TP 3: Perceptrón Simple y Multicapa

Grupo 6

Alberto Abancens (62581)
Kevin Catino (61643)
Agustín Galarza (61481)
Abril Occhipinti (61159)
Maiwenn Boizumault (65988)
Agustin Benvenuto (61448)

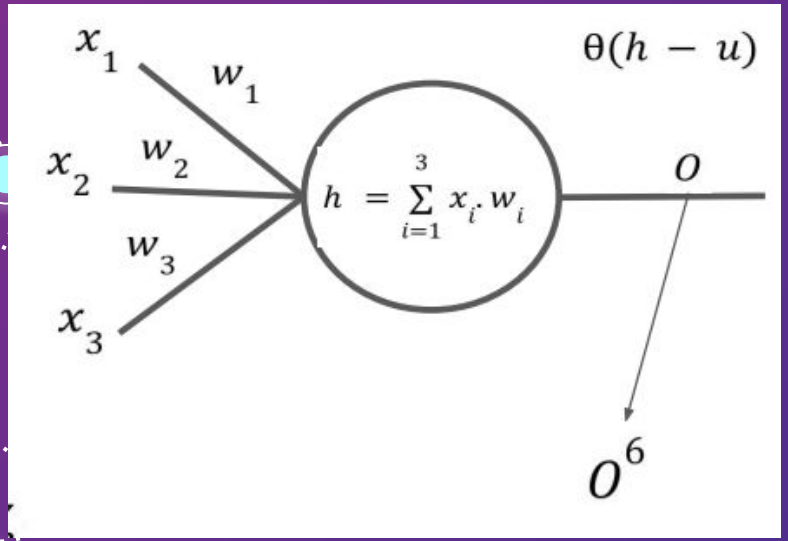
Contenidos

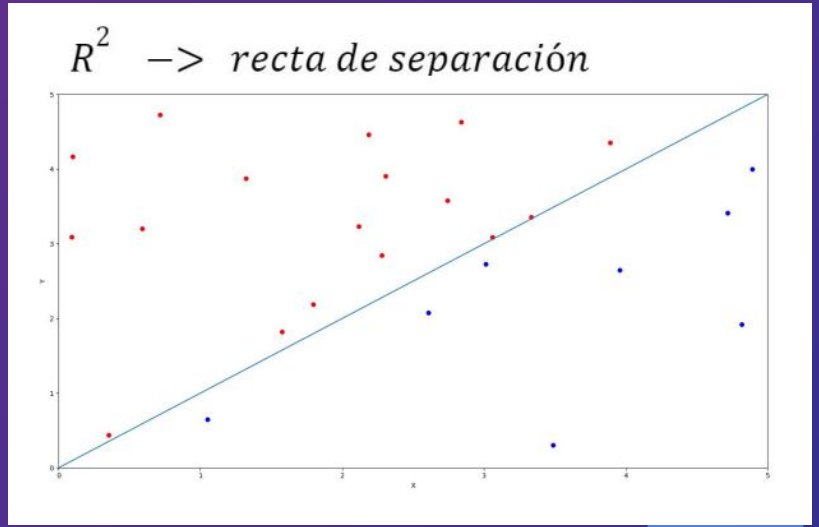
01

Ejercicio 1

Descripción del problema

Implementar el algoritmo de perceptrón simple con función de activación escalón y utilizar el mismo para aprender los siguientes problemas:





Descripción del problema

- Función lógica “Y” con entradas:
- $x = \{\{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\}\}$
- salida esperada: $y = \{-1, -1, -1, 1\}$

Entradas		Salida
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

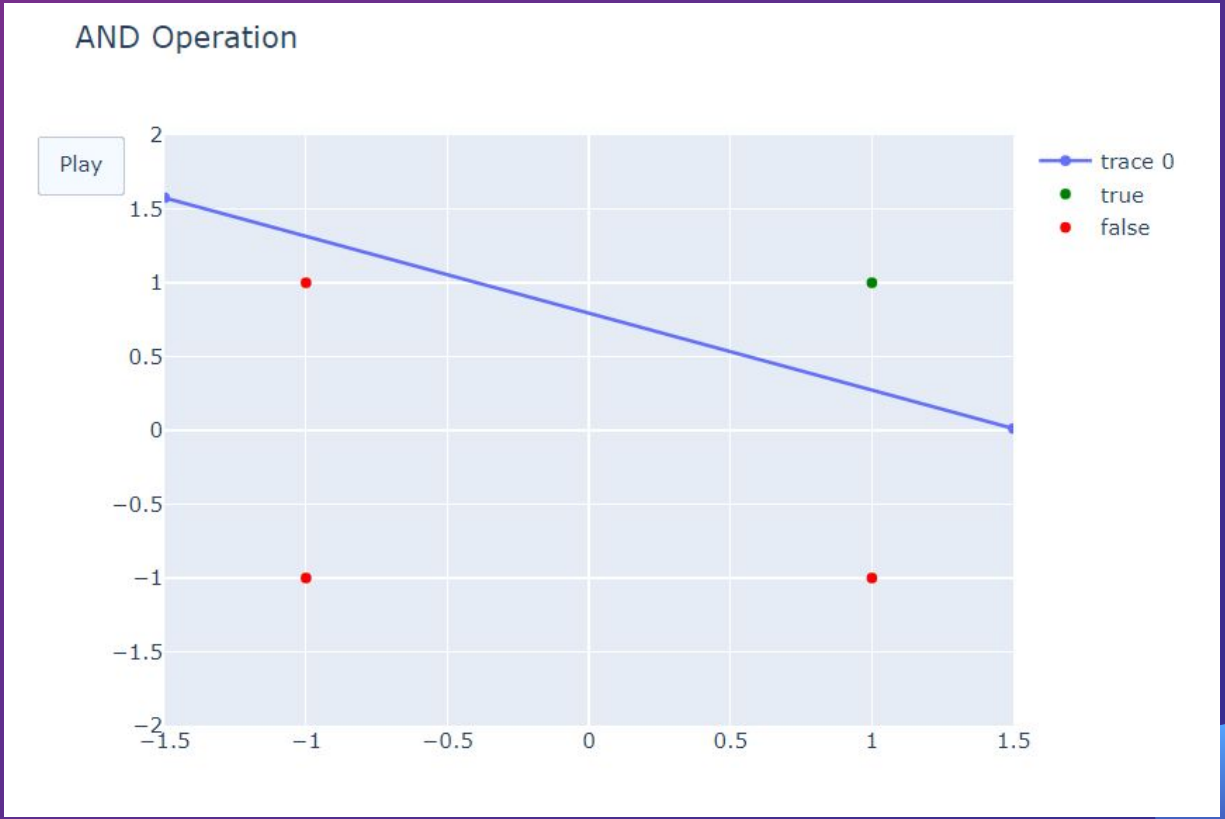
Descripción del problema

- Función lógica “O exclusivo” con entradas
- $x = \{\{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\}\}$
- salida esperada: $y = \{1, 1, -1, -1\}$

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Resultado operacion AND

learning_rate : 0.1

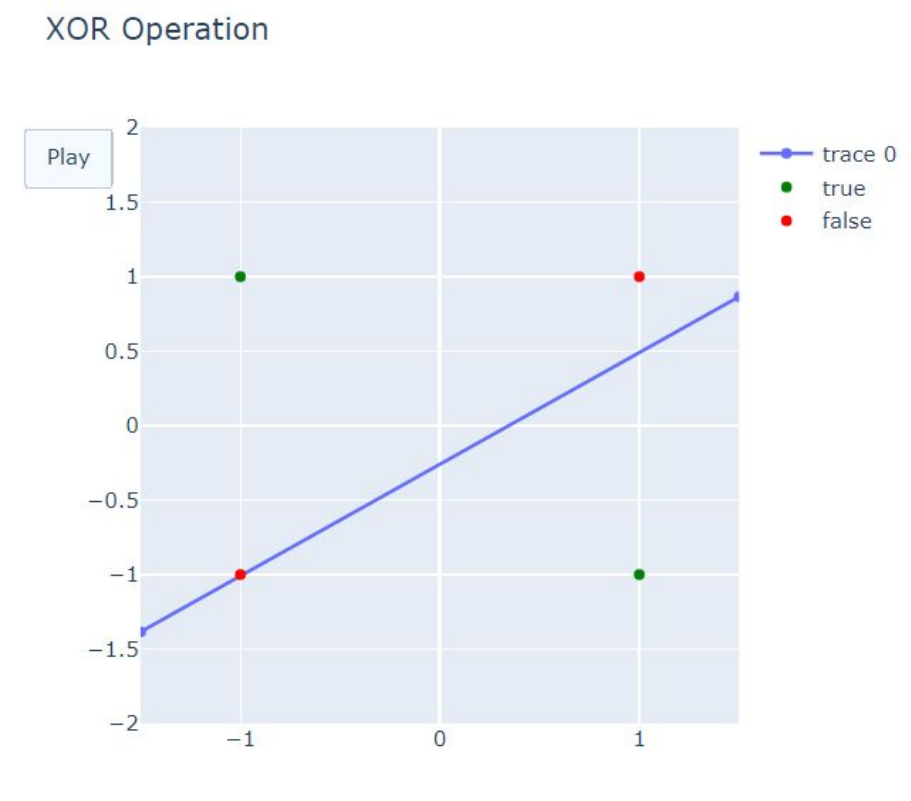


Video Operacion AND

learning_rate : 0.1

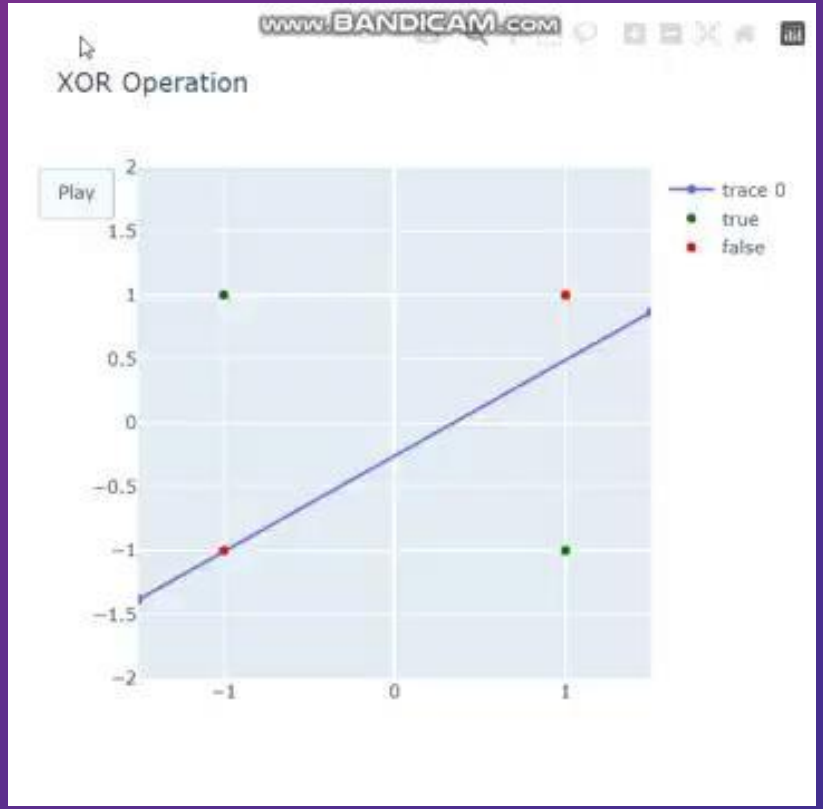
Resultado operacion XOR

learning_rate : 0.03



Video operacion XOR

learning_rate : 0.03



Resultados

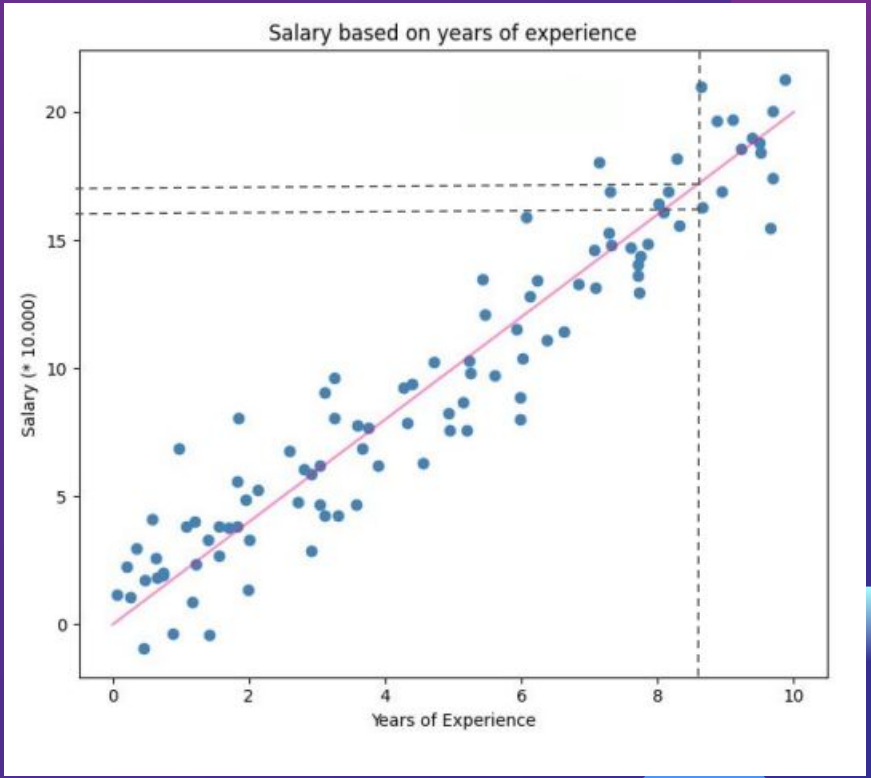
- Vimos gráficamente que la operación lógica AND es linealmente separable y por ello el perceptrón simple encuentra una solución
- Por otro lado la operación lógica XOR resultó ser no linealmente separable y por ello el perceptrón simple no encuentra una solución

02

Ejercicio 2

Perceptron Lineal

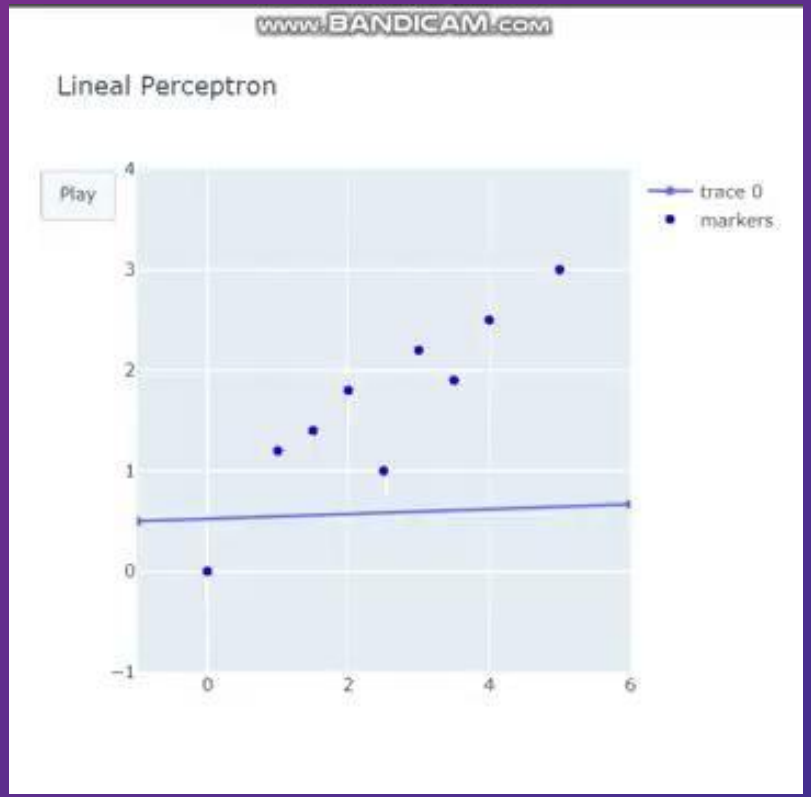
- Cambia la funcion de activacion
- Cambian las actualizaciones de los pesos
- Se parece a una regresión lineal
- O(h) = h



Perceptron Lineal

Ejemplo de juguete

learning_rate : 0.01



Perceptron Lineal - Ej2.1

El conjunto “TP2-ej2-conjunto.csv” tiene 3 dimensiones de entrada por lo que no podemos visualizar el problema espacialmente

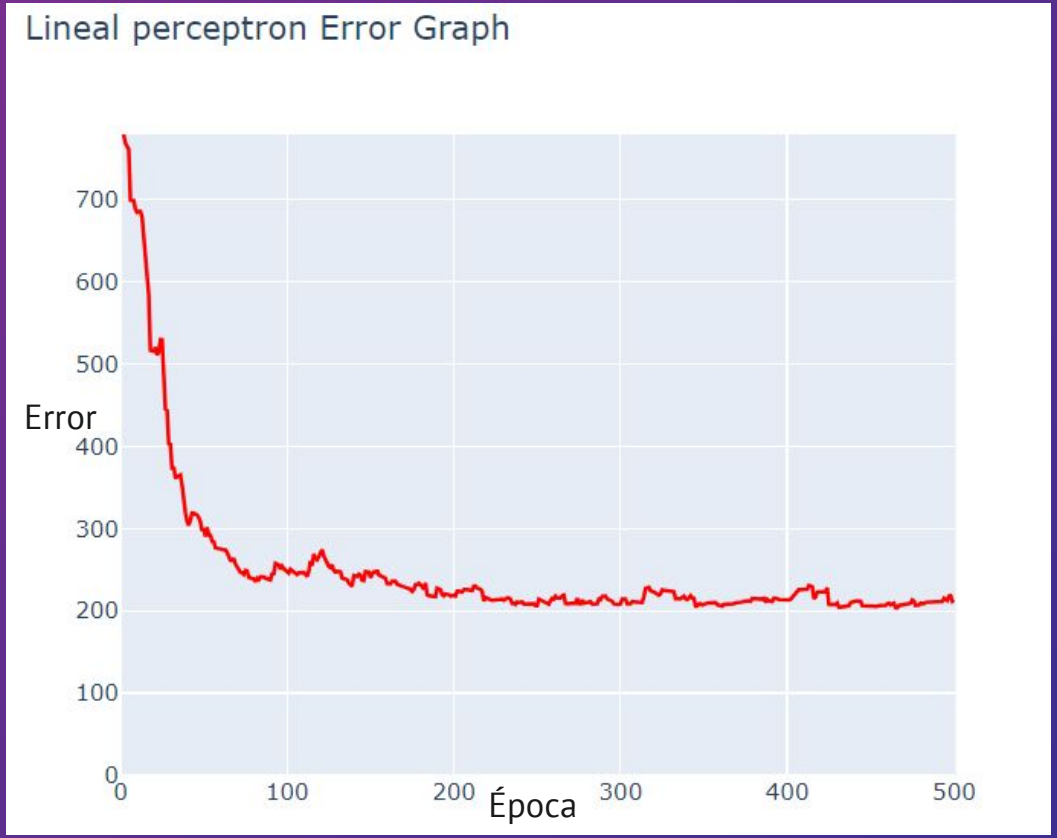
Visualizamos el aprendizaje a través de la evolución del error

$$E(O) = \frac{1}{2} \sum_{\mu=0}^{p-1} (\zeta^{\mu} - O^{\mu})^2$$

Perceptron Lineal - Error

learning_rate : 0.03

Min value: 204.40

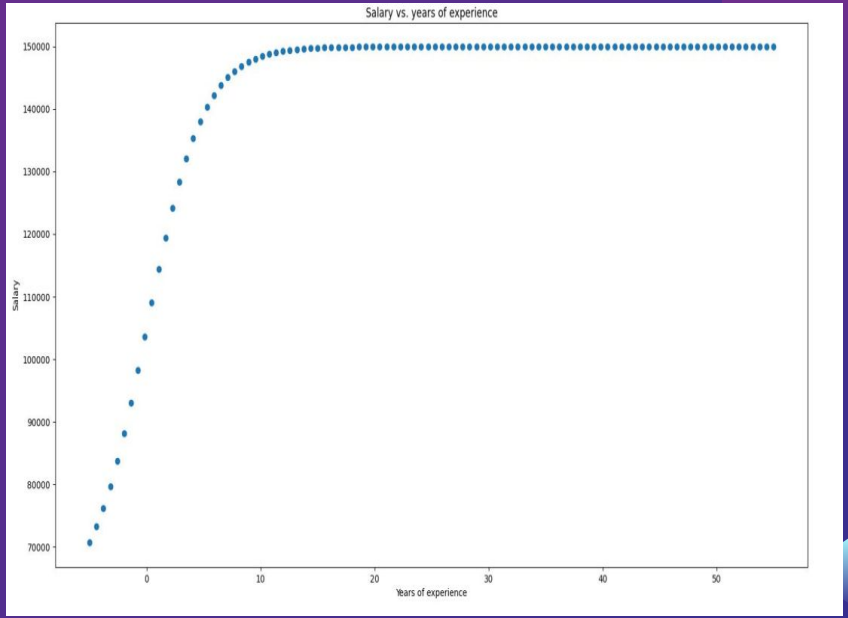


Perceptron No lineal

- Si la distribución tiene esta pinta la función identidad resulta limitada
- Cambio funcion de activacion y su derivada por una sigmoidea
- Elegimos:

$$\theta(h) = tanh(\beta h)$$

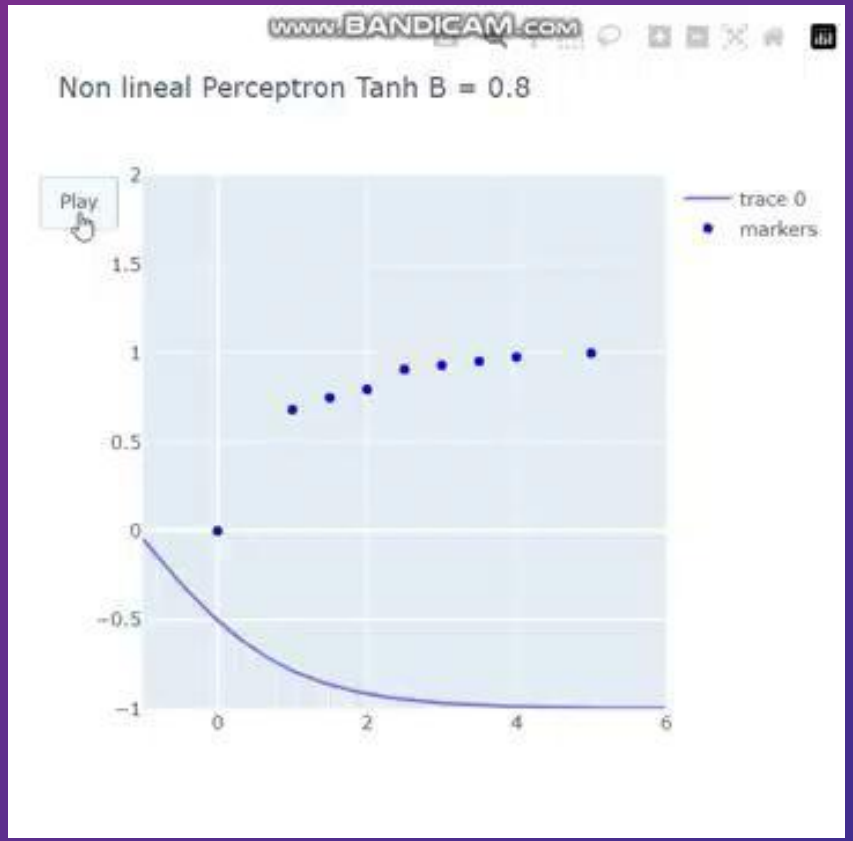
$$\theta'(h) = \beta(1 - \theta^2(h))$$



Perceptron No lineal

Ejemplo de juguete

learning_rate : 0.01



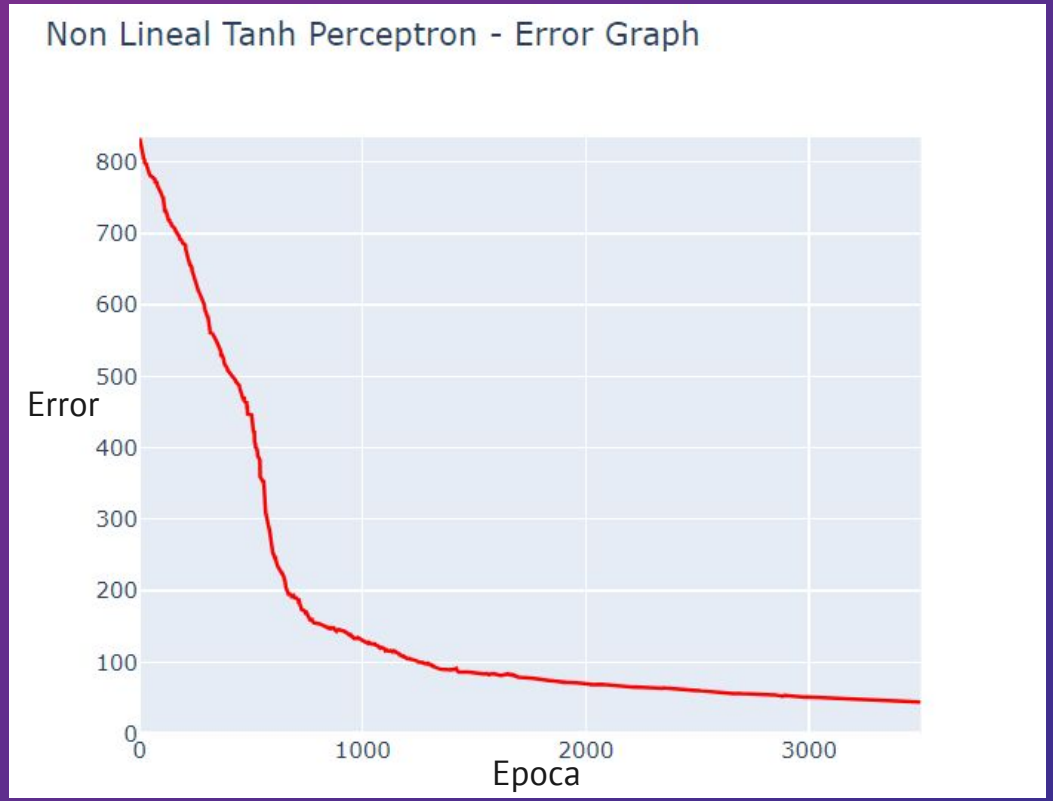
Perceptron No lineal - EJ2.1

- El conjunto de datos tiene valores de salida en R
- La función tanh tiene salida en [-1, 1]
- Por lo tanto escalamos los datos entre -1 y 1
- Luego para calcular el error lo volvemos a los valores originales
- Para poder comparar con el lineal

