SIA

TP 3: Perceptrón Simple y Multicapa

GRUPO 1:

SOL VICTORIA ANSELMO

JULIÁN SASSO

AGUSTÍN MATTIUSSI

CAMILA SIERRA PÉREZ

IAN JAMES ARNOTT

JUAN ADOLFO ROSAUER HERRMANN

11:11PM

Ejercicio 1

Perceptrón simple con función de activación escalón

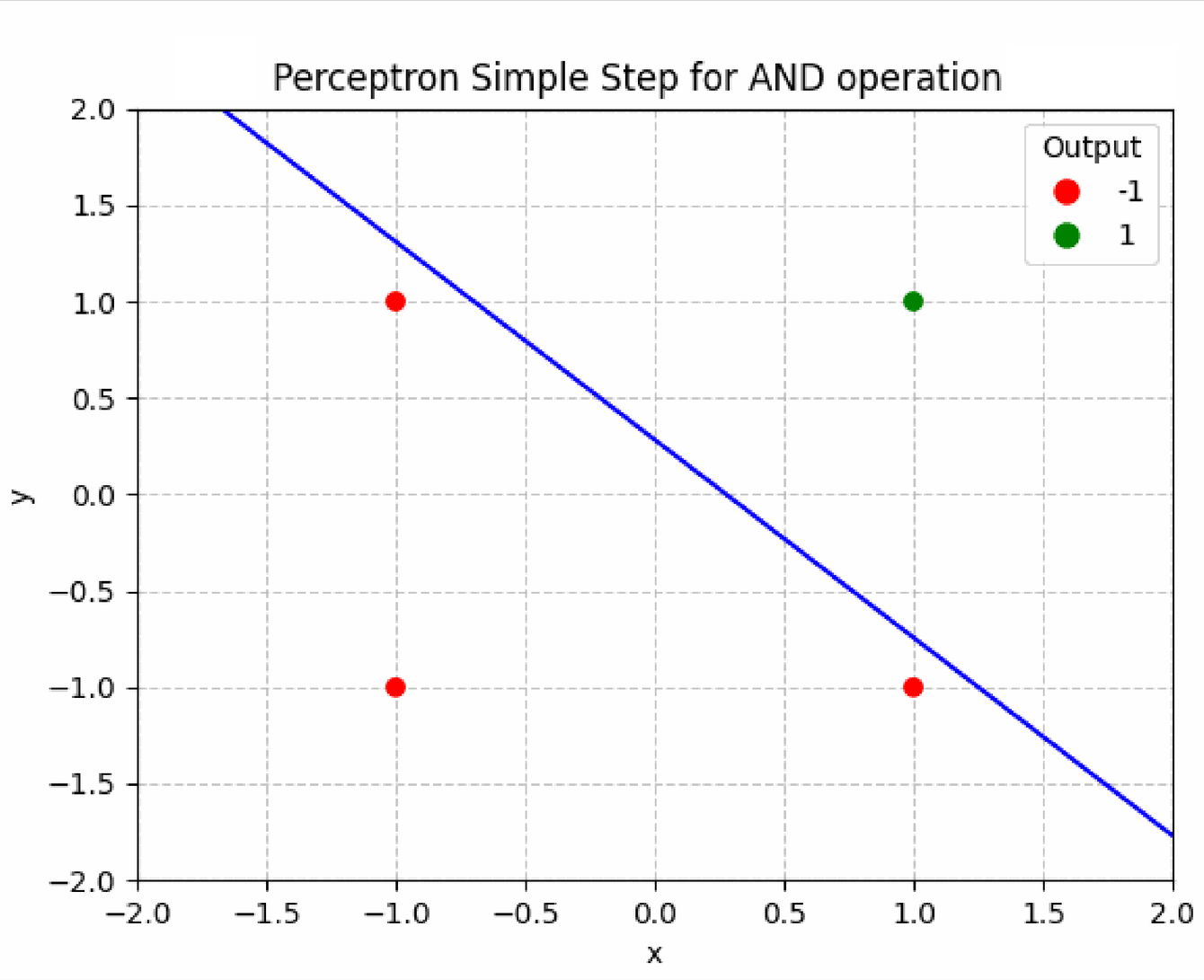
Funciones a implementar

-Operación lógica AND

-Operación lógica XOR



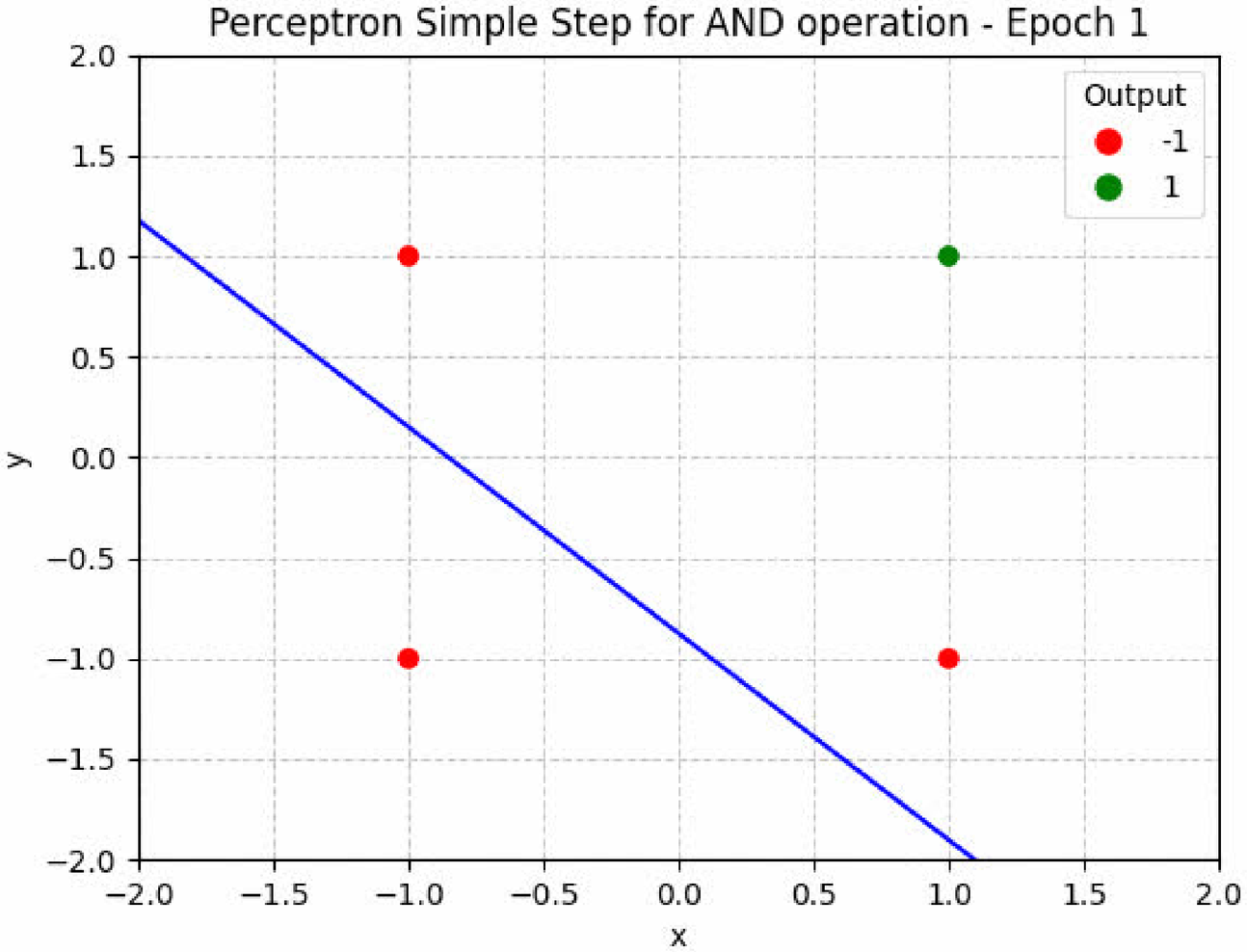
Operación AND



Cantidad de épocas: 500

Learning Rate: 0.1

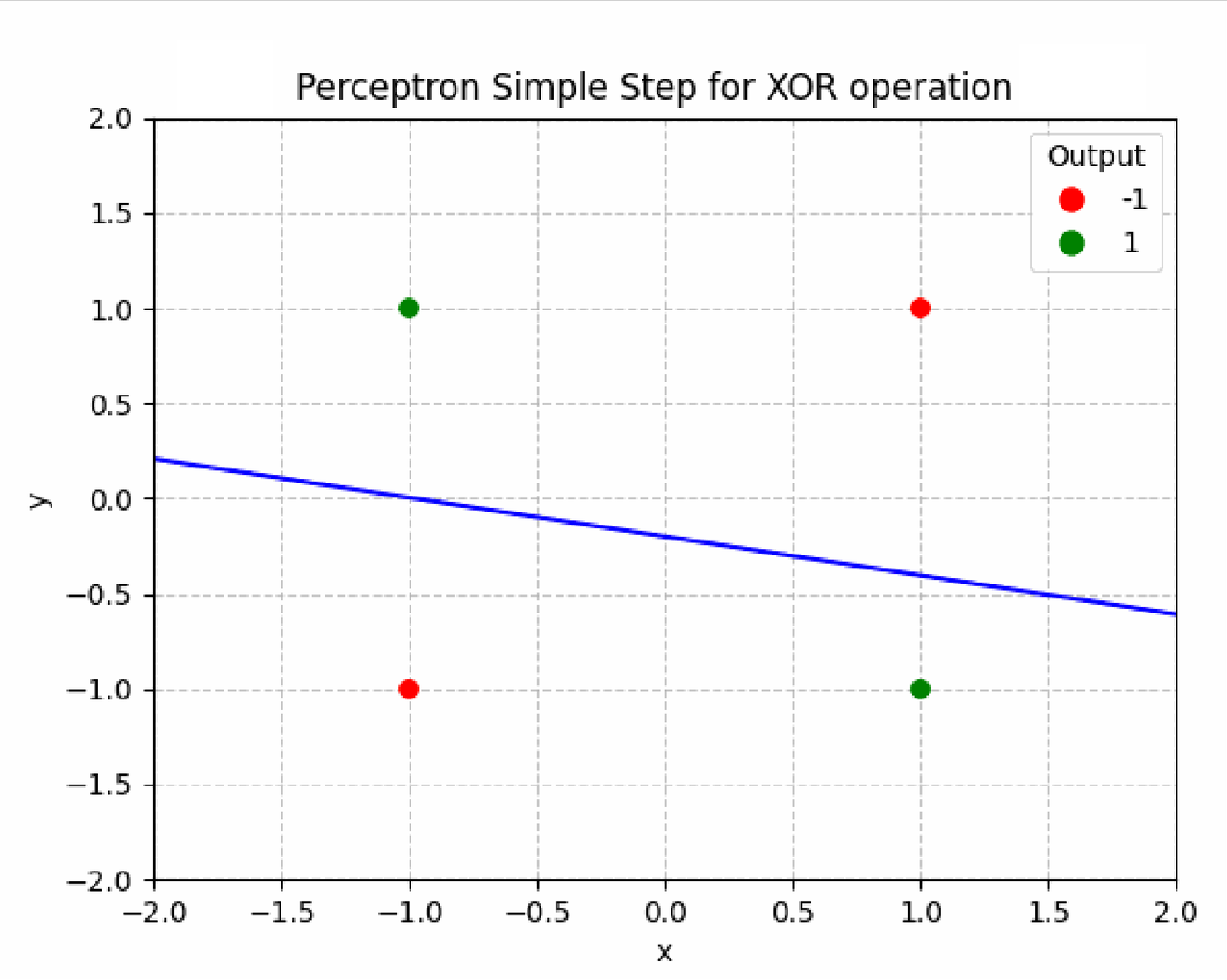
Operación AND



Cantidad de épocas: 500

Learning Rate: 0.1

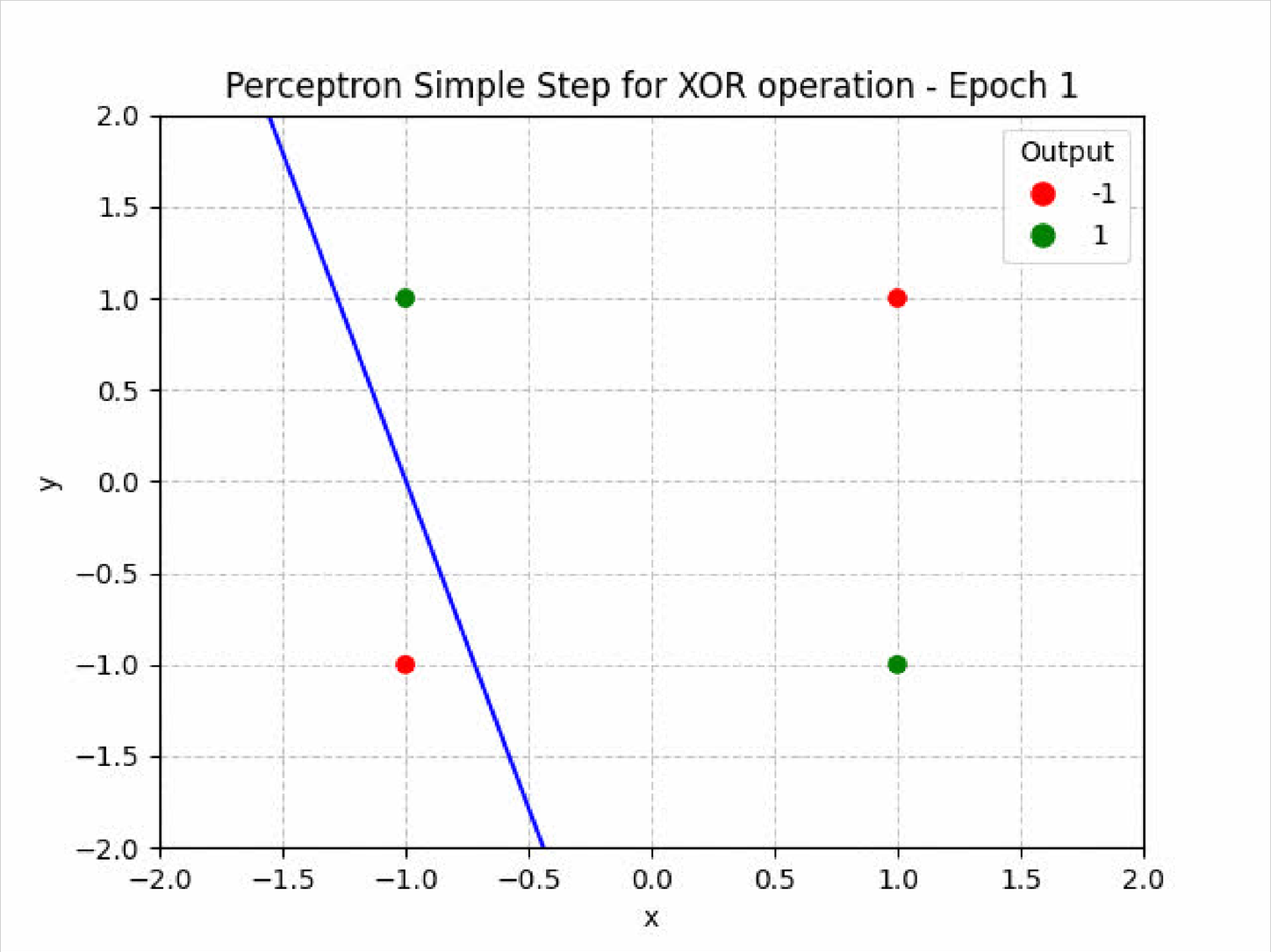
Operación XOR



Cantidad de épocas: 500

Learning Rate: 0.1

Operación XOR



Cantidad de épocas: 500

Learning Rate: 0.1

Conclusiones

El perceptrón simple escalón:

