Sistemas de Inteligencia Artificial

TP3: Perceptrón Simple y Multicapa

Grupo 6

- Desiree Melisa Limachi
- Joseph Rouquette
- Magdalena Flores Levalle
- Matias Ezequiel Daneri

Ejercicio 1: Perceptrón Simple Escalonado

AND y XOR

AND:

Devuelve verdadero (1) si ambas entradas son 1; de lo contrario, devuelve falso (0)

XOR:

Devuelve verdadero (1) si exactamente una de las entradas es 1; de lo contrario, devuelve falso (0)

Parametros de entrada/salida

AND y XOR

AND:

Entrada: x={{−1,1},{1,−1},{−1,−1},{1,1}}

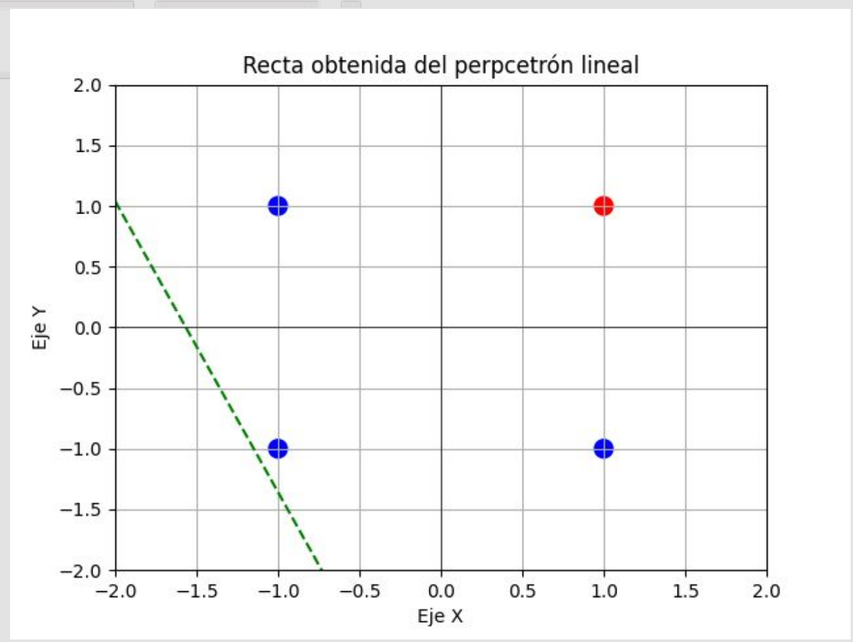
Salida esperada: y={−1,−1,−1,1}

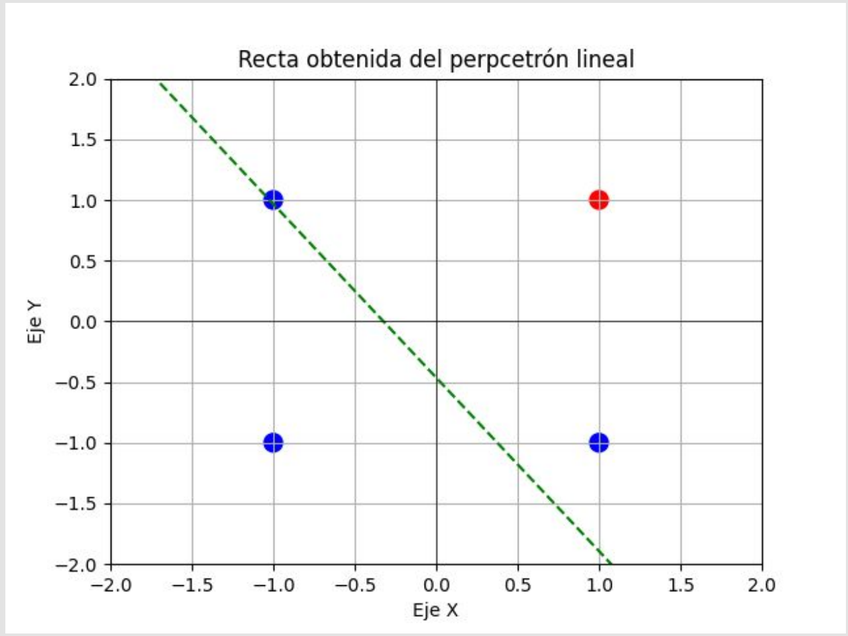
XOR:

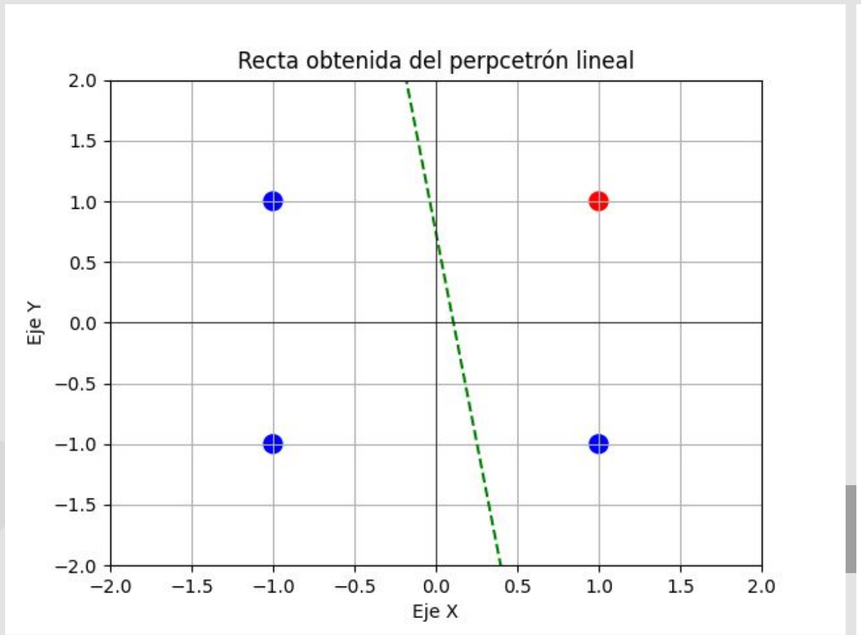
Entrada: x={{−1,1},{1,−1},{−1,−1},{1,1}}

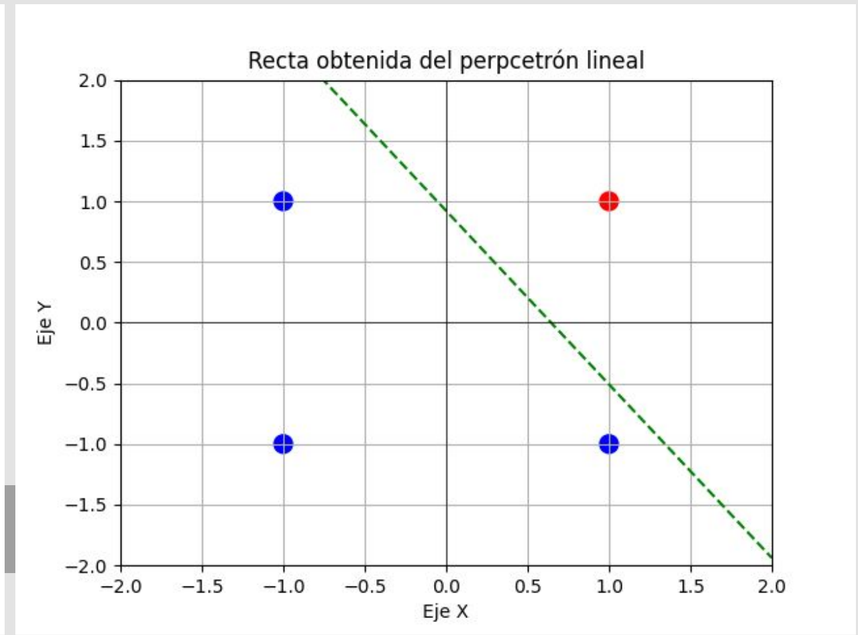
Salida esperada: y={1,1,−1,−1}

AND - Comparación de epochs







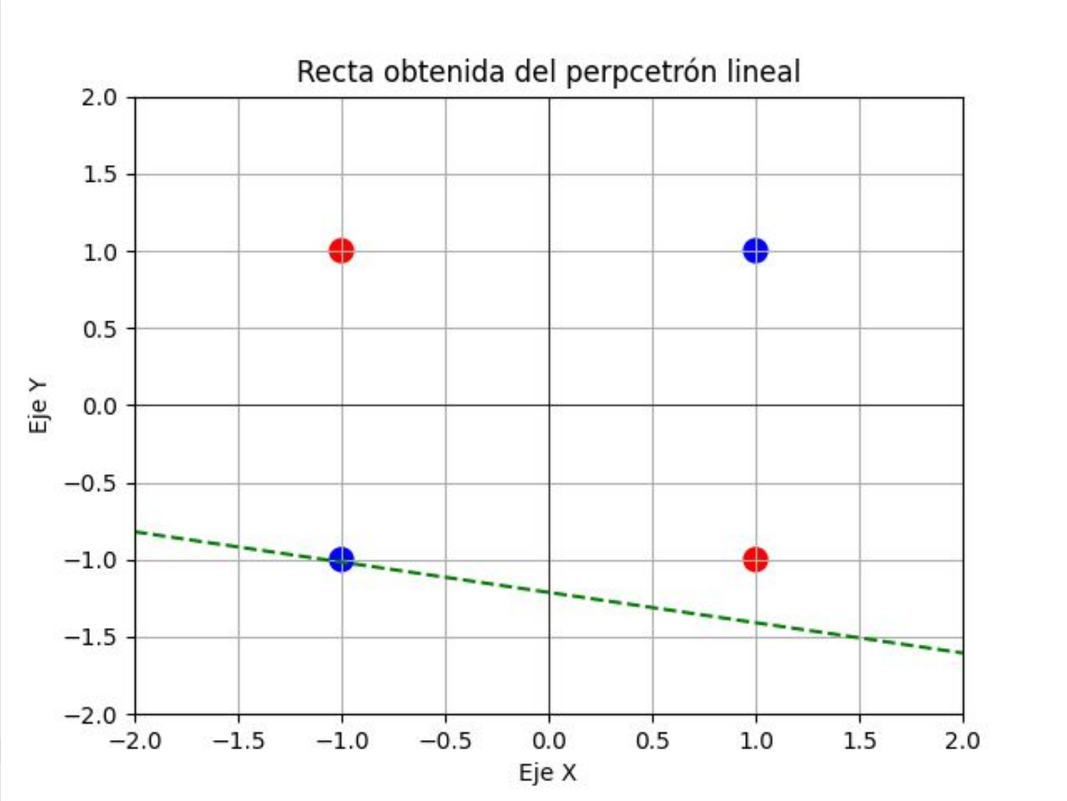


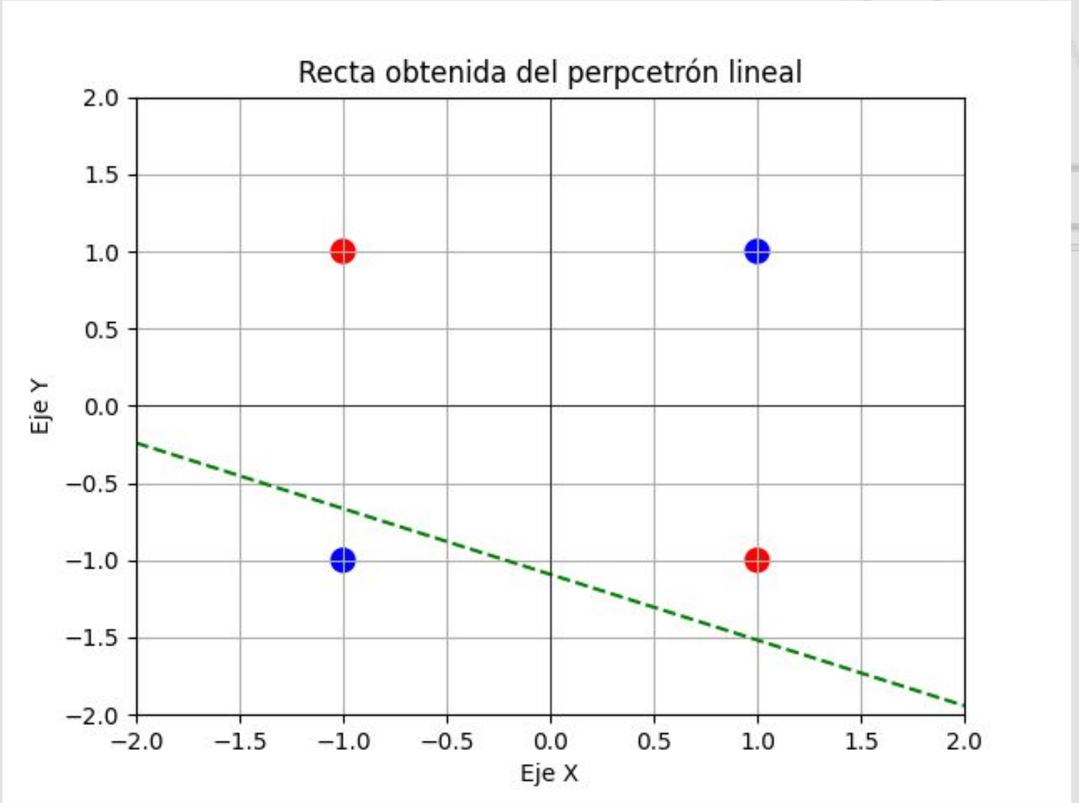
output:

-1

1

XOR - Gráfico


Recta obtenida del perpcetrón lineal
Eje Y
Eje X
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Recta obtenida del perpcetrón lineal
Eje Y
Eje X
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Conclusiones

AND

Es un problema **linealmente separable** (se puede trazar una recta que separa las dos clases de datos).