Perceptron No lineal - EJ2.1

learning_rate : 0.01

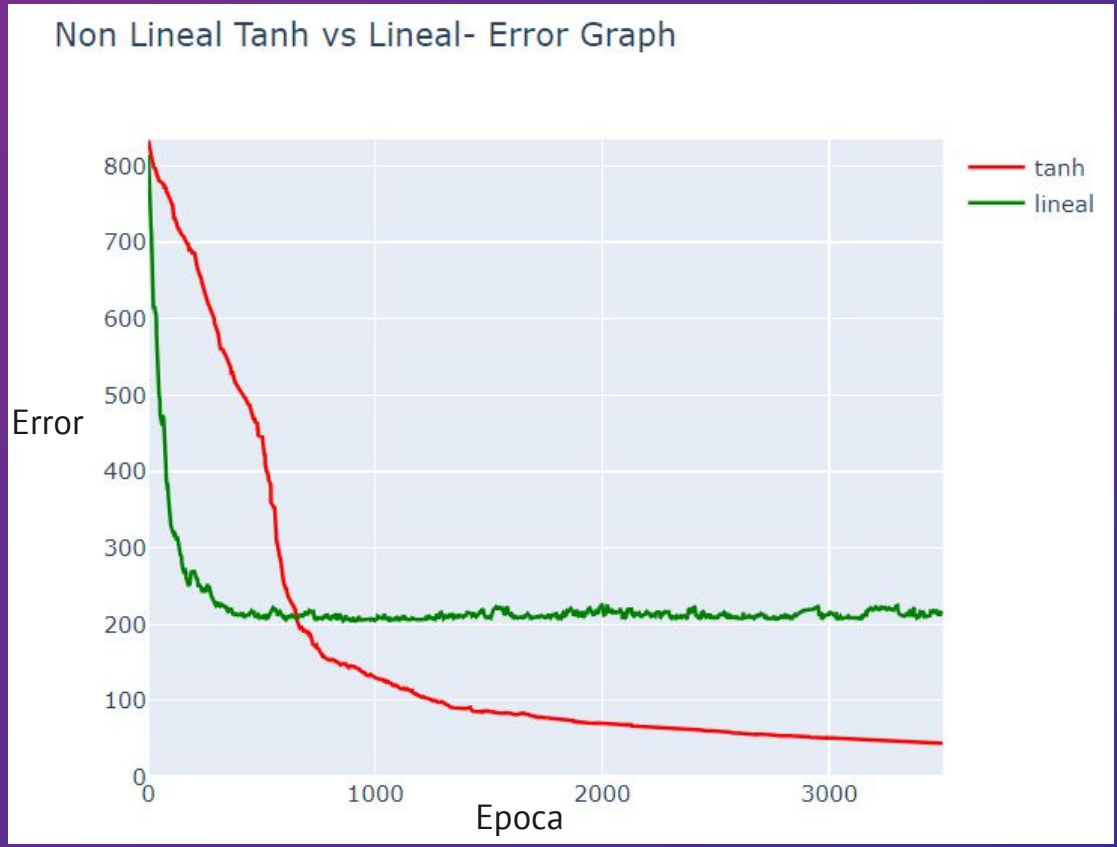
Min_value: 39.07



Perceptron No lineal vs Lineal - EJ2.1

Comparativo

learning_rate : 0.01



Perceptron No lineal - EJ2.1
Train vs Test

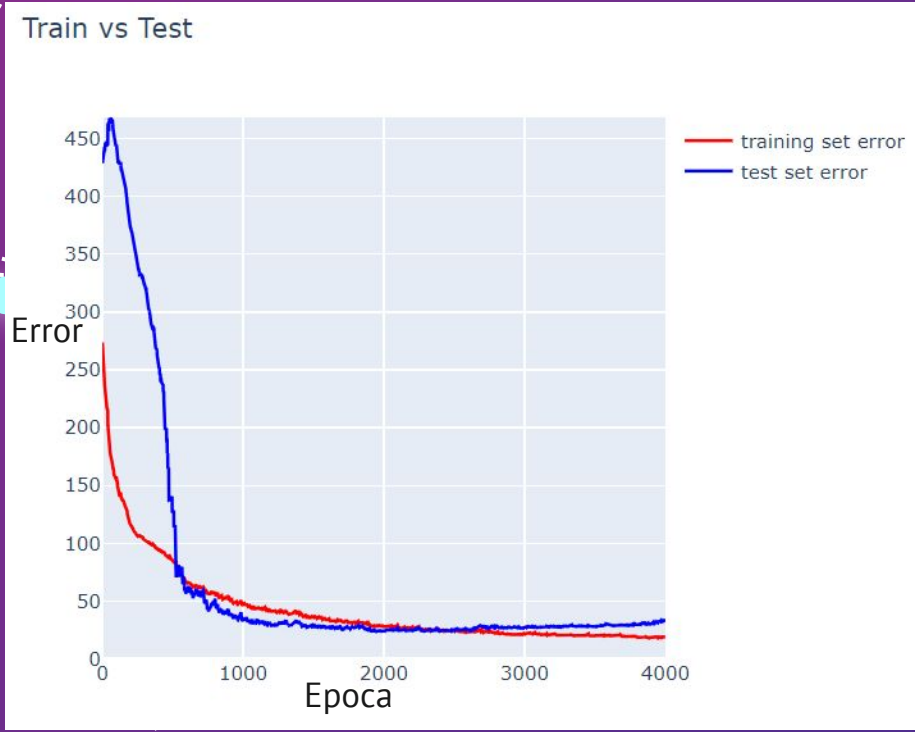
- Definimos P: Proporcion del dataset que le corresponde a Train
- Al test set le corresponde 1 - P

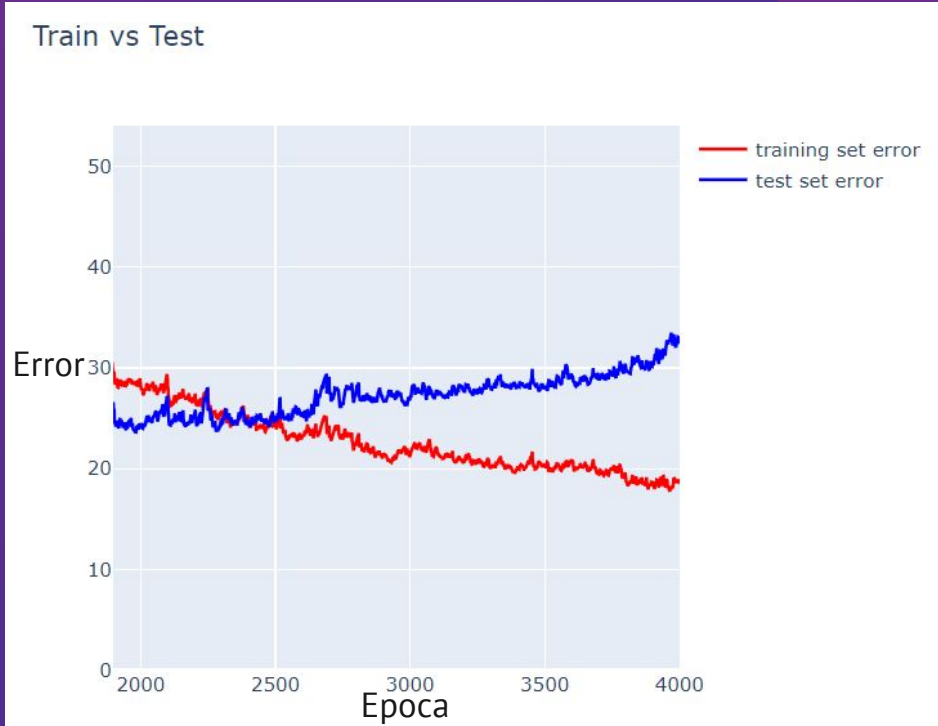
Perceptron No lineal Train vs Test

learning_rate : 0.01

P = 0.3

Min_generalize = 24.11



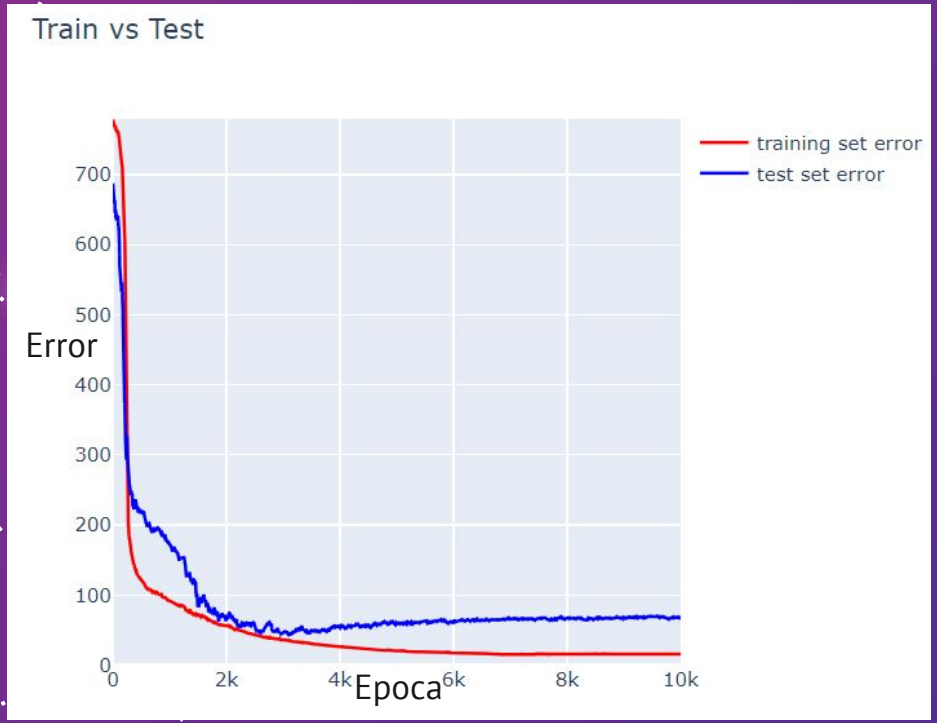


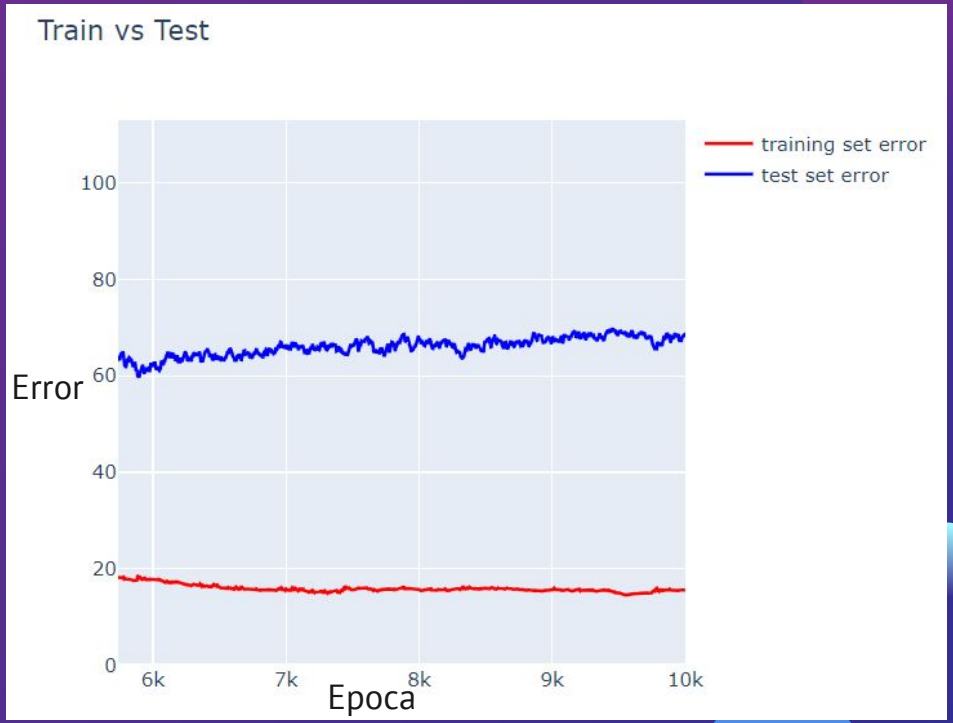
Perceptron No lineal Train vs Test

learning_rate : 0.01

P = 0.5

Min_generalize = 42.67



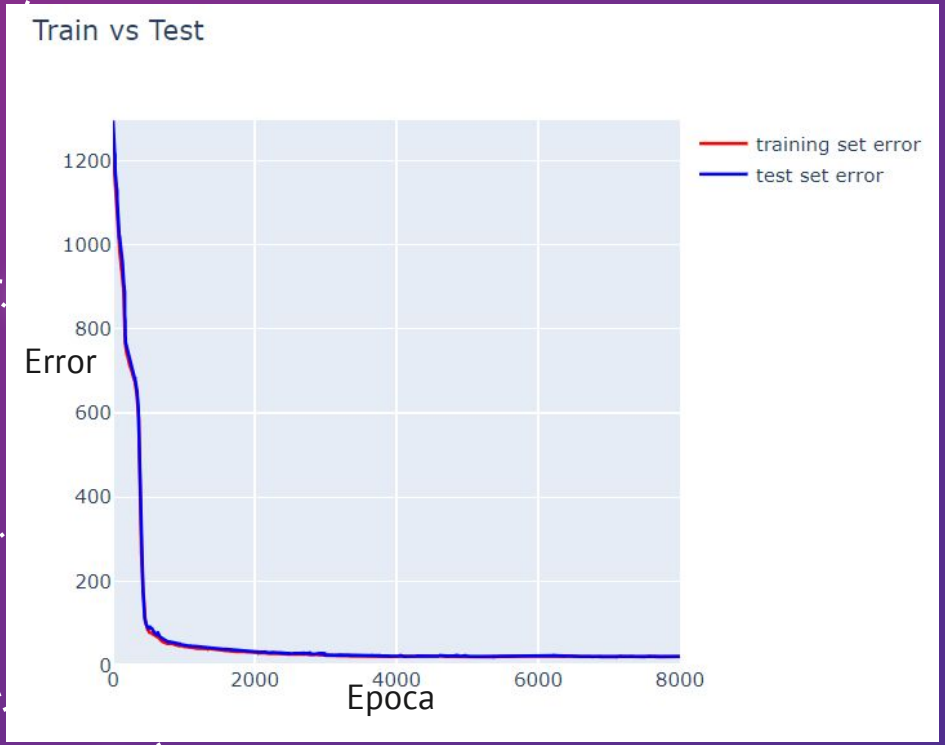


Perceptron No lineal Train vs Test

learning_rate : 0.01

P = 0.8

Min_generalize = 19.90



Resultados

- Un training set pequeño con poca variedad de datos resulta en una mala generalización del modelo GIGO (Garbage in Garbage Out)
- El perceptrón no lineal es considerablemente mejor para resolver el problema en cuestión comparado al lineal. Variando el valor de B se puede transformar en un escalón o en una recta

03

Ejercicio 3

Parte A

Implementar el algoritmo de perceptrón multicapa y utilizarlo para aprender con la Función lógica "O exclusivo".

Conjunto de Entrenamiento: Subconjunto de [[-1,-1],[-1,1],[1,-1],[1,1]]

Conjunto de Testeo: Subconjunto de [[-1,-1],[-1,1],[1,-1],[1,1]]

Layers: [4]

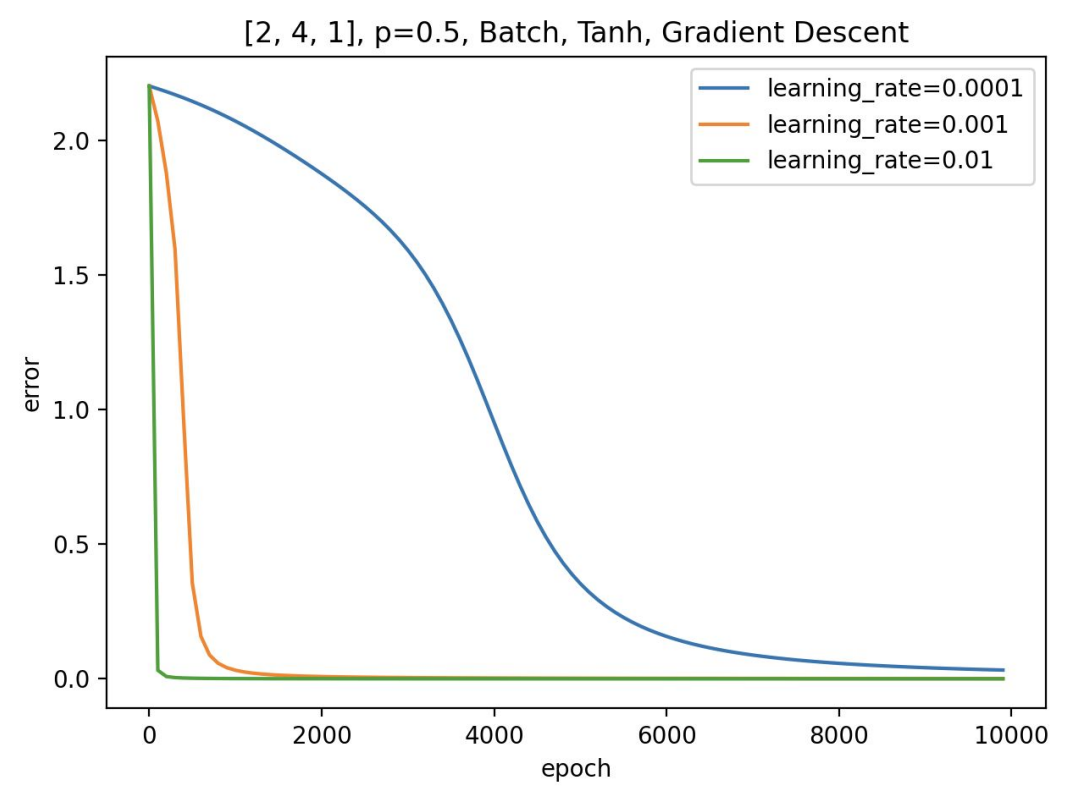
Método de Optimización: Gradiente Descendente / ADAM

Función de Activación: Sigmoid / Tanh

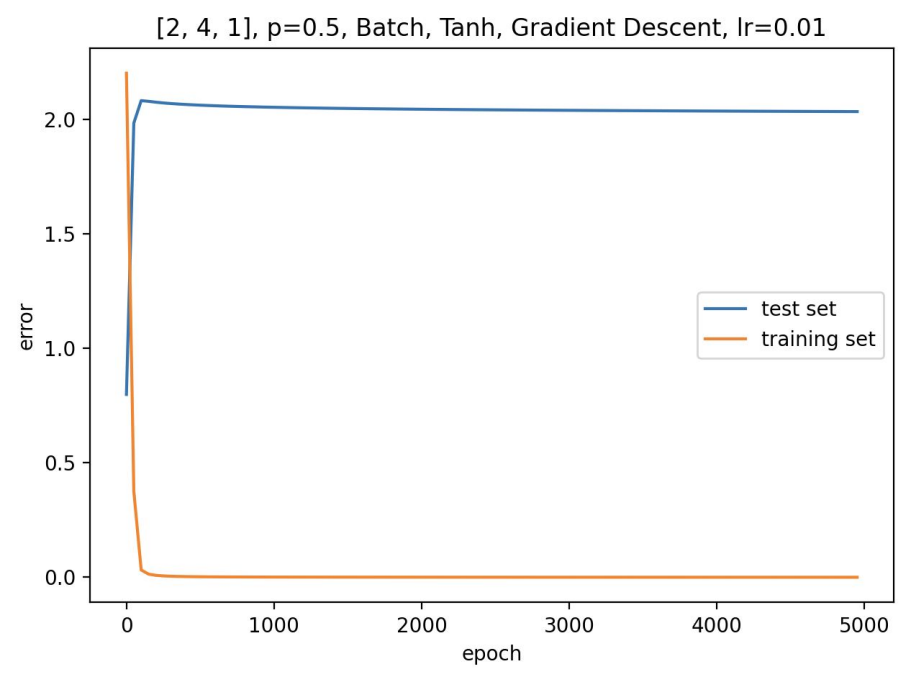
Learning rate: 0.01/0.001/0.0001

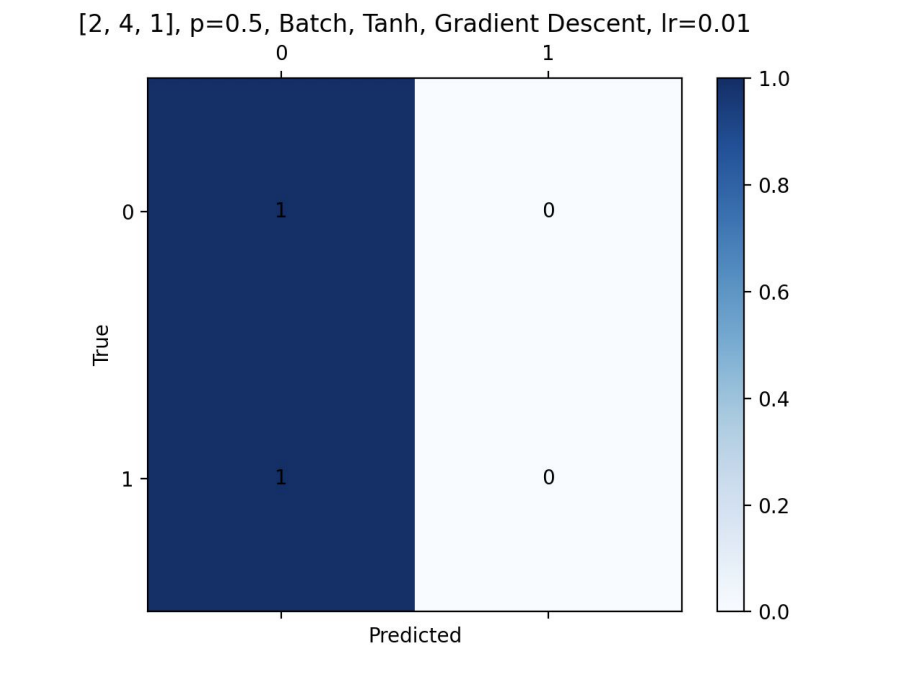
Los conjuntos de entrenamiento y testeo se definen a partir de una proporción de entrenamiento p en el intervalo [0,1]