- Encuentra un hiperplano que separa las dos clases en la operación “AND” ya que las coordenadas son **linealmente separables**.

- No encuentra un hiperplano que separe bien las dos clases en la operación “XOR” ya que los coordenadas **NO son linealmente separables.**

Ejercicio 2

Perceptrón simple lineal y perceptrón simple no lineal

Objetivos

- Evaluar capacidad de aprendizaje
- Evaluar capacidad de generalización

11:11PM

Consideraciones

Para los perceptrones no lineales las imágenes de las funciones están acotadas:

- (-1;1) para tanh
- (0;1) para logística

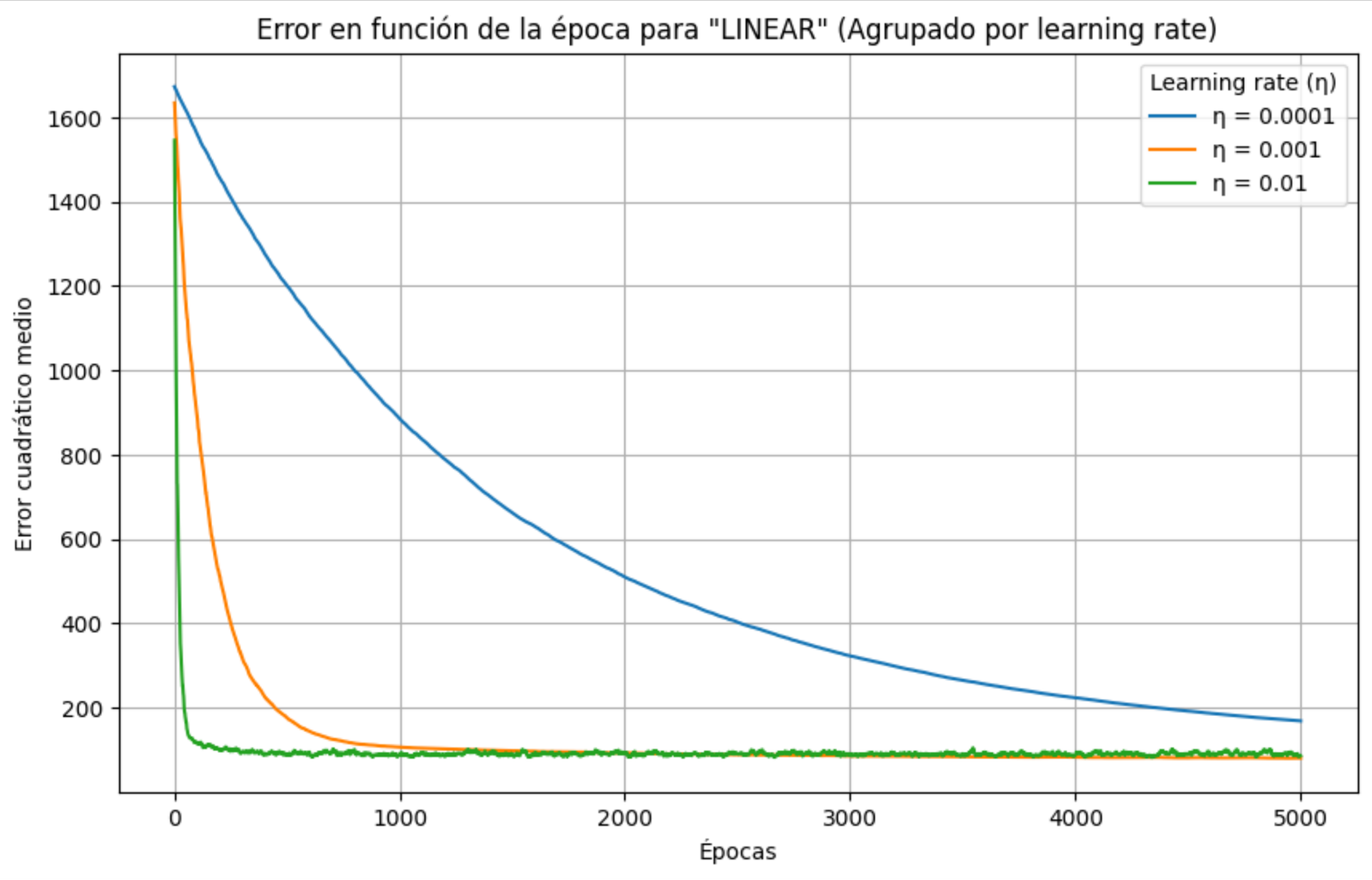
Como las salidas esperadas de nuestros perceptrones se encuentran en todos los reales, se normalizó el valor de θ con el metodo Min-Max Feature Scaling.