XOR

No es un problema que se pueda resolver con este tipo de perceptrón, se requiere de dos rectas que separen las dos categorías.

Ejercicio 2: Perceptron Simple

Lineal, No Lineal, Hyperbólico, Logarítmico

Consideraciones:

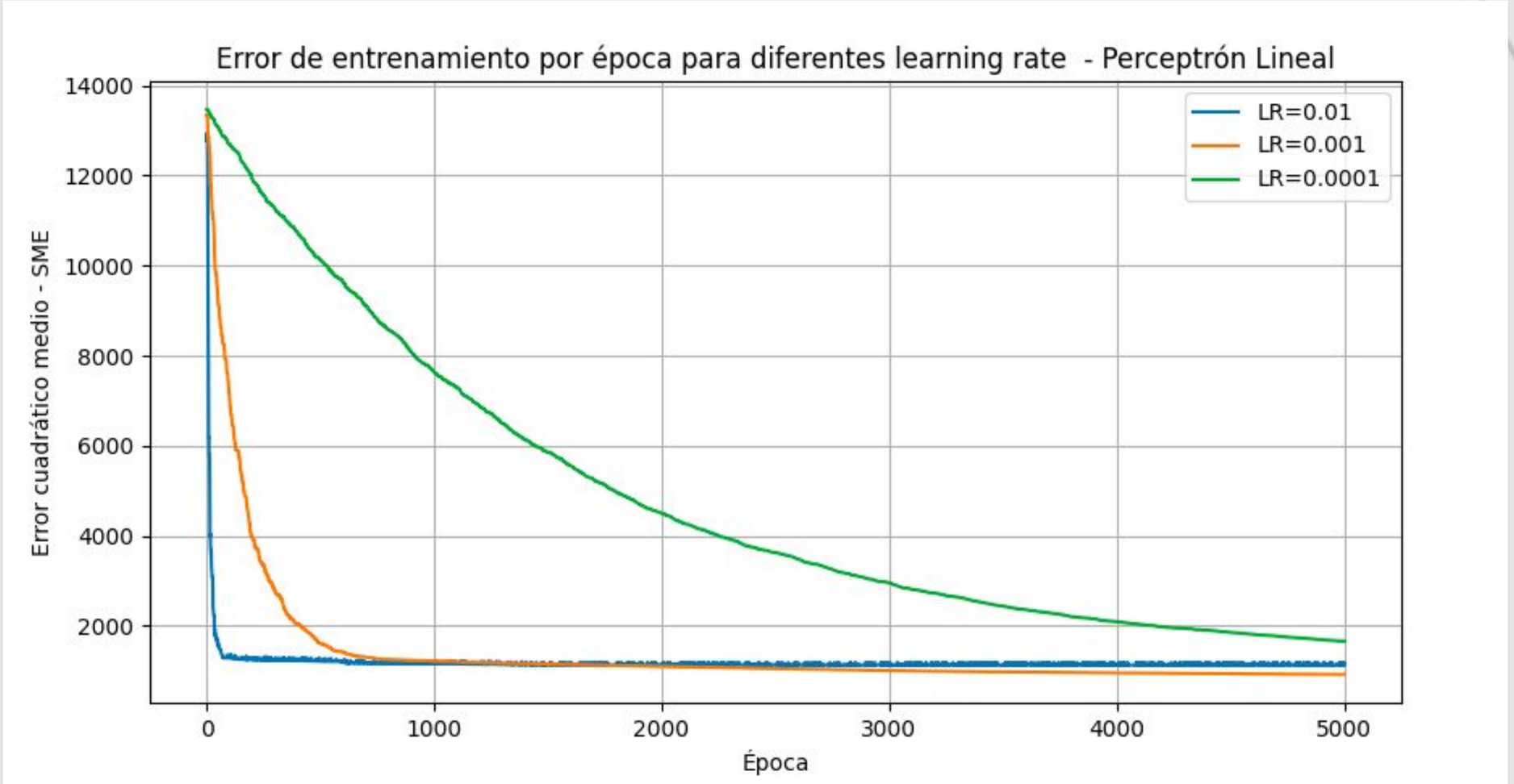
- Para los perceptrones no lineales se normalizaron, con el metodo Min-Max Feature Scaling, los valores esperados debido a la función de activación, cuya imagen está contenida en [-1, 1] para tanh y [0;1] para logarithmic
- Como la función de activación de nuestro perceptrones lineales utiliza la función de activación como la identidad, no se debió normalizar.

Nos preguntamos

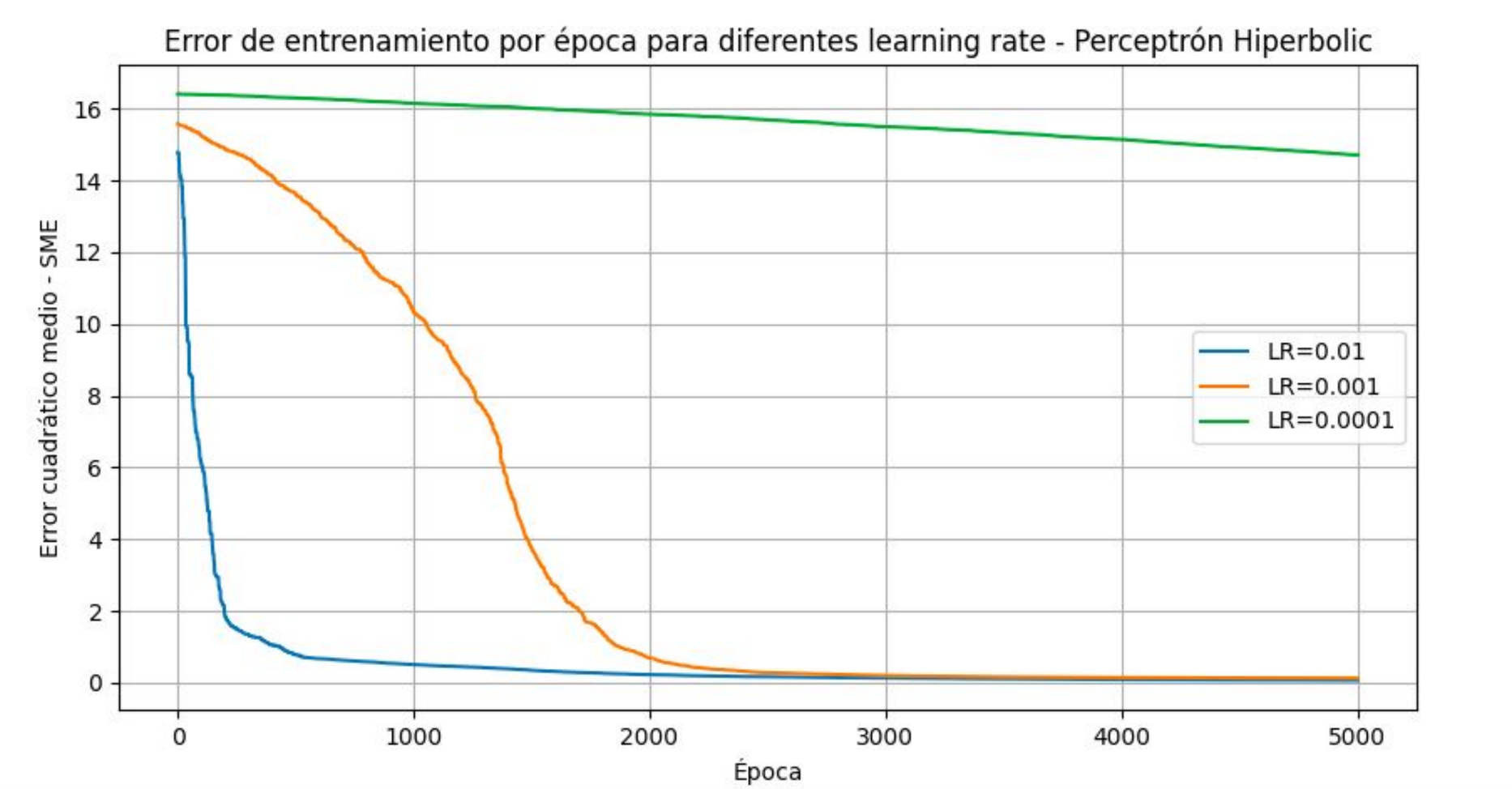
¿Cuál es el error cuadrático en función de las épocas agrupadas por learning rate?

```
"train_percentage": 0.8
"max_epochs": 5000
"beta": 1
"epsilon": 0.01
```

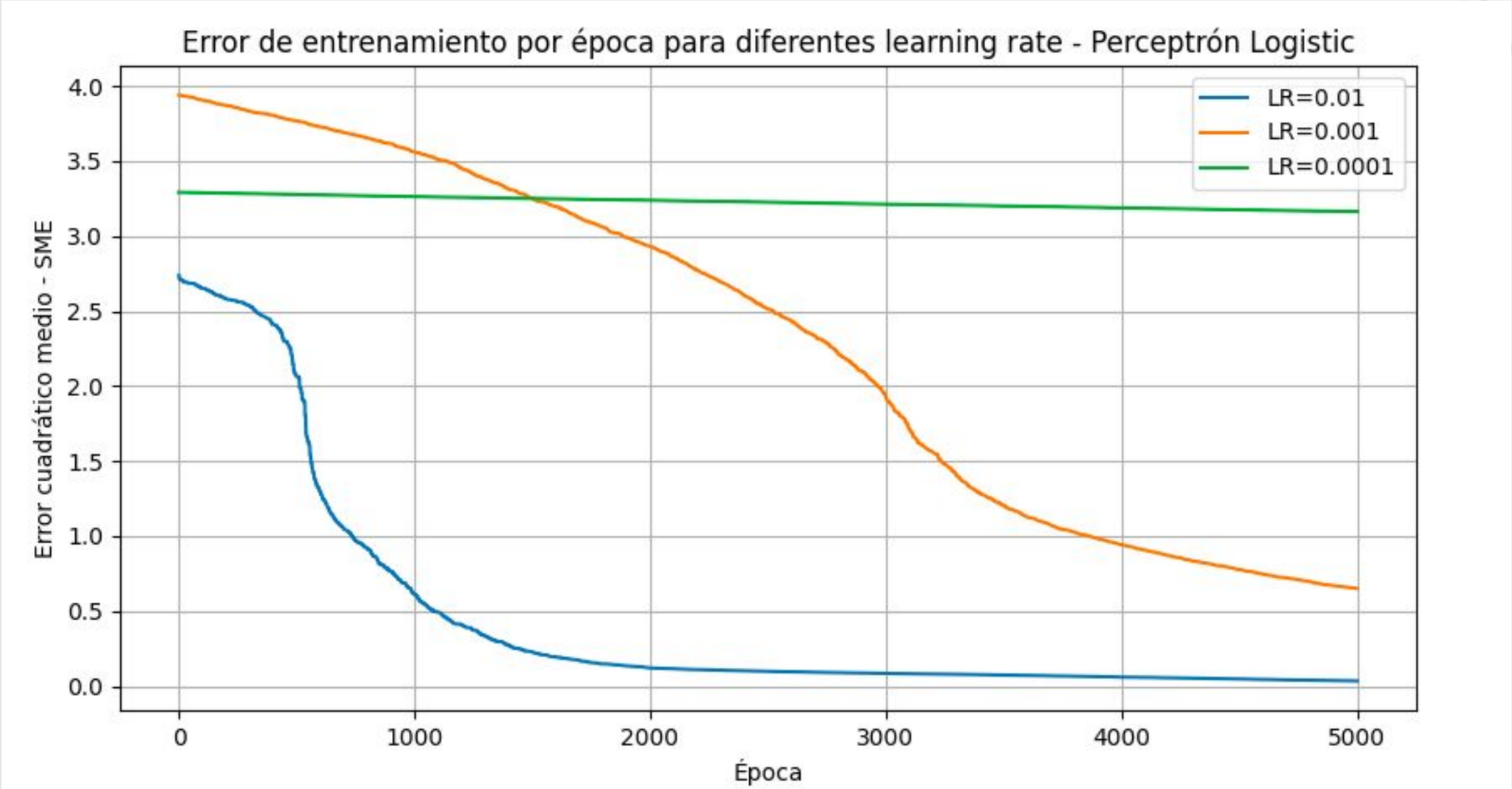
Gráfico


Error de entrenamiento por época para diferentes learning rate - Perceptrón Lineal
Error cuadrático medio - SME
Época
LR=0.01
LR=0.001
LR=0.0001
14000
12000
10000
8000
6000
4000
2000
0
1000
2000
3000
4000
5000