p = 0.5, Grad. desc

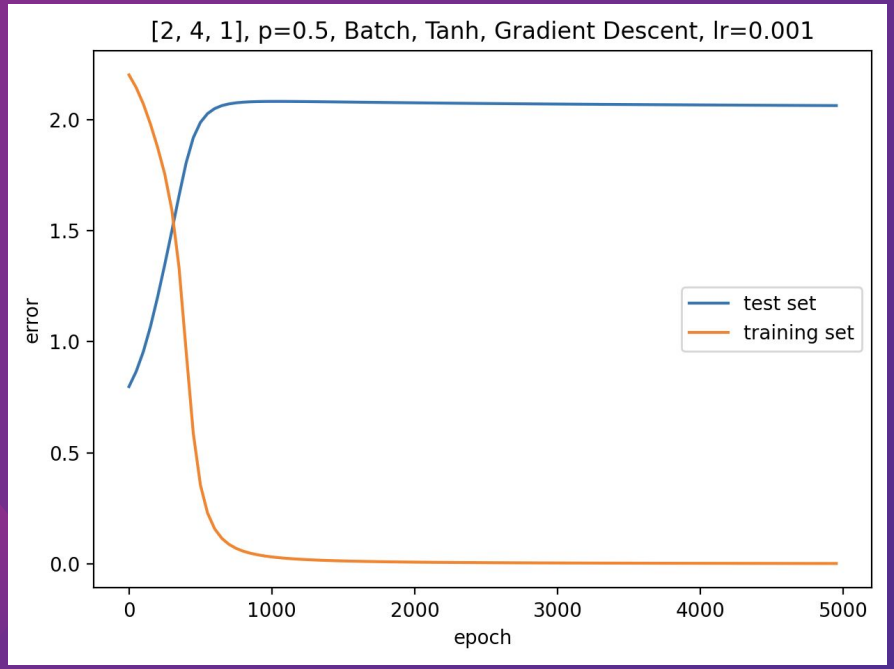


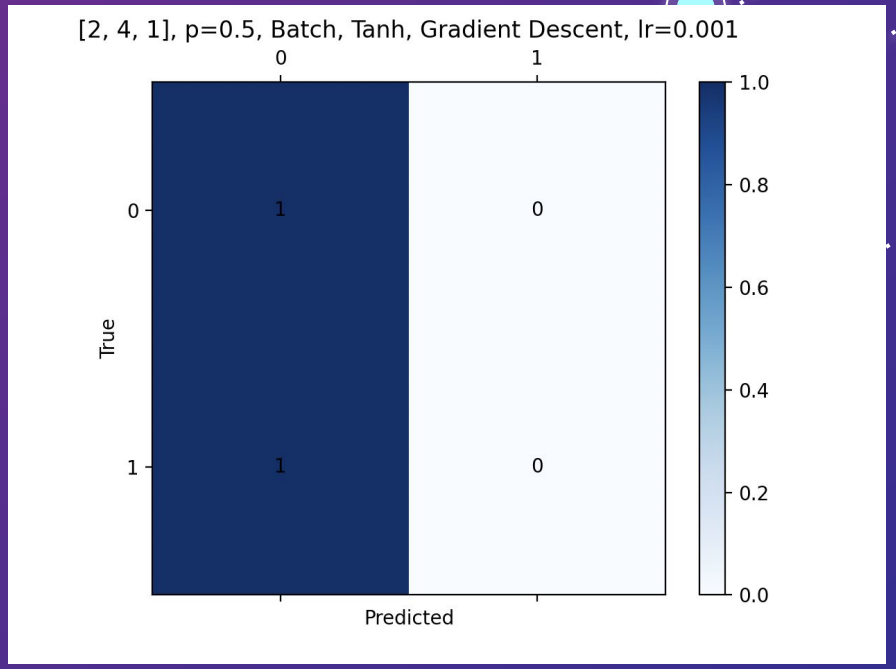
p = 0.5, Grad. desc, lr=0.01



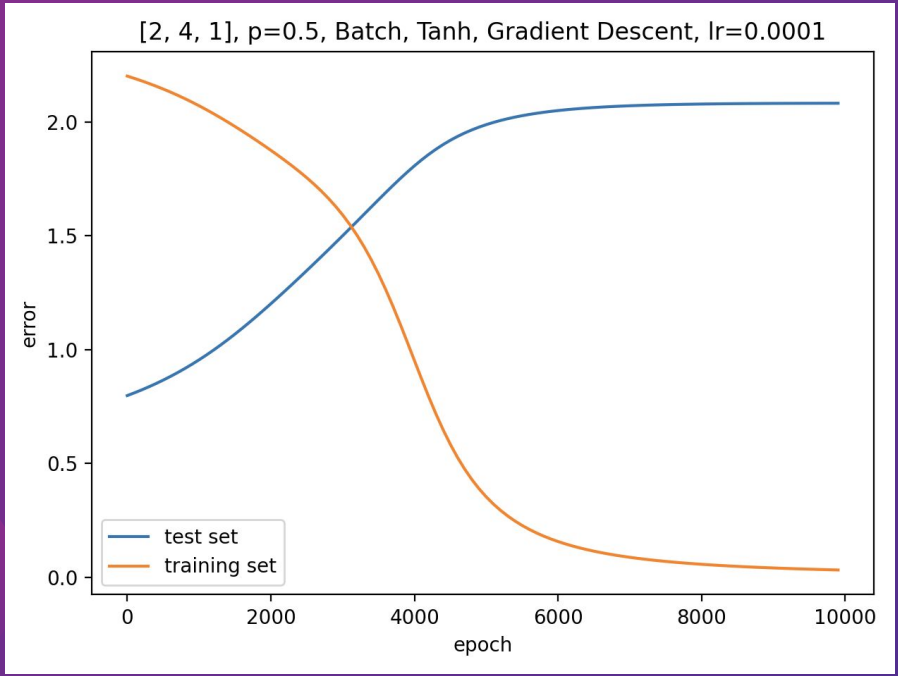


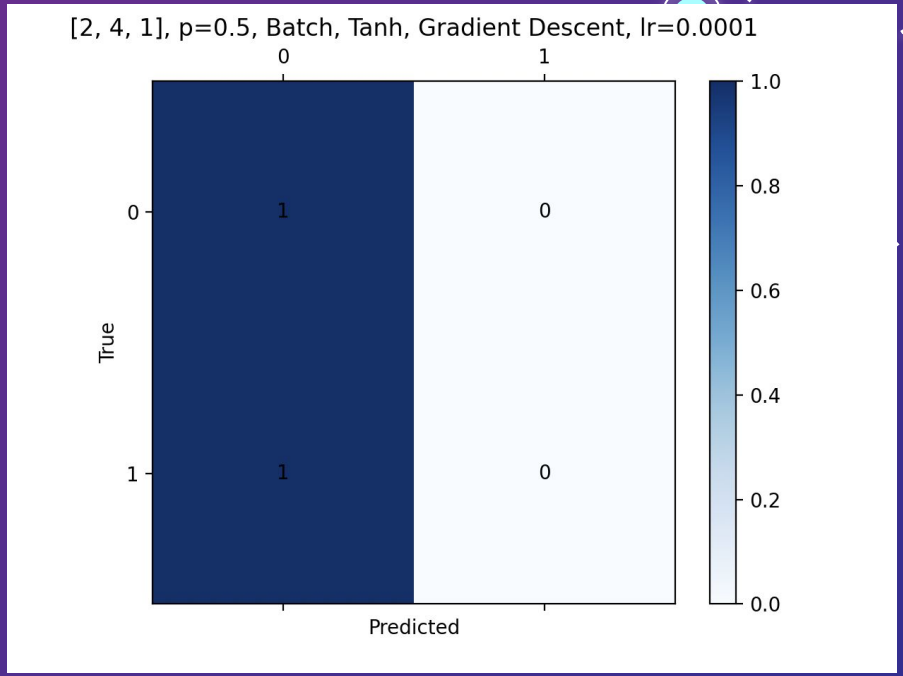
p = 0.5, Grad. desc, lr=0.001



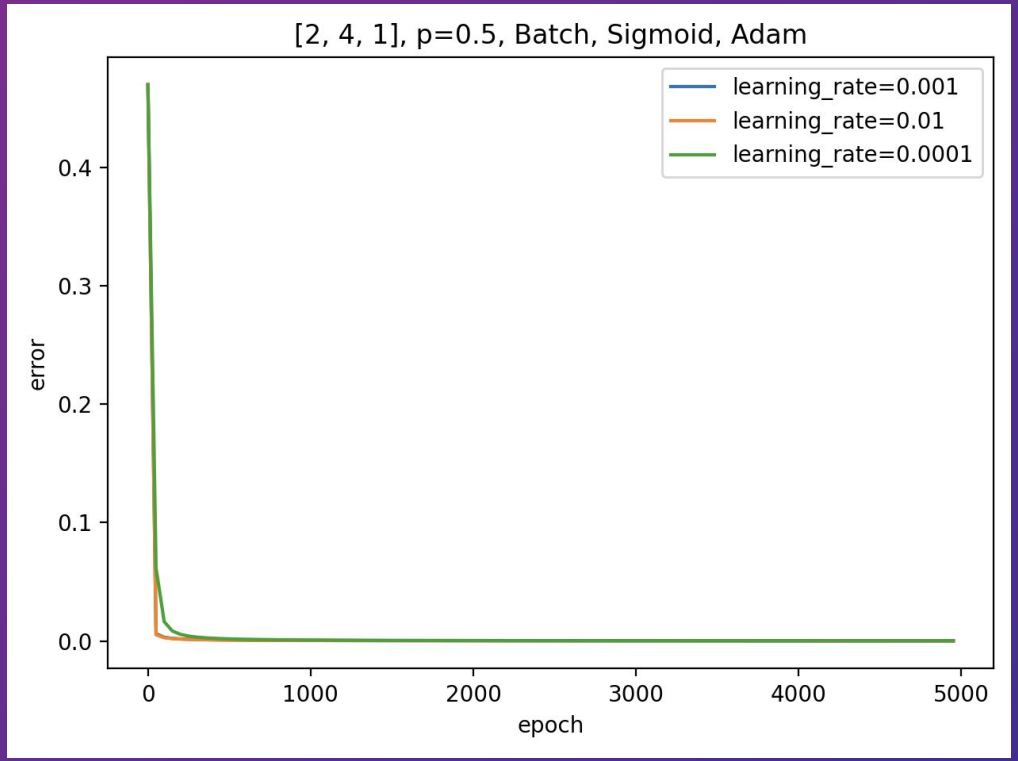


p = 0.5, Grad. desc, lr=0.0001

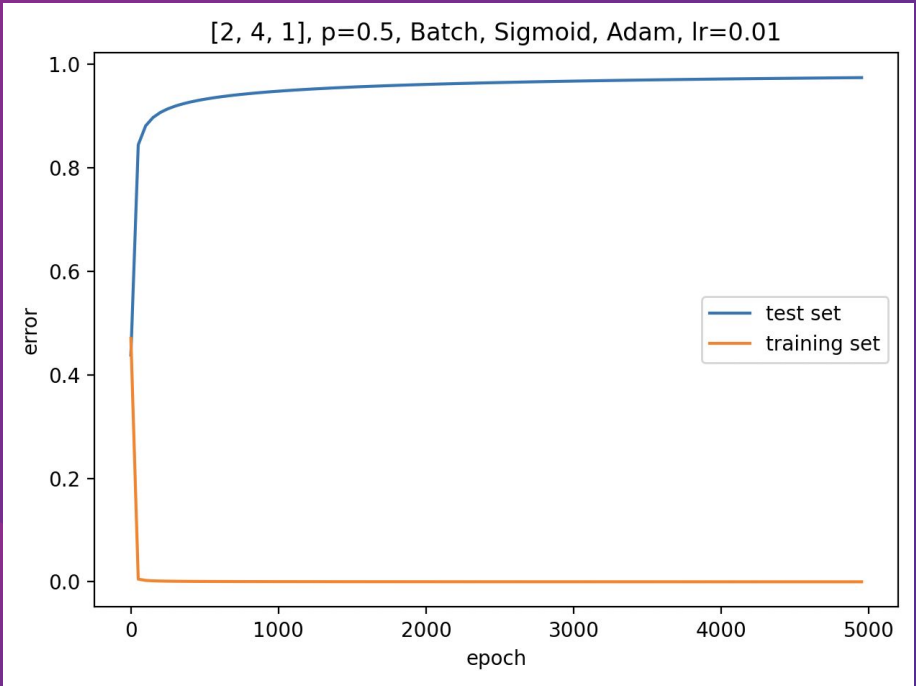


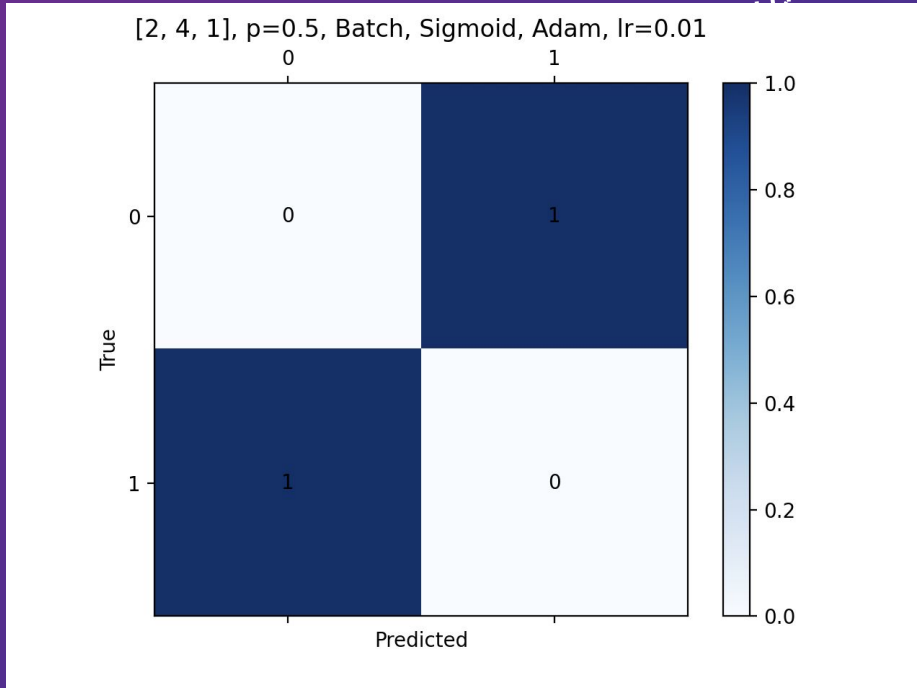


p = 0.5, ADAM

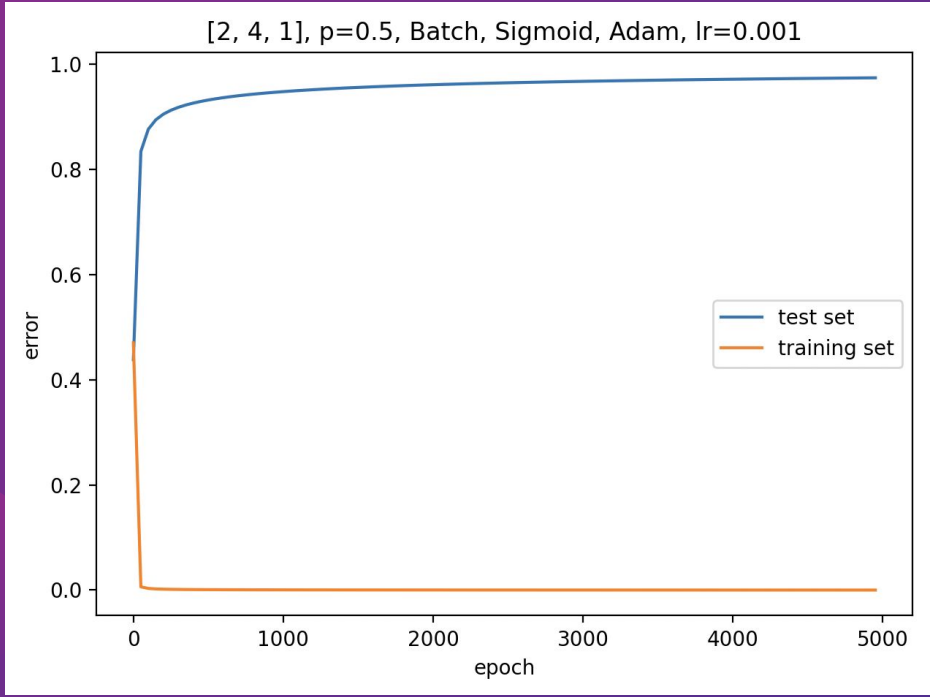

[2, 4, 1], p=0.5, Batch, Sigmoid, Adam
learning_rate=0.001
learning_rate=0.01
learning_rate=0.0001
error
0.0
0.1
0.2
0.3
0.4
0
1000
2000
3000
4000
5000
epoch