SIA

VEAMOS EL ERROR EN FUNCIÓN DE LA ÉPOCA AGRUPANDO POR LEARNING RATE

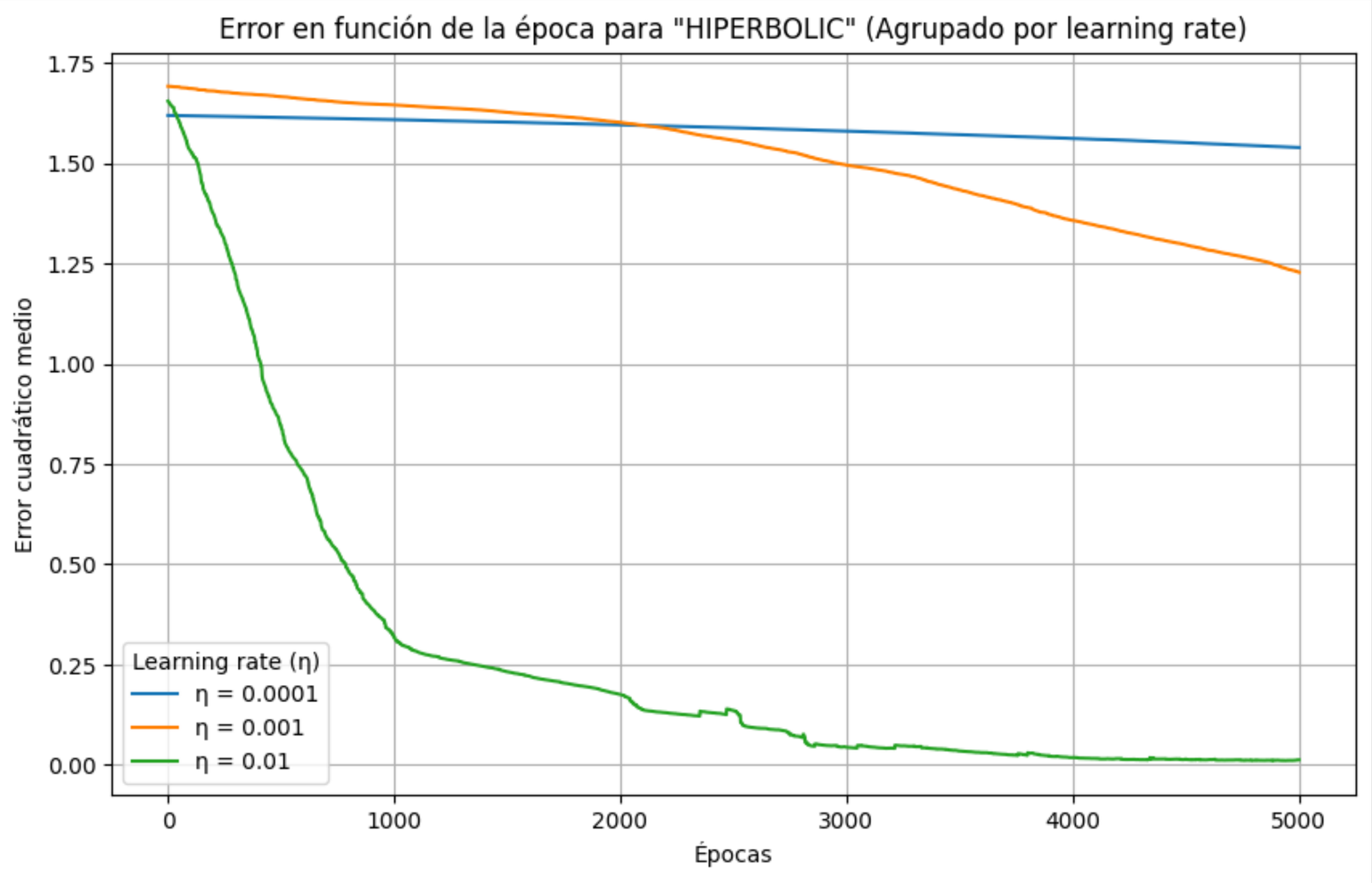
11:11PM

Perceptrón Lineal



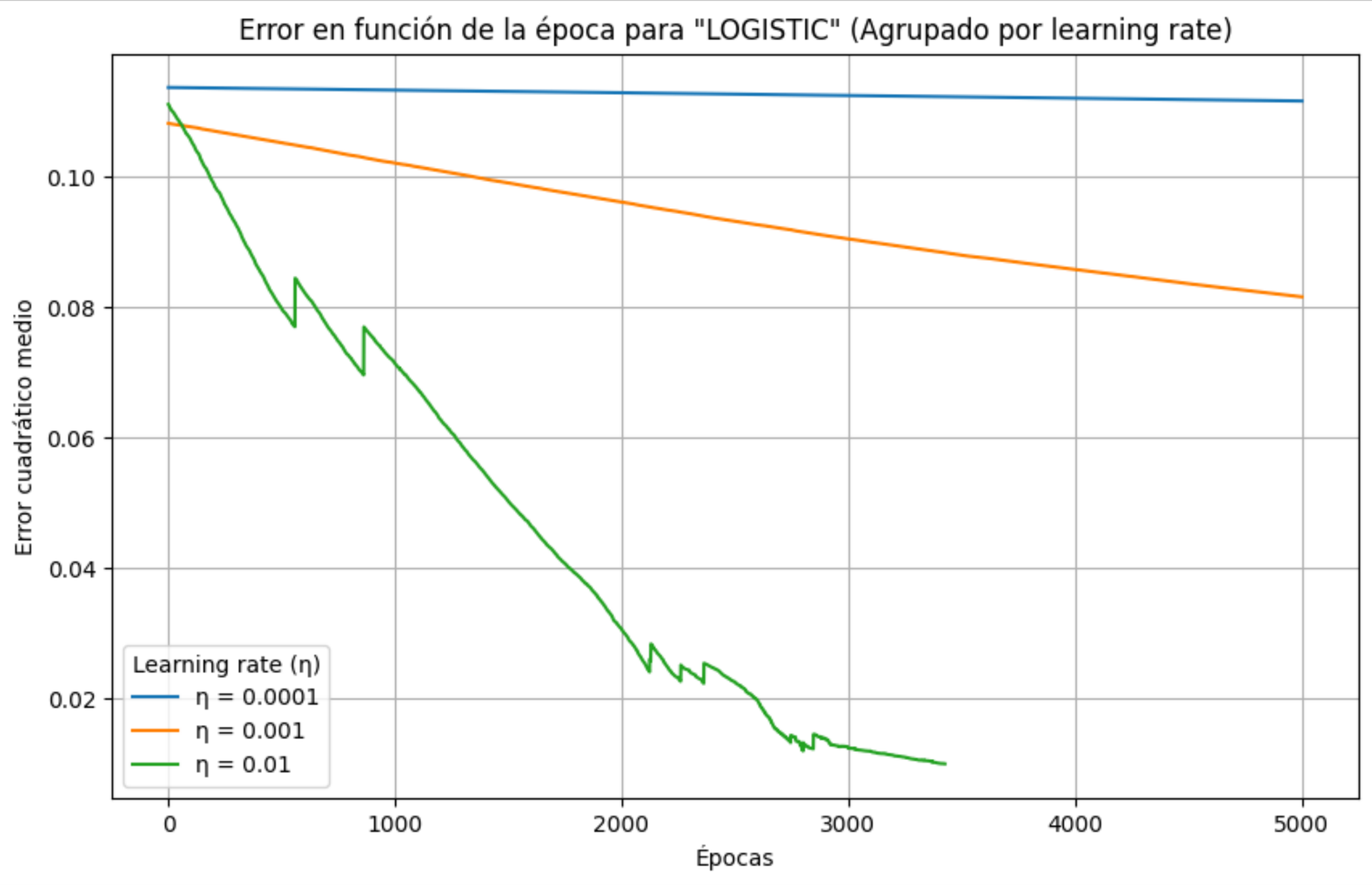
Training Percentage: 0.8
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Perceptrón No Lineal



HYPERBOLIC
Training percentage: 0.8
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Perceptrón No Lineal



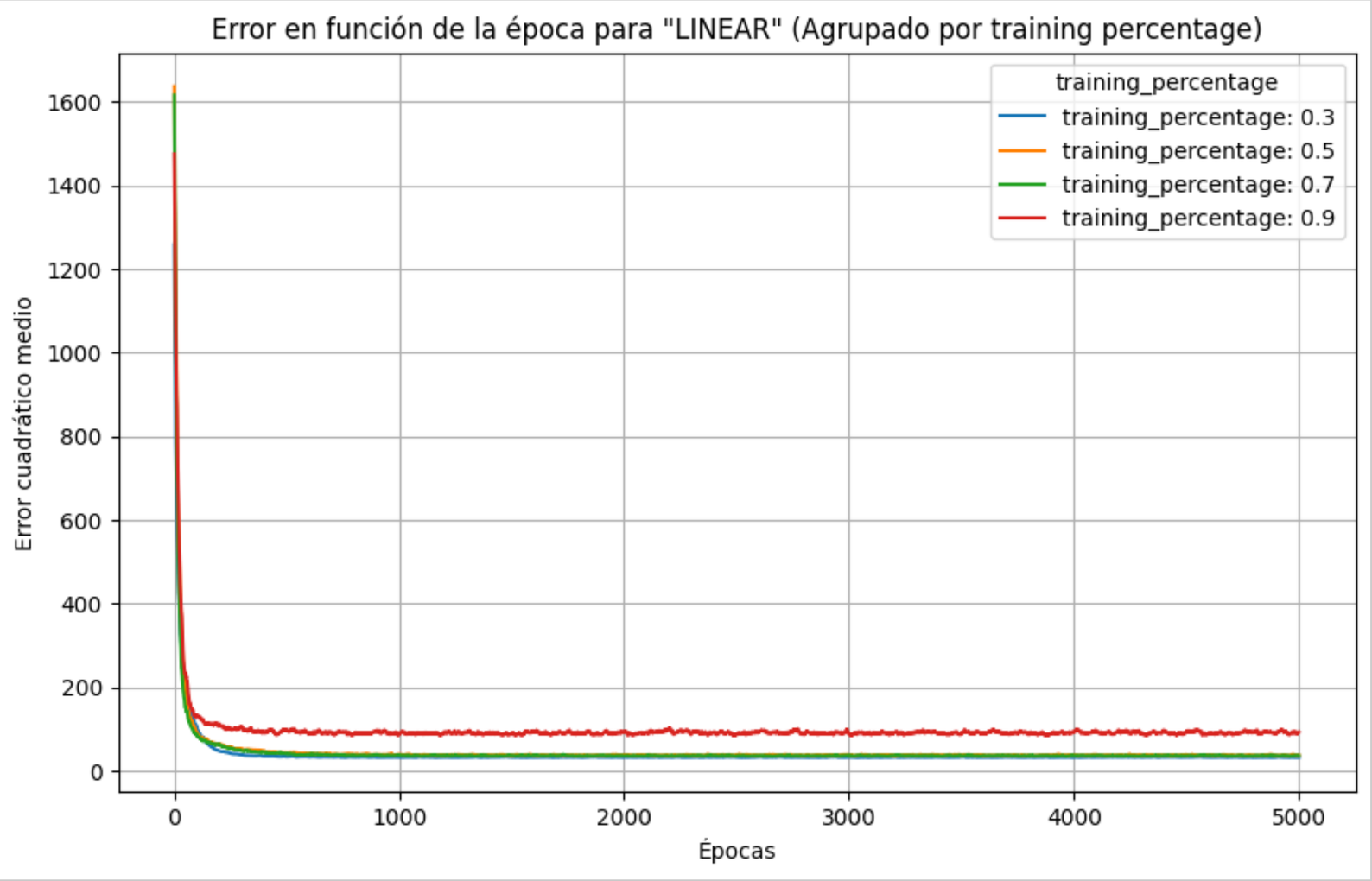
LOGISTIC
Training percentage: 0.8
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