Gráfico


Error de entrenamiento por época para diferentes learning rate - Perceptrón Hiperbolic
Error cuadrático medio - SME
Época
0
2
4
6
8
10
12
14
16
0
1000
2000
3000
4000
5000
LR=0.01
LR=0.001
LR=0.0001

Gráfico

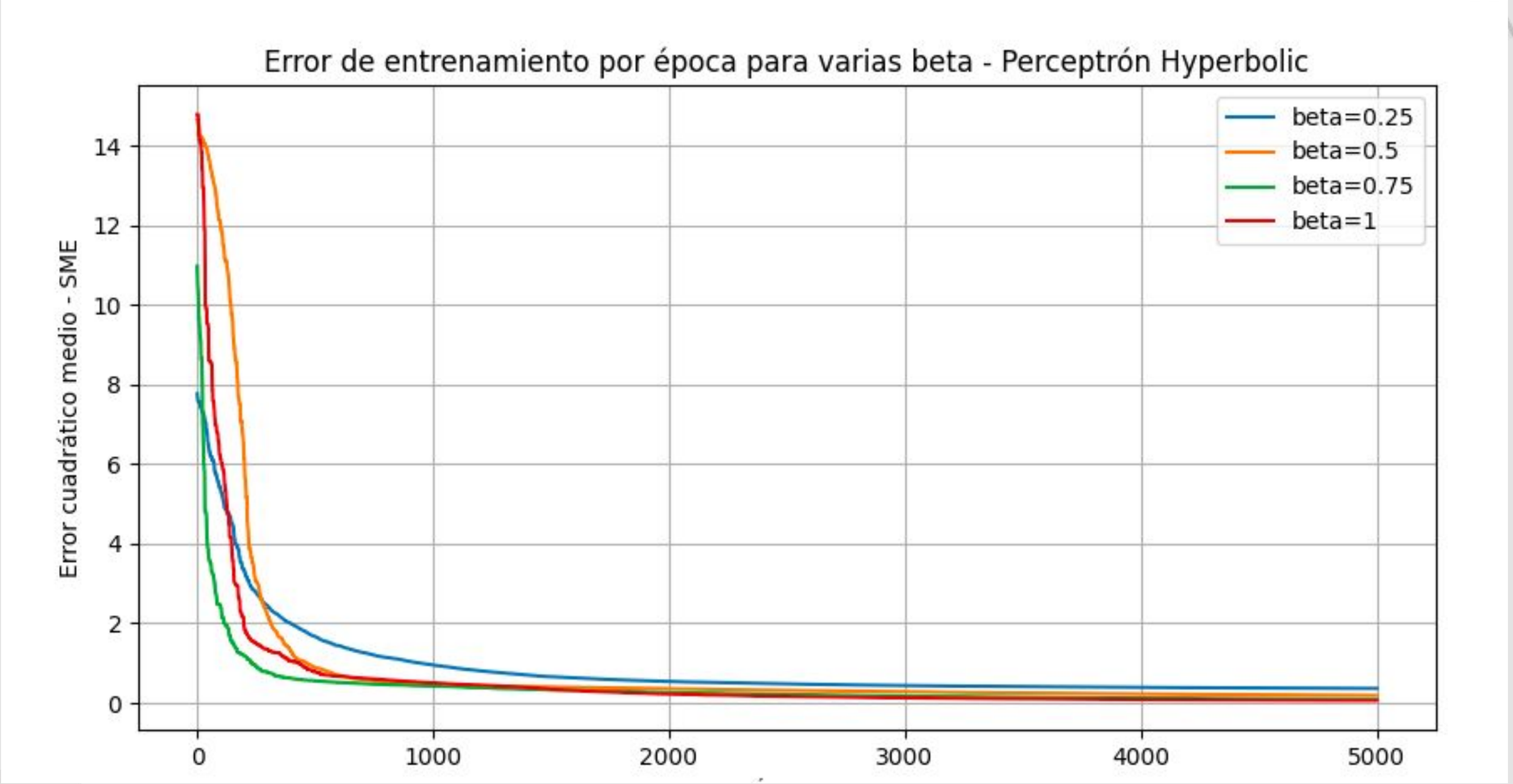

Error de entrenamiento por época para diferentes learning rate - Perceptrón Logistic
Error cuadrático medio - SME
Época
LR=0.01
LR=0.001
LR=0.0001
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000

Nos preguntamos

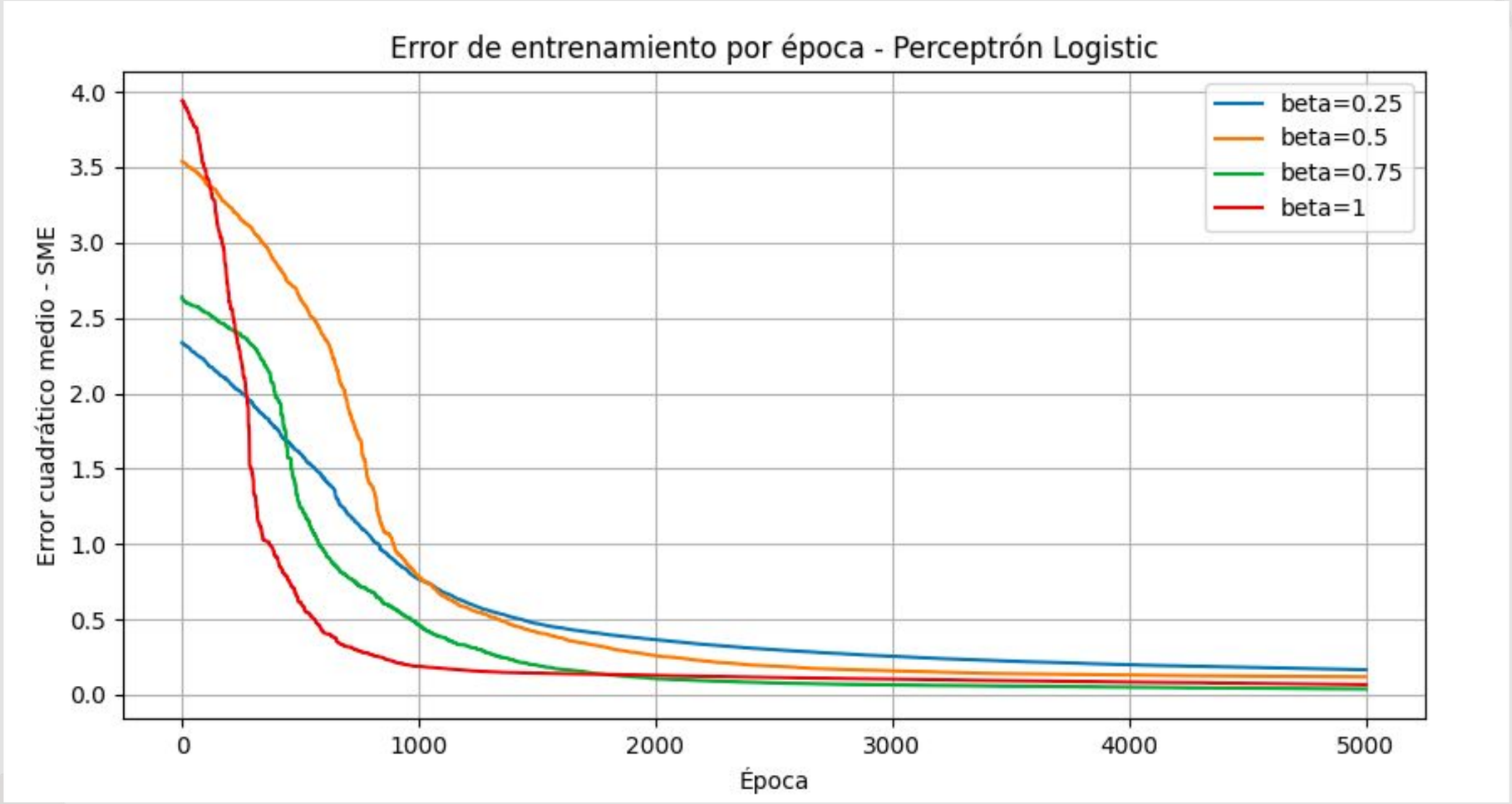
Para los perceptrones No Lineal. ¿Cuál es el error cuadrático en función de las épocas agrupadas por beta?

```
"train_percentage": 0.8
   "max_epochs": 5000
"learning_rate": 0.01
   "epsilon": 0.01
```

Gráfico


Error de entrenamiento por época para varias beta - Perceptrón Hyperbolic
Error cuadrático medio - SME
beta=0.25
beta=0.5
beta=0.75
beta=1
0
2
4
6
8
10
12
14
0
1000
2000
3000
4000
5000

Gráfico

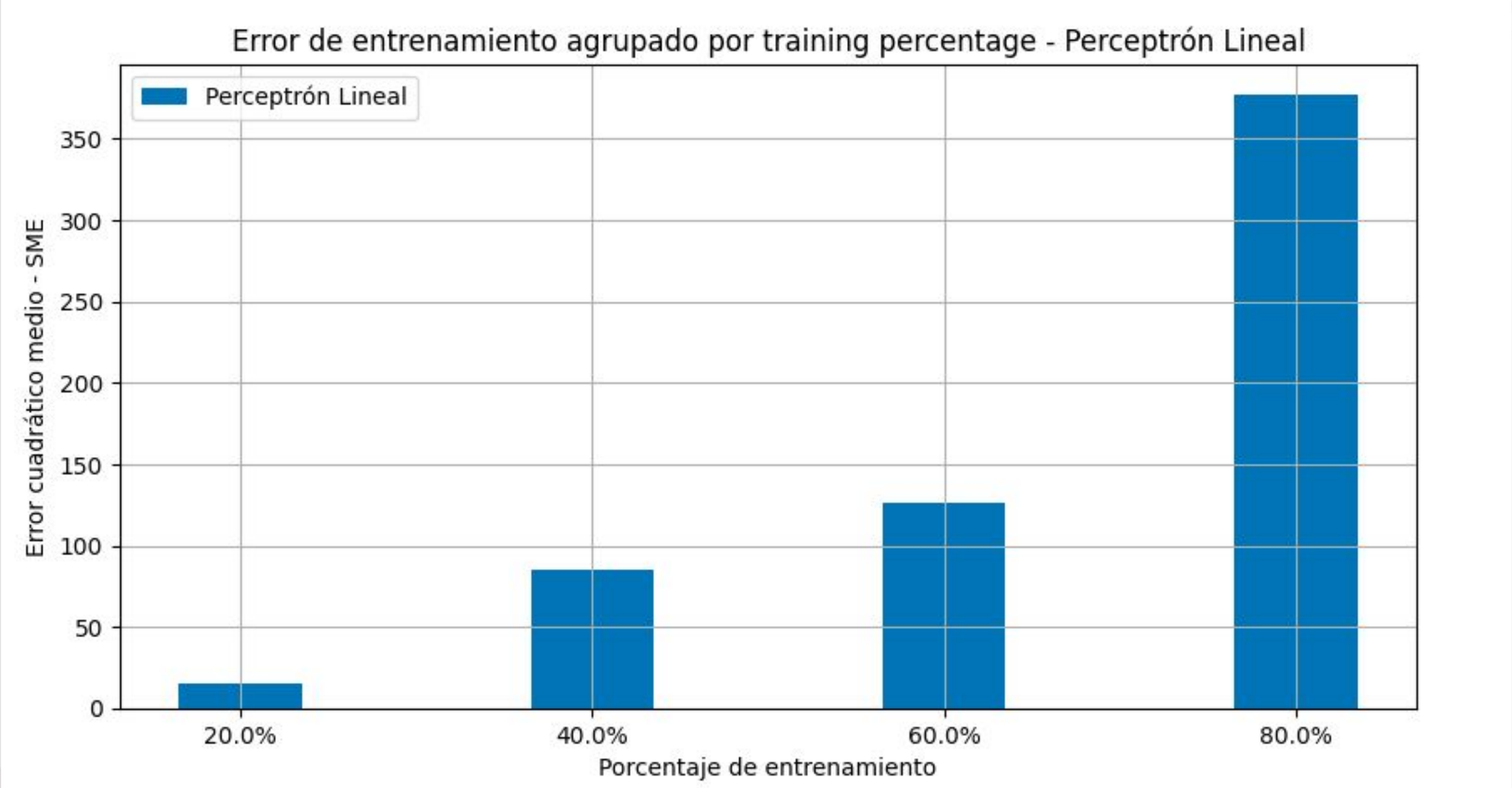

Error de entrenamiento por época - Perceptrón Logistic
Error cuadrático medio - SME
Época
beta=0.25
beta=0.5
beta=0.75
beta=1
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000