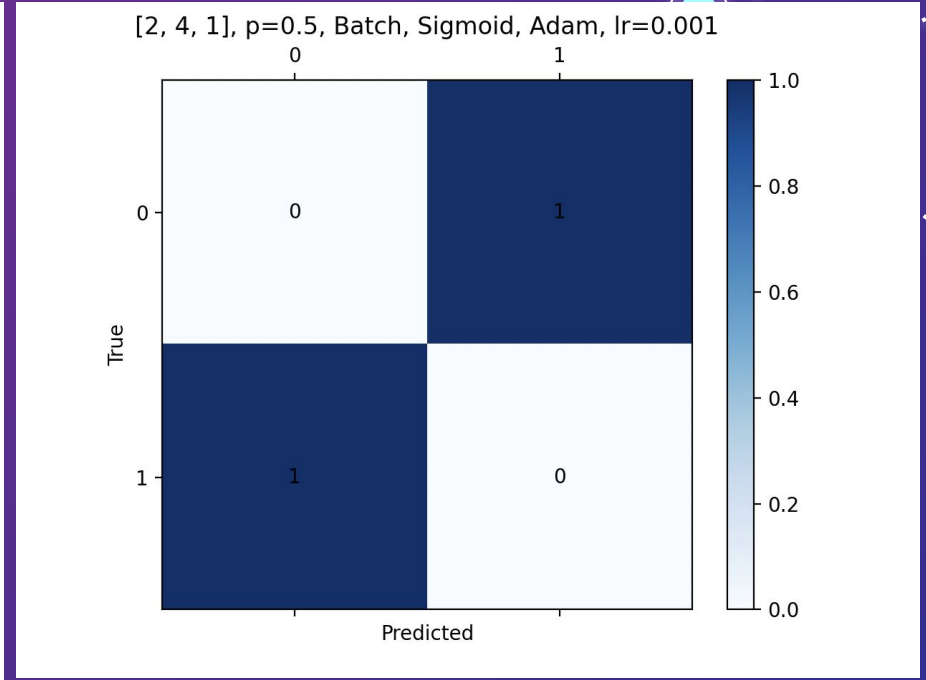
p = 0.5, ADAM, lr=0.01



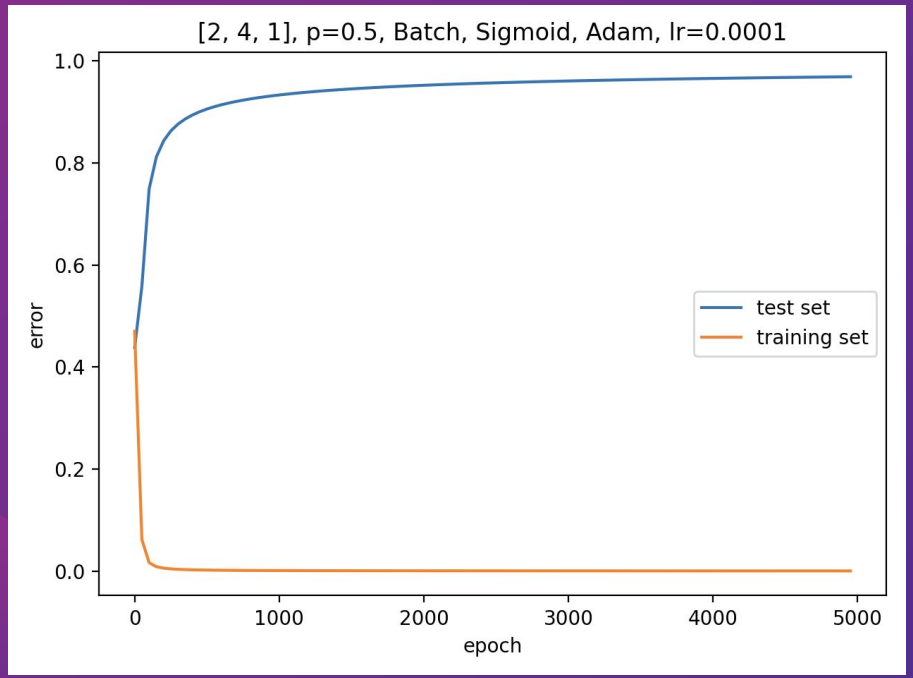


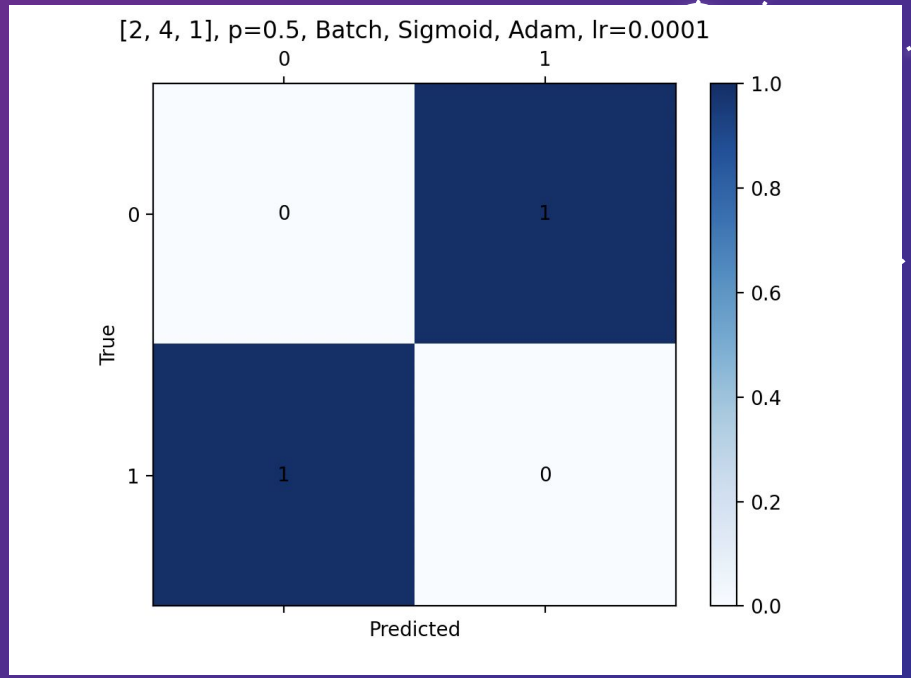
p = 0.5, ADAM, lr=0.001



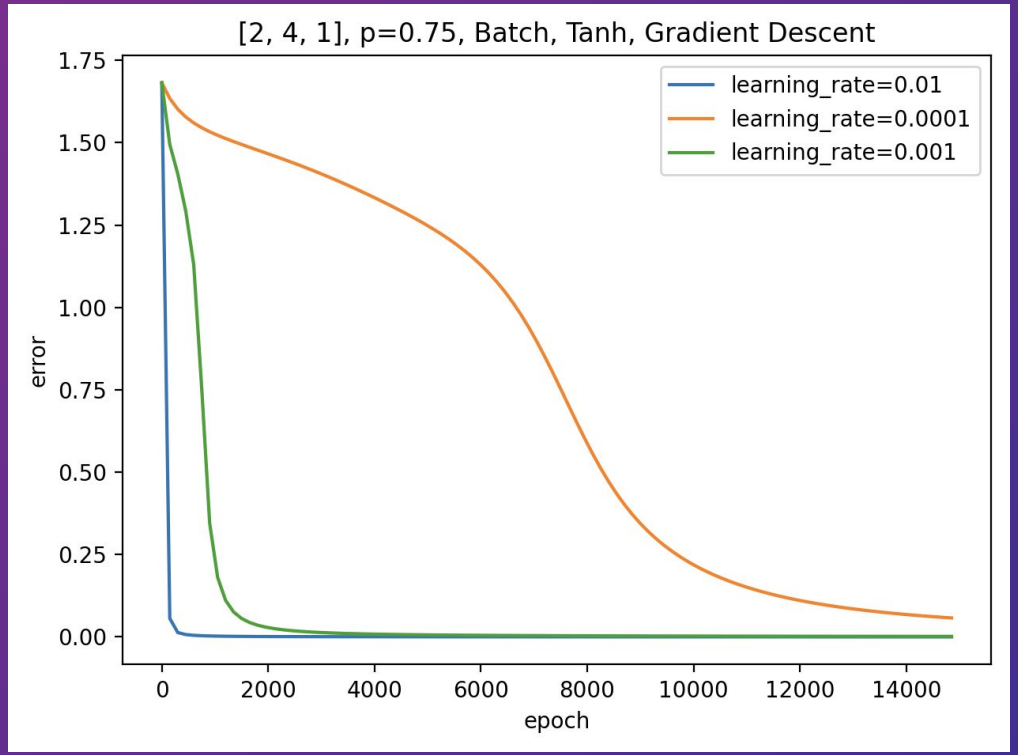


p = 0.5, ADAM, lr=0.0001

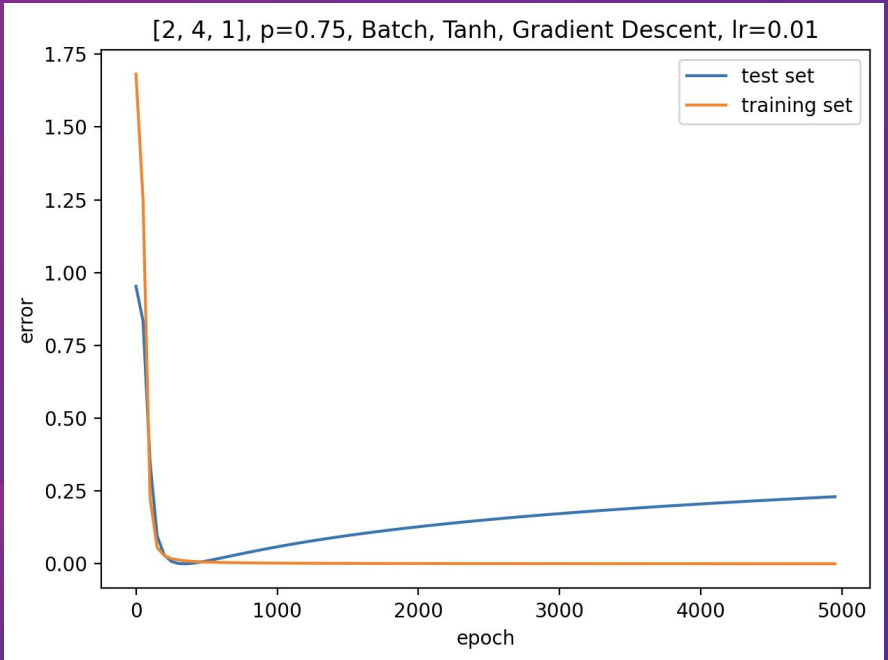


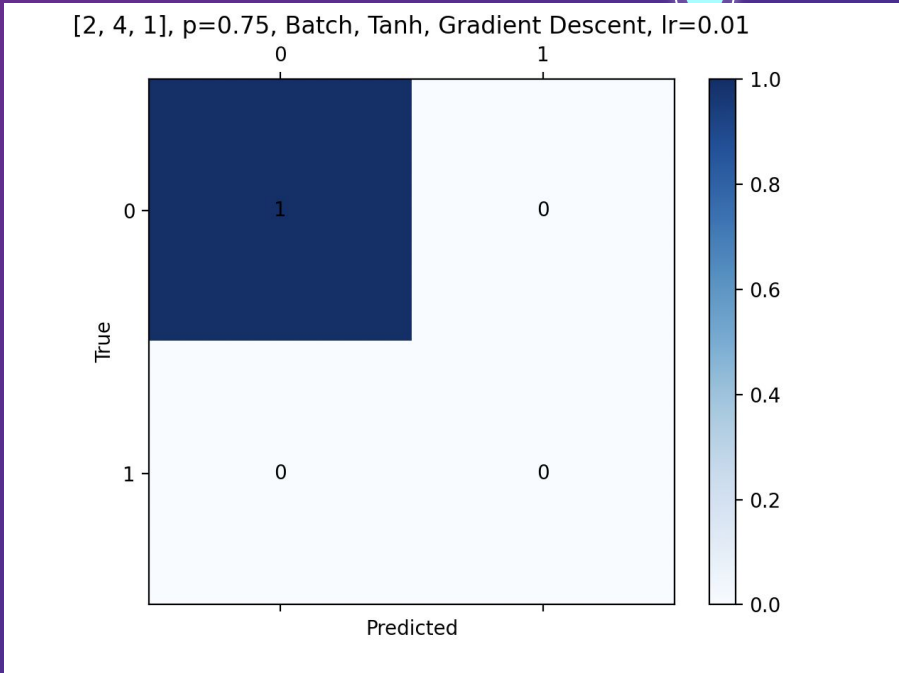


p = 0.75, Grad. desc

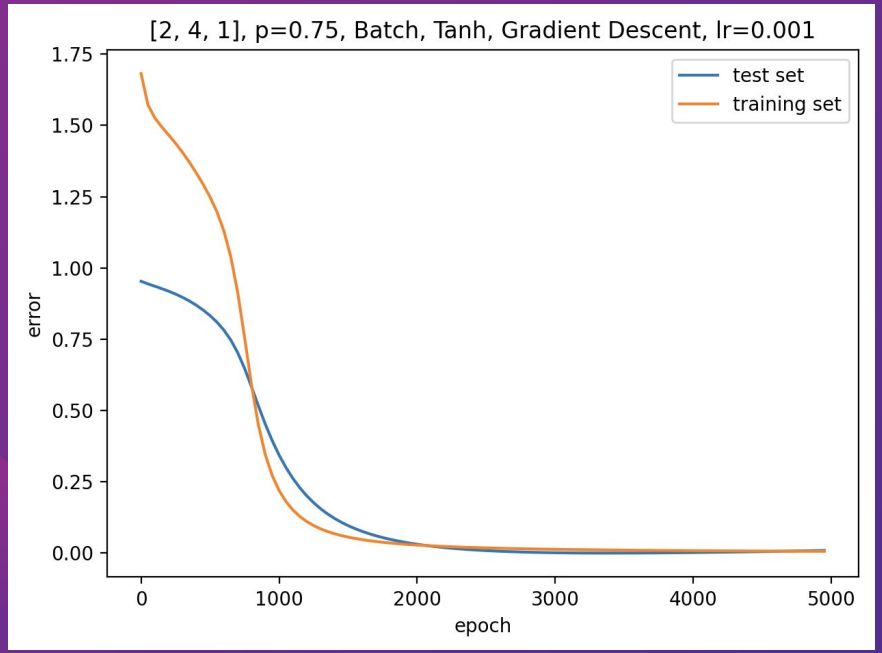


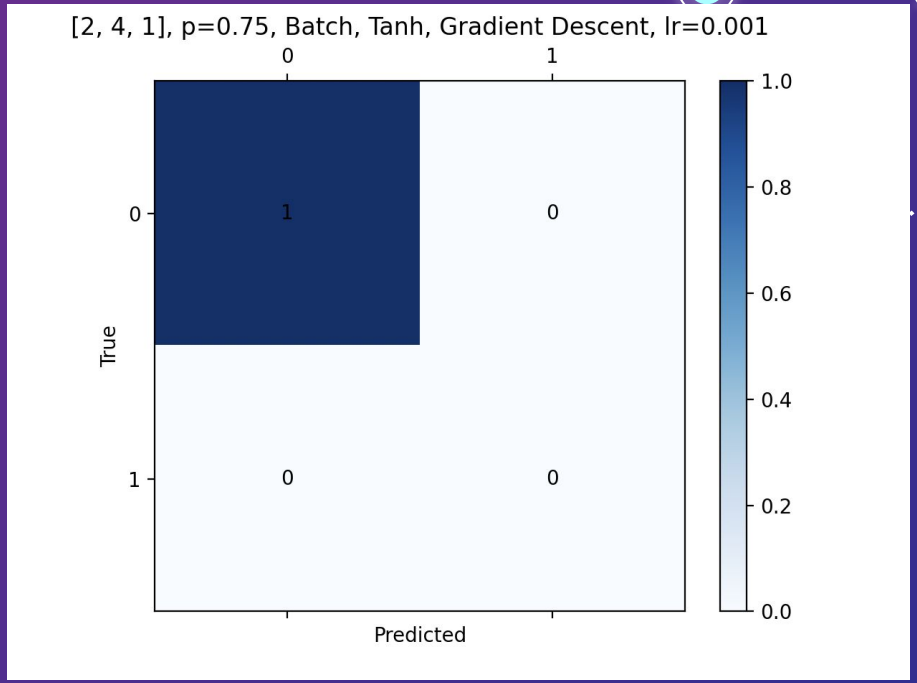
p = 0.75, Grad. desc, lr=0.01



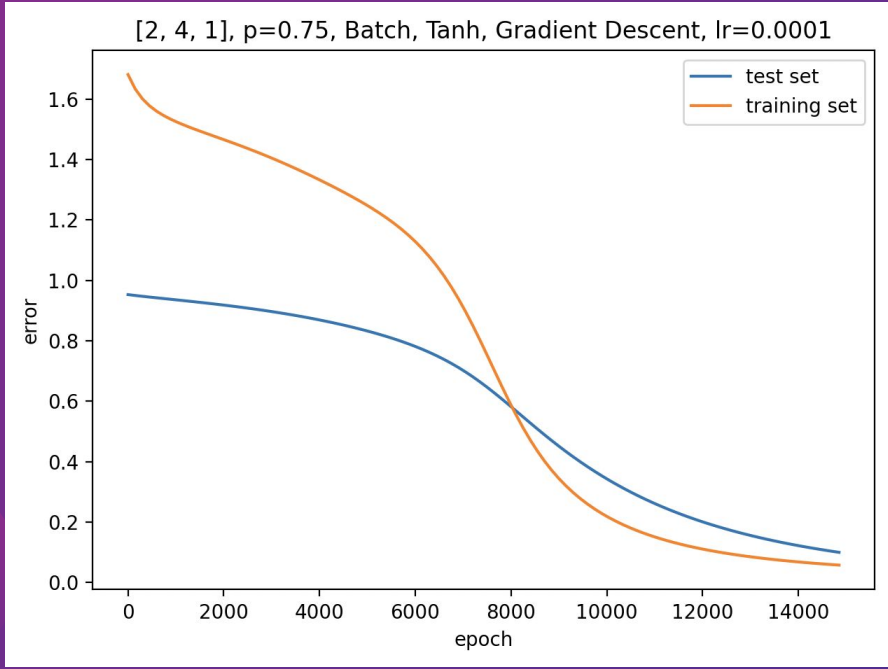


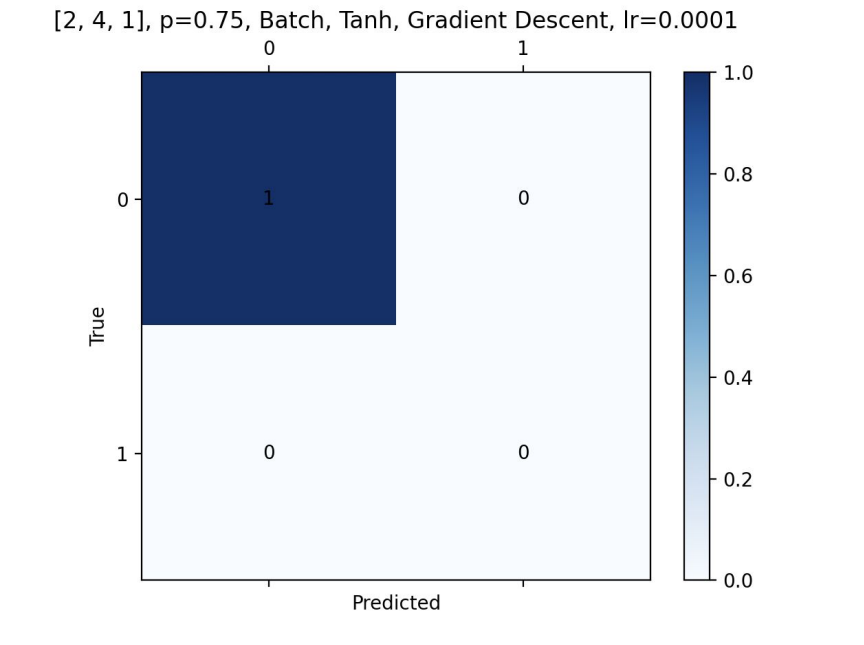
p = 0.75, Grad. desc, lr=0.001



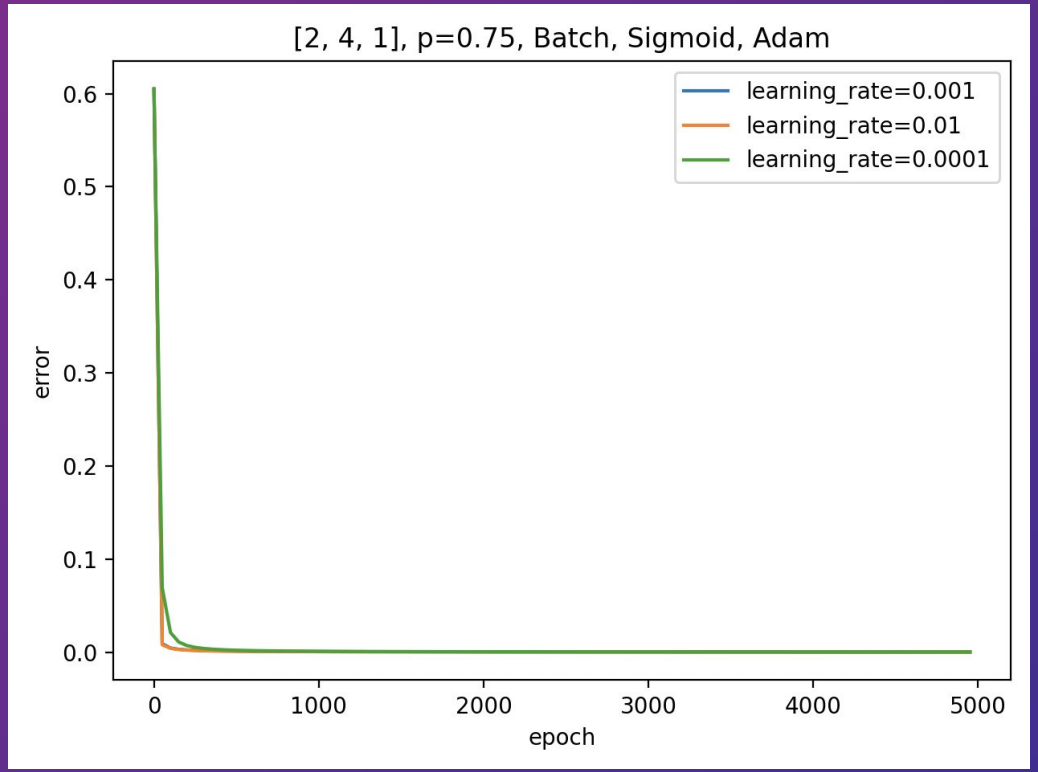


p = 0.75, Grad. desc, lr=0.0001

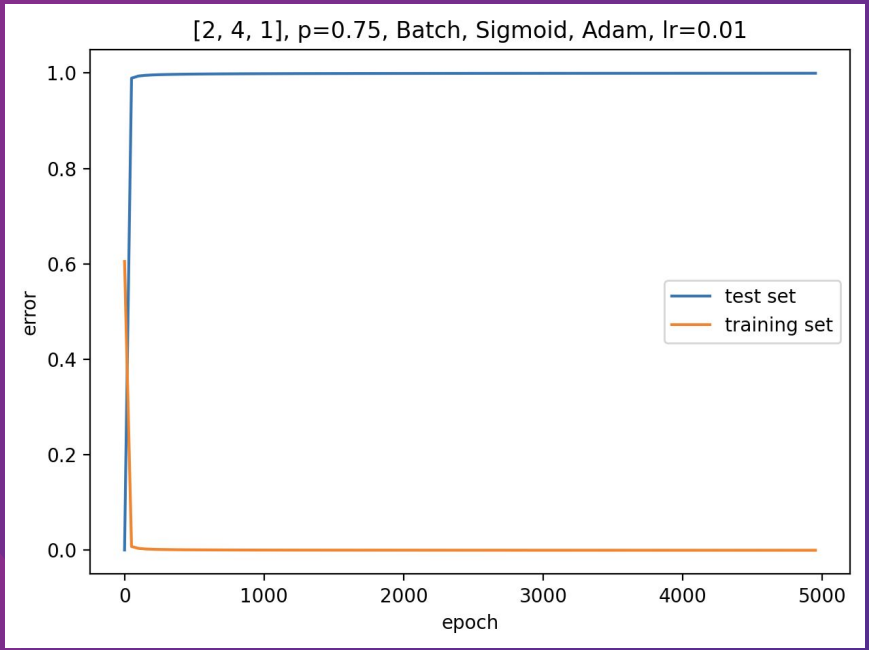


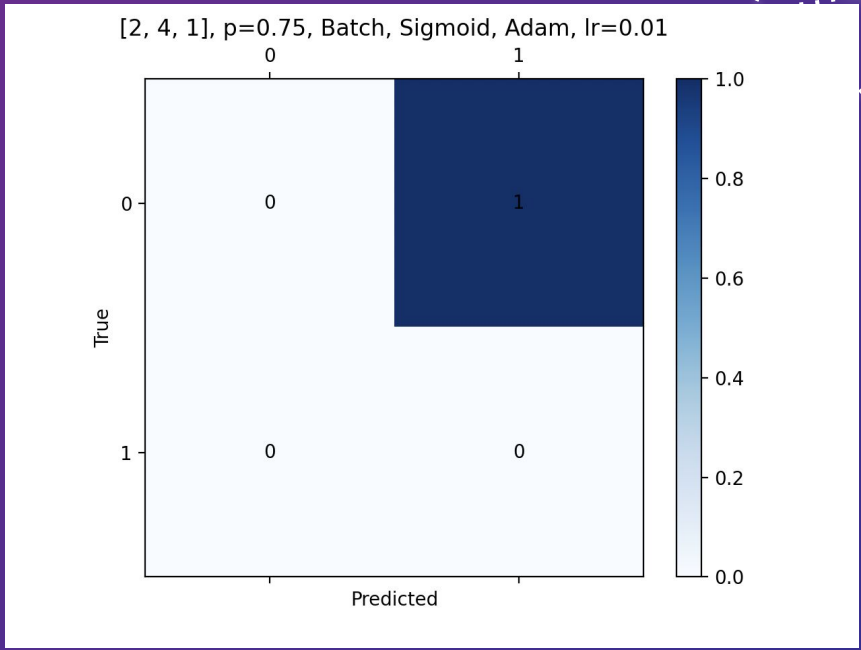


p = 0.75, ADAM

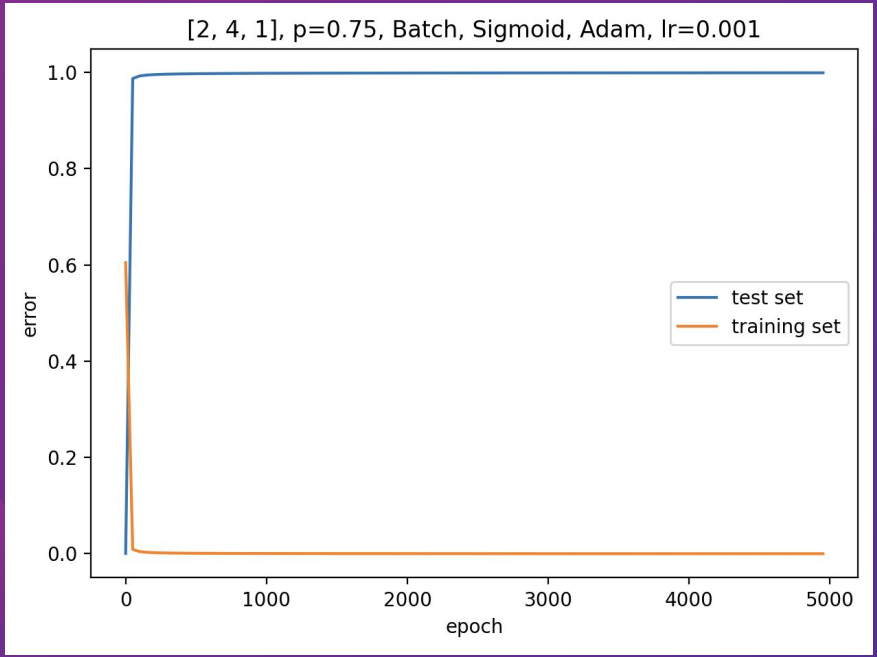

[2, 4, 1], p=0.75, Batch, Sigmoid, Adam
learning_rate=0.001
learning_rate=0.01
learning_rate=0.0001
error
epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1000
2000
3000
4000
5000

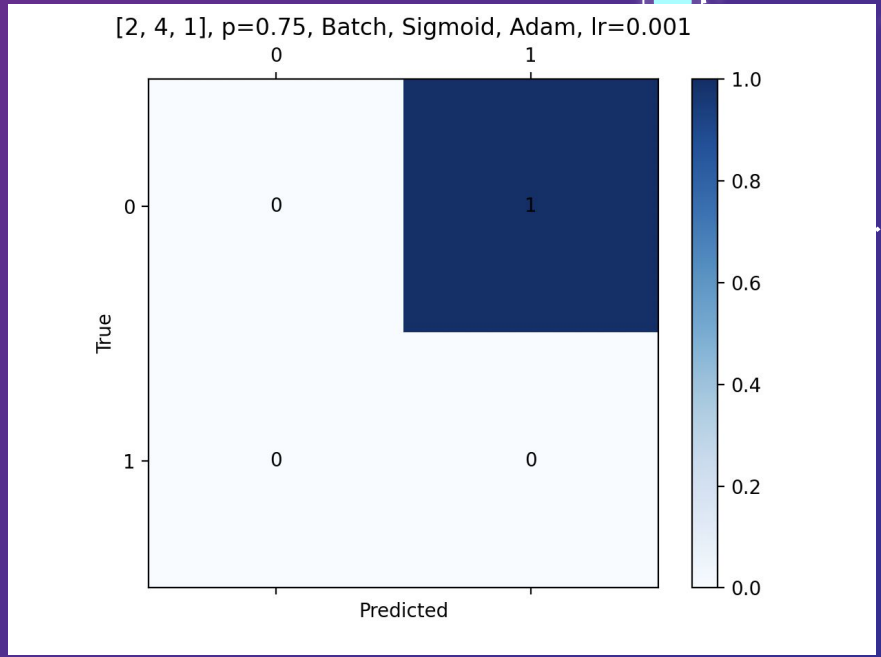
p = 0.75, ADAM, lr=0.01



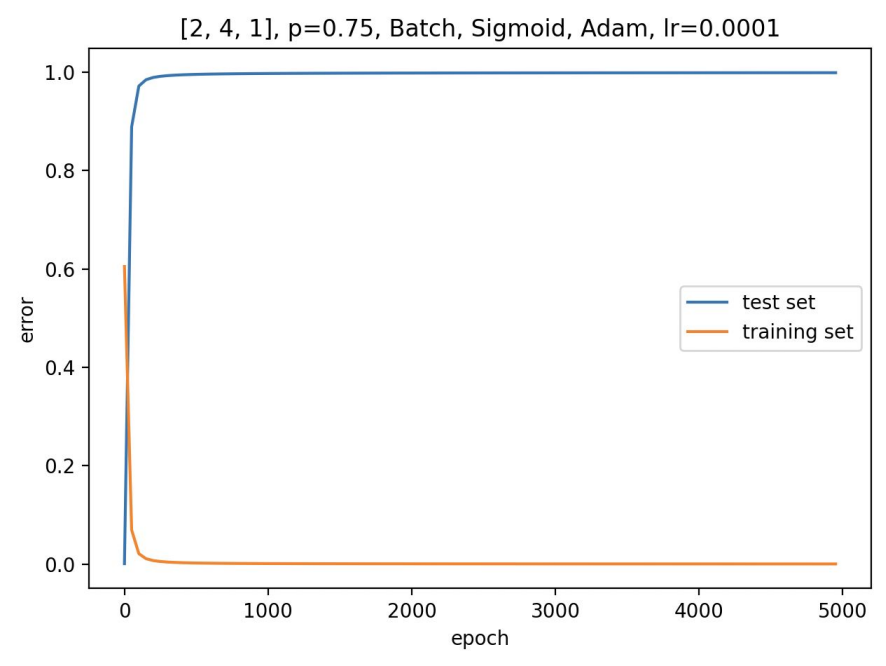


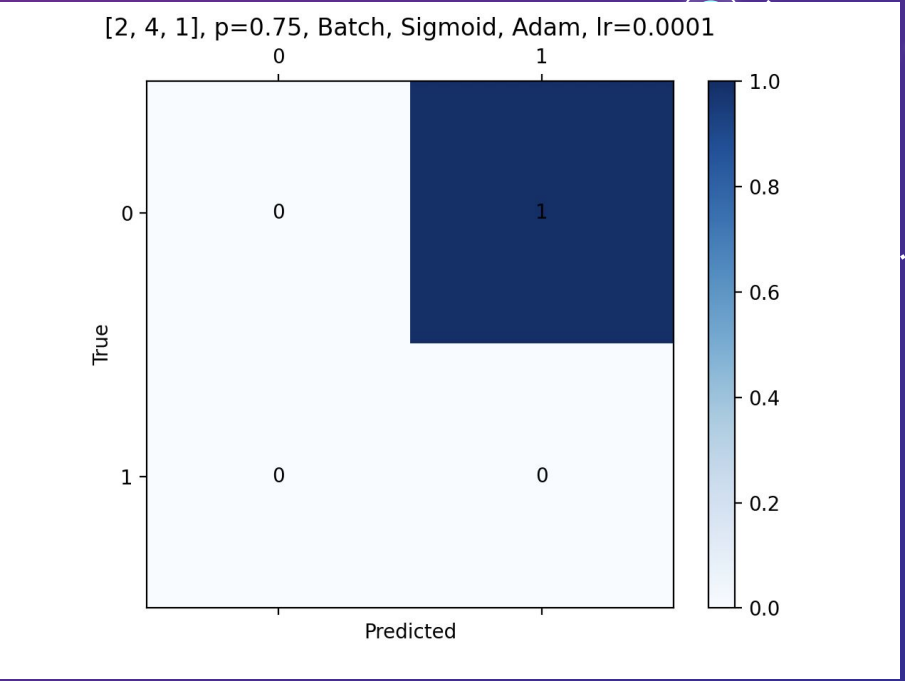
p = 0.75, ADAM, lr=0.001



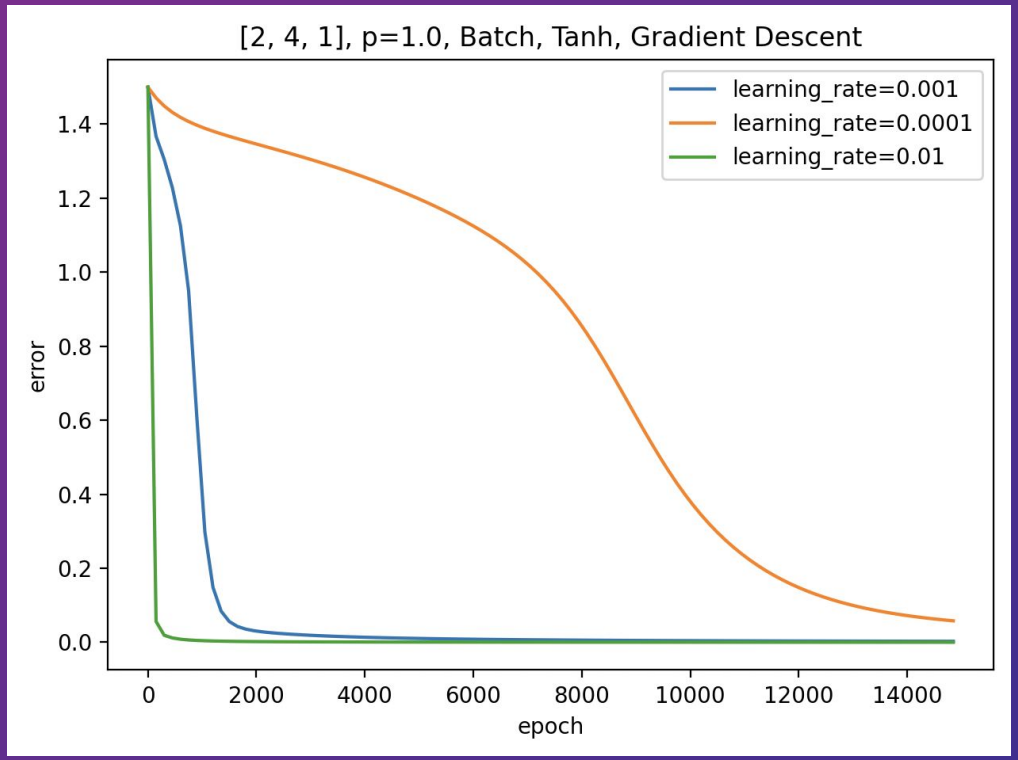


p = 0.75, ADAM, lr=0.0001

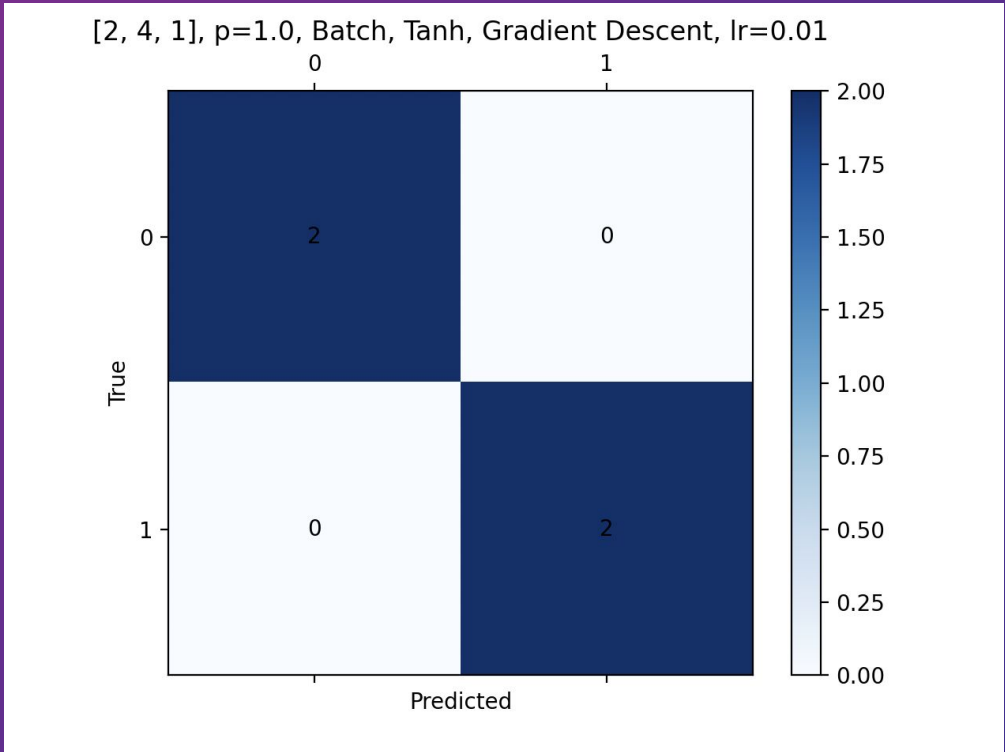




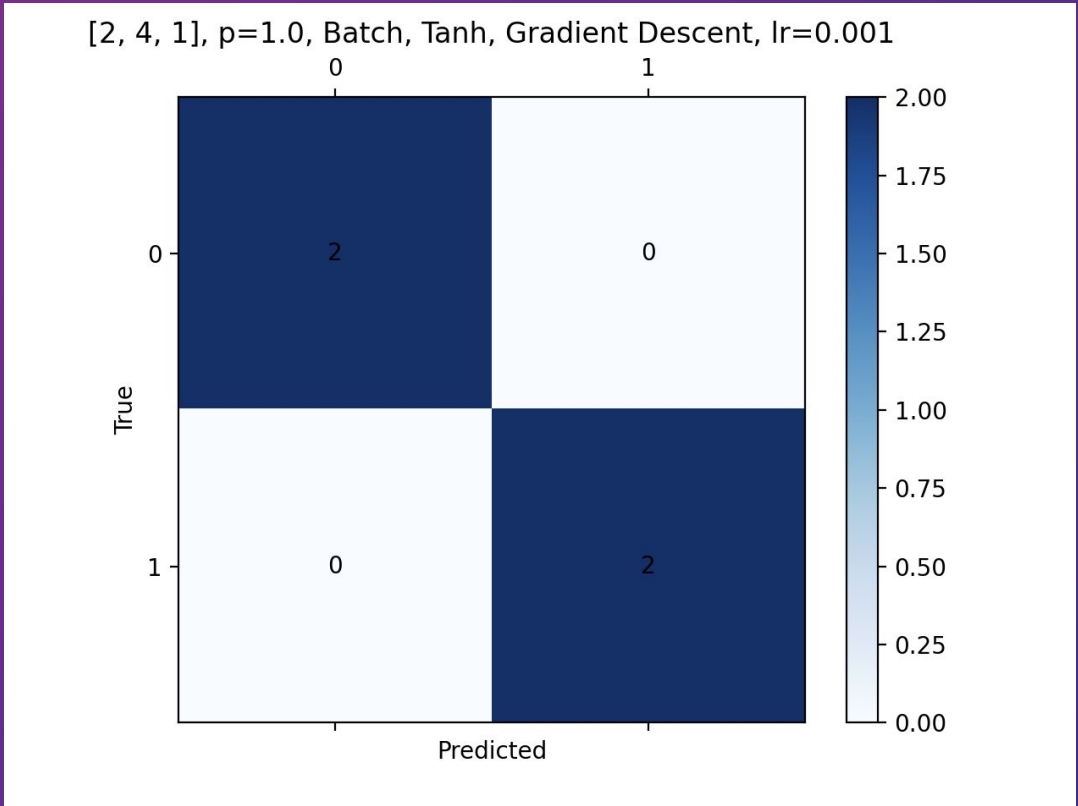
p = 1, Grad. desc


[2, 4, 1], p=1.0, Batch, Tanh, Gradient Descent
learning_rate=0.001
learning_rate=0.0001
learning_rate=0.01
error
epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
2000
4000
6000
8000
10000
12000
14000