SIA

VEAMOS EL ERROR EN FUNCIÓN DE LA ÉPOCA AGRUPANDO POR TRAINING PERCENTAGE

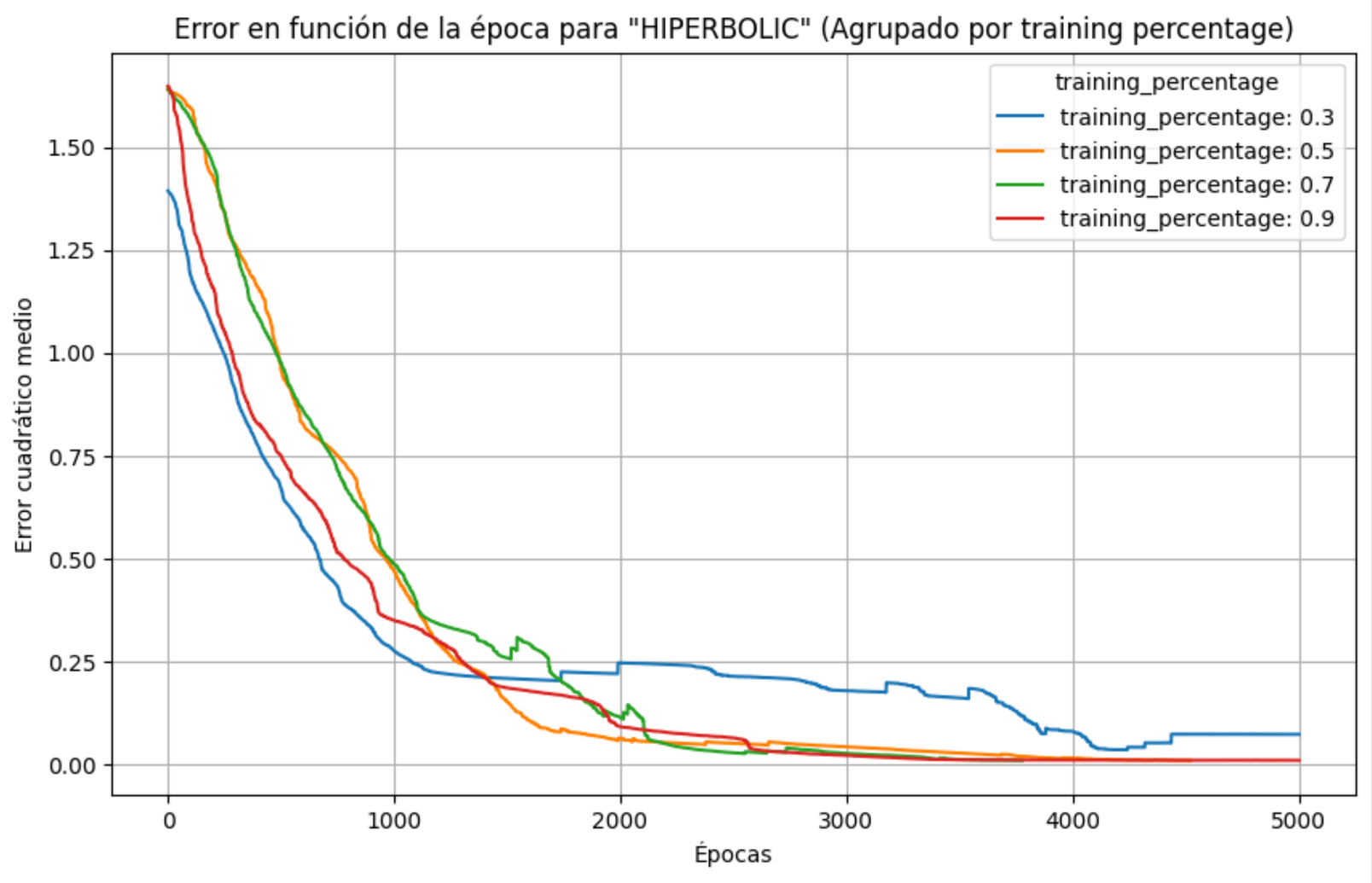
11:11PM

Perceptrón Lineal



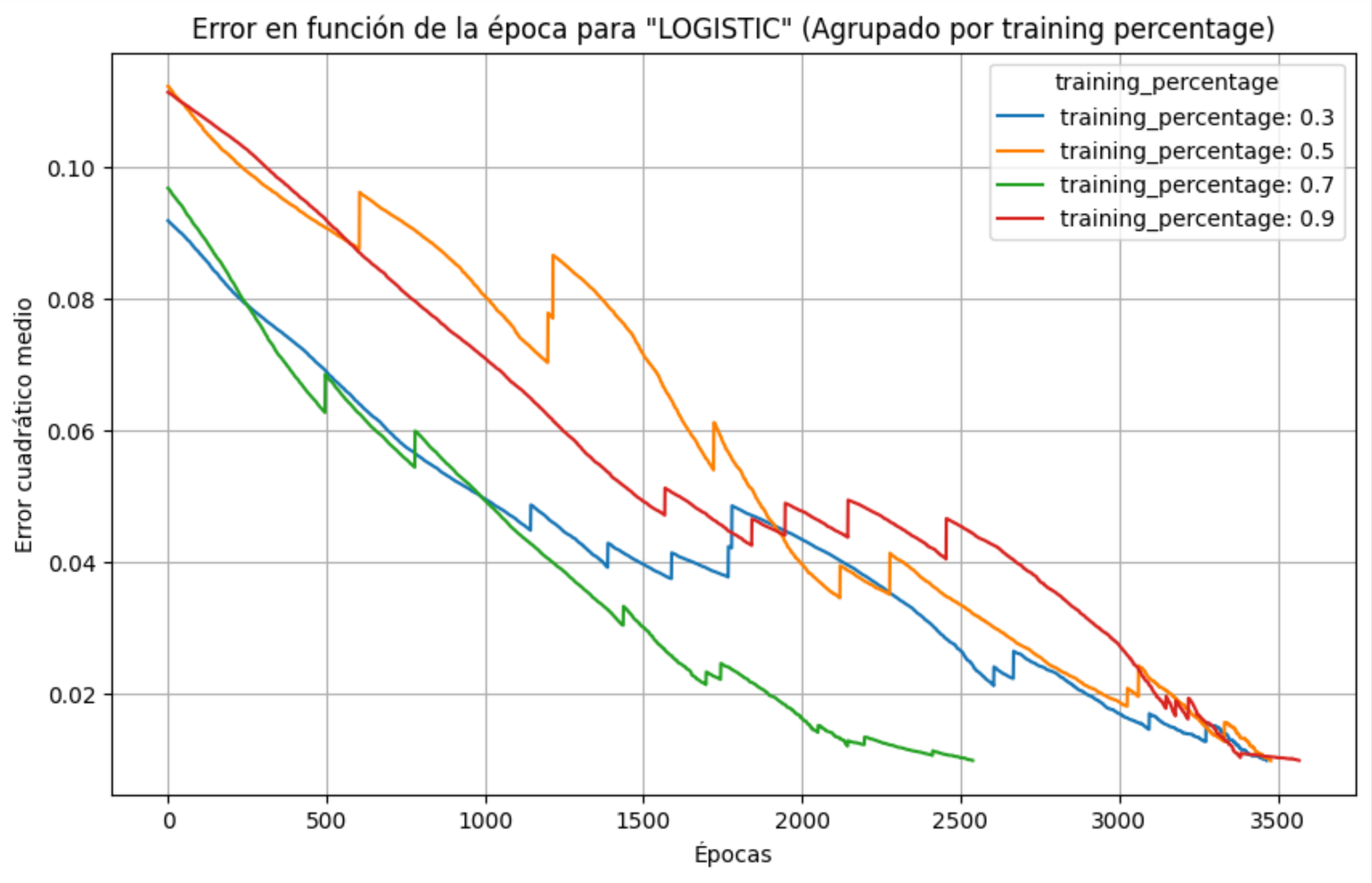
Learning Rate: 0.01
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Perceptrón No Lineal



HYPERBOLIC
Learning Rate: 0.01
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Perceptrón No Lineal



LOGISTIC
Learning Rate: 0.01
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01



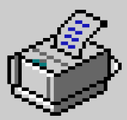





SIA

VEAMOS EL ERROR EN FUNCIÓN DE LA ÉPOCA AGRUPANDO POR BETA

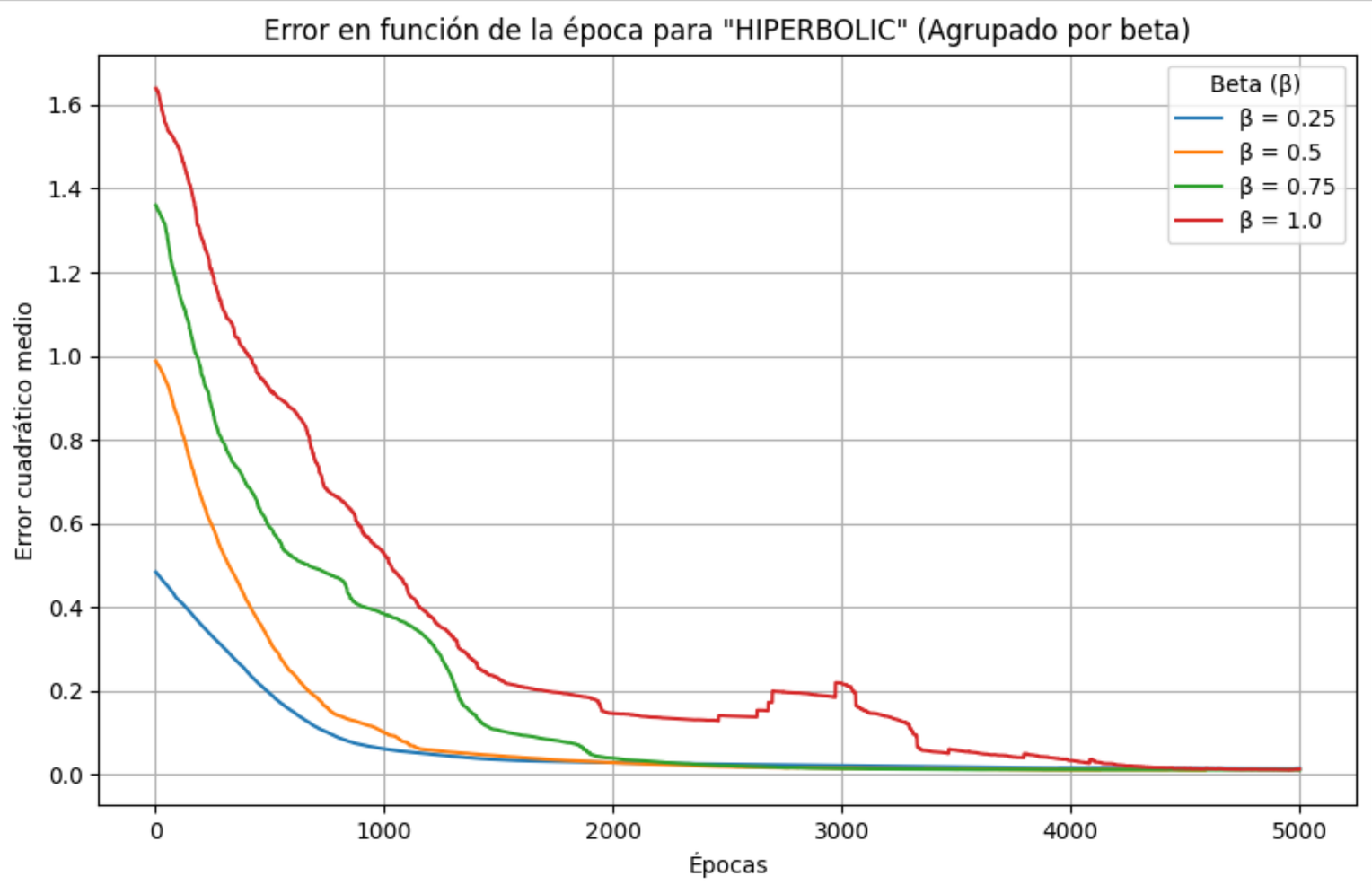






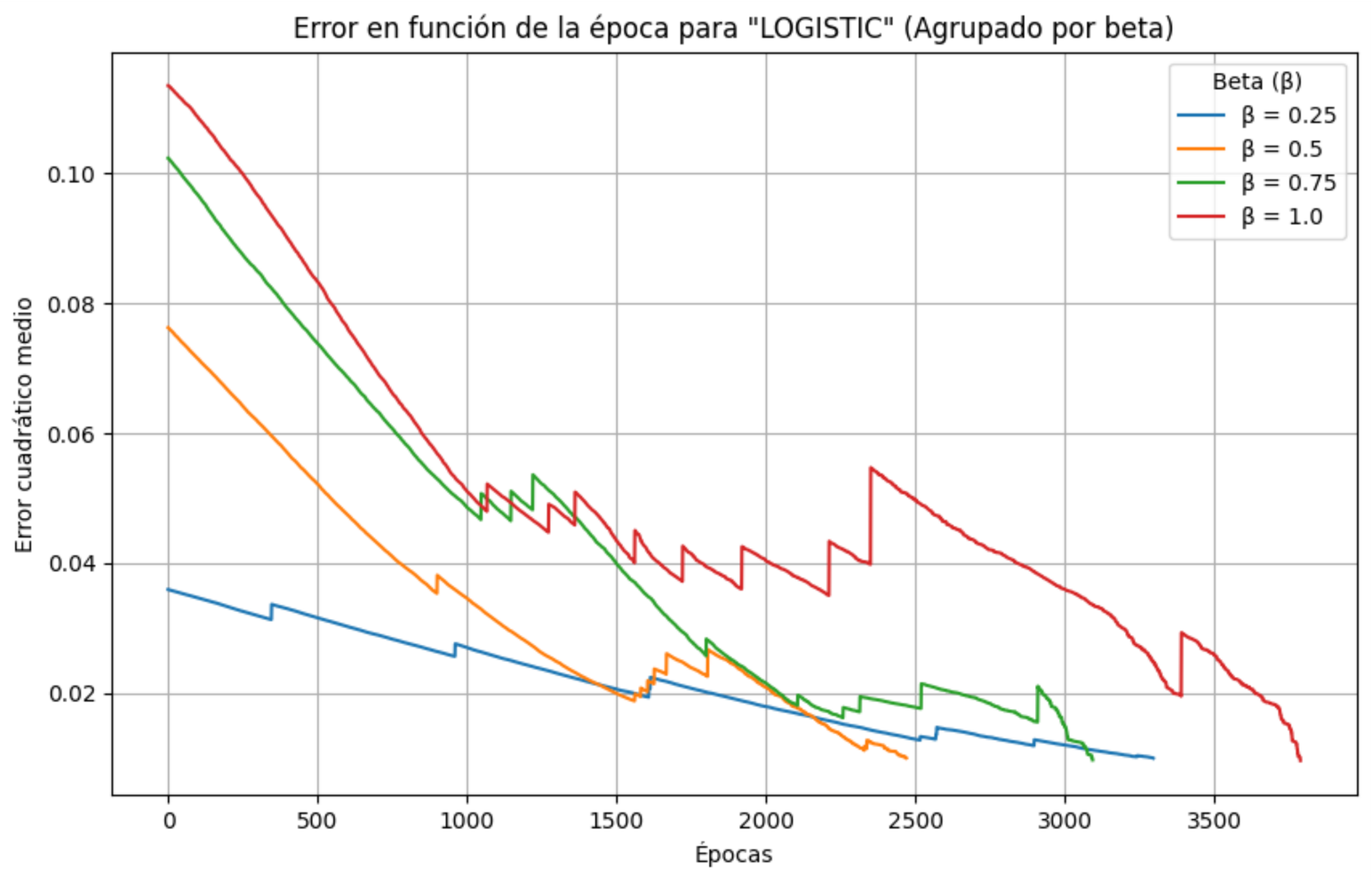
11:11PM

Perceptrón No Lineal



HYPERBOLIC
Learning Rate: 0.01
Training percentage: 0.8
Epochs: 5000
Bias: 1
Epsilon: 0.01