Nos preguntamos

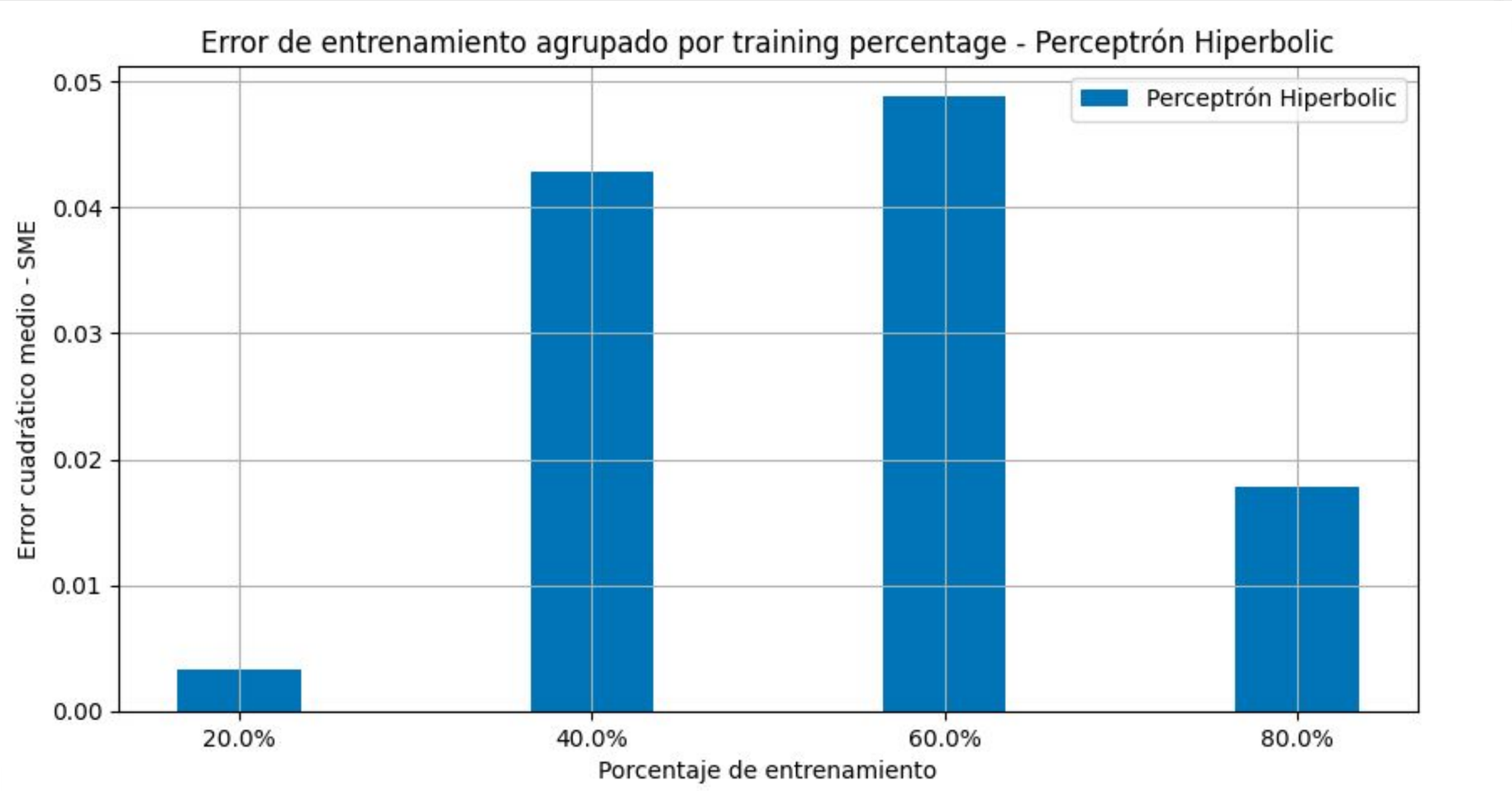
¿Cúal es la capacidad de cada uno de los perceptrones para aprender la función cuyas muestras están en los archivos?

```
"max_epochs": 5000
"learning_rate": 0.01
"beta": 1
"epsilon": 0.01
```

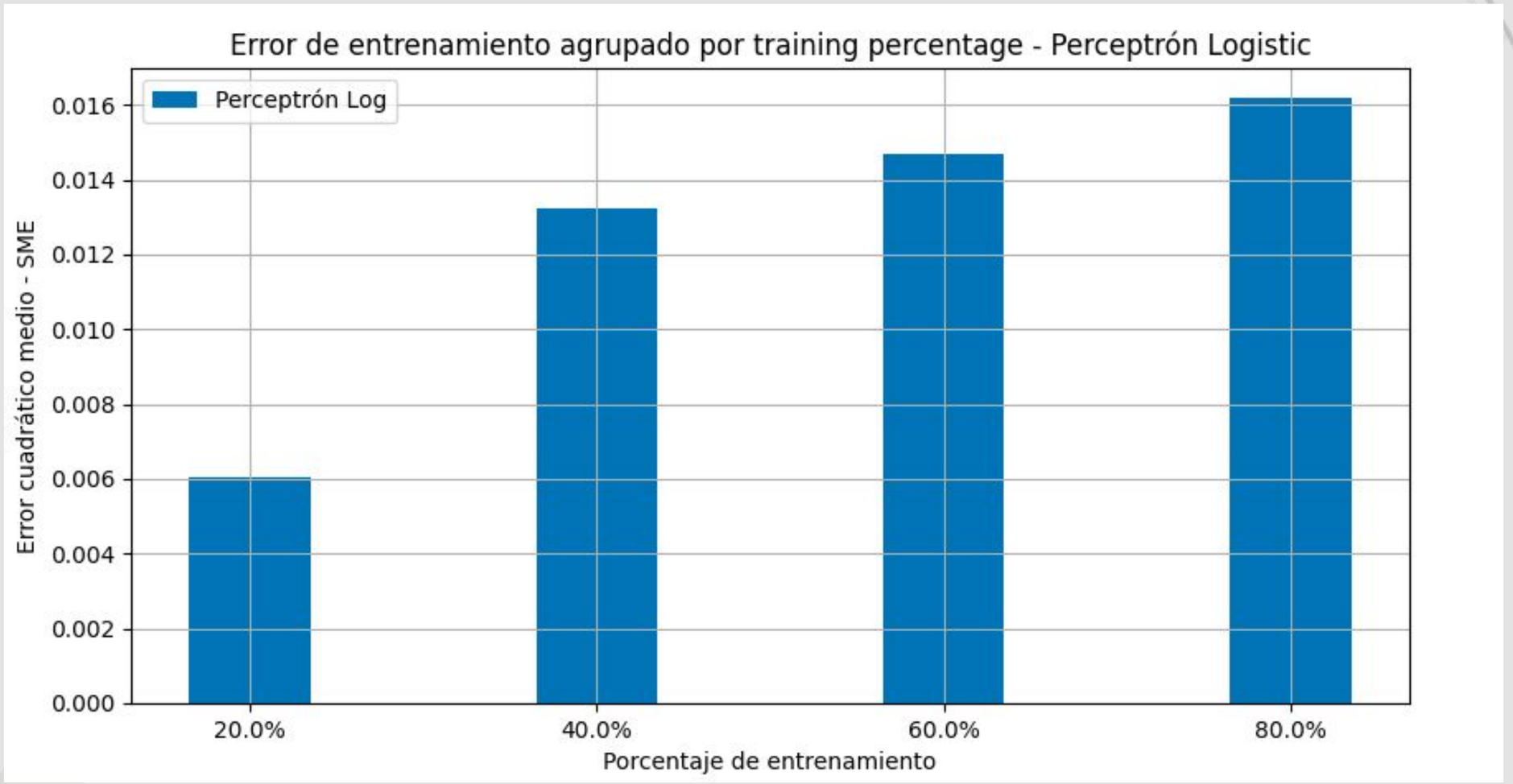
Gráfico


Error de entrenamiento agrupado por training percentage - Perceptrón Lineal
Perceptrón Lineal
Error cuadrático medio - SME
0
50
100
150
200
250
300
350
20.0%
40.0%
60.0%
80.0%
Porcentaje de entrenamiento

Gráfico


Error de entrenamiento agrupado por training percentage - Perceptrón Hiperbolic
Perceptrón Hiperbolic
Error cuadrático medio - SME
0.05
0.04
0.03
0.02
0.01
0.00
20.0%
40.0%
60.0%
80.0%
Porcentaje de entrenamiento

Gráfico


Error de entrenamiento agrupado por training percentage - Perceptrón Logistic
Perceptrón Log
Error cuadrático medio - SME
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
20.0%
40.0%
60.0%
80.0%
Porcentaje de entrenamiento

Conclusion

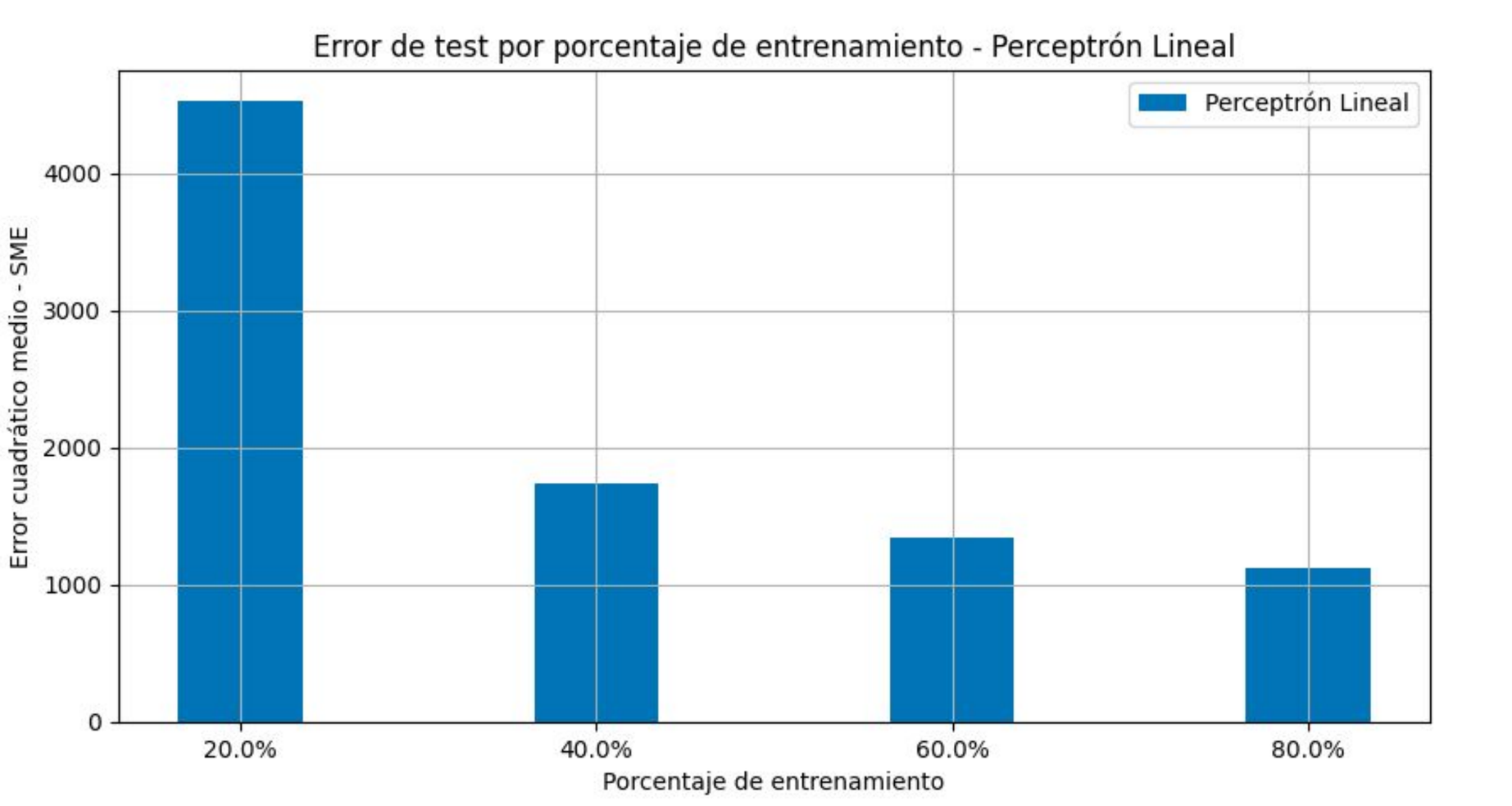
- La capacidad de aprendizaje de los dos tipos de perceptrones varía con la cantidad de datos de entrenamiento.
- El perceptrón lineal parece beneficiarse de la adición de datos hasta cierto punto, después del cual puede sufrir de sobreajuste.
- El perceptrón logístico demuestra una capacidad más robusta para aprender y generalizar a partir de diversos conjuntos de datos, sugiriendo que podría ser más adecuado para tareas donde la variabilidad de los datos es mayor.
- La selección de los datos y la gestión del sobreajuste son esenciales para optimizar el rendimiento de estos modelos.