Matriz de Confusión
p = 1, Grad. desc, lr=0.01

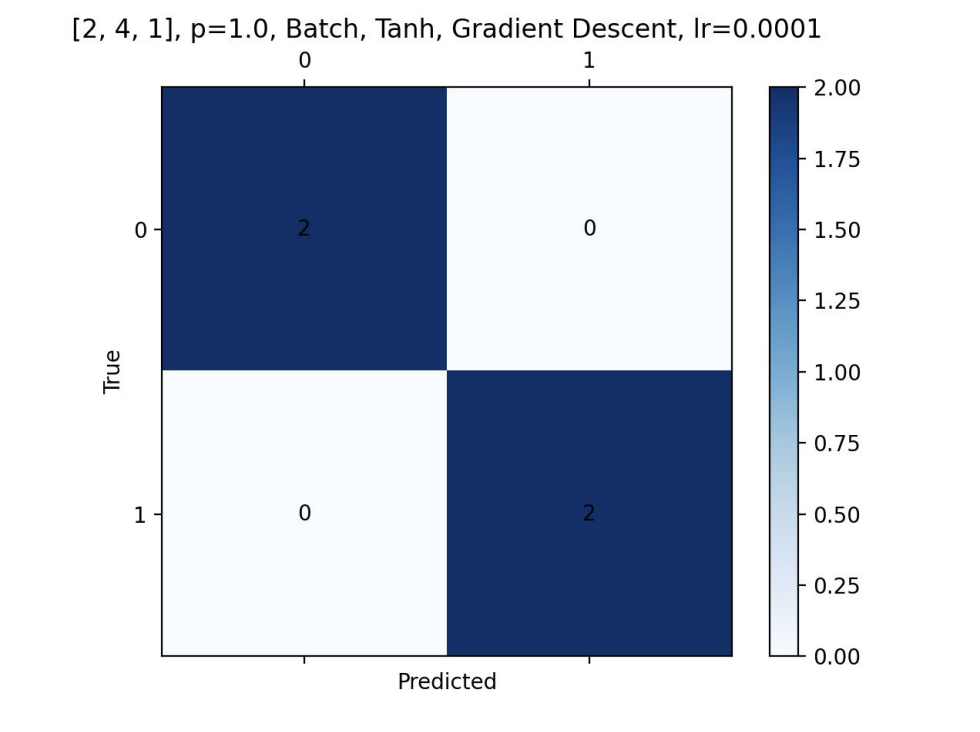


Matriz de Confusión
p = 1, Grad. desc, lr=0.001

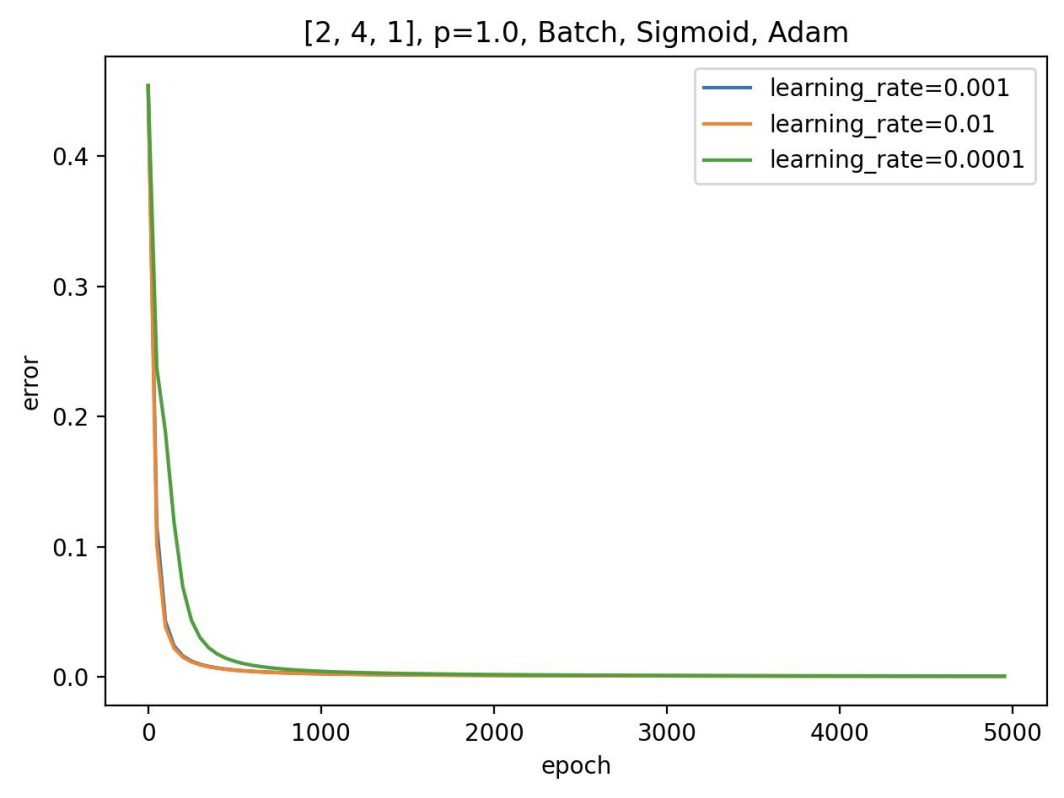


Matriz de Confusión

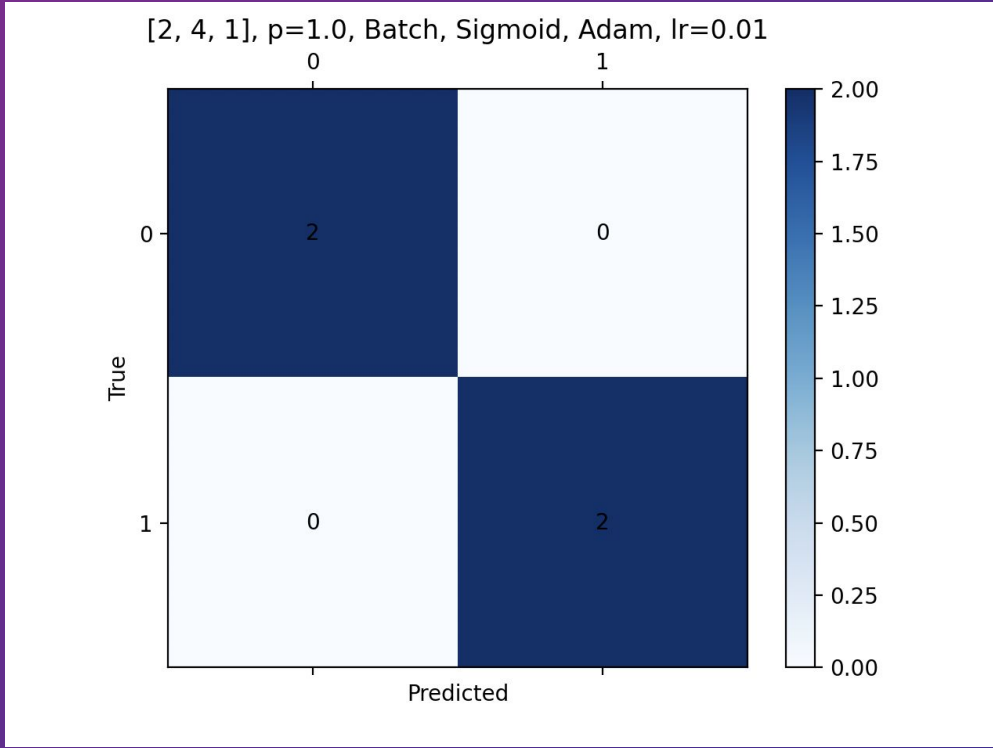
p = 1, Grad. desc, lr=0.0001



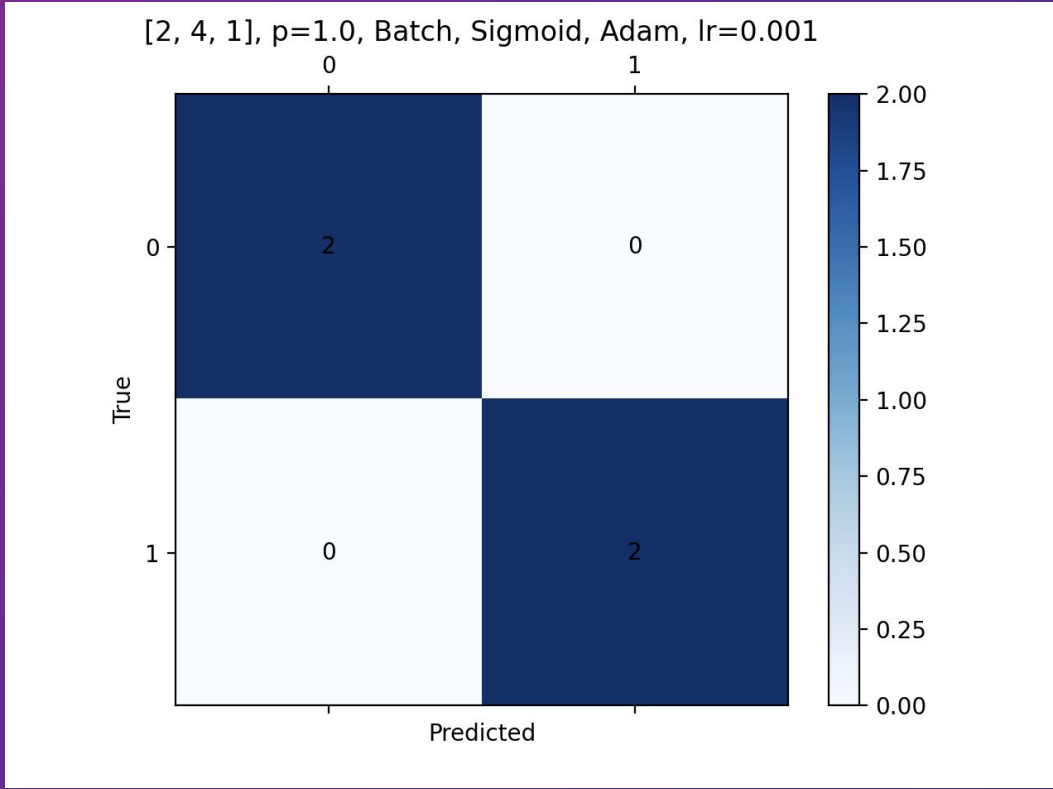
p = 1, ADAM


[2, 4, 1], p=1.0, Batch, Sigmoid, Adam
learning_rate=0.001
learning_rate=0.01
learning_rate=0.0001
error
0.0
0.1
0.2
0.3
0.4
0
1000
2000
3000
4000
5000
epoch

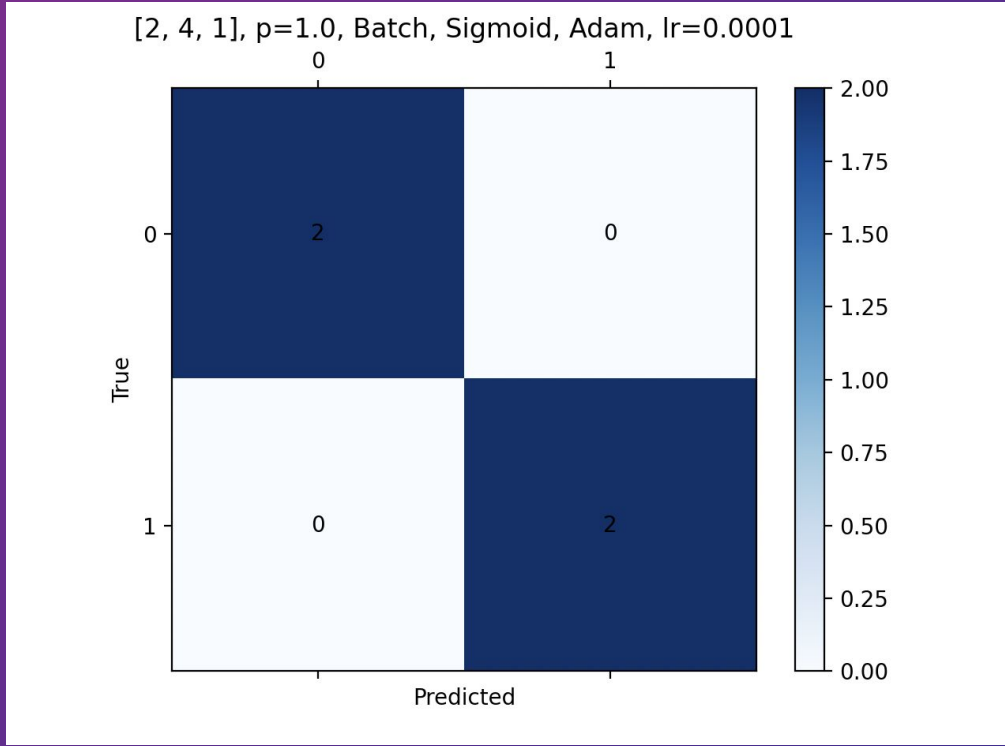
p = 1, ADAM, lr=0.01



p = 1, ADAM, lr=0.001



p = 1, ADAM, lr=0.0001



Conclusiones

- Se puede resolver el XOR ya que el perceptrón multicapa permite resolver problemas complejos que no son linealmente separables.
- Para este problema, el método de ADAM converge más rápido que el gradiente descendiente.
- La red no puede inferir el resultado de la operación XOR para entradas que no formaron parte del training set.

Parte B

Implementar el algoritmo de perceptrón multicapa y utilizarlo para discriminar si un número es "par", con entradas dadas por el conjunto de números decimales del 0 al 9 representados en grillas de 7x5.

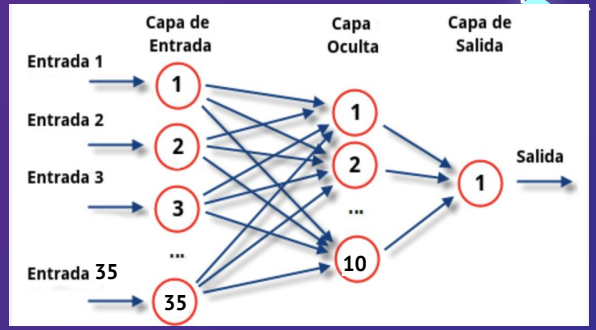
Entrada: [0,1,2,3,4,5,6,7,8,9], donde cada número es su representación gráfica en una grilla de 7x5. En la red es un vector de $[0,1]^{35}$

Layers: [10]

Método de Optimización: Gradiente Descendente / ADAM

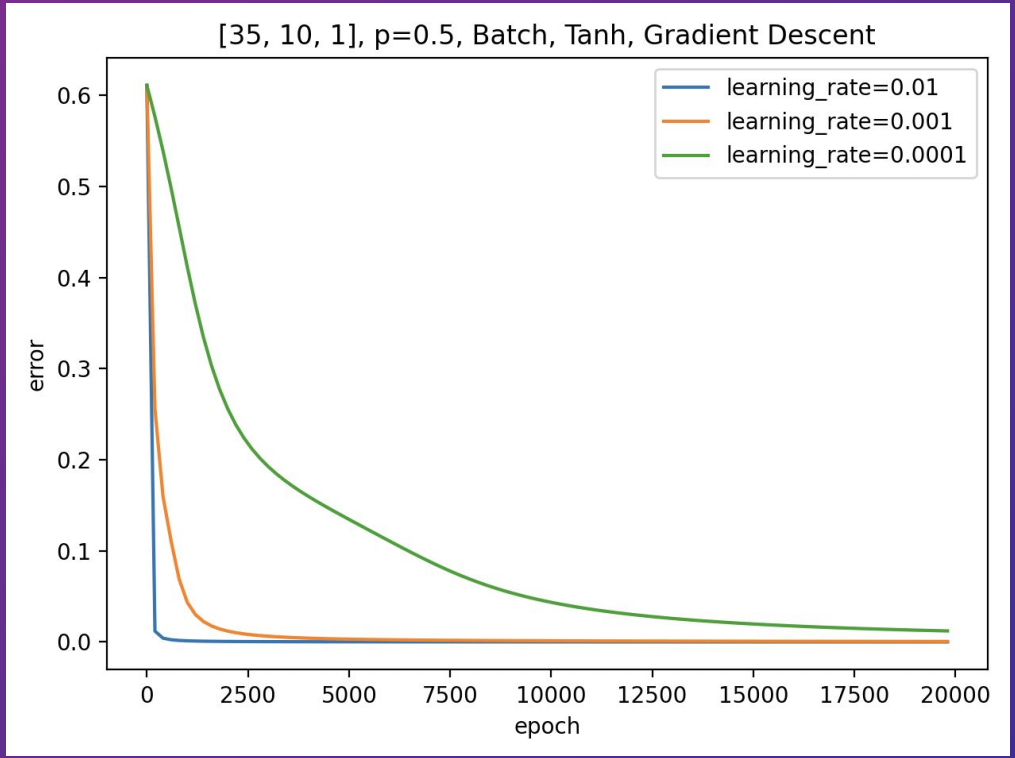
Función de Activación: Sigmoid / Tanh

Learning rate: 0.01/0.001/0.0001

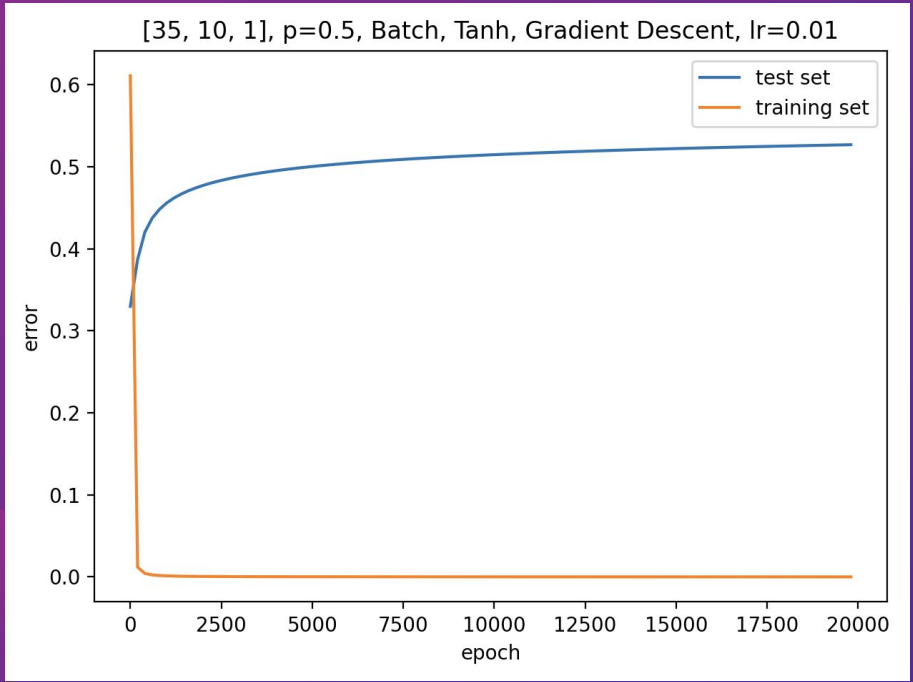


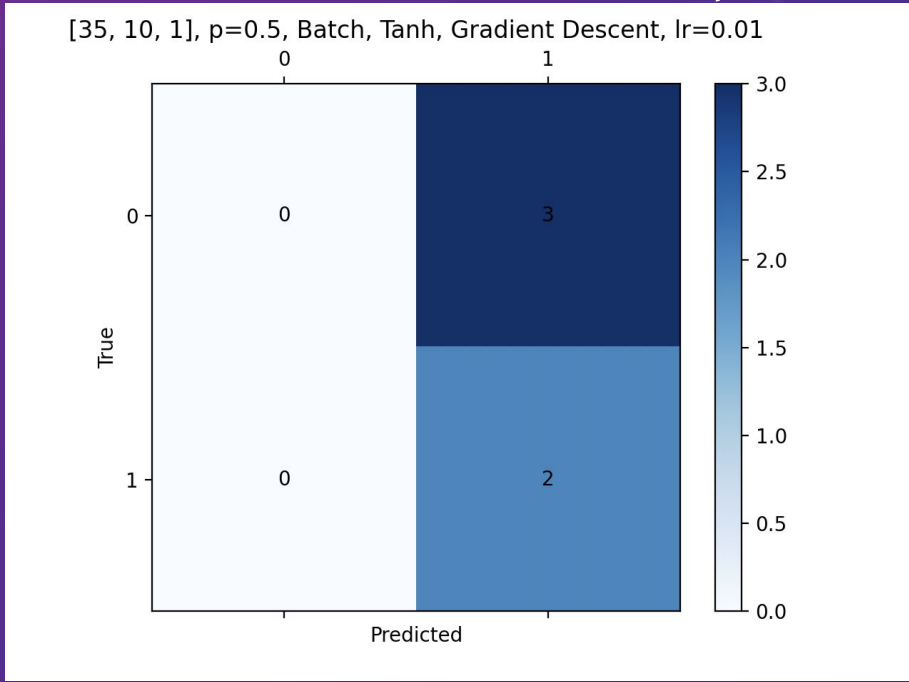
Los conjuntos de entrenamiento y testeo se definen a partir de una proporción de entrenamiento p en el intervalo [0,1]

