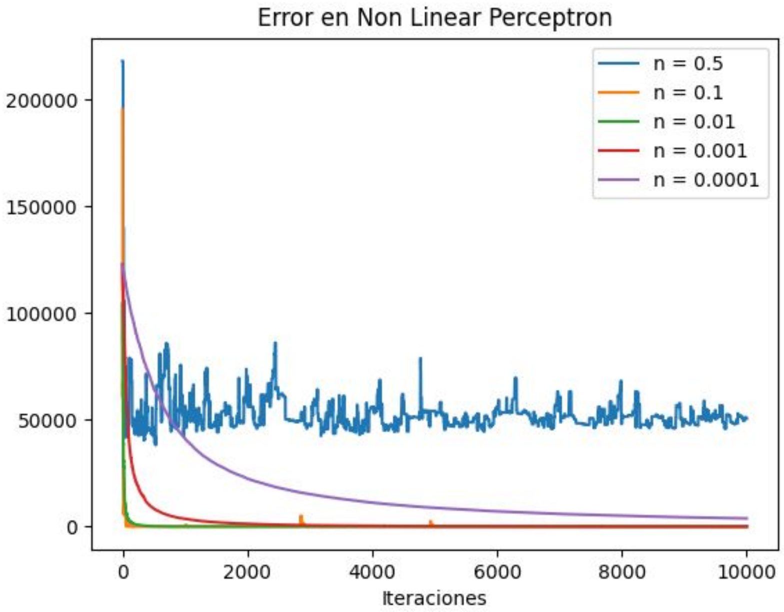
$\eta = 0{,}01 - \beta = 1$

Podemos apreciar que el perceptrón lineal **no aprende** el problema.

Se concluye que el mapeo de las entradas a las salidas **no se trata de una transformación lineal**.

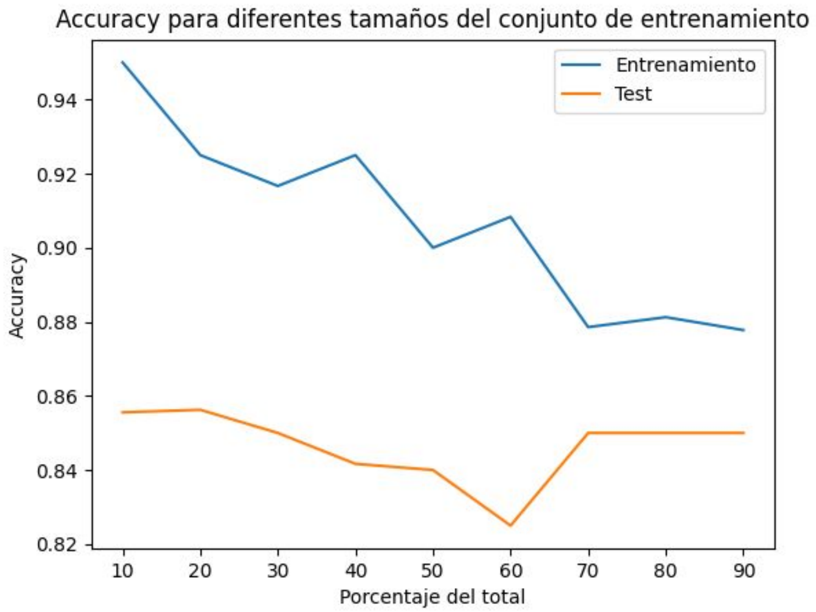
Por otro lado, el perceptrón no lineal **sí aprende** el problema, brindando una solución satisfactoria al mismo. Estudiaremos más a fondo el comportamiento de dicho perceptrón.

¿Qué impacto tiene η?



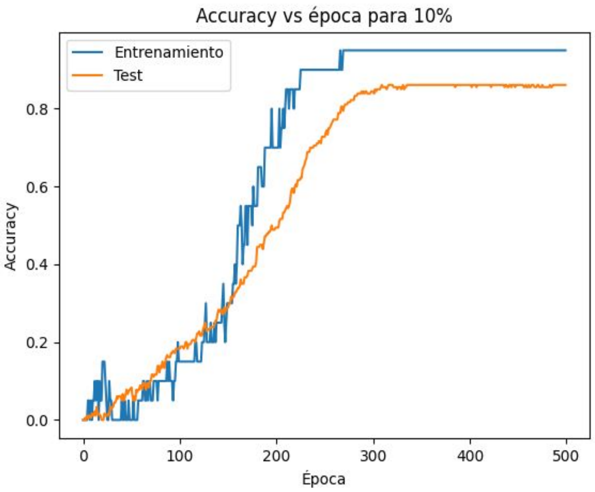
Conjunto de Entrenamiento vs Conjunto de Test