Gráfico

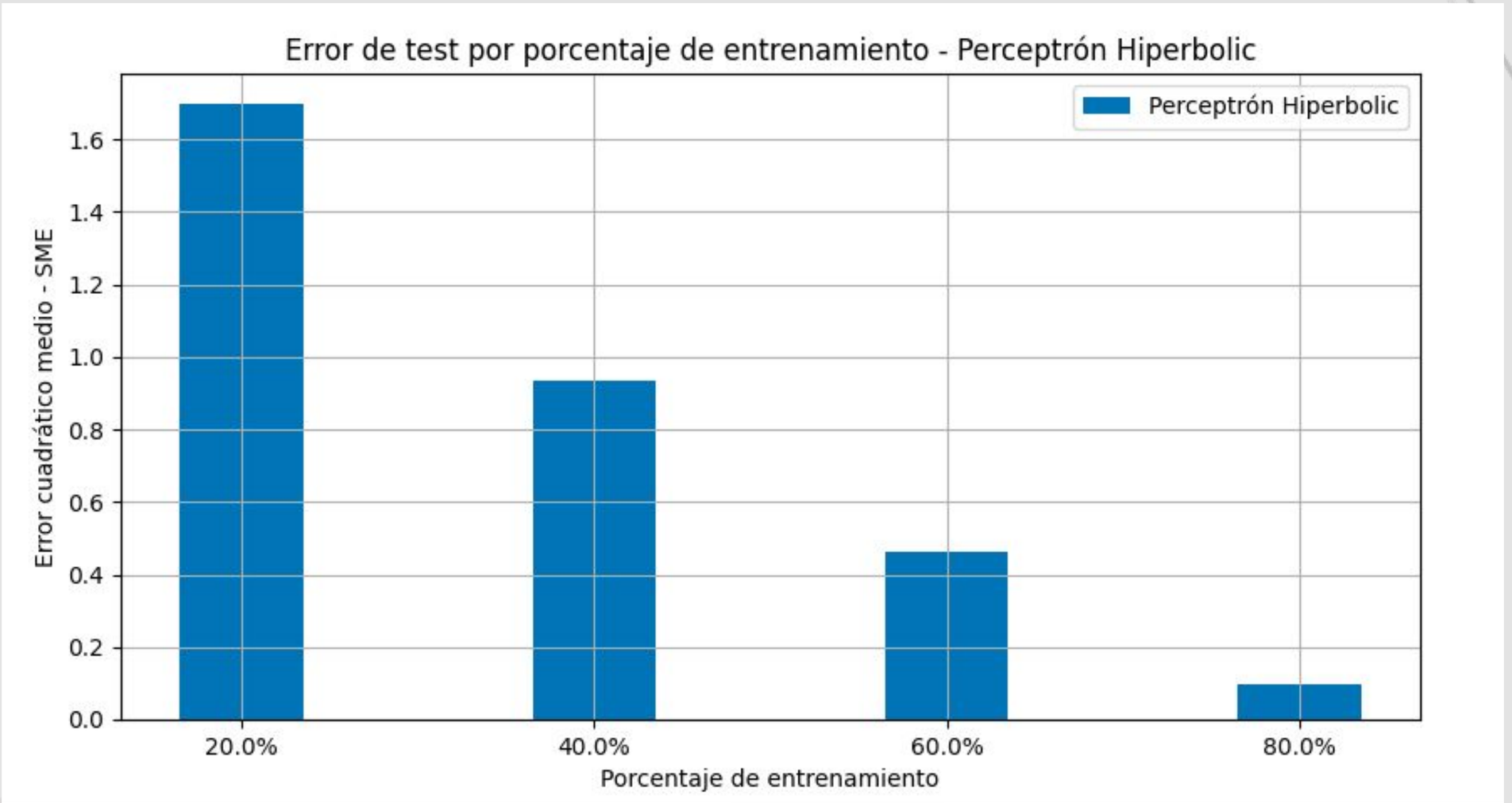
¿Cúal es la capacidad de generalización del perceptron simple no lineal utilizando, de los datos provistos,un subconjunto para entrenar y otro para testear ?


```
"max_epochs": 5000
"learning_rate": 0.01
"beta": 1
"epsilon": 0.01
```

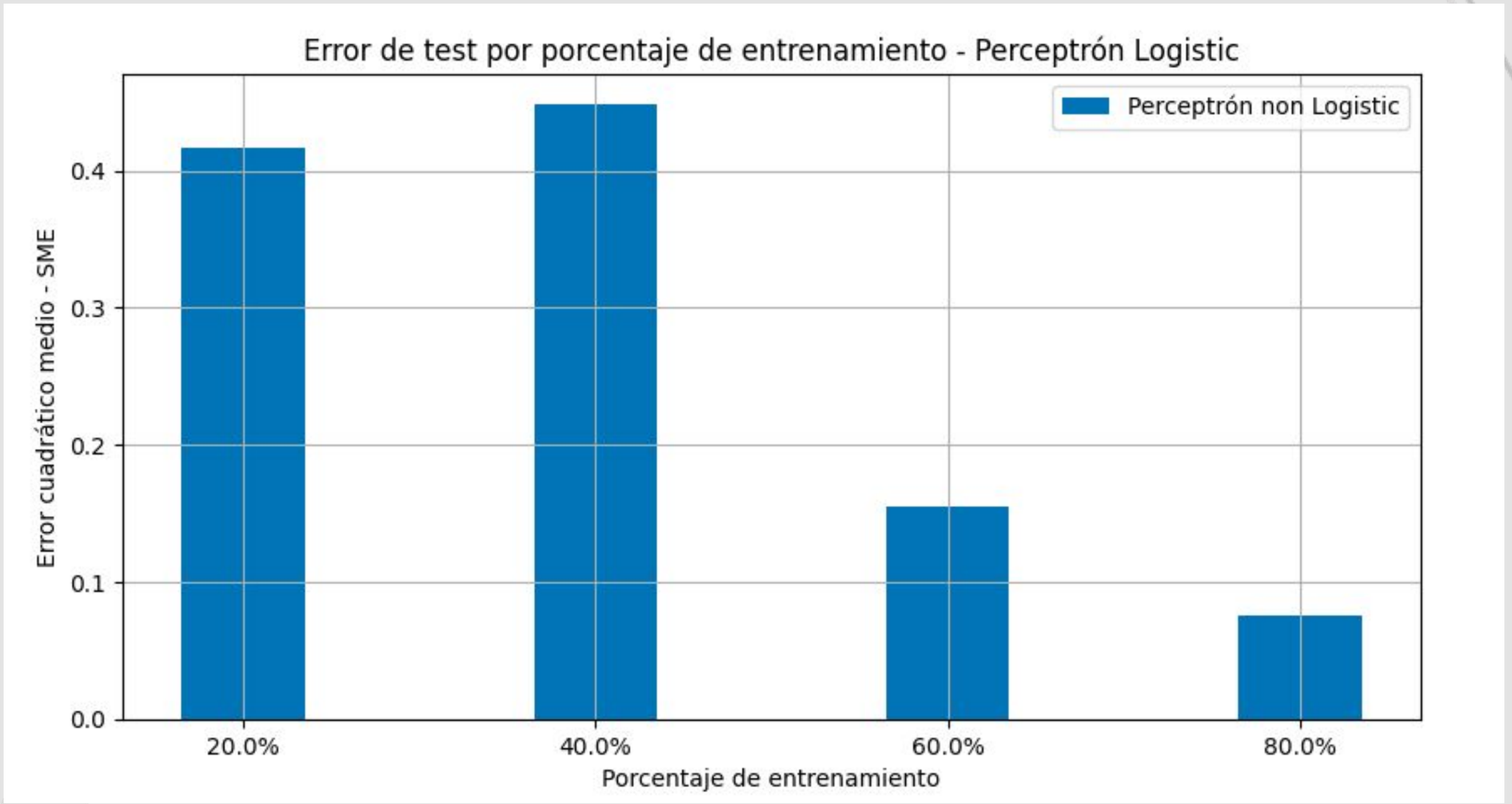
Gráfico


Error de test por porcentaje de entrenamiento - Perceptrón Lineal
Perceptrón Lineal
Error cuadrático medio - SME
0
1000
2000
3000
4000
20.0%
40.0%
60.0%
80.0%
Porcentaje de entrenamiento

Gráfico


Error de test por porcentaje de entrenamiento - Perceptrón Hiperbolic
Perceptrón Hiperbolic
Error cuadrático medio - SME
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
20.0%
40.0%
60.0%
80.0%
Porcentaje de entrenamiento

Gráfico


Error de test por porcentaje de entrenamiento - Perceptrón Logistic
Perceptrón non Logistic
Error cuadrático medio - SME
0.0
0.1
0.2
0.3
0.4
20.0%
40.0%
60.0%
80.0%
Porcentaje de entrenamiento

Conclusion

- Un aumento en el porcentaje de entrenamiento reduce los errores al generalizar.
- La selección de datos parece ser un aspecto crítico que influye significativamente en la habilidad de los perceptrones para generalizar.
- Esto resalta la importancia de considerar cuidadosamente tanto la cantidad como la calidad de los datos de entrenamiento

Nos Preguntamos

¿Cómo elegimos cuál es el mejor conjunto de entrenamiento?

```
"max_epochs": 400
"learning_rate": 0.01
"beta": 1
"epsilon": 0.01
“k”: 6
```