p = 0.5, Grad. desc

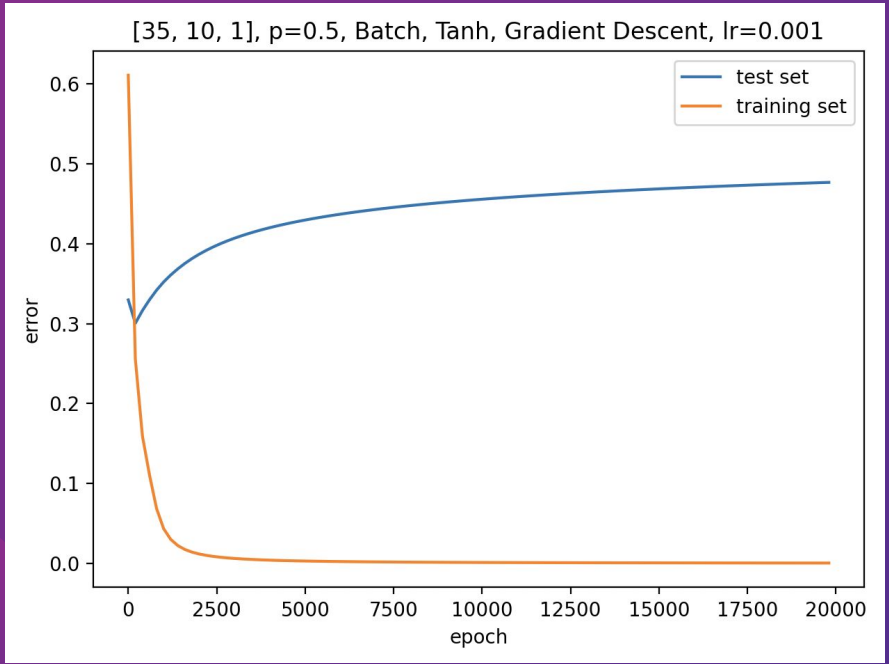


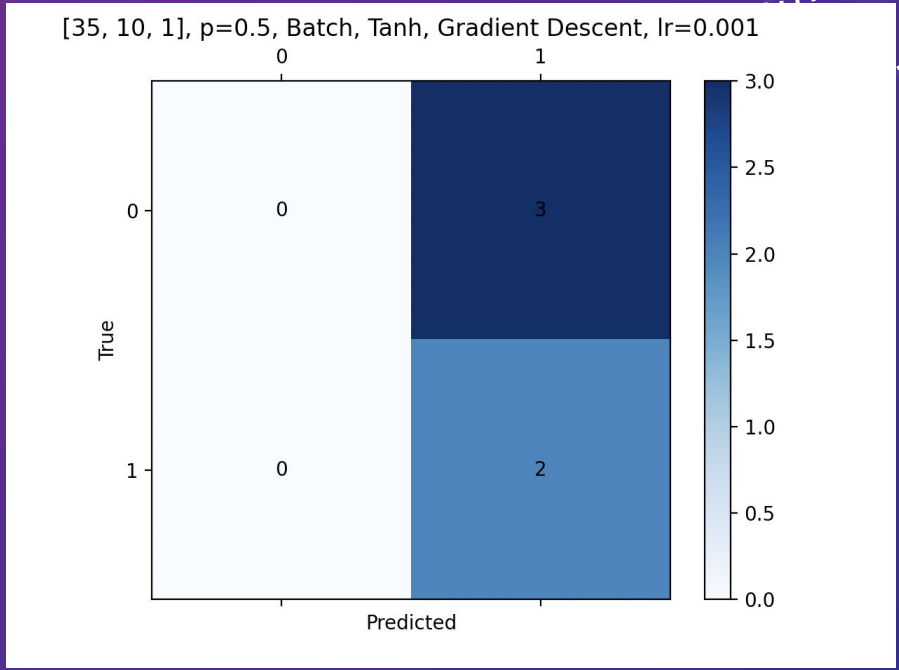
p = 0.5, Grad. desc, lr=0.01



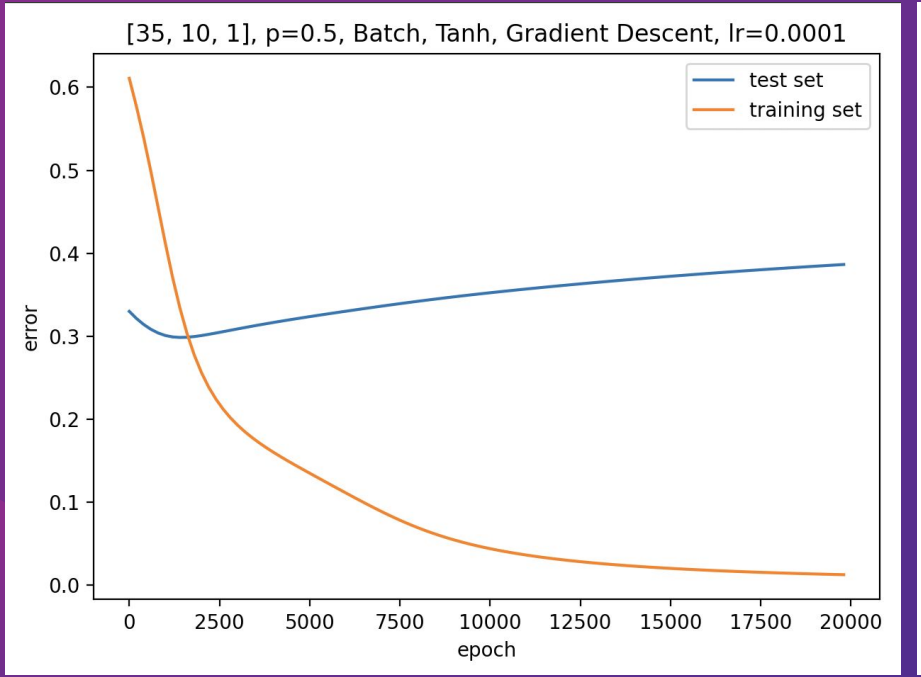


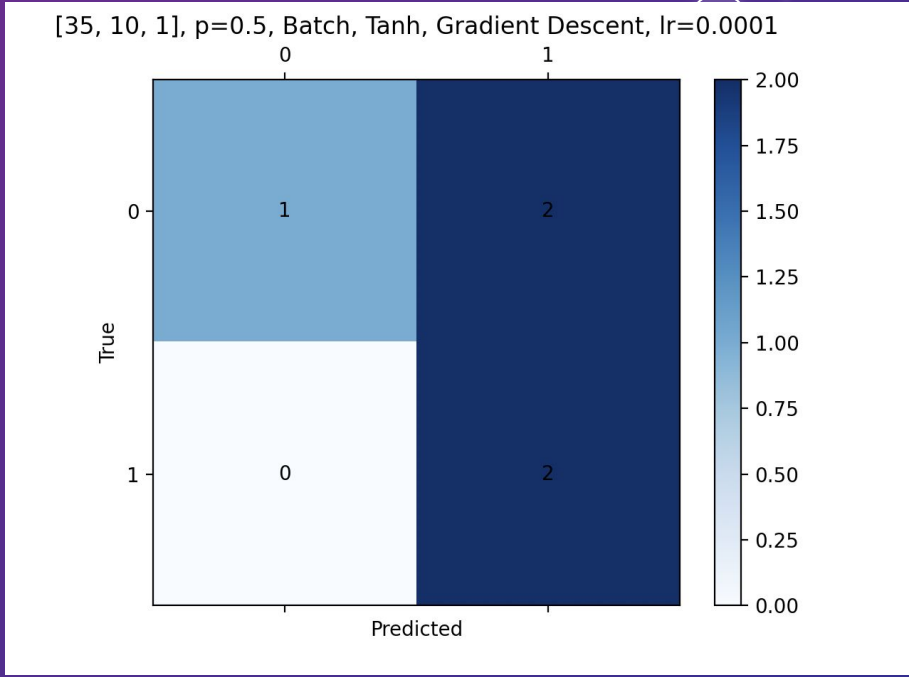
p = 0.5, Grad. desc, lr=0.001



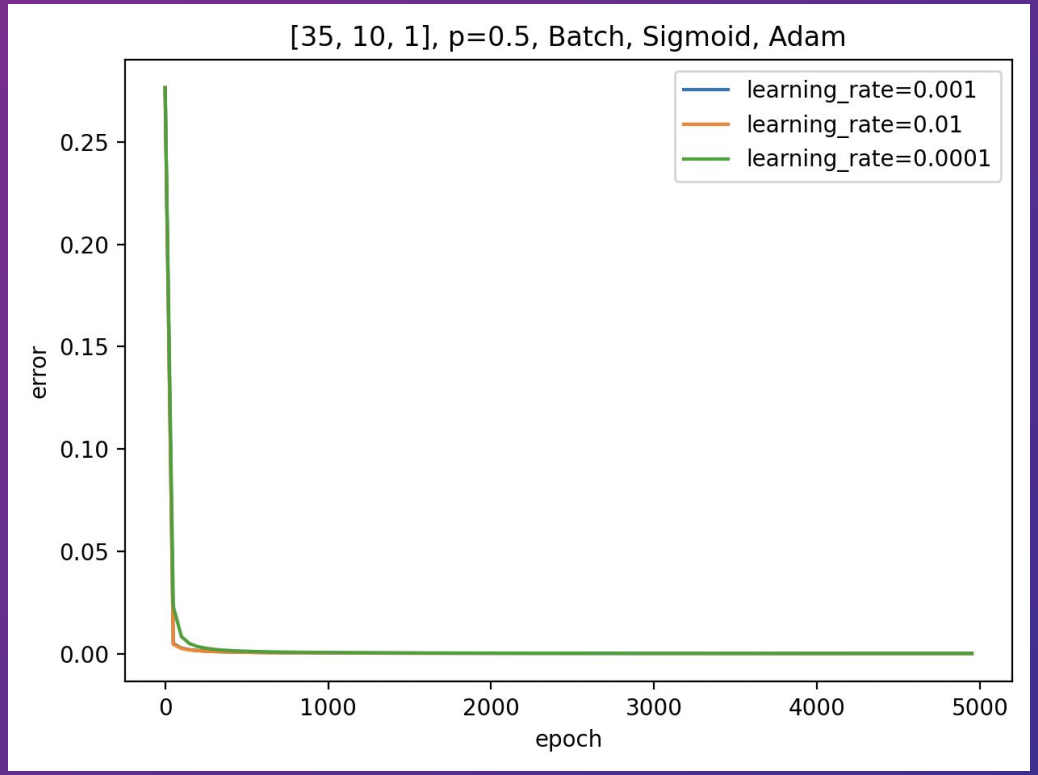


p = 0.5, Grad. desc, lr=0.0001

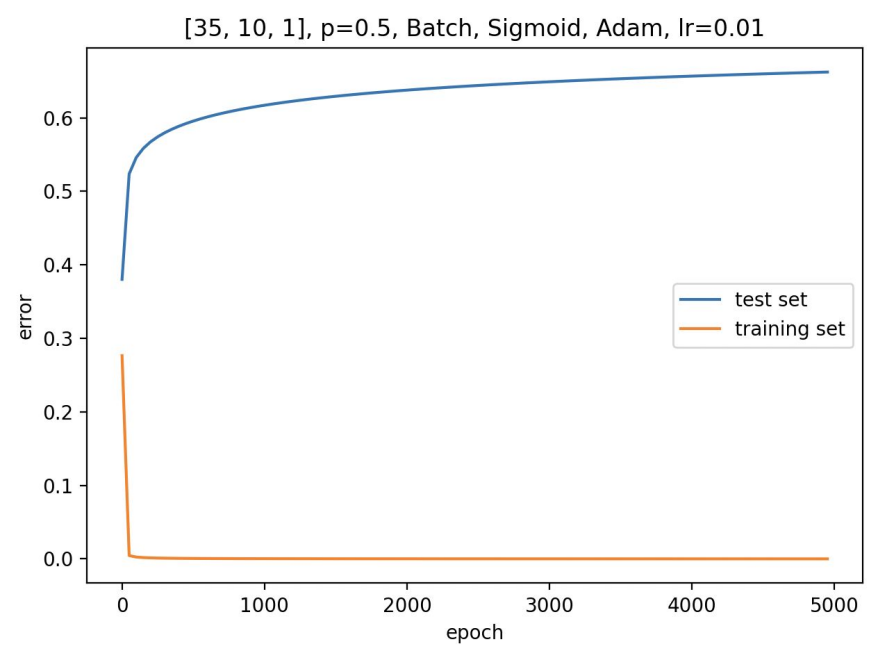


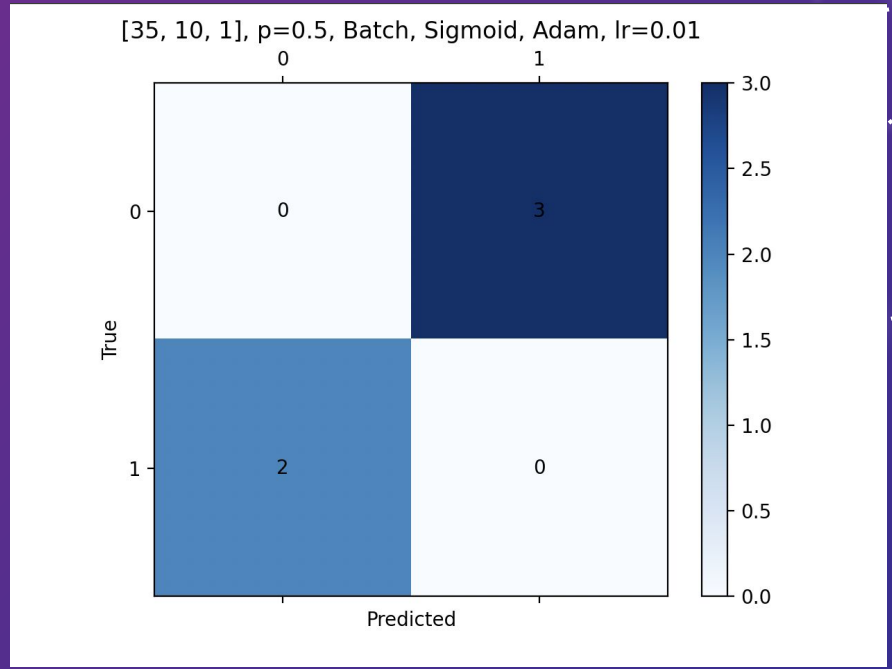


p = 0.5, ADAM

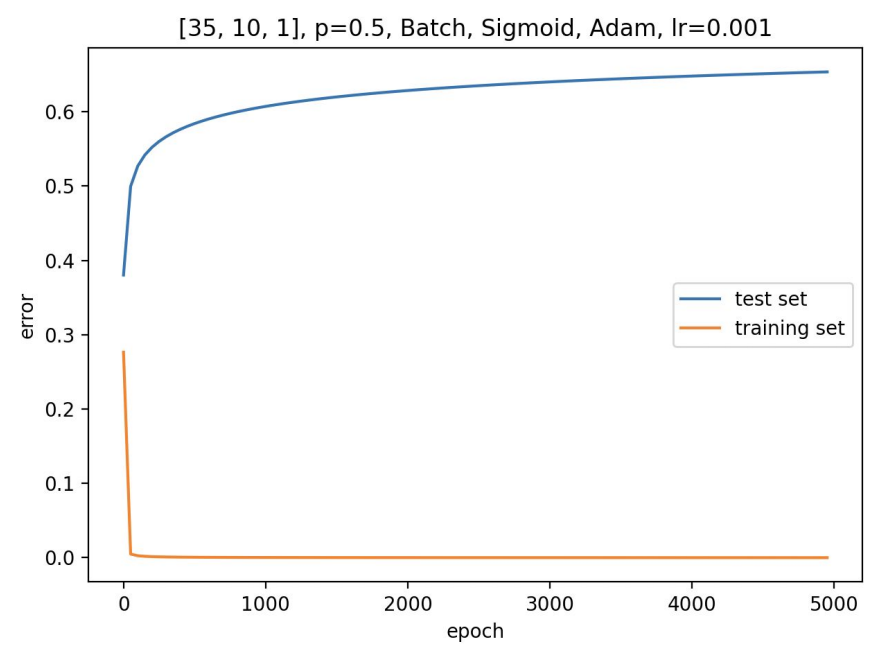

[35, 10, 1], p=0.5, Batch, Sigmoid, Adam
learning_rate=0.001
learning_rate=0.01
learning_rate=0.0001
error
0.25
0.20
0.15
0.10
0.05
0.00
0
1000
2000
3000
4000
5000
epoch