Perceptrón No Lineal



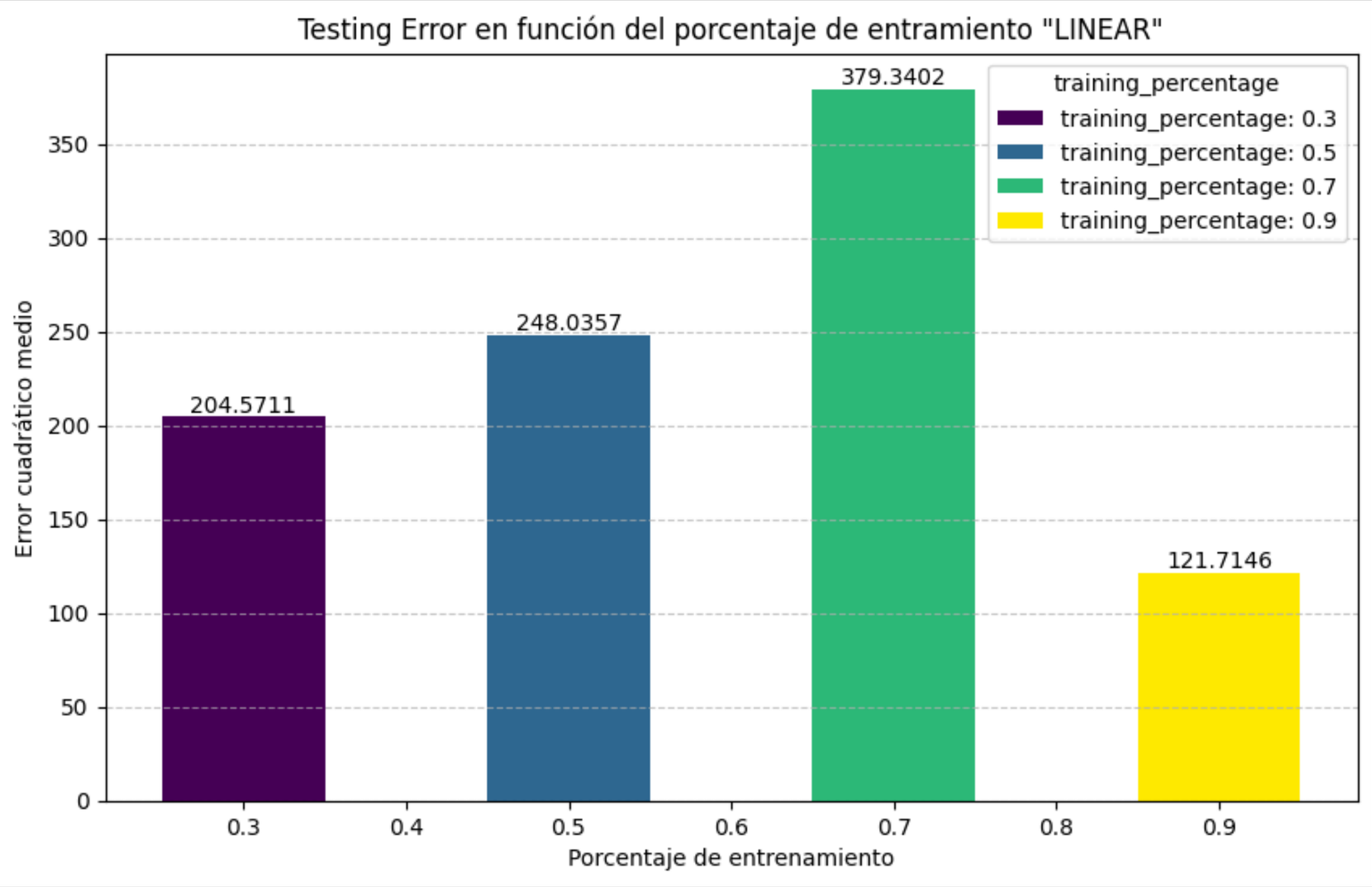
LOGISTIC
Learning Rate: 0.01
Training percentage: 0.8
Epochs: 5000
Bias: 1
Epsilon: 0.01

SIA

VEAMOS EL ERROR DE TESTEO EN FUNCIÓN DEL TRAINING PERCENTAGE

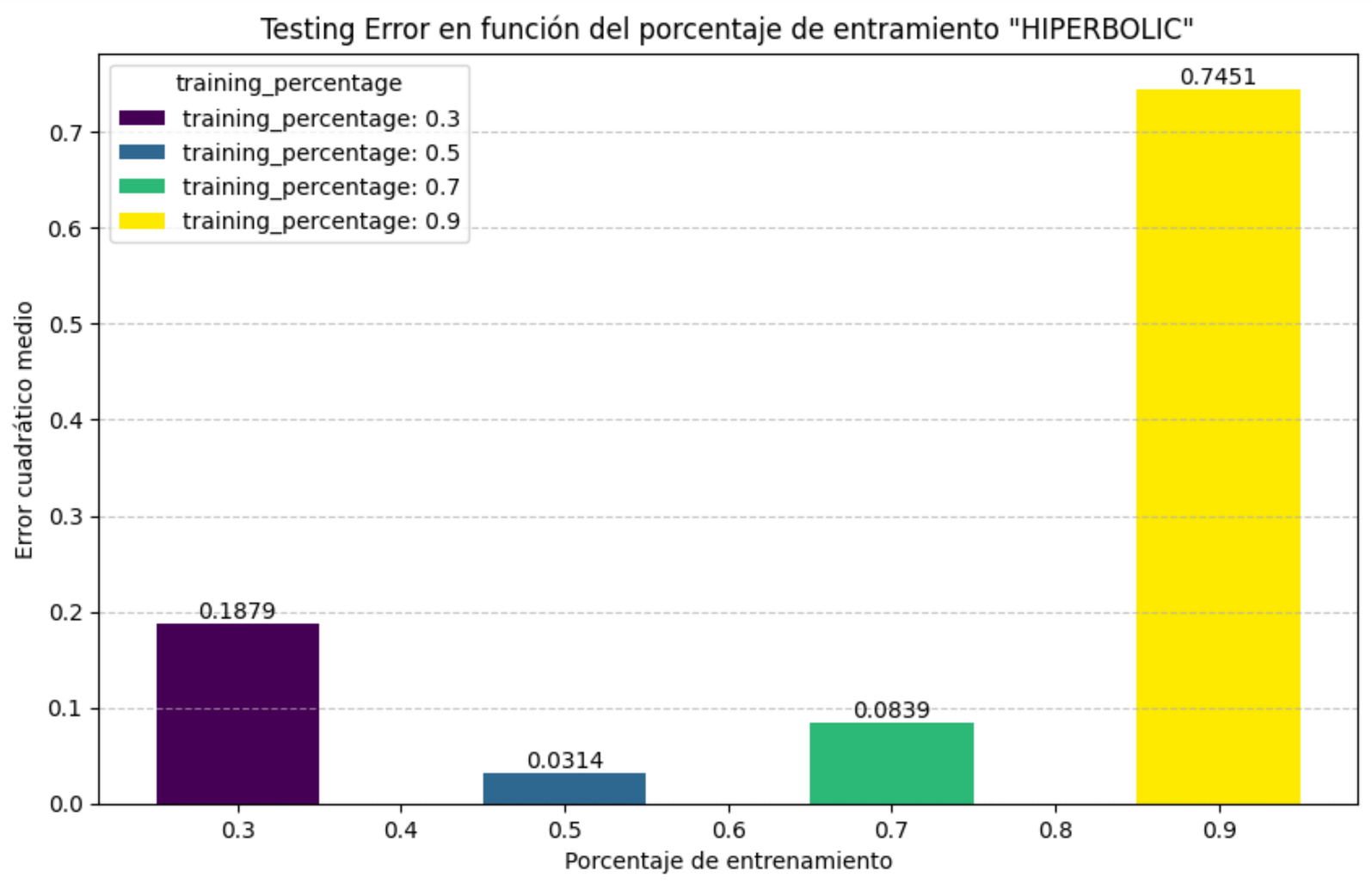
11:11PM

Perceptrón Lineal



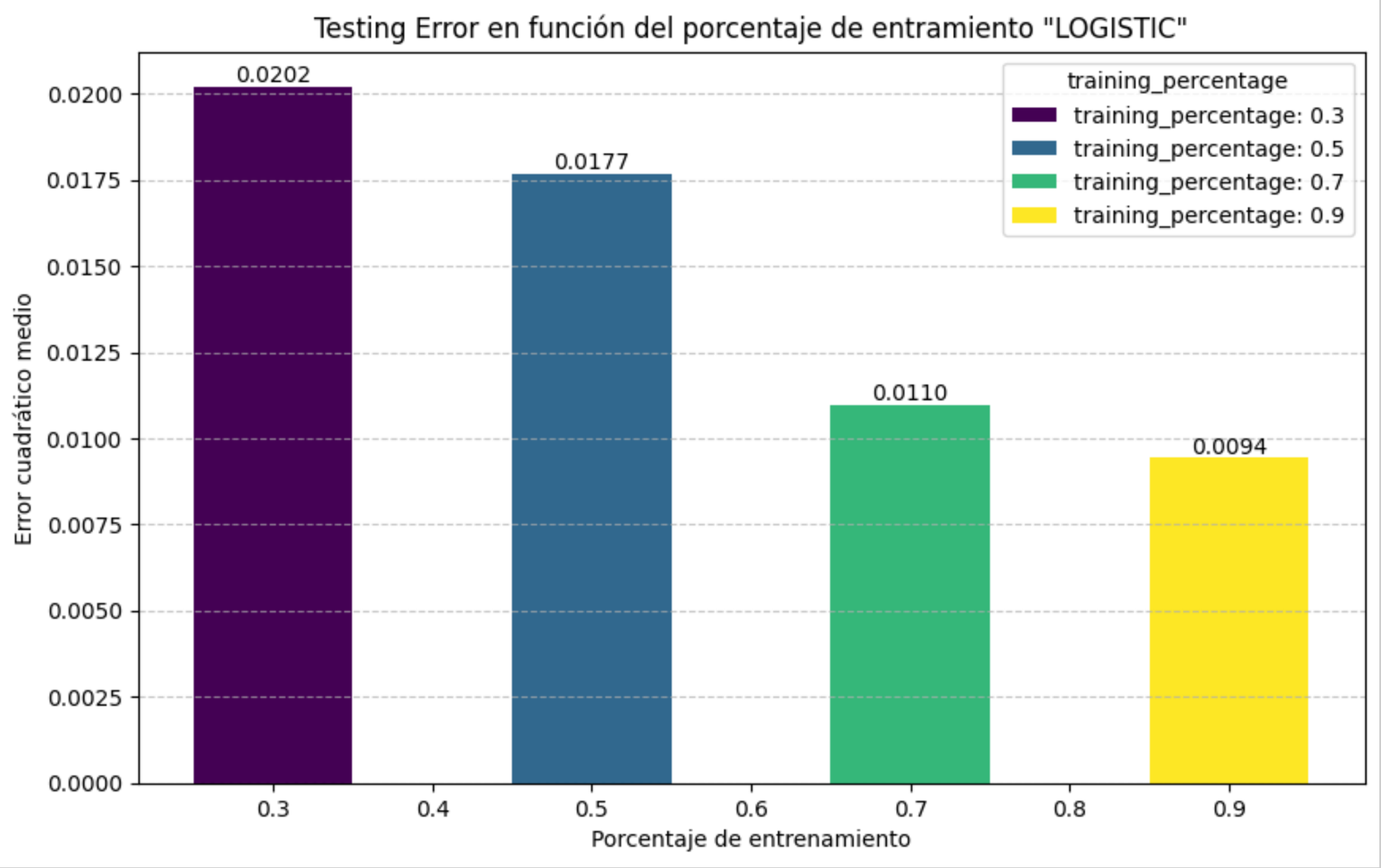
Learning Rate: 0.01
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Perceptrón No Lineal



HYPERBOLIC
Learning Rate: 0.01
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Perceptrón No Lineal



LOGISTIC
Learning Rate: 0.01
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.01

Conclusiones

- No hay una receta para elegir la mejor combinación de parámetros para una red neuronal.
  - Depende mucho del problema y del conjunto de datos de entrenamiento que se le pase al perceptrón

- Los dataset para aprendizaje y testeo tiene una gran influencia en el entrenamiento del perceptrón.
  - Un buen dataset, diverso y representativo, resultará en mejores resultados.
  - Con porcentajes muy bajos o muy altos se produce una pérdida de capacidad de generalización del perceptrón.