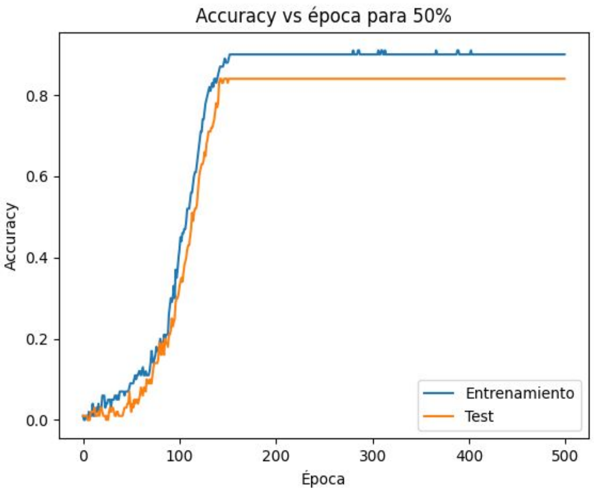
Separamos el conjunto de entrada en dos partes: **Conjunto de Entrenamiento** (con el que se entrena la red), y **Conjunto de Test**, con el que se evalúa la **capacidad de generalización** del perceptrón. Se estudia cuál sería el tamaño ideal del Conjunto de Entrenamiento.

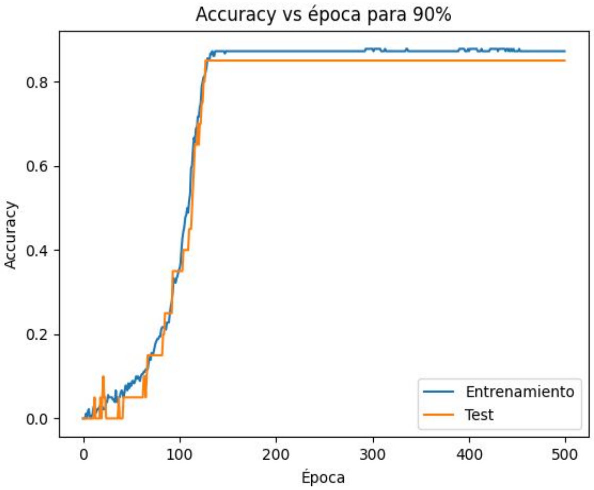


Accuracy = #hits / #datos de entrada

Si |salida deseada - salida obtenida| < 0,25 → hit!


Accuracy vs época para 10%
Entrenamiento
Test
Accuracy
Época
0.0
0.2
0.4
0.6
0.8
0
100
200
300
400
500


Accuracy vs época para 50%
Entrenamiento
Test
Accuracy
Época
0.0
0.2
0.4
0.6
0.8
0
100
200
300
400
500


Accuracy vs época para 90%
Entrenamiento
Test
Accuracy
Época
0.0
0.2
0.4
0.6
0.8
0
100
200
300
400
500

K-fold Cross Validation

Dividimos el conjunto de entrada en 5 subconjuntos de 40 elementos, viendo cuál da mejores resultados al ser elegido como Conjunto Test.

Conjunto	Accuracy
1	0,925
2	0,9
3	0,875
4	0,8
5	0,85

Conclusiones

- El problema se trata de una **transformación no lineal**, por lo que el perceptrón lineal es incapaz de aprenderlo, mientras que el no lineal se ajusta bien a él.
- El parámetro η repercute en la velocidad de convergencia a valores mínimos.
- El perceptrón muestra una buena **capacidad de generalización**, dado que predice correctamente la salida para entradas no pertenecientes al conjunto de entrenamiento.
- Los datos del problema muestran un buen equilibrio, posibilitando dicha generalización.

Ejercicio 3.2: Perceptrón multicapa - par o impar

¿Cómo evaluamos la capacidad de generalización?