Gráfico mejor conjunto de entrenamiento

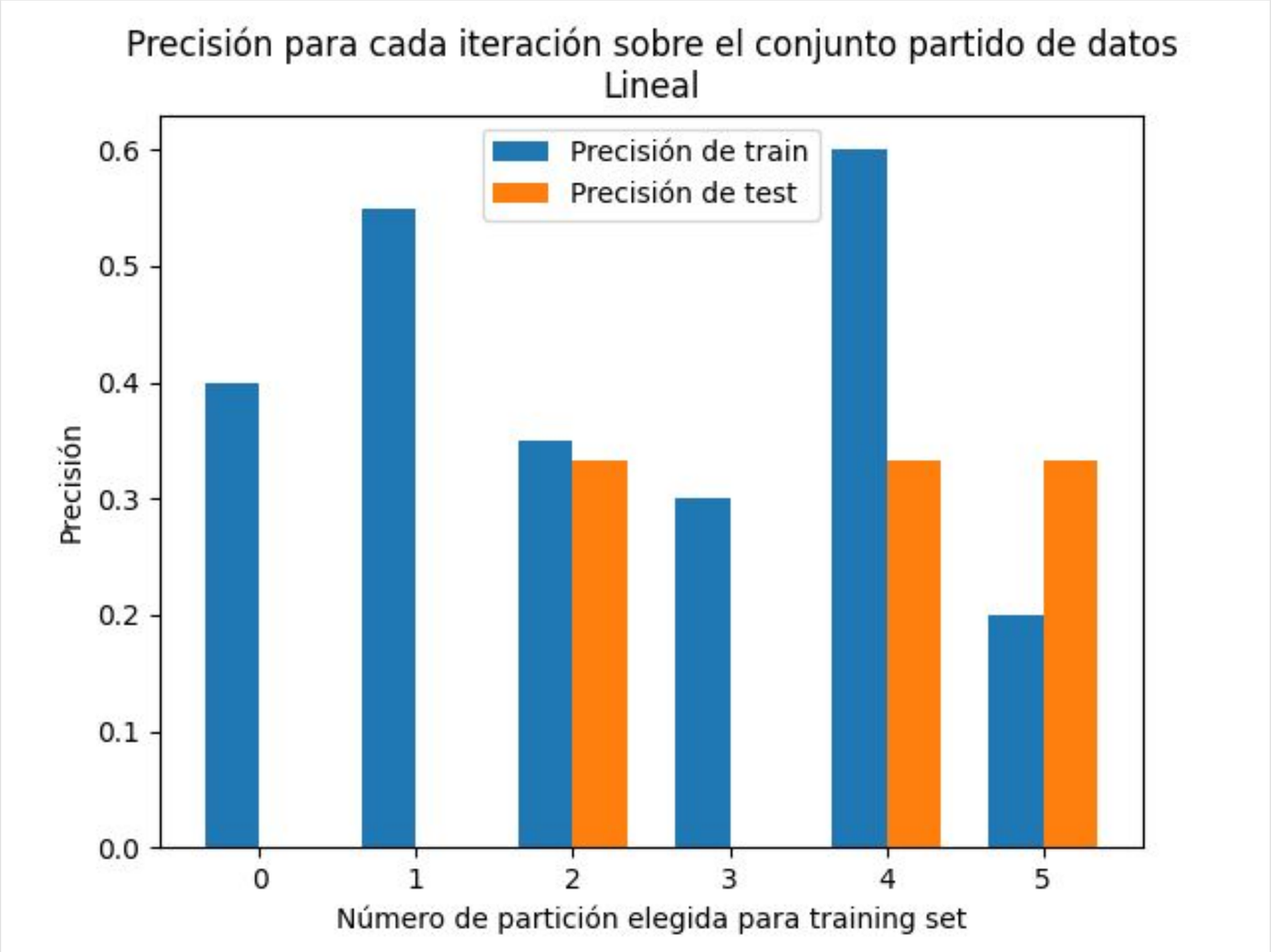


Gráfico mejor conjunto de entrenamiento

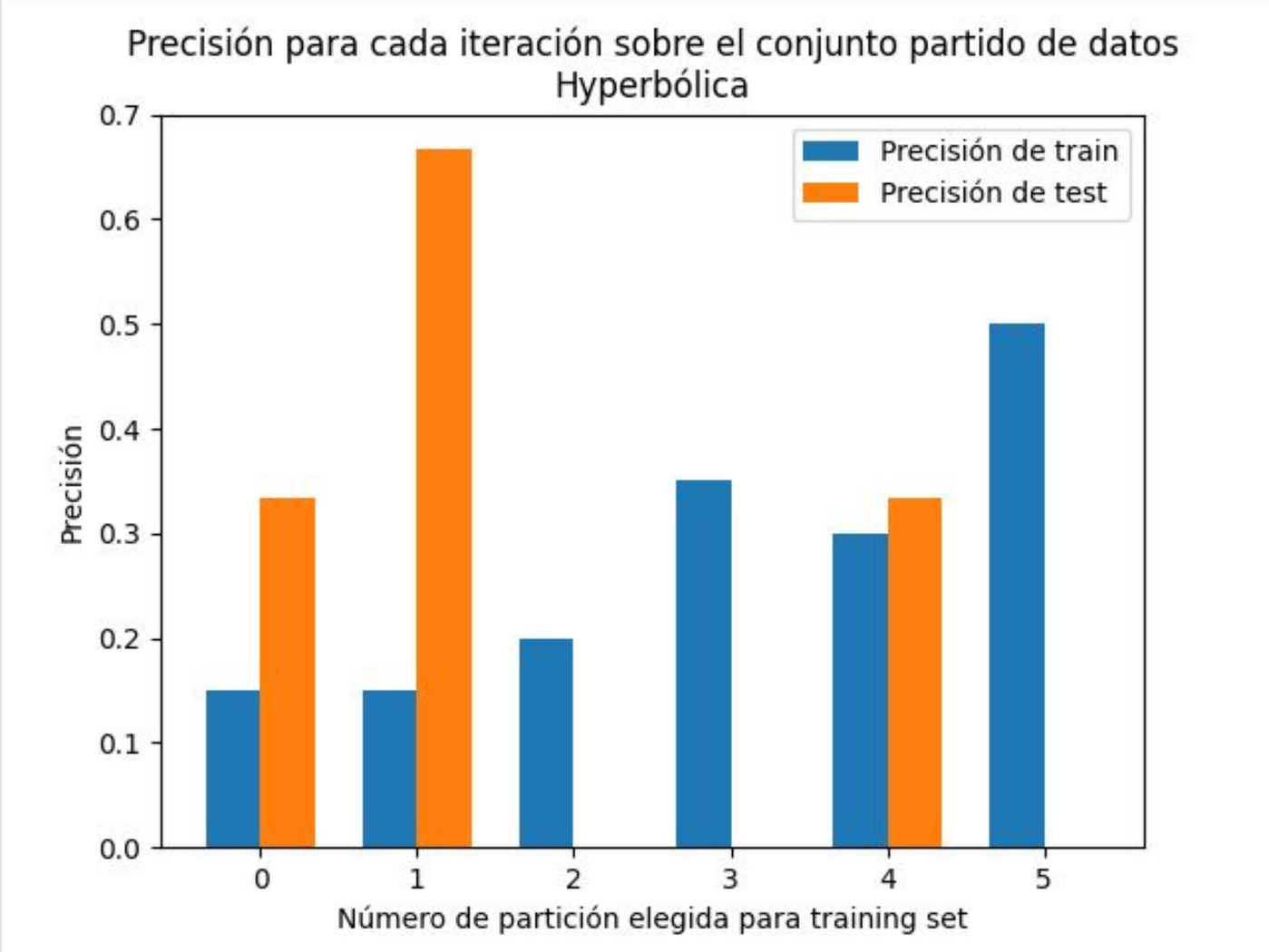
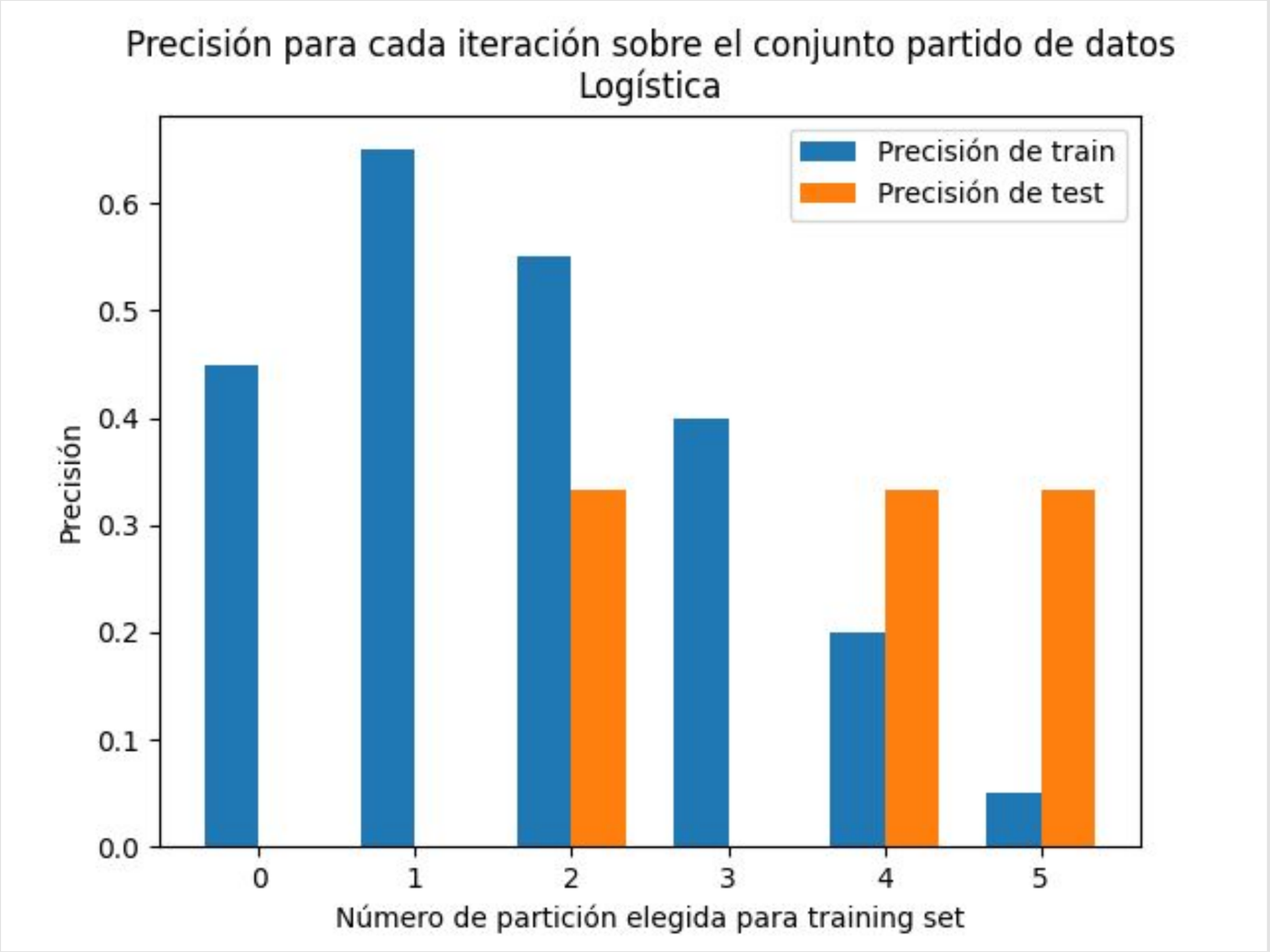


Gráfico mejor conjunto de entrenamiento



Conclusion

- Vemos mejor accuracy en los sets de entrenamiento para el perceptron lineal y logístico.
- El valor de k es muy importante, ya que si es un valor no divisible no deja las partes divididas de igual manera.

Ejercicio 3: Perceptrón Multicapa

XOR, PAR y NUMBER

Funciones implementadas:

- XOR: Función para determinar si es verdadero si exactamente una de las entradas es igual; de lo contrario, devuelve falso
- PAR: Función para discriminar si un número es “par”
- NUMBER: Función para determinar qué dígito se corresponde con la entrada a la red

Ejercicio 3: Parametros de entrada/salida

XOR, PAR y NUMBER

Funciones implementadas:

- XOR: Entrada: x={{−1,1},{1,−1},{−1,−1},{1,1}}

  Salida esperada: y={1,1,−1,−1}

- PAR: Entrada: conjunto datos que forman números decimales del 0 al 9

  Salida esperada: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

- NUMBER: Entrada: conjunto datos que forman números decimales del 0 al 9

  Salida: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Gráfico XOR

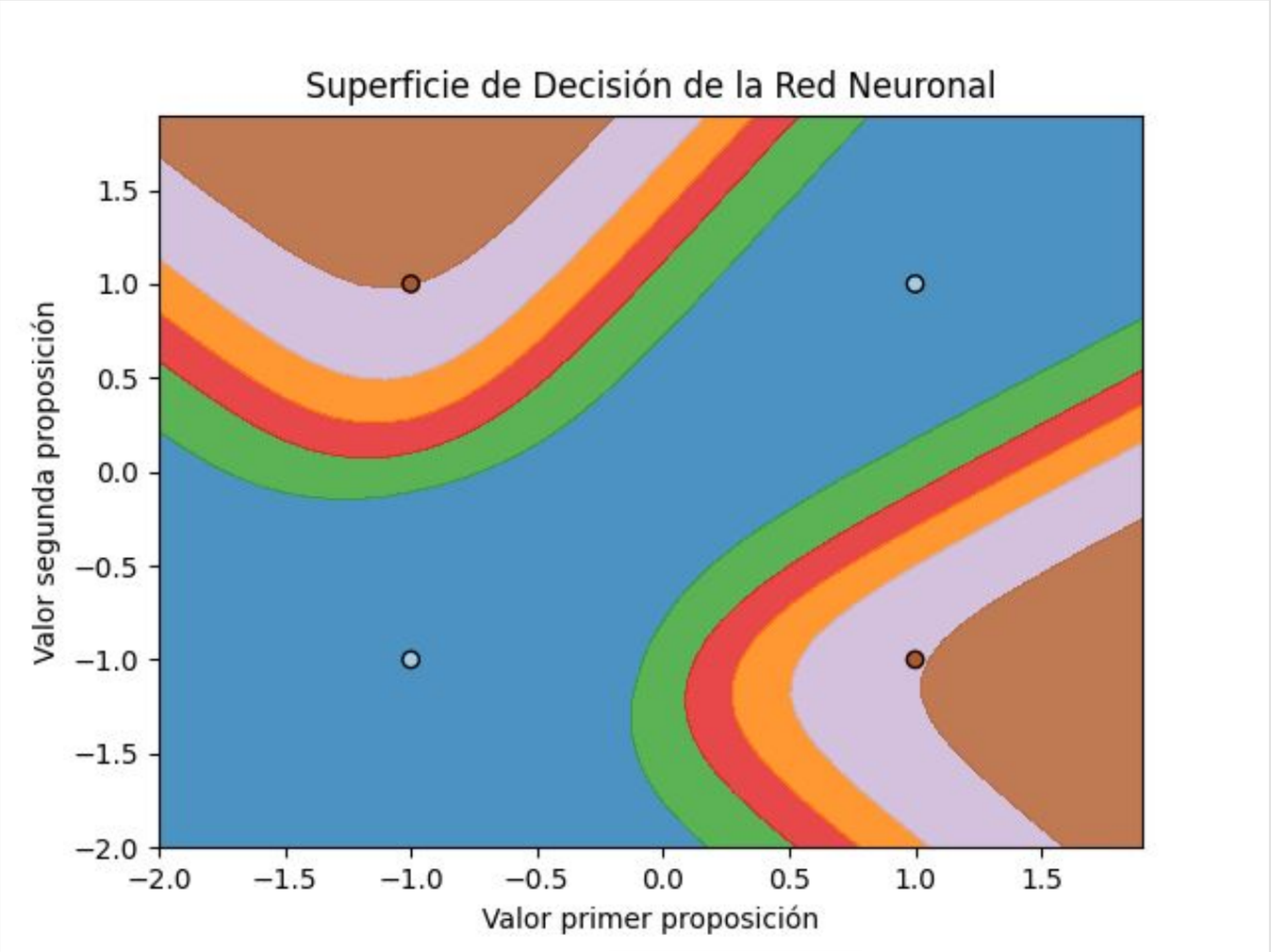

Superficie de Decisión de la Red Neuronal
Valor segunda proposición
Valor primer proposición
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
output:
1
-1