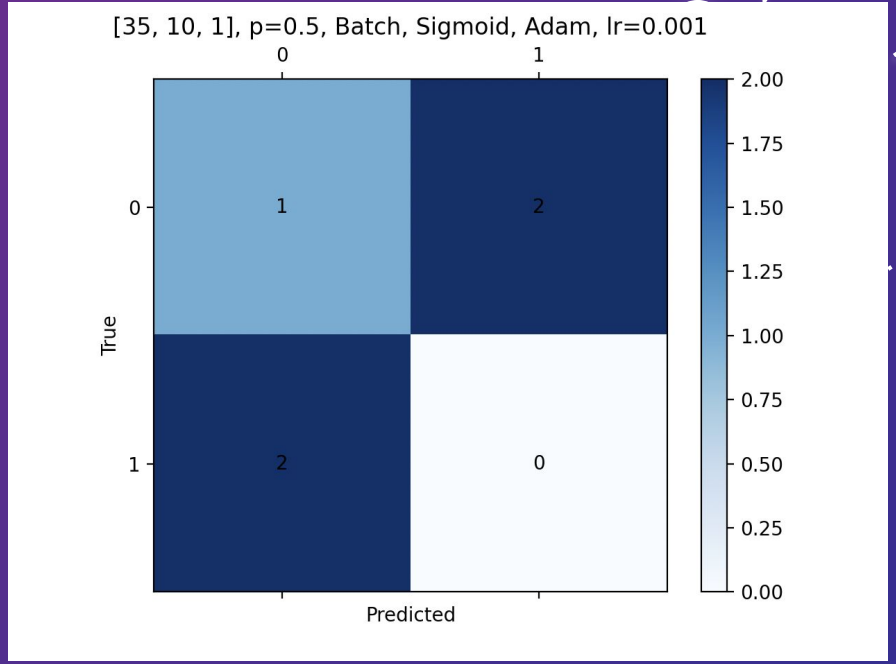
p = 0.5, ADAM, lr=0.01



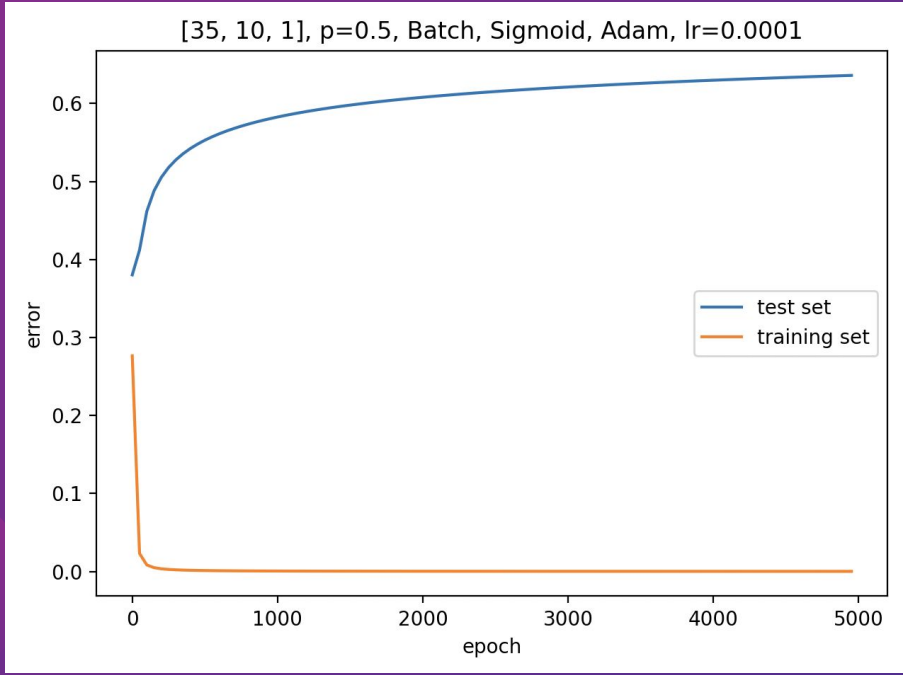


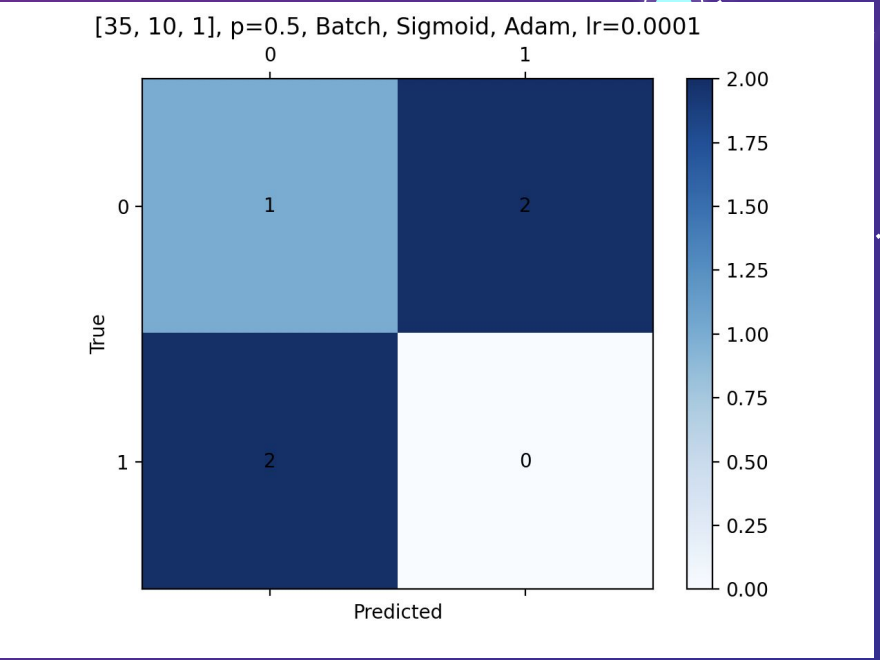
p = 0.5, ADAM, lr=0.001



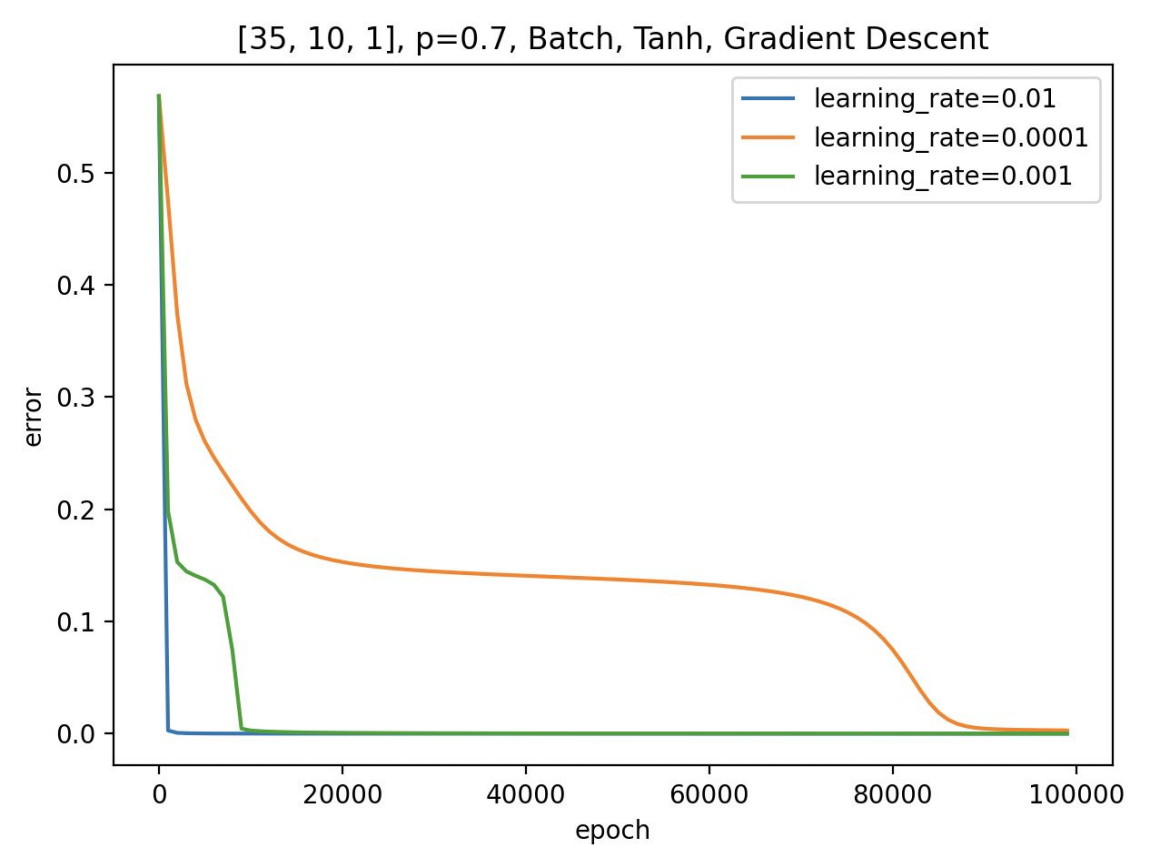


p = 0.5, ADAM, lr=0.0001

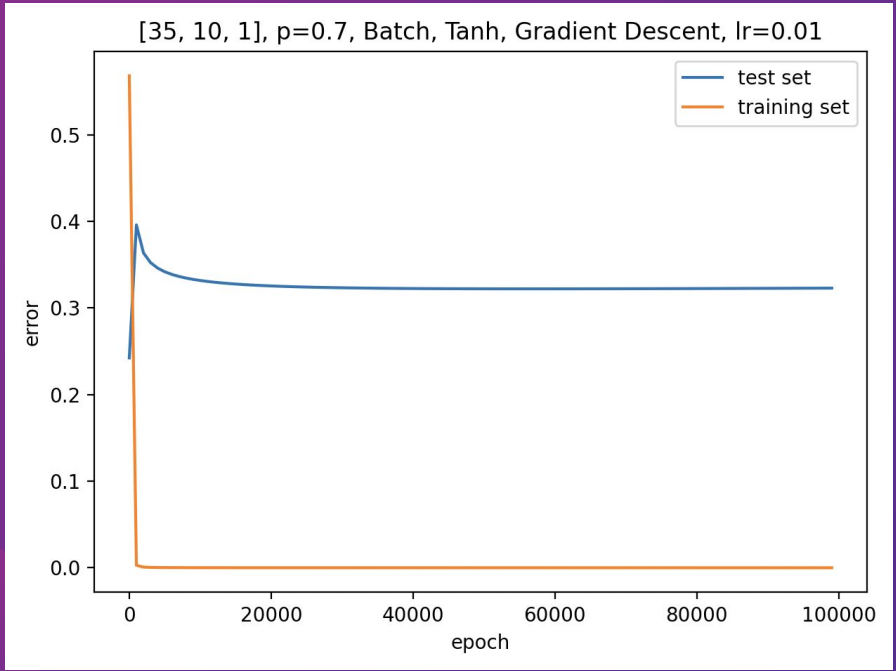


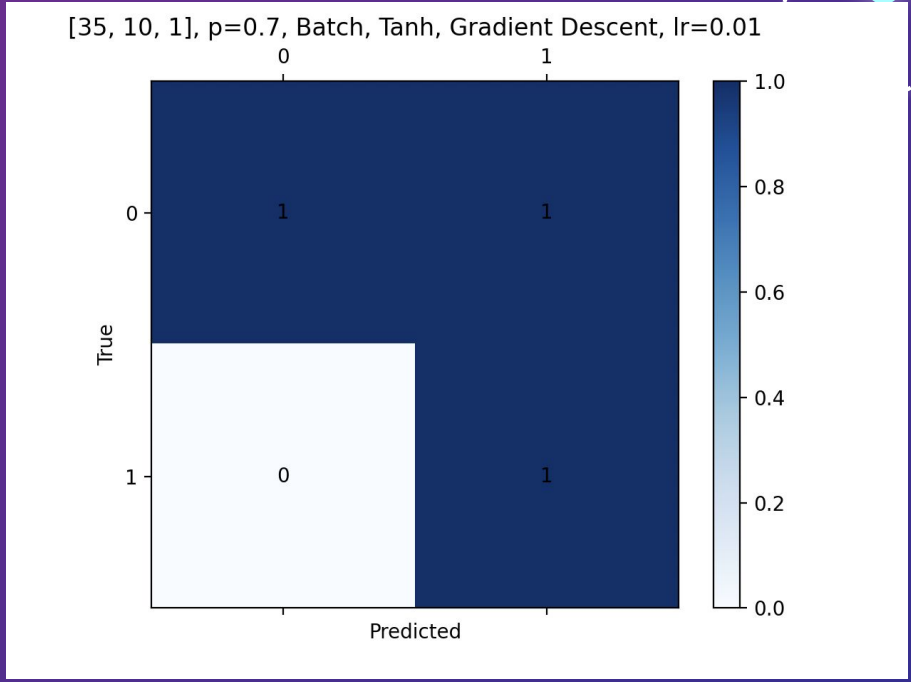


p = 0.7, Grad. desc

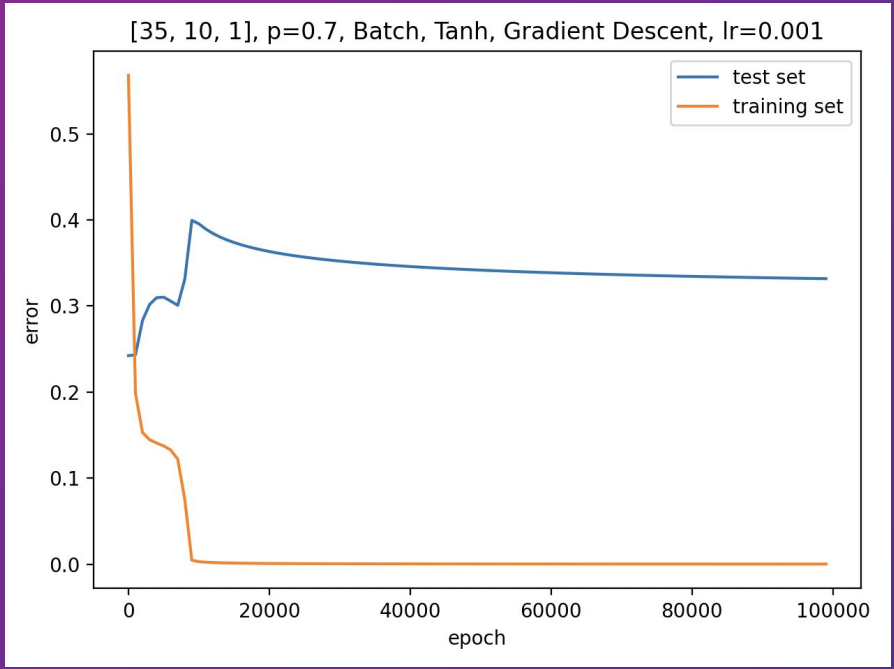


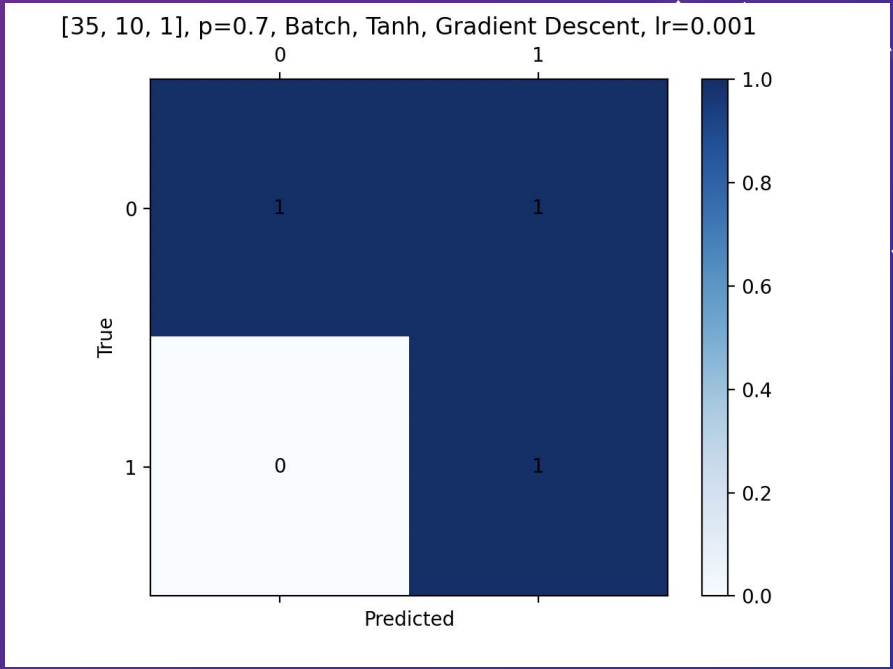
p = 0.7, Grad. desc, lr=0.01



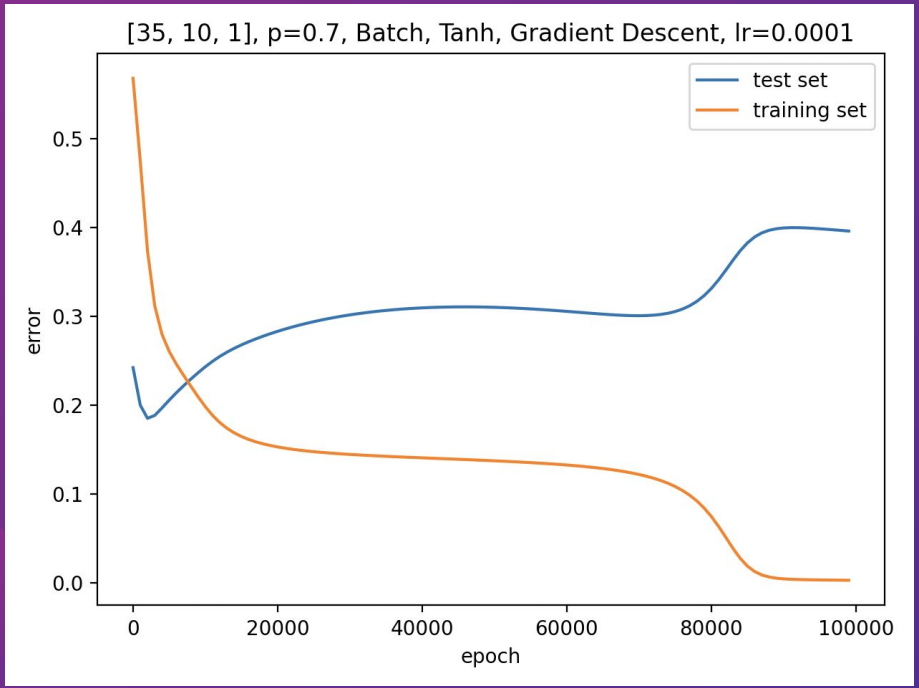


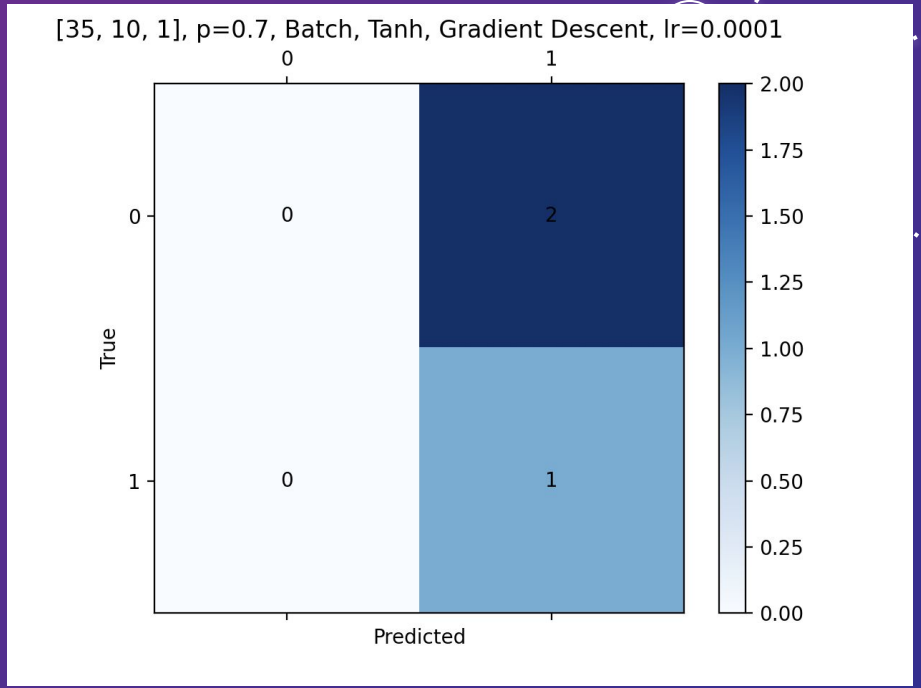
p = 0.7, Grad. desc, lr=0.001



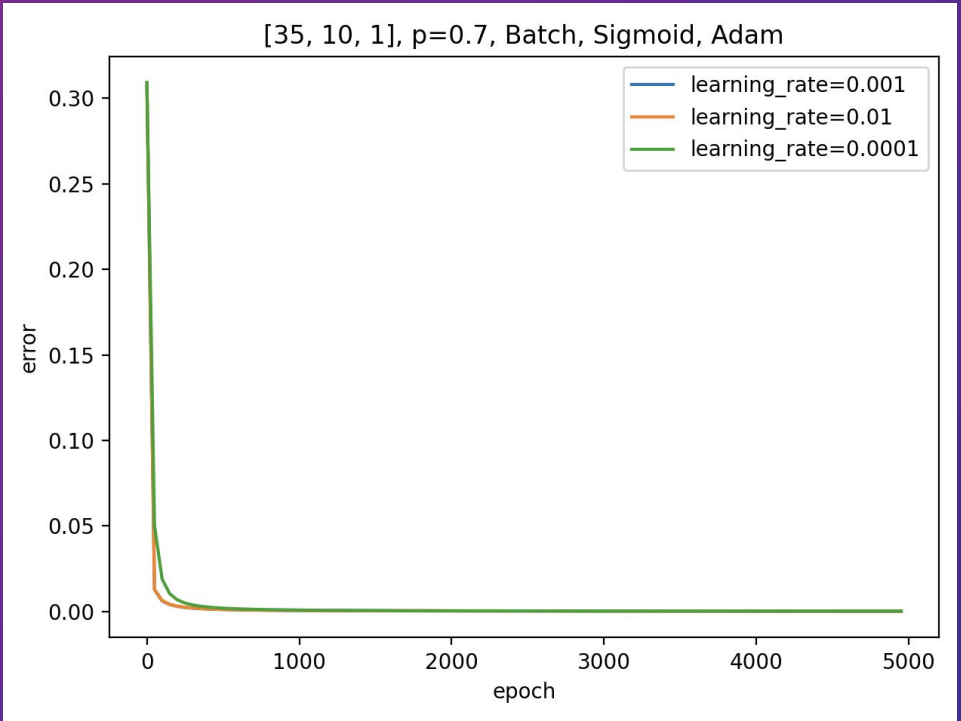


p = 0.7, Grad. desc, lr=0.0001

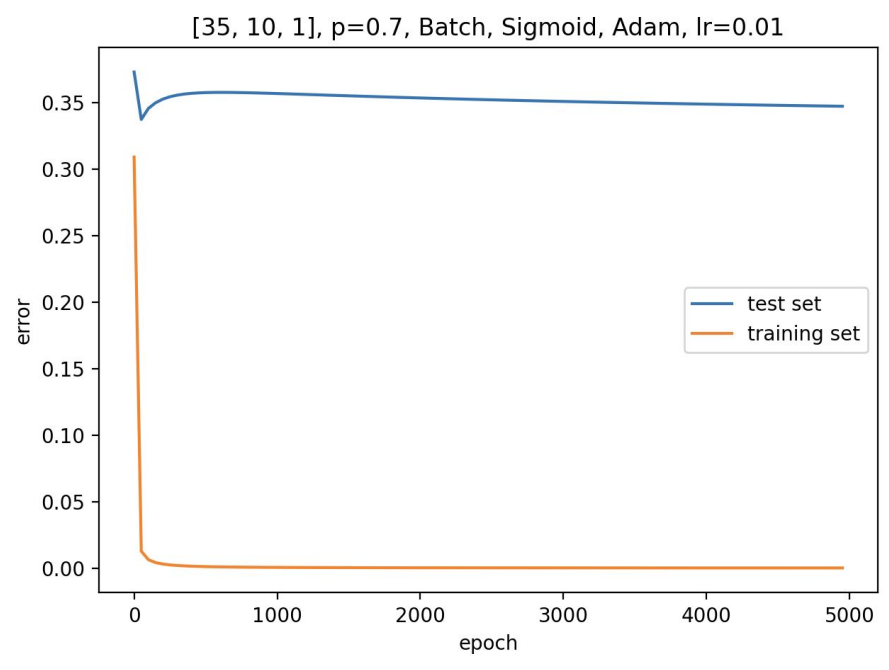


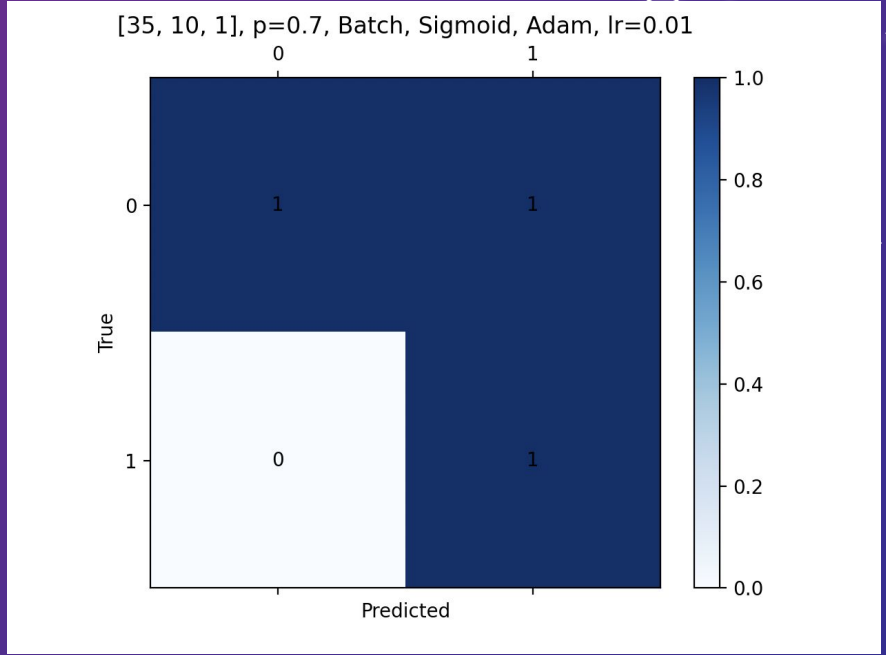


p = 0.7, ADAM

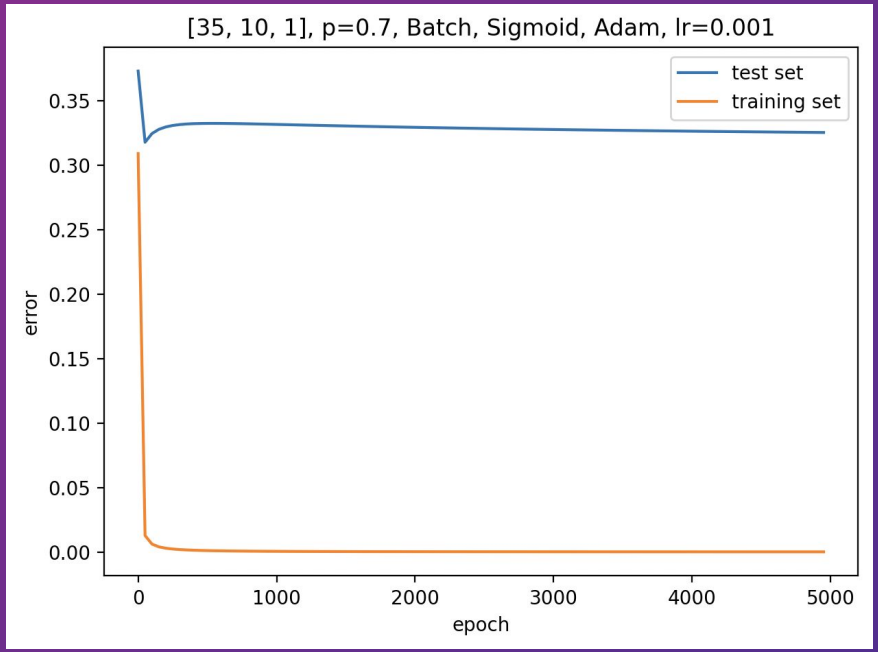

[35, 10, 1], p=0.7, Batch, Sigmoid, Adam
learning_rate=0.001
learning_rate=0.01
learning_rate=0.0001
error
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1000
2000
3000
4000
5000
epoch

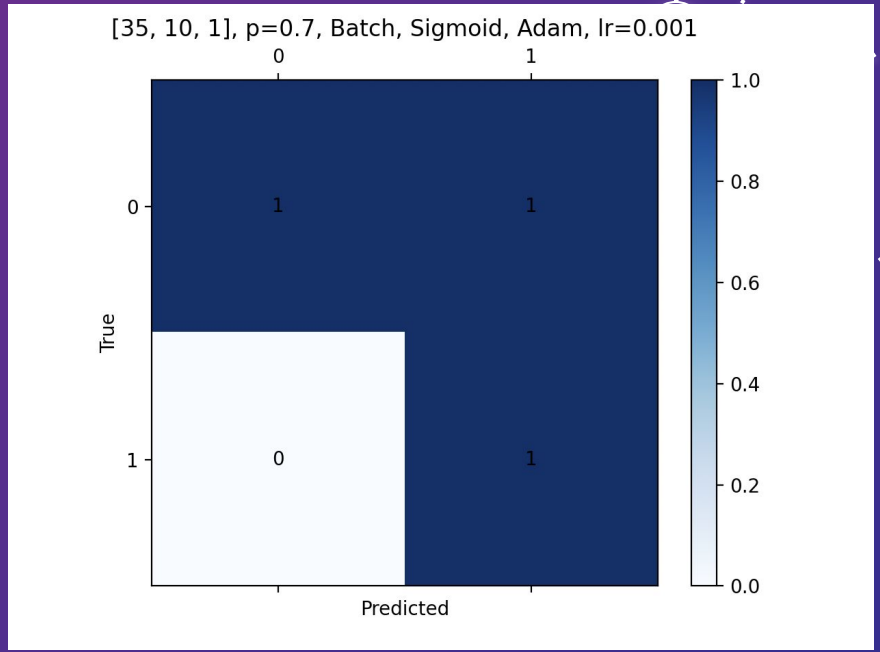
p = 0.7, ADAM, lr=0.01



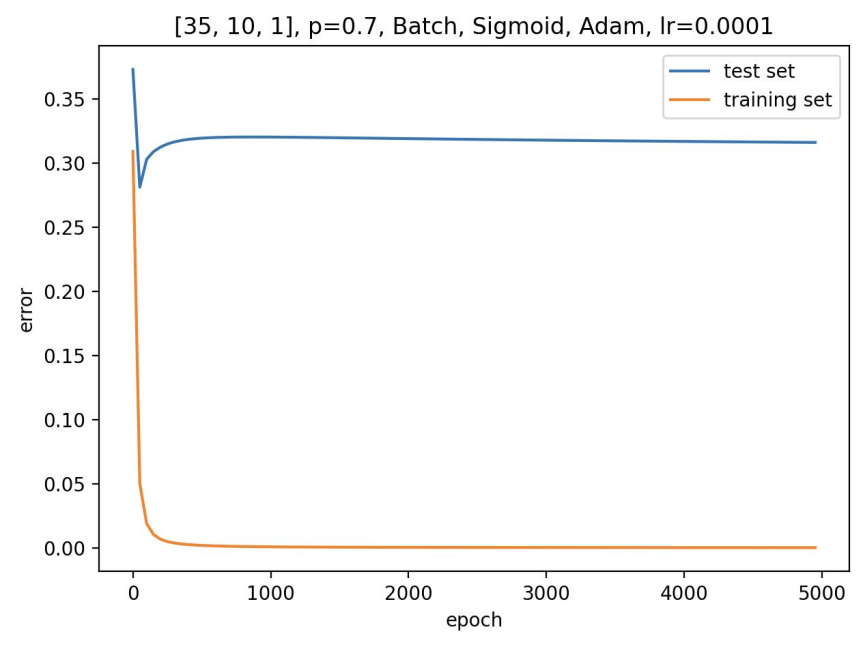


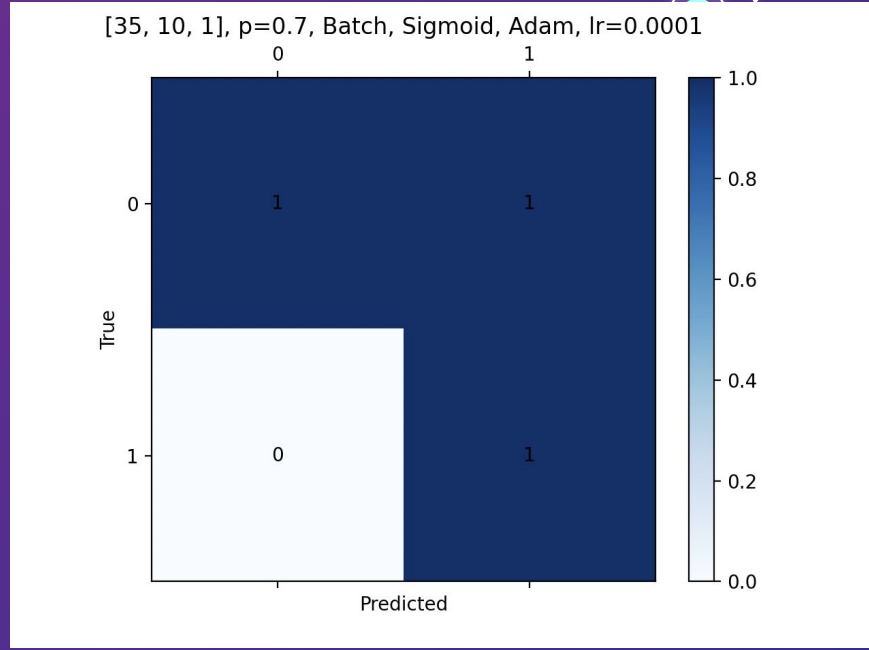
p = 0.7, ADAM, lr=0.001



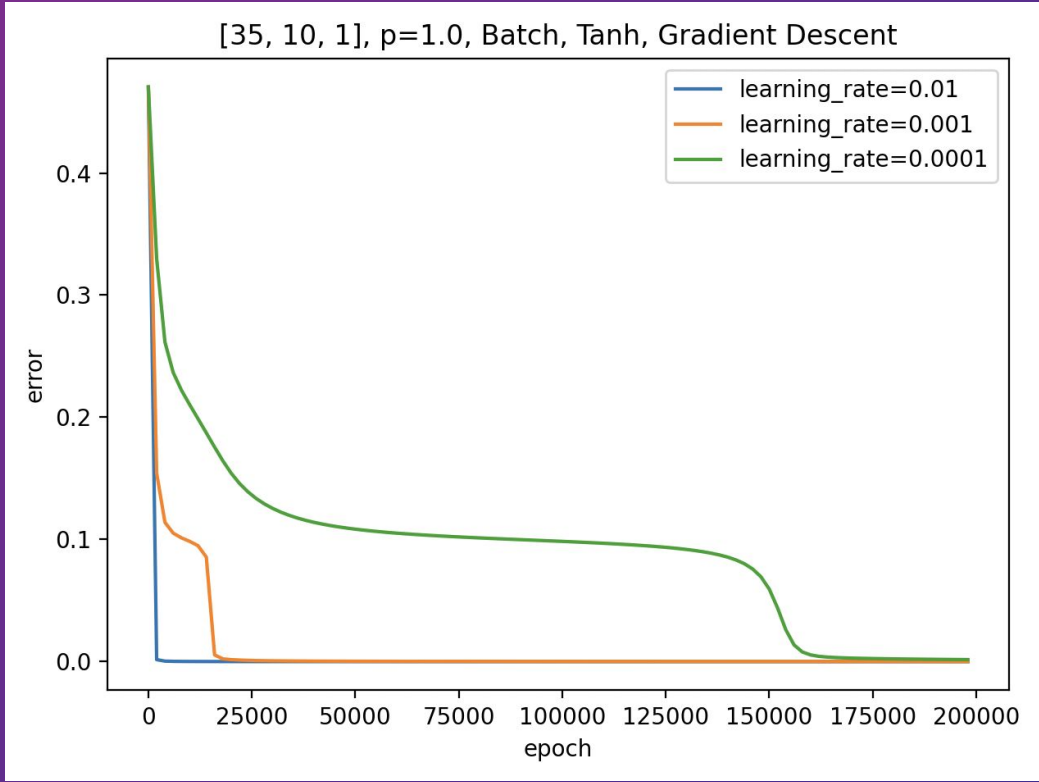


p = 0.7, ADAM, lr=0.0001



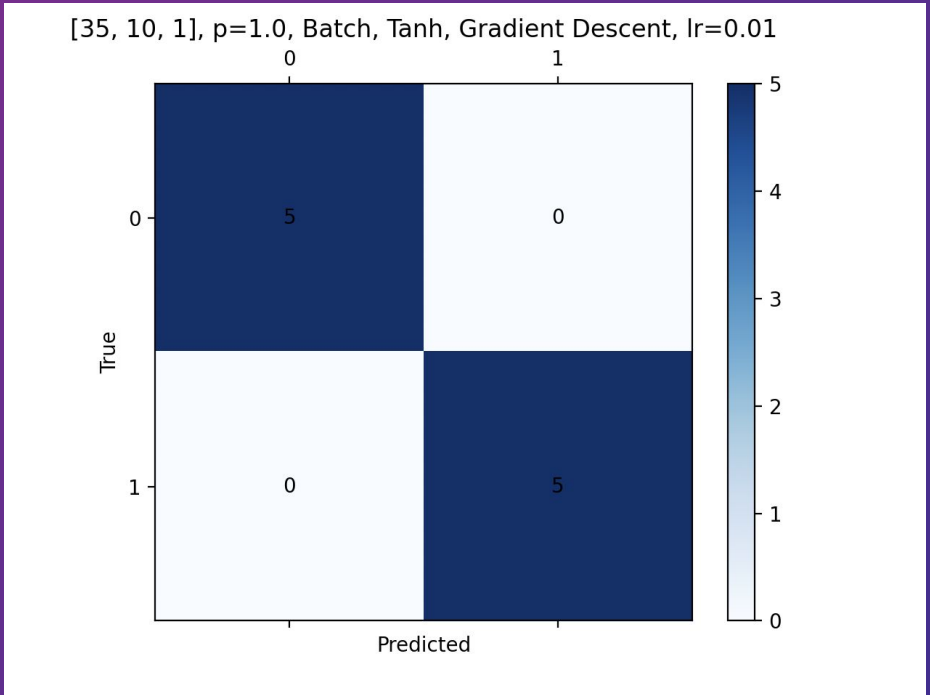


p = 1, Grad. desc


[35, 10, 1], p=1.0, Batch, Tanh, Gradient Descent
learning_rate=0.01
learning_rate=0.001
learning_rate=0.0001
error
0.0
0.1
0.2
0.3
0.4
0
25000
50000
75000
100000
125000
150000
175000
200000
epoch