Perceptron Lineal vs Perceptron No lineal

- ¿Cómo se diferencia la capacidad de generalización para perceptrones lineales vs. no lineales?
  - El perceptrón lineal es más “naive”, pues intenta ajustarse utilizando una función de activación lineal, cuando en la mayoría de los problemas reales los datos están dispersos y no se encuentran alineados.
  - Los perceptrones no lineales son capaces de ajustar sus pesos siguiendo los valores de funciones más “flexibles” (sigmoideales) como lo son la tangente hiperbólica y la logística, lo que permite una mayor adaptabilidad frente a los datos de testing, que no estuvieron presentes durante su entrenamiento.

Ejercicio 3

Perceptrón multicapa

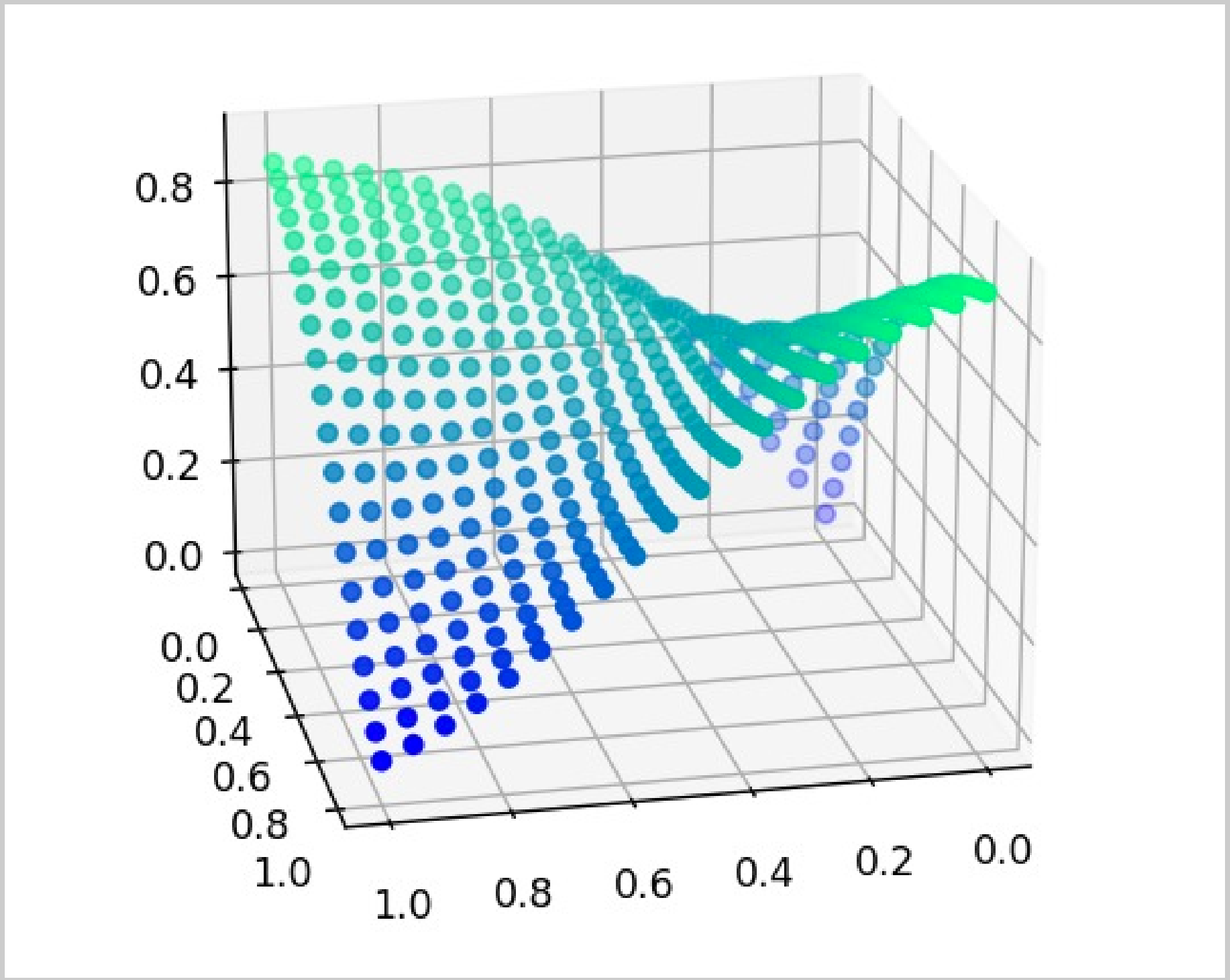
Funciones a implementar

- Función lógica “O exclusivo”
- Discriminación de números pares
- Determinar si el dígito se corresponde con la entrada a la red