Métricas ➔

- Accuracy
- Precision
- F1-Score
- Recall

- Accuracy: $\frac{TP+TN}{TP+TN+FN+FP}$. Los que están bien clasificados sobre todos.
- Precision: $\frac{TP}{TP+FP}$. Mide los positivos verdaderos sobre todos los que dieron positivo.
- Recall: $\frac{TP}{TP+FN}$. Mide los positivos verdaderos sobre todos los que son positivos.
- F_1-score: $\frac{2 * Precision * Recall}{Precision + Recall}$.

Separación de conjunto de entrenamiento y prueba

K-Fold Cross Validation

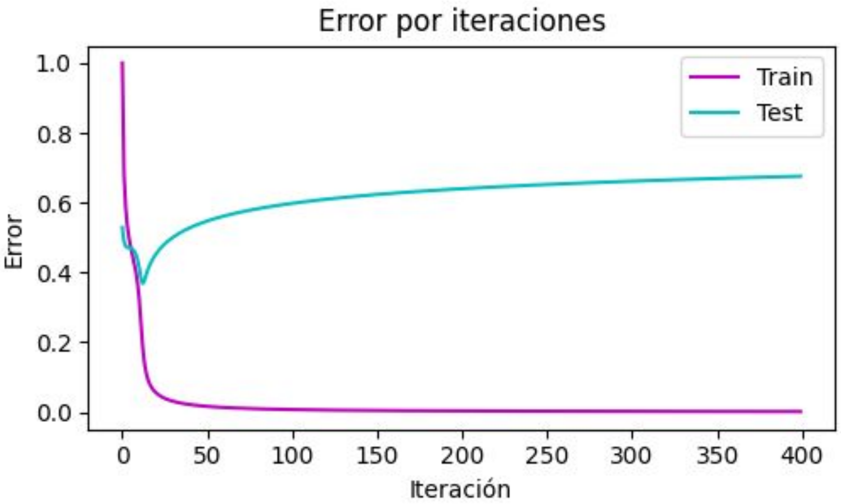
1) Se divide aleatoriamente el conjunto de entrenamiento en k partes de igual tamaño.
2) Se hacen k entrenamientos con sus correspondientes evaluaciones, en cada paso, se usa como test una de las partes y como entrenamiento las $k - 1$ restantes.
3) Para $j \in [1,K]$, se entrena con el j-ésimo conjunto, se prueba con el conjunto designado para probar, y se calculan las métricas correspondientes.
   a) Se elige el conjunto tal que dé el resultado más preciso

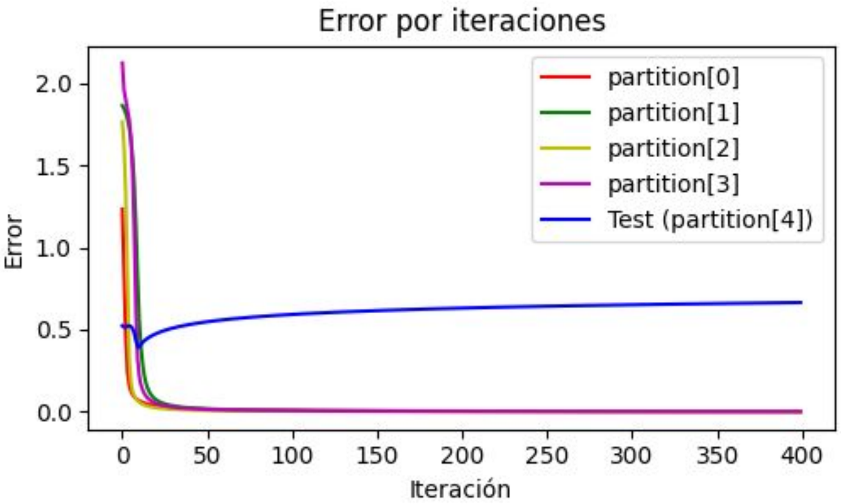
Resultados

$$\Theta(x) = tanh(\beta x)$$

Test = [8, 9]   Train = [0, 1], [2, 3], [4, 5], [6, 7]