Gráfico error convergente con función XOR

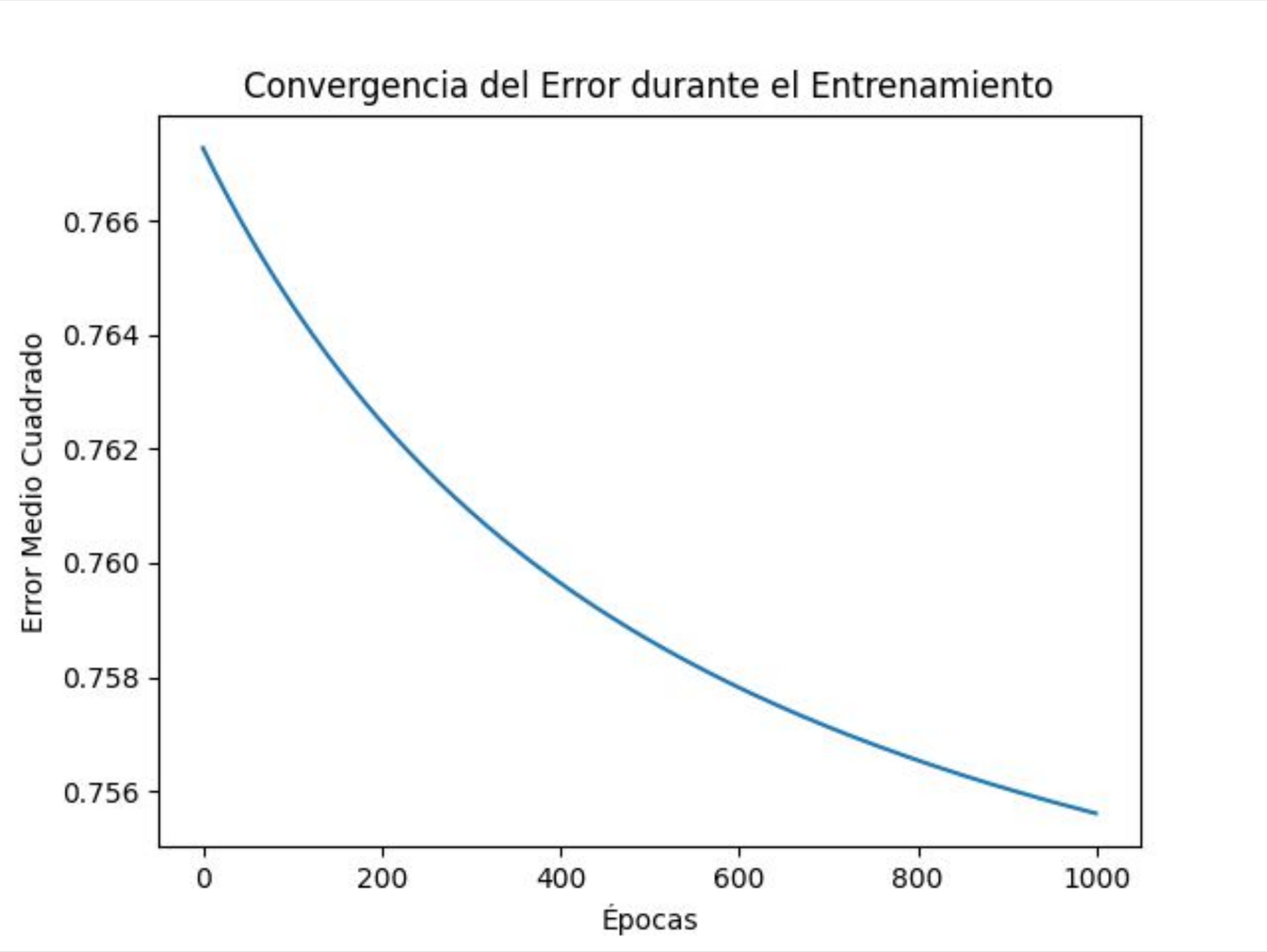


Gráfico Error de entrenamiento Lineal

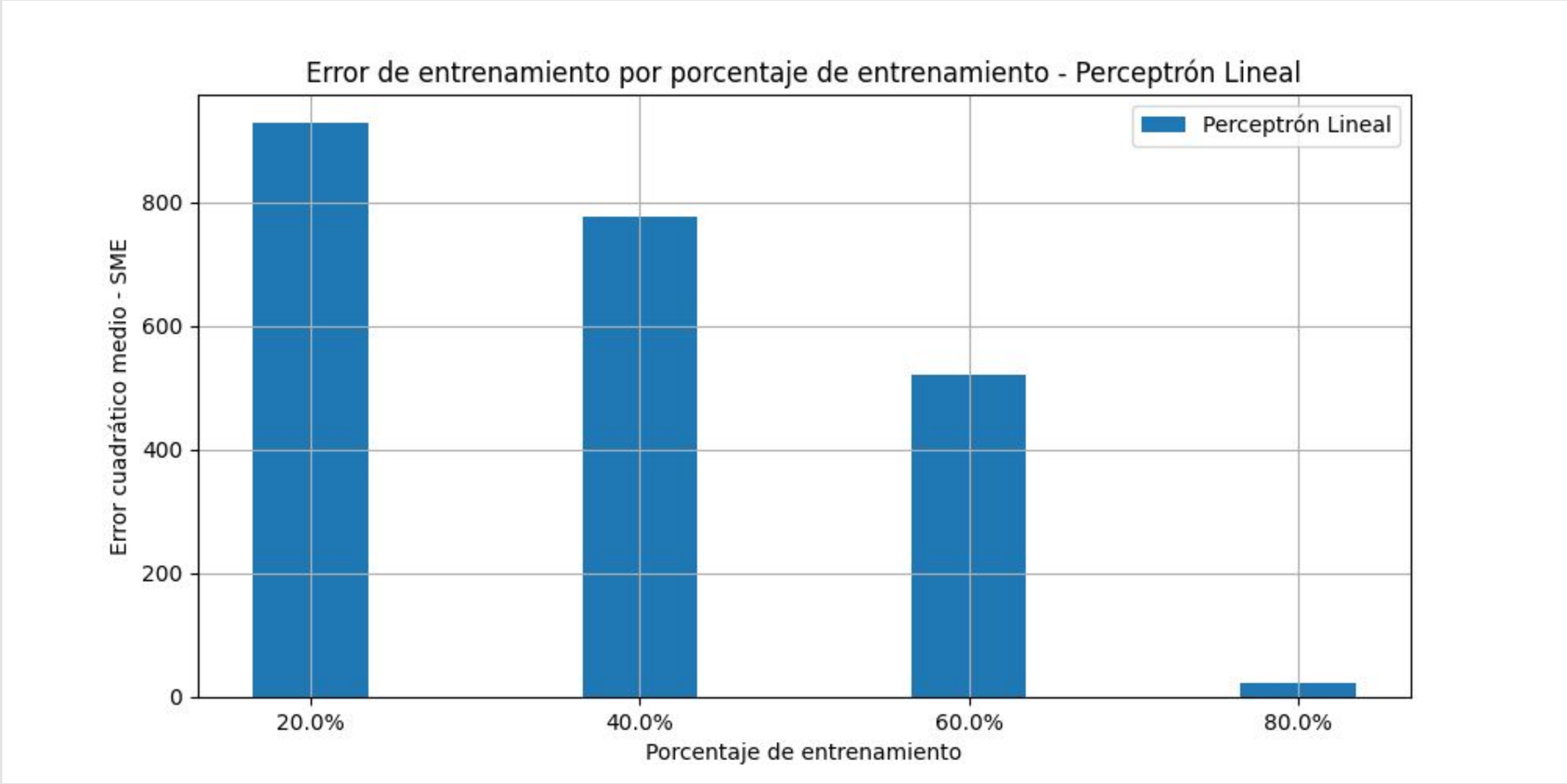


Gráfico Error de entrenamiento No Lineal

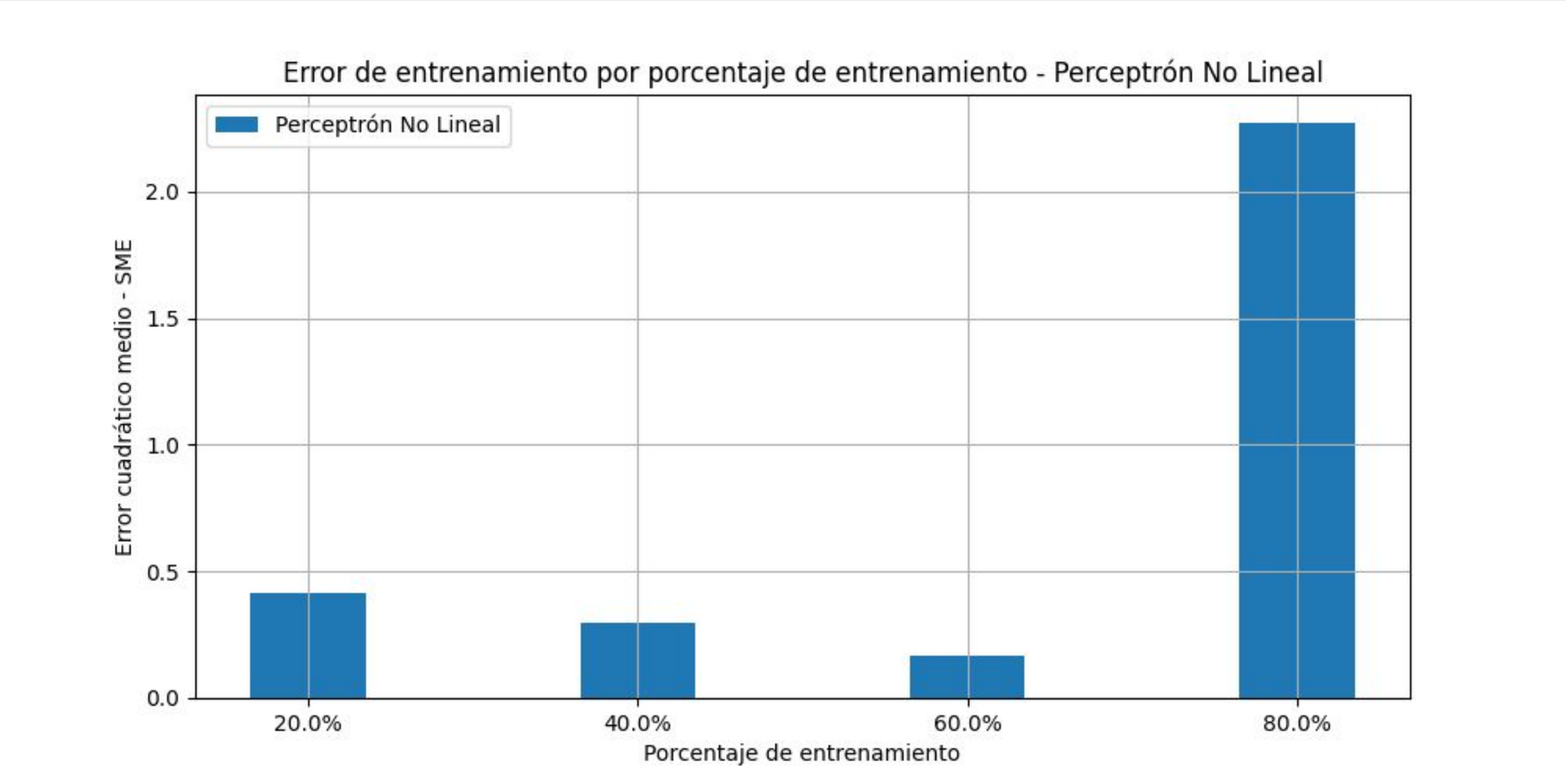
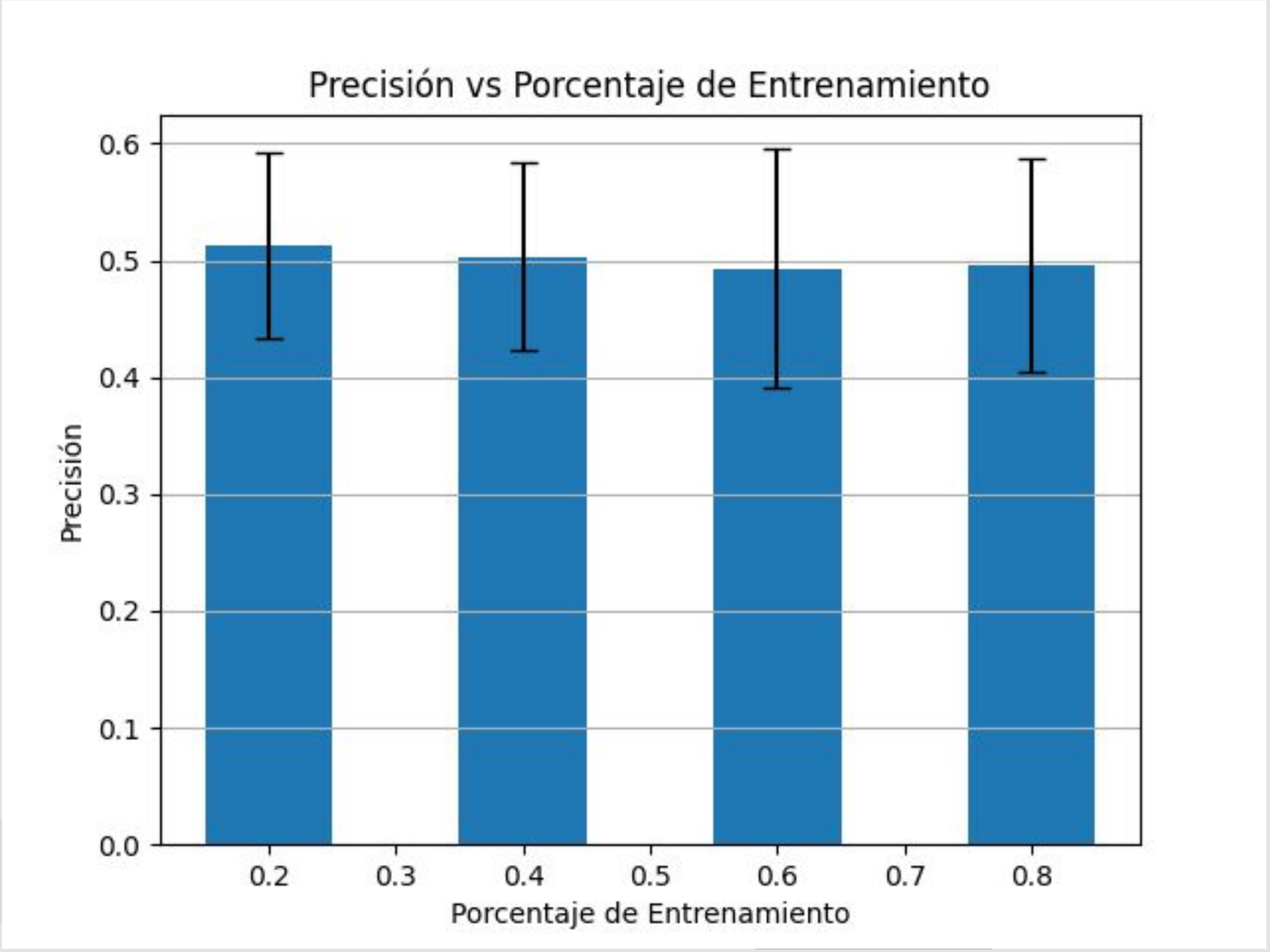


Gráfico Función Par

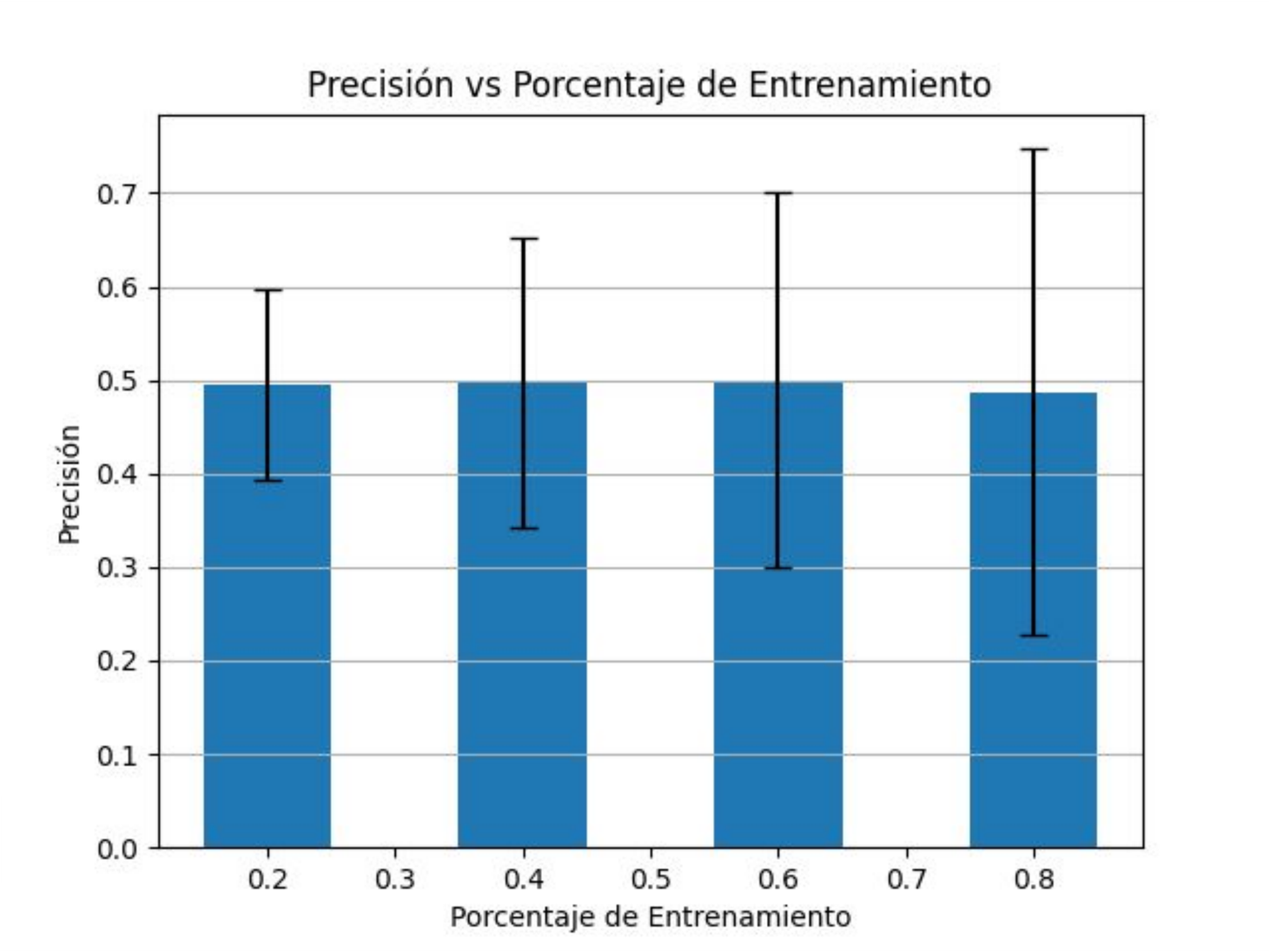


Config:

Optimizador: Gradiente descendente

learning_rate=0.001

Gráfico Función Par



Config:

Optimizador: Adam

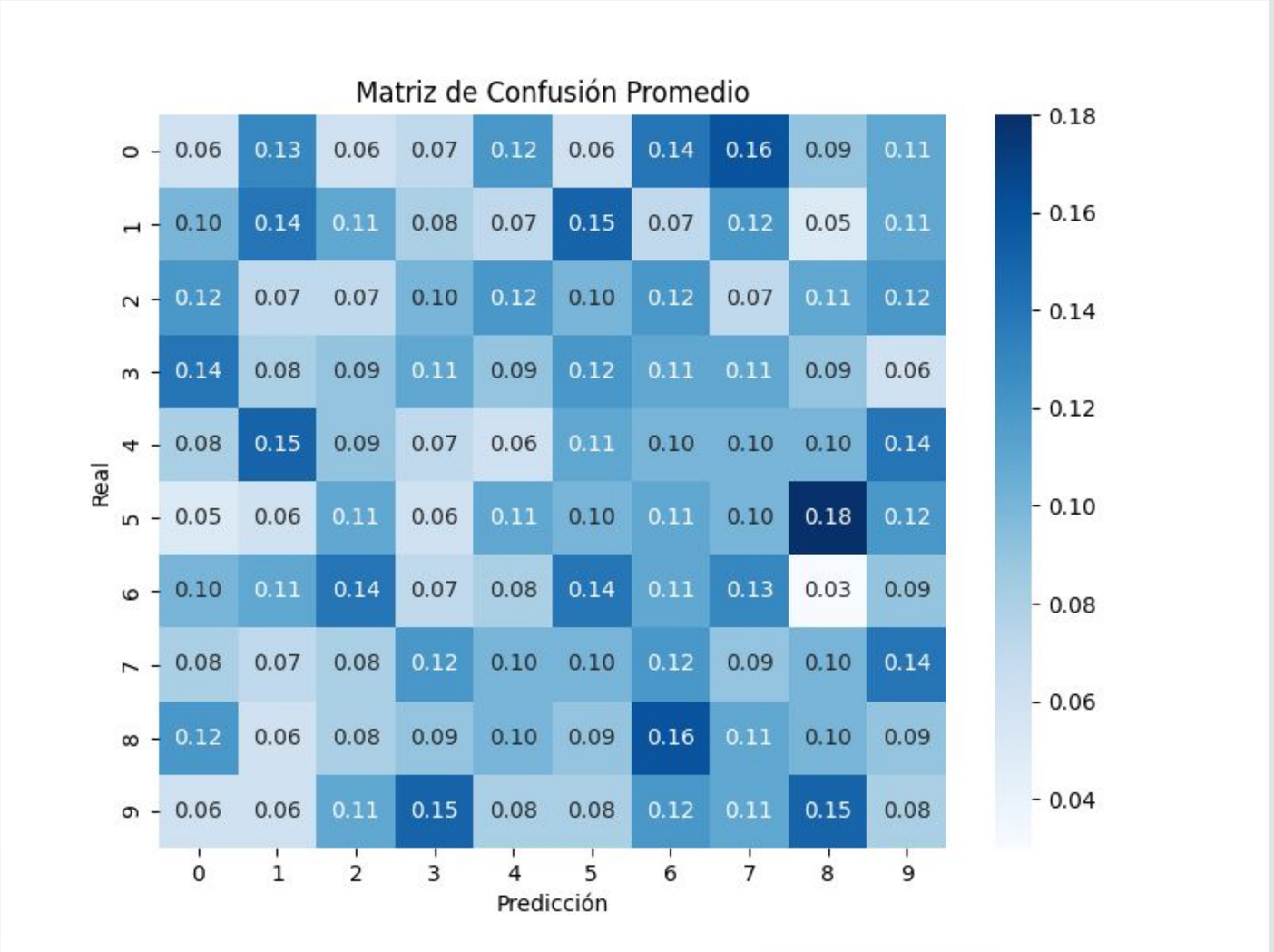
learning_rate=0.001,

beta1=0.9,

beta2=0.999,

epsilon=0.1

Gráfico Función Number con ruido



"max_epochs":

300

"learning_rate":

0.01

"Optimizador":

Gradiente

descendente

"epsilon": 0.1

"Neuronas": 10

Conclusiones

- Se puede representar la operación XOR usando dos neuronas de entrada que eventualmente convergen en una neurona de salida
- Un conjunto acotado de datos no permite que la red entrene bien
- Al tener un conjunto acotado de datos, podemos insertar ruido para modificarlo, agrandando el conjunto, y así permitir un mejor entrenamiento del perceptrón
- Se debe ir experimentando con la combinación de capas para conseguir la ideal
- En el ejercicio 3 (B y C) podemos observar que los resultados obtenidos son relativamente aleatorios


Gracias!