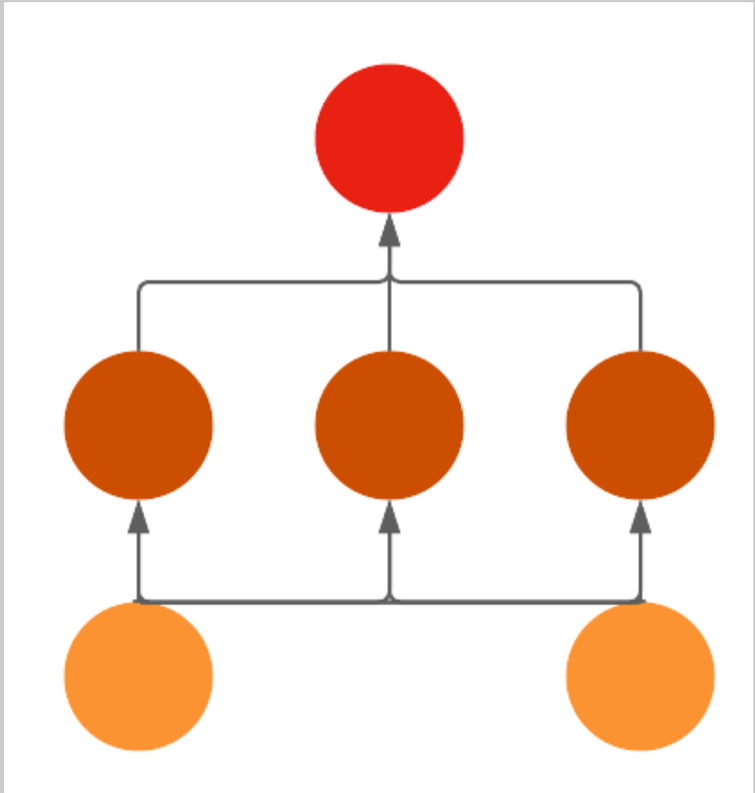


Perceptrón Multicapa

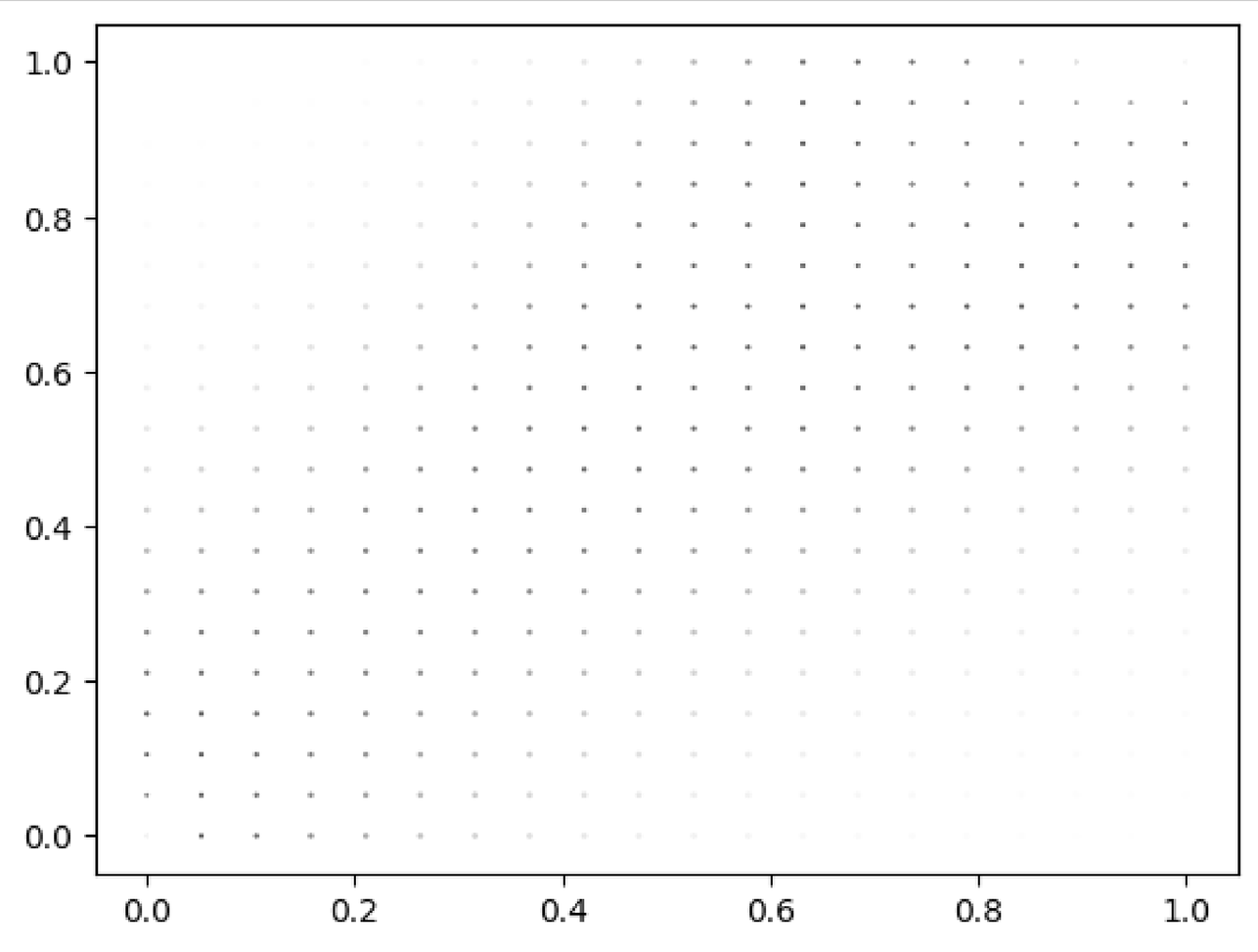


FUNCIÓN XOR

AHORA SE RESUELVE

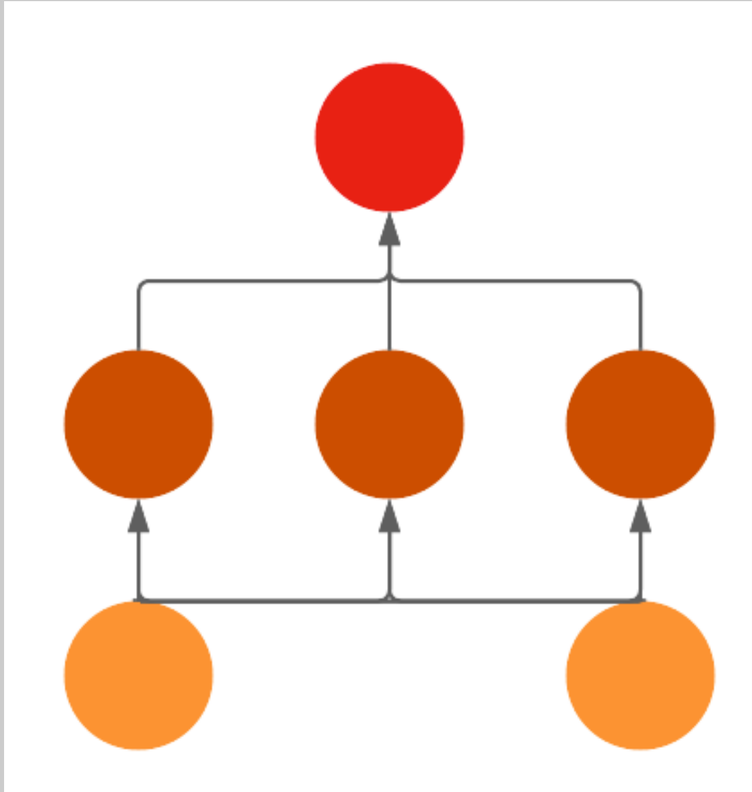


Perceptrón Multicapa



FUNCIÓN XOR

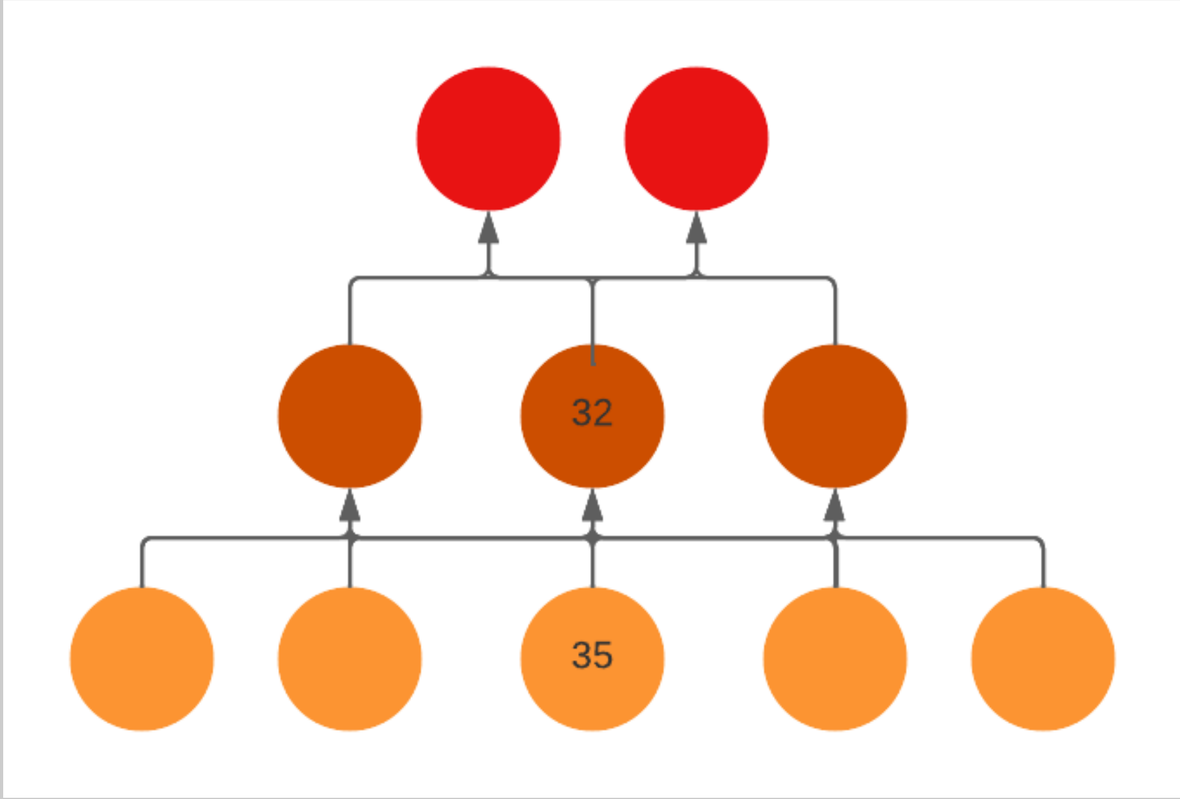
AHORA SE RESUELVE



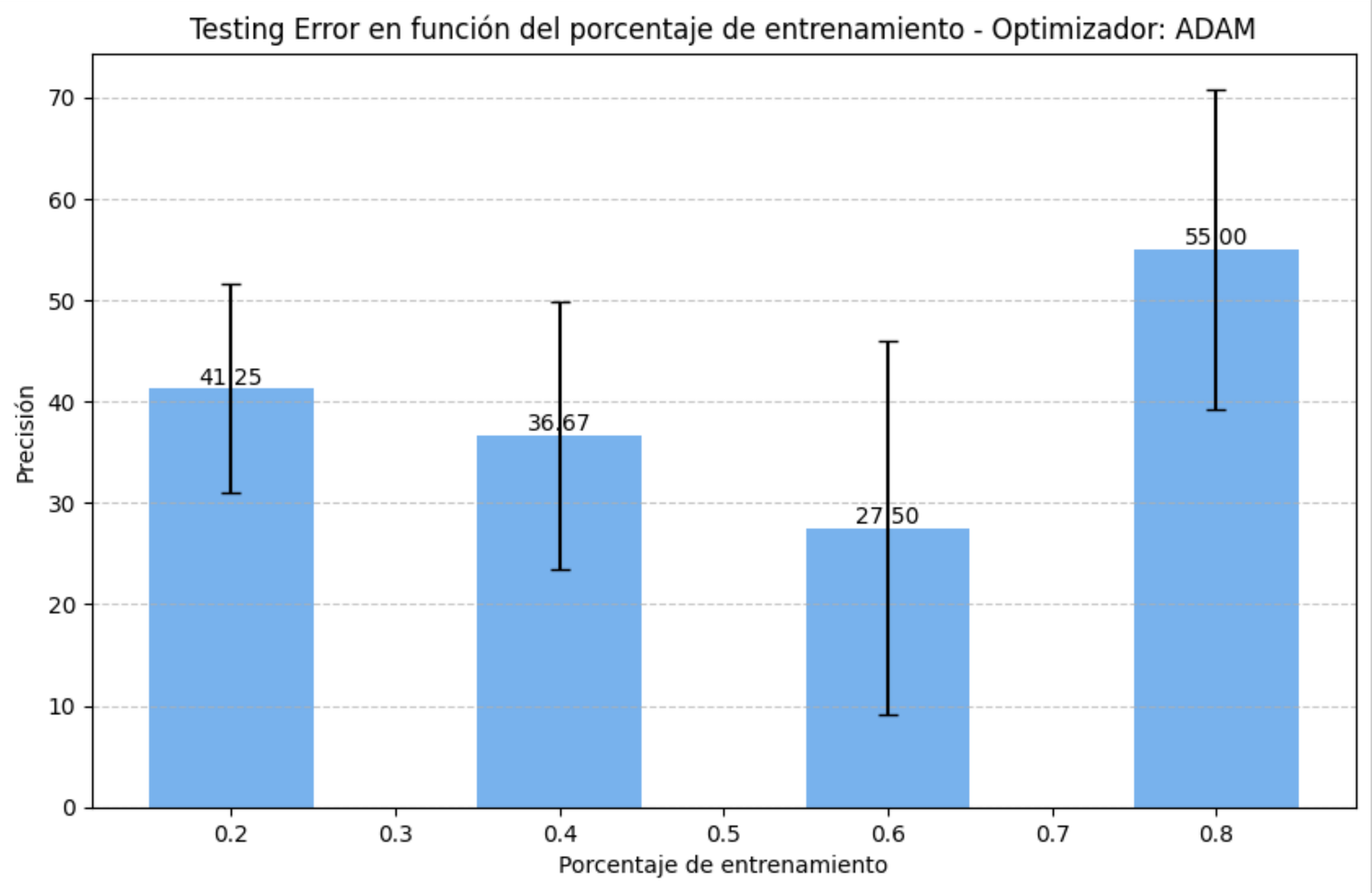
Perceptrón Multicapa

DISCRIMINACIÓN DE NÚMEROS PAR

Learning Rate: 0.01
Training Percentage: 0.5
Epochs: 5000
Bias: 1
Beta: 1
Epsilon: 0.5
Activation: Sigmoid



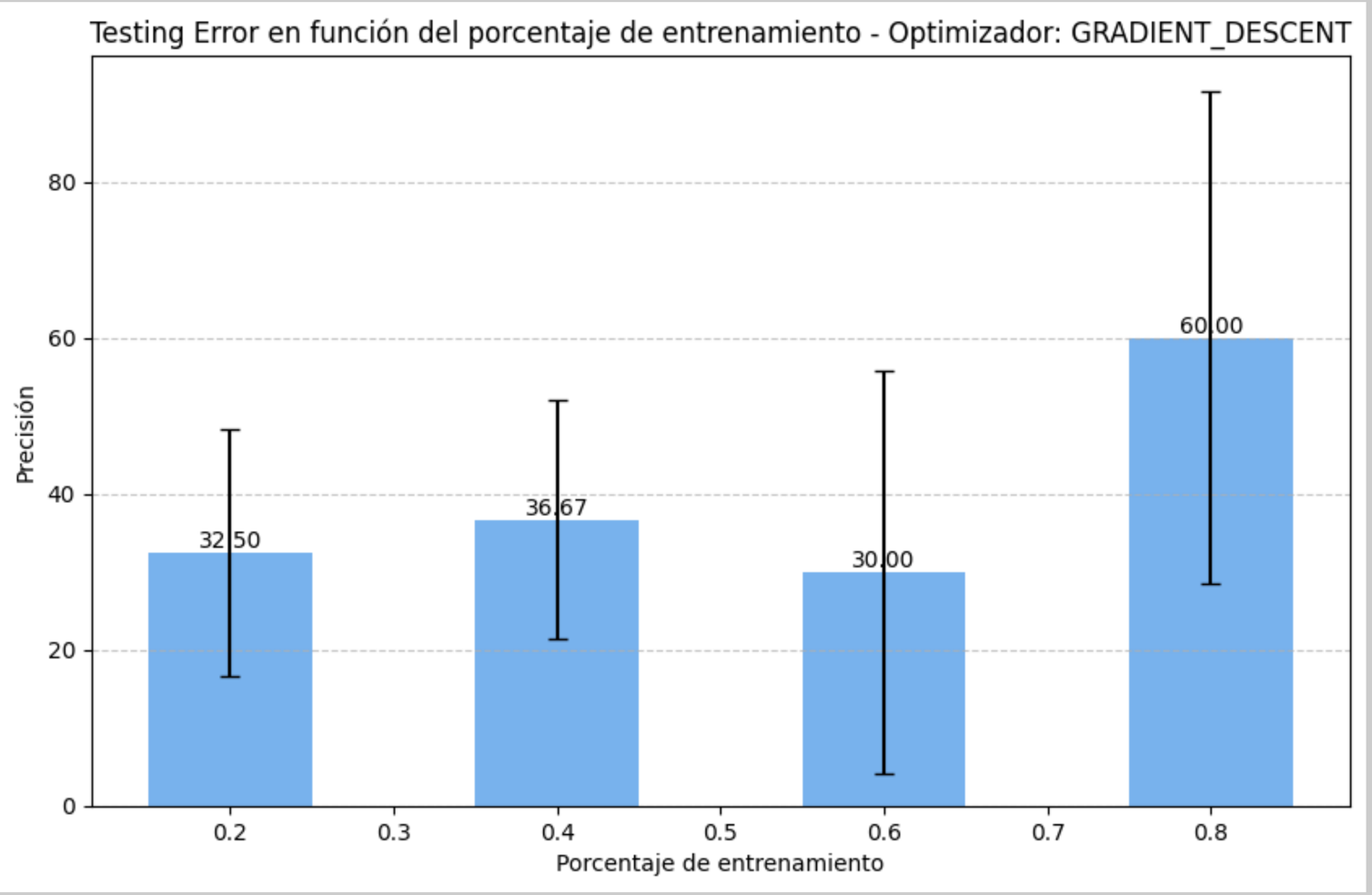
Perceptrón Multicapa



DISCRIMINACIÓN DE NÚMEROS PAR

Al dividir el conjunto de datos en entramiento y testeo, estamos privando a la red de que aprenda la paridad de todos los números. Por lo que le cuesta ser eficiente a la hora de predecir la paridad de los números no aprendidos.