η = 0.01, β = 0,5



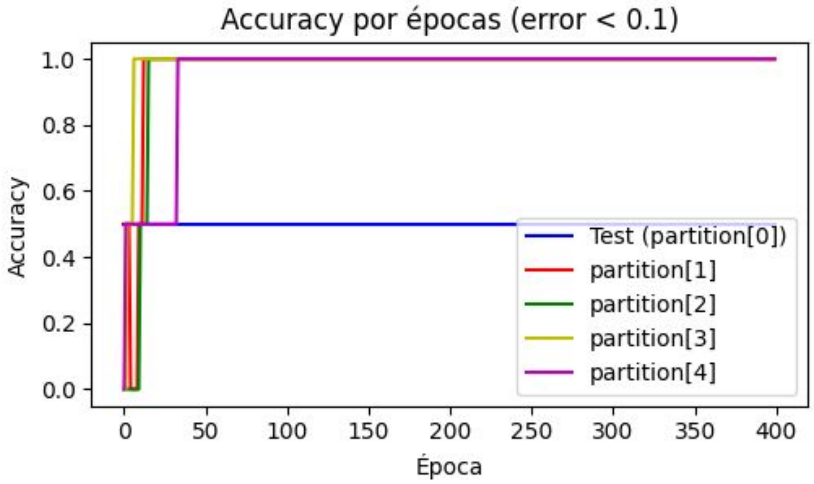


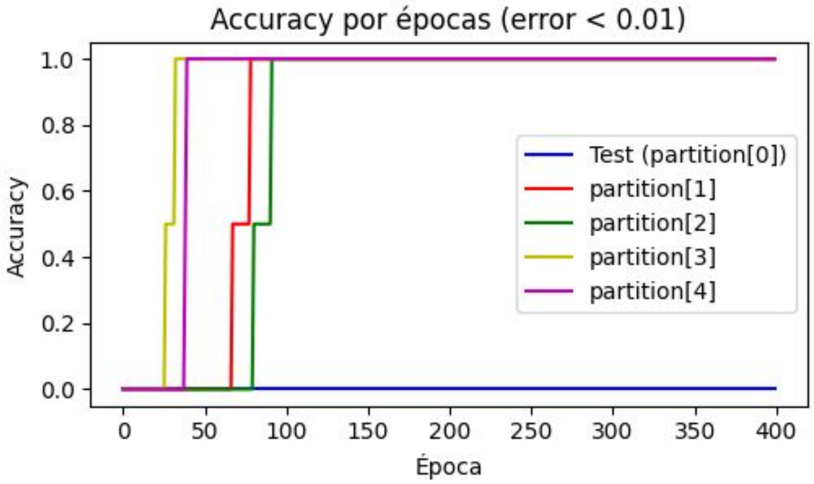
Resultados

$$\Theta(x) = tanh(\beta x)$$

Test = [0, 1]   Train = [2, 3], [4, 5], [6, 7], [8, 9]