Perceptrón Multicapa



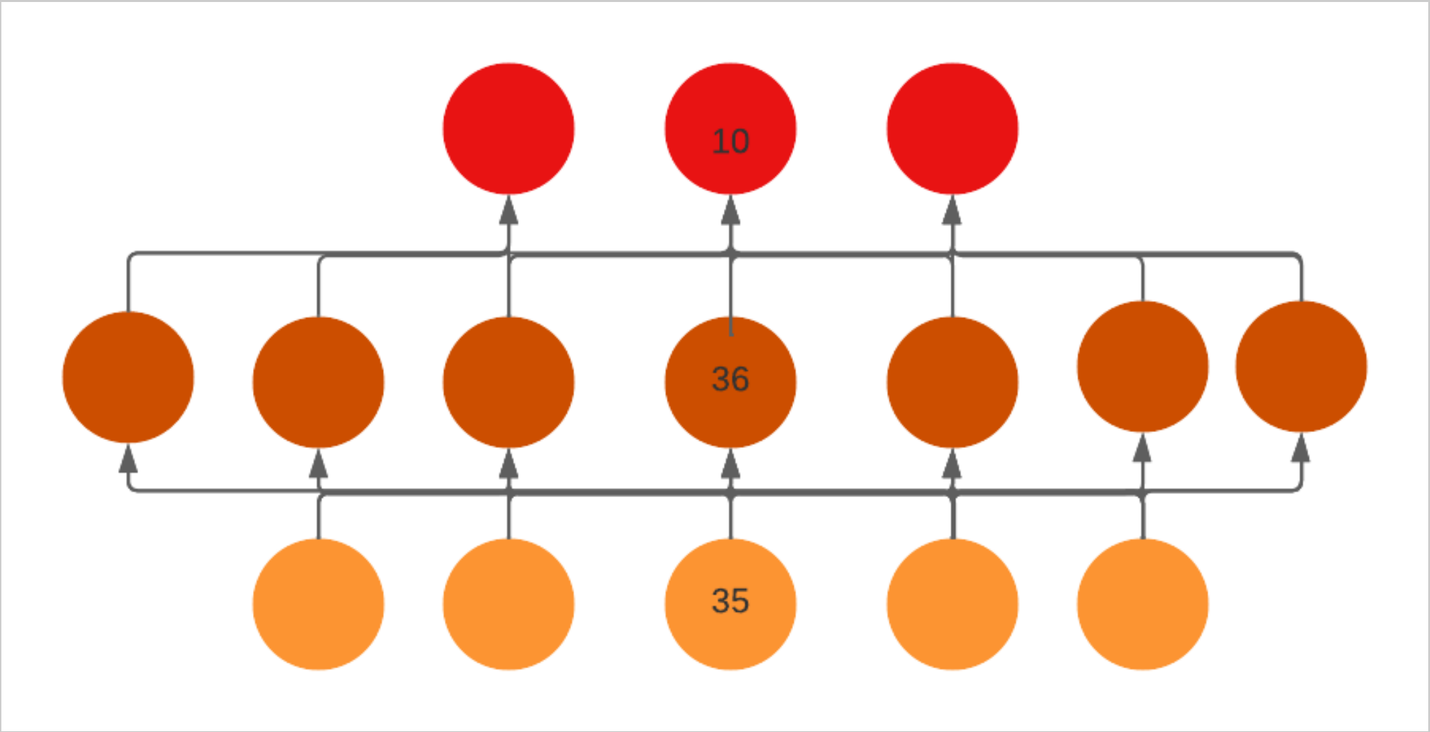
DISCRIMINACIÓN DE NÚMEROS PAR

Cuando se testea un número que nunca entreno es prácticamente aleatorio que logre acertar la paridad del mismo.

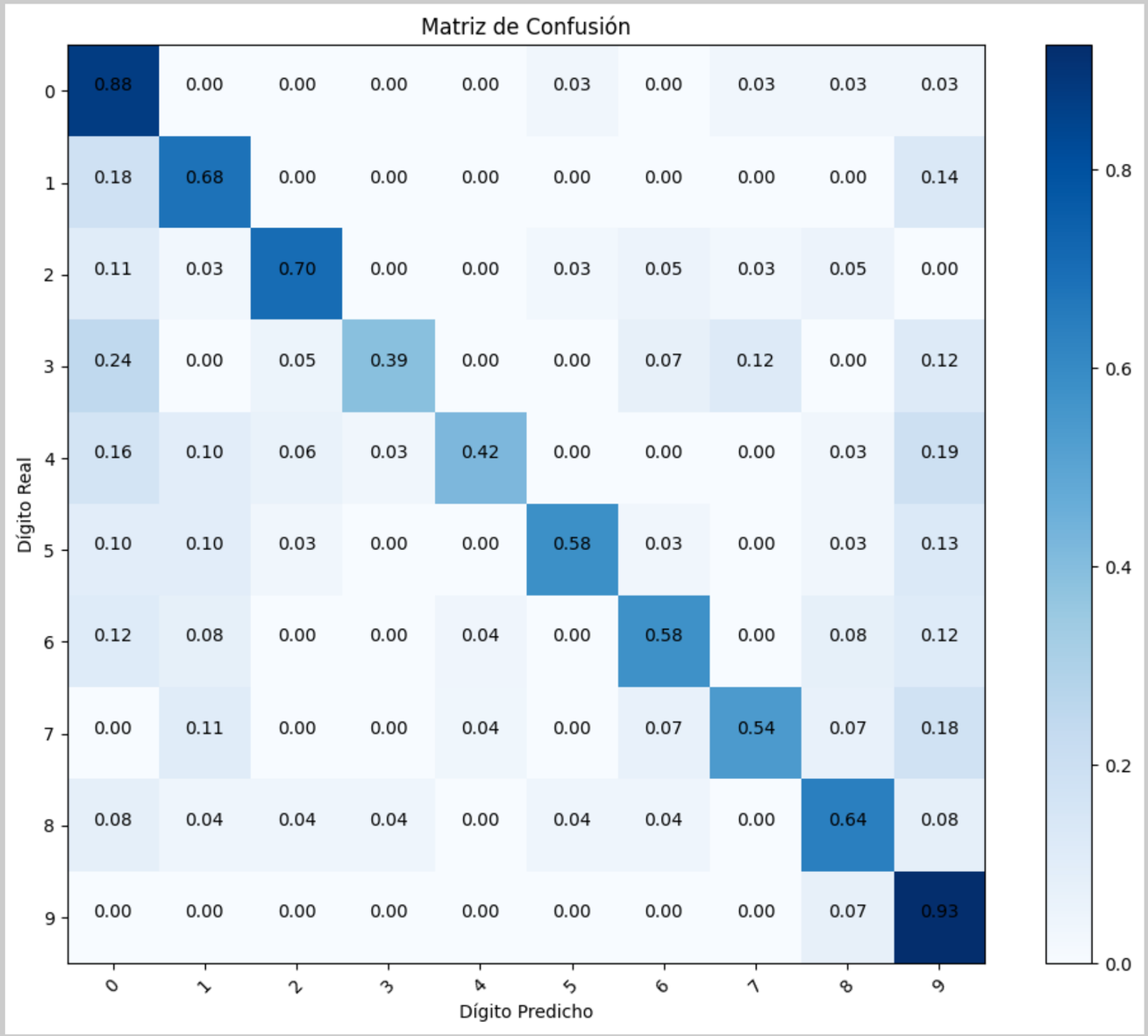
Perceptrón Multicapa

DISTINCIÓN DE NÚMEROS (0-9)

Learning Rate: 0.001
Training Percentage: 1*
Epochs: 1000
Bias: 1
Beta: 1
Epsilon: 0.5
Optimizer: "ADAM"



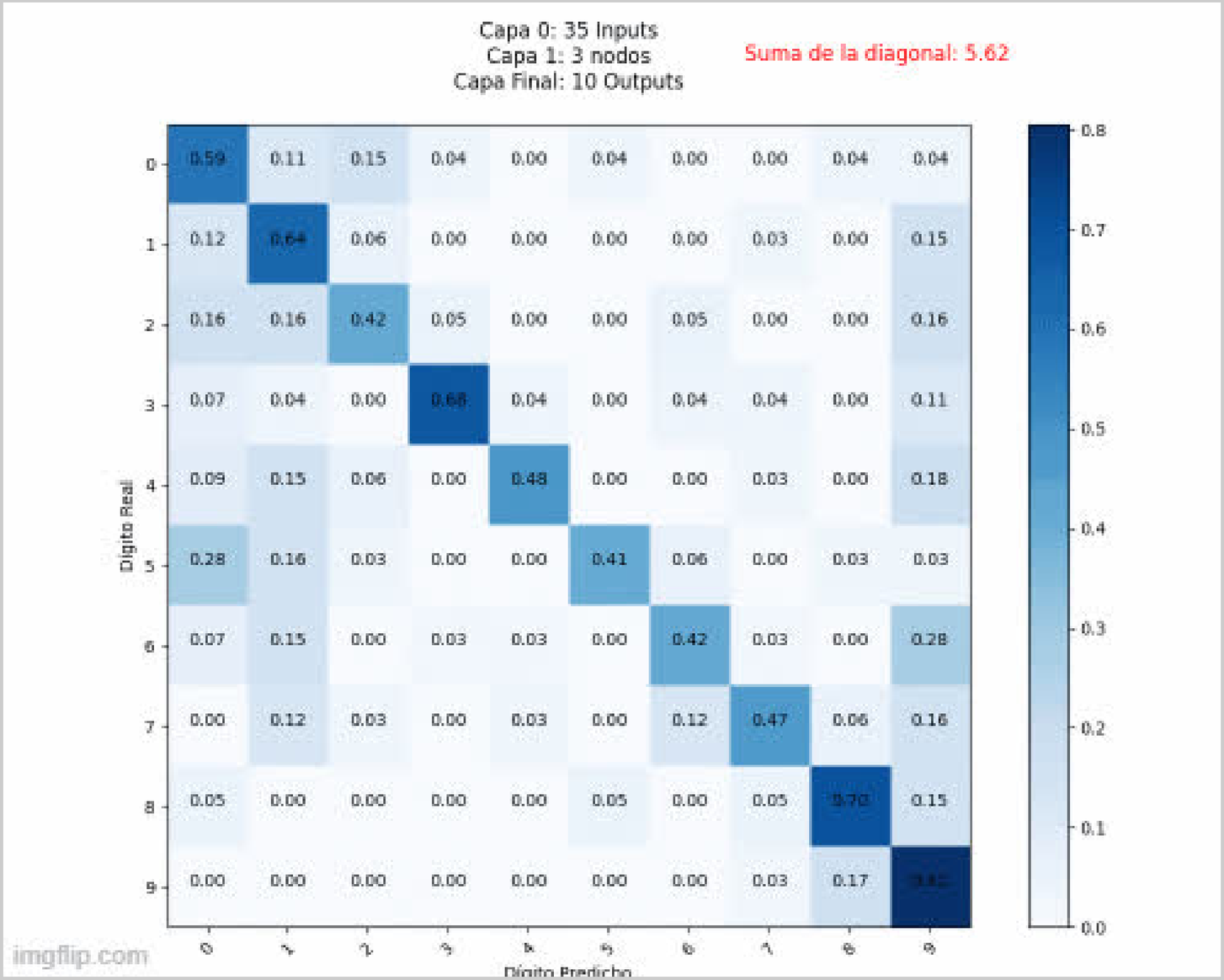
* Se testea con los mismos dígitos + ruido. El ruido implica invertir el valor de hasta 5 bits aleatorios



Perceptrón Multicapa

DISTINCIÓN DE NÚMEROS (0-9)

Con un poco de ruido, la red fue capaz de utilizar todos los datos como entrenamiento y testearse con datos nuevos.



Perceptrón Multicapa

DISTINCIÓN DE NÚMEROS (0-9)

Notamos que agregar más nodos y/o capas no implica una mayor precisión a la hora de clasificar.

Conclusiones

- Un conjunto acotado de datos no permite que la red entrene bien.
- En caso de tener un conjunto de datos acotado, podemos transformarlo un poco (ruido) para hacer más grande el conjunto disponible y asi permitir un mejor entrenamiento del perceptrón.
- La combinación de capas ideal de la red neuronal debe ser experimentada. No hay una receta (por ejemplo, más neuronas = mejor resultado)

Más Conclusiones

Tanto en el ejercicio 3.2 como en el 3.3, se observa que los resultados obtenidos son relativamente aleatorios.

- En el ejercicio 3.2, resulta lógico que los aciertos obtenidos se encuentren alrededor del 50%, ya que “adivinar” la paridad de un número es análogo a “lanzar una moneda”.

- En el ejercicio 3.3, al ser imágenes de tan baja resolución (35 bits en total), agregarles ruido (cambiar hasta cinco 0's por 1's, y viceversa) afecta mucho el desempeño del modelo a la hora de discriminar un número. Por esto, a pesar de que la diagonal en la matriz de confusión se encuentre resaltada, en promedio se tiene un acierto de entre el 50 y 60%, lo que no es significativamente distinto del azar.