η = 0.01, β = 0,5



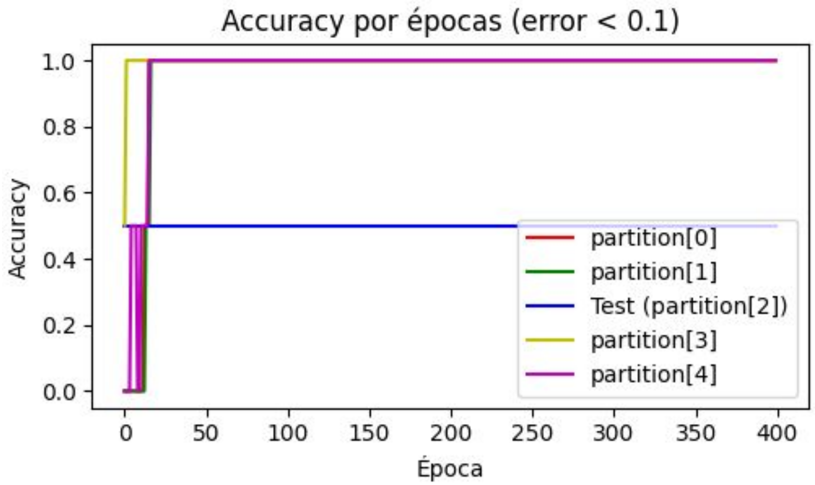


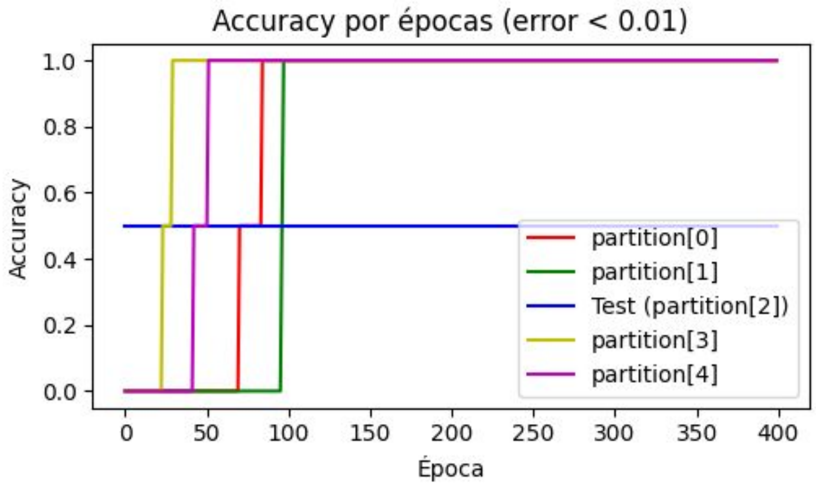
Resultados

$$\Theta(x) = tanh(\beta x)$$

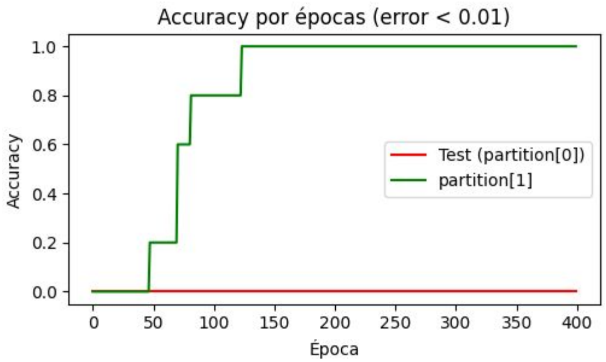
Test = [4, 5]   Train = [0, 1], [2, 3], [6, 7], [8, 9]

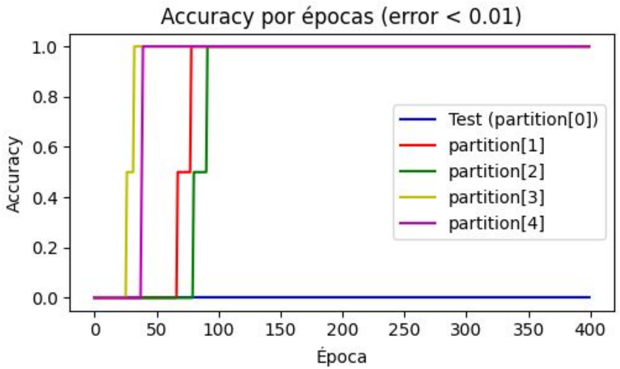
η = 0.01, β = 0,5

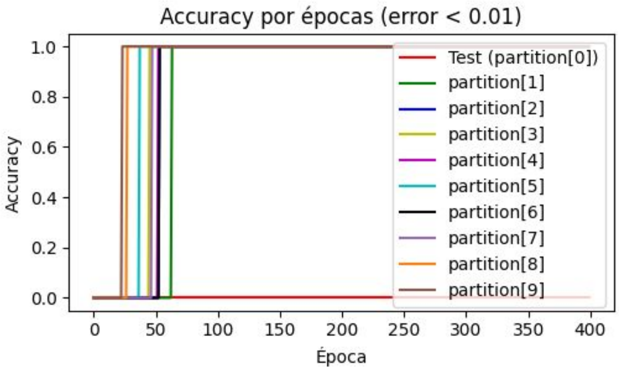




Resultados







$\Theta(x) = tanh(\beta x)$ η = 0.01, β = 0,5

Conclusiones

- La cota de error exigida tiene una influencia **directa** sobre la **accuracy** y el **error**
  - Con un epsilon de 0.01, la red tiene dificultades notorias para aprender el problema, mientras que con 0.1 muestra mayores niveles de aprendizaje
- La red **no posee** capacidad de generalización. Tras haber intentado con distintas subdivisiones de tamaño k del conjunto de entrenamiento y testeo mediante la metodología para reducir el sobreajuste conocida como “K-Fold Cross Validation”, pudimos observar que no reconoce los patrones del conjunto de testeo, a pesar de que los de entrenamiento los reconoce perfecto.