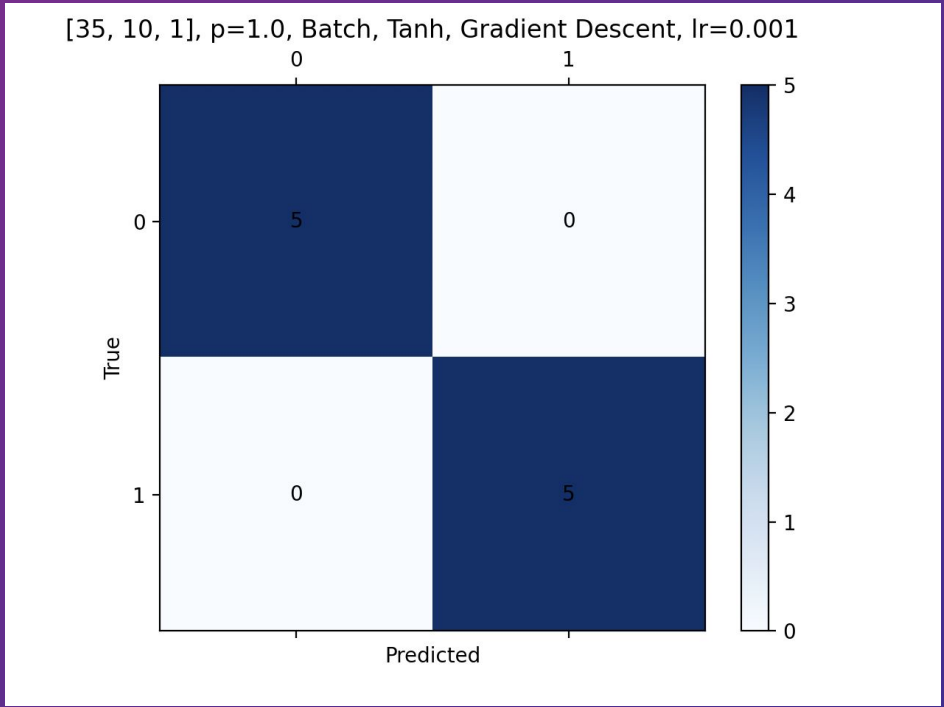
Matriz de Confusión

p = 1, Grad. desc, lr=0.01



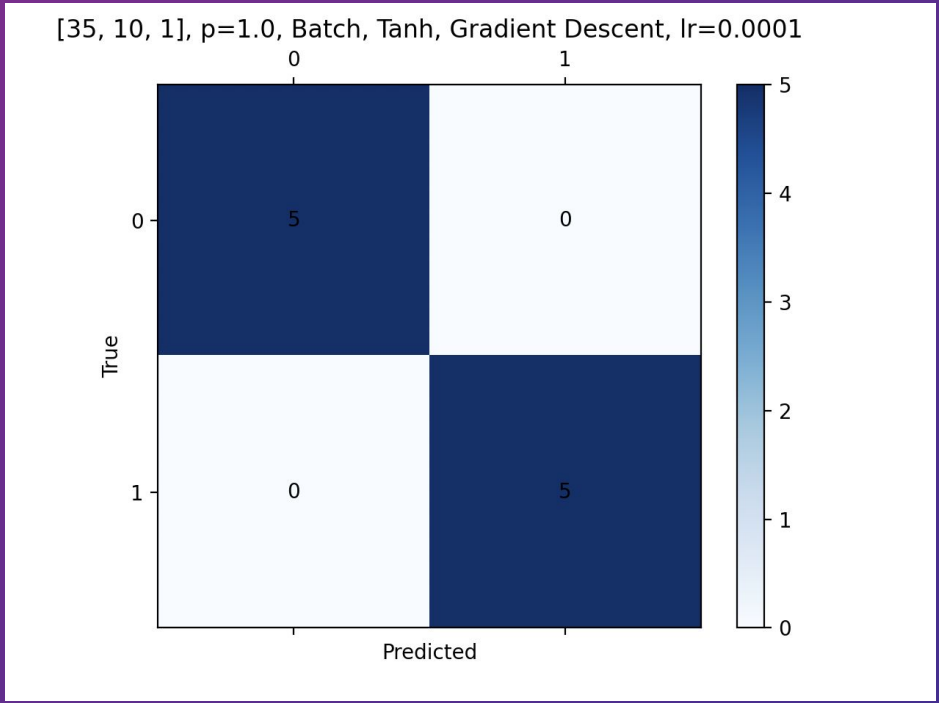
Matriz de Confusión

p = 1, Grad. desc, lr=0.001

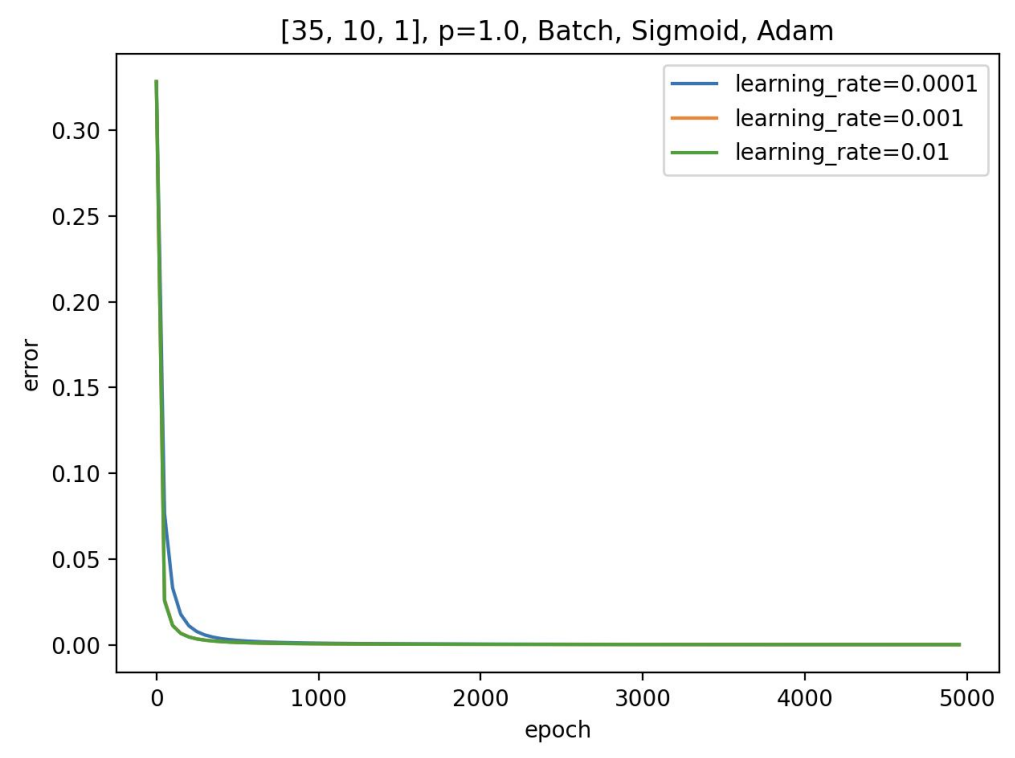


Matriz de Confusión

p = 1, Grad. desc, lr=0.0001

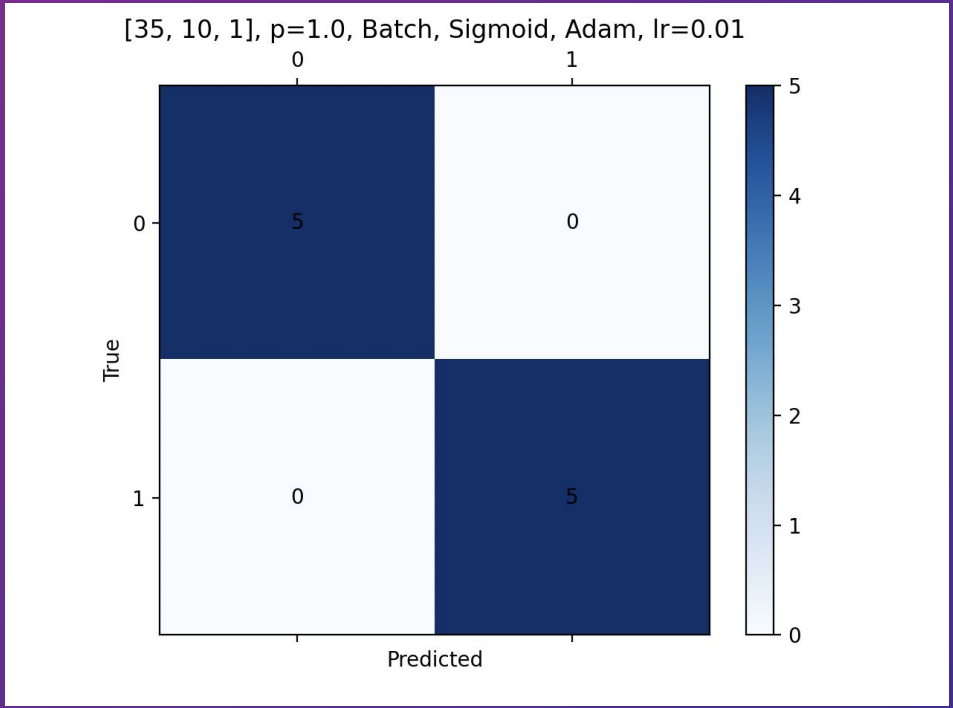


p = 1, ADAM


[35, 10, 1], p=1.0, Batch, Sigmoid, Adam
learning_rate=0.0001
learning_rate=0.001
learning_rate=0.01
error
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1000
2000
3000
4000
5000
epoch

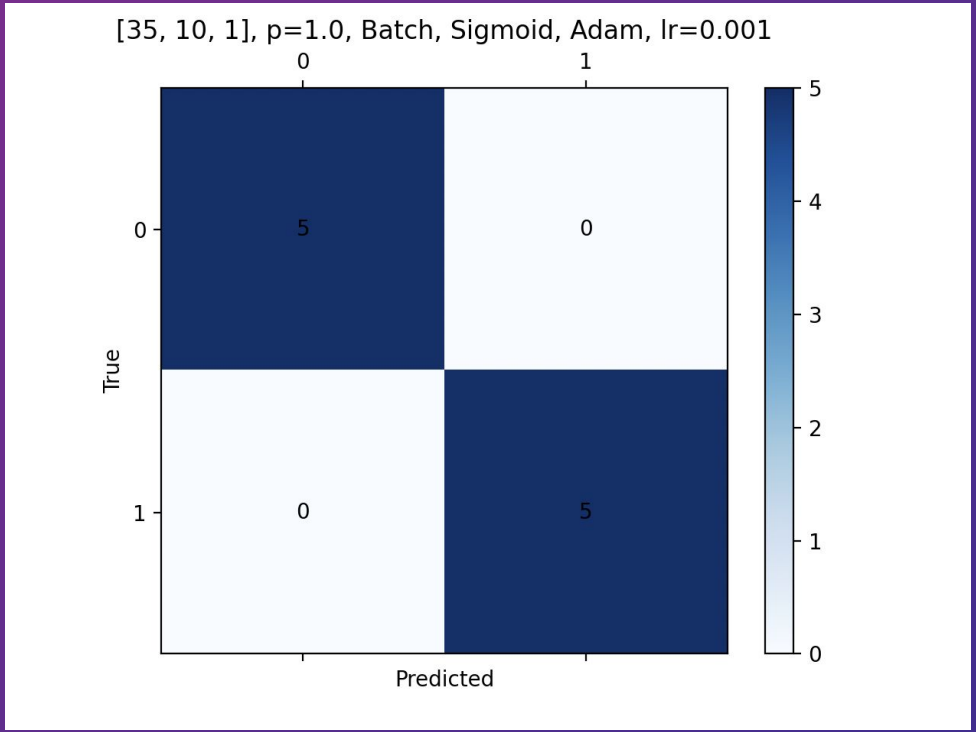
Matriz de Confusión

p = 1, ADAM, lr=0.01



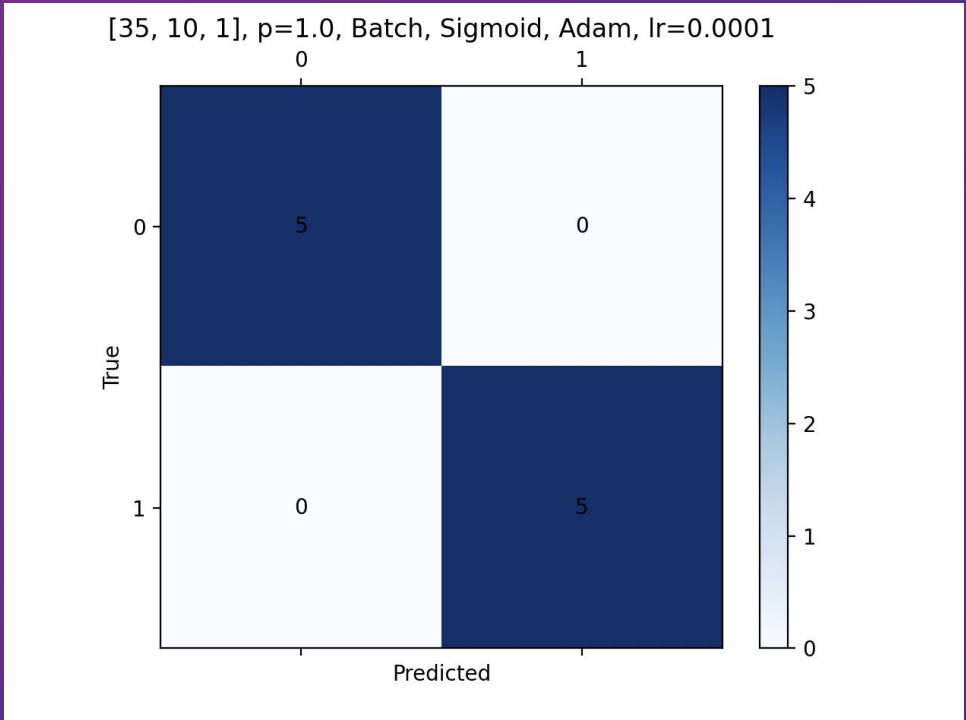
Matriz de Confusión

p = 1, ADAM, lr=0.001



Matriz de Confusión

p = 1, ADAM, lr=0.0001



Layers: [10]

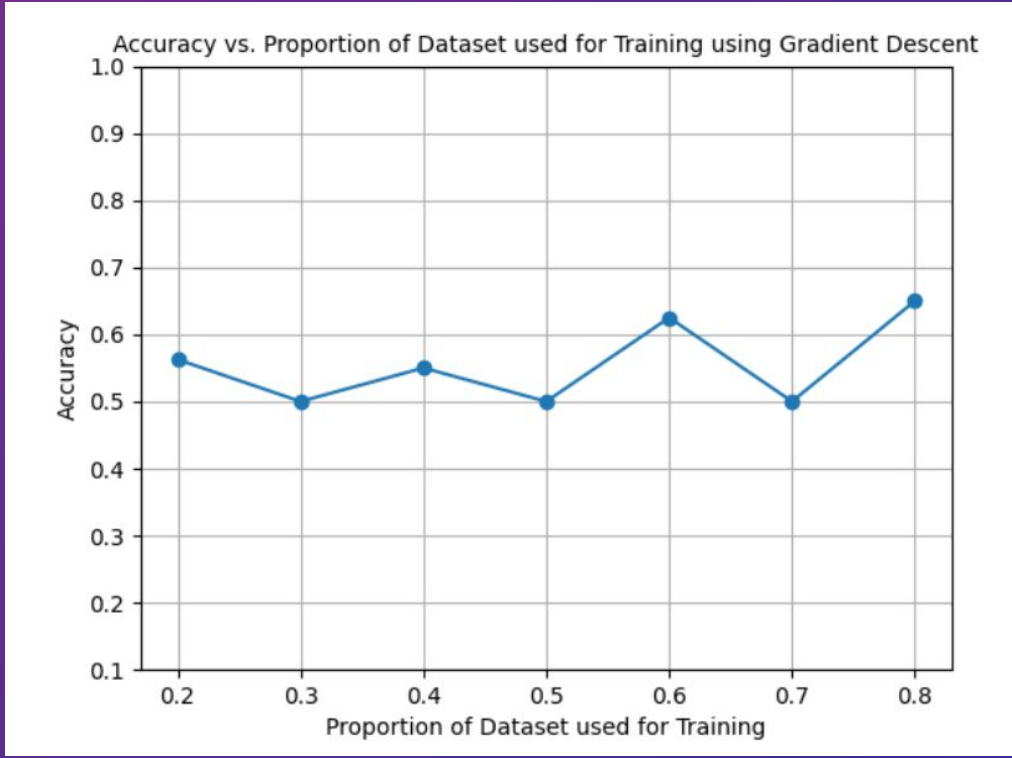
Función Activación: Tanh

Lr: 0.01

Método de Optimización: Gradiente Descendente

Epoch: 5000

Iterations: 10



Layers: [10]

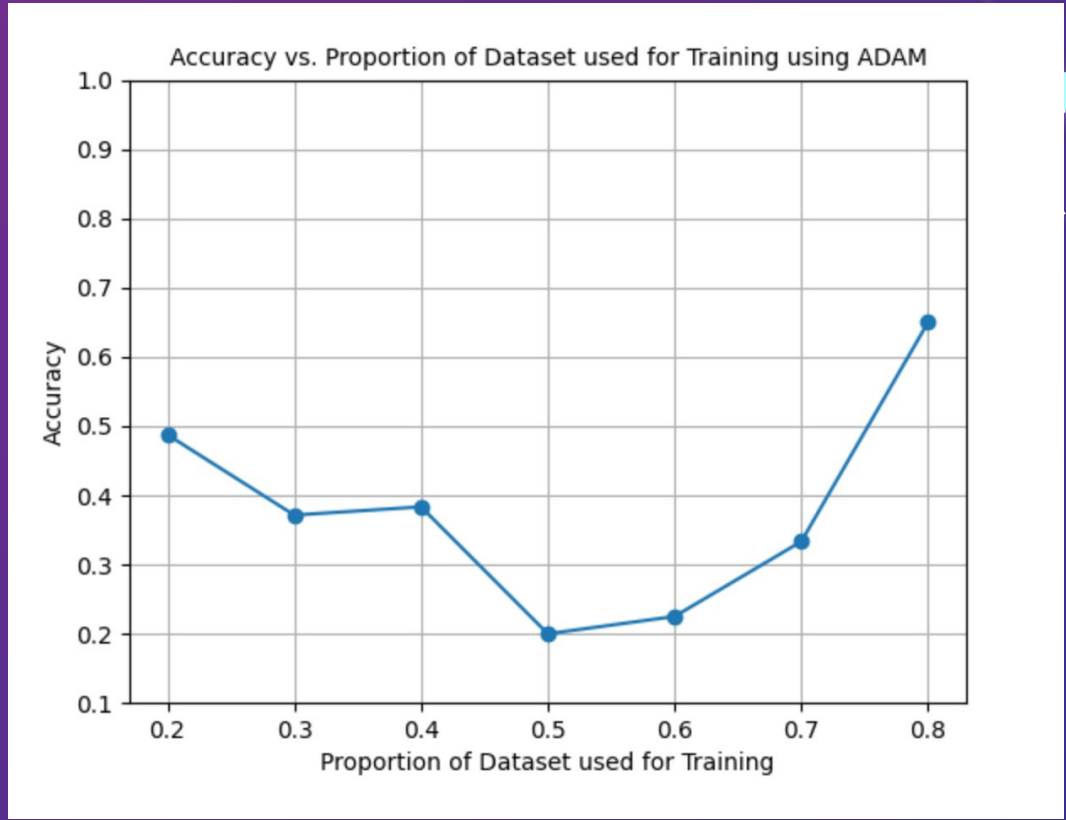
Función Activación: Logística

Lr: 0.01

Método de Optimización: ADAM

Epoch: 20000

Iterations: 10

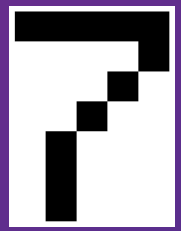


Conclusiones

- Para este problema, el método de ADAM converge más rápido para el training set que el gradiente descendiente.
- La red no puede inferir el resultado de la paridad de un número, esto se debe a que la imágen de un número no tiene relación con su paridad.

Parte C

Determinar qué dígito se corresponde con la entrada a la red. Por ejemplo, si alimentamos al perceptrón multicapa con una imagen del dígito “7”, la salida esperada será “7” (la salida puede tomar valores entre 0 y 9). Nótese que se utiliza el mismo perceptrón multicapa, con una salida de 10 neuronas.
Una vez que la red haya aprendido, utilizar patrones correspondientes a los dígitos del conjunto de datos, con sus píxeles afectados por ruido. Evaluar los resultados.



```
For the digit: 7, the perceptron guessed 7
```

Parte C

Conjunto de entrada: [0,1,2,3,4,5,6,7,8,9], donde cada número es su representación gráfica en una grilla de 7x5. En la red es un vector de $[0,1]^{35}$

Layers: [10]

Método de Optimización: Gradiente Descendente / ADAM

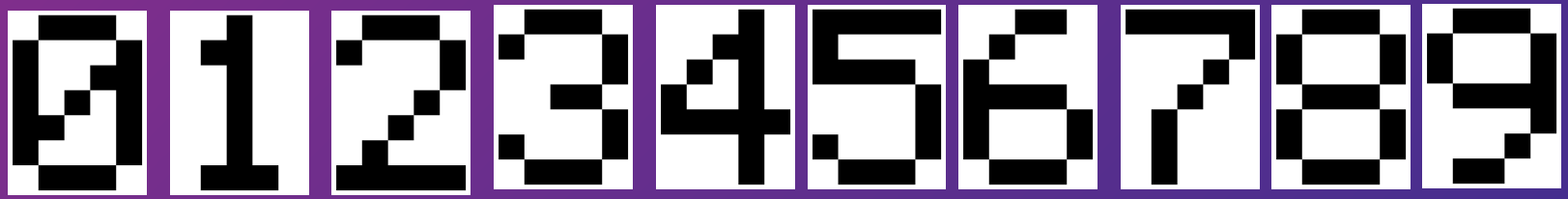
Función de Activación: Sigmoid / Tanh

Learning rate: 0.01/0.001/0.0001

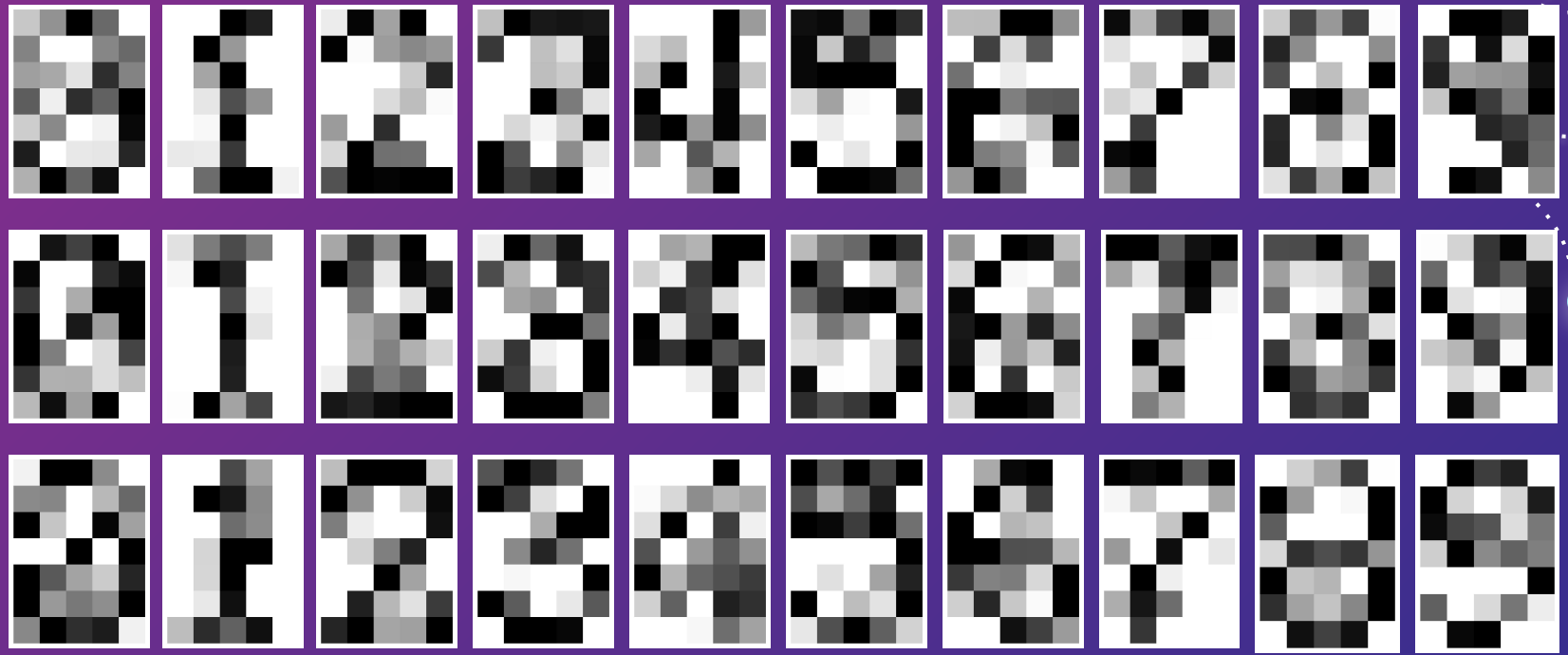
Conjunto de entrenamiento: conjunto de números del 0 al 9 sin ruido.

Conjunto de testeo: conjunto de 30 números del 0 al 9 con una máscara de ruido de intensidad 0.3.

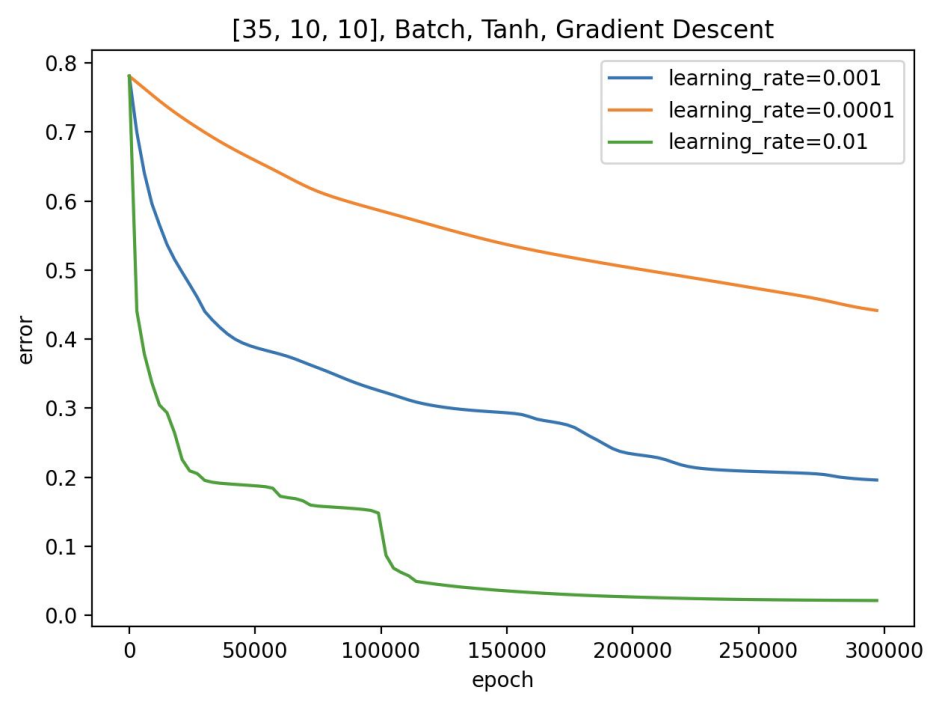
Training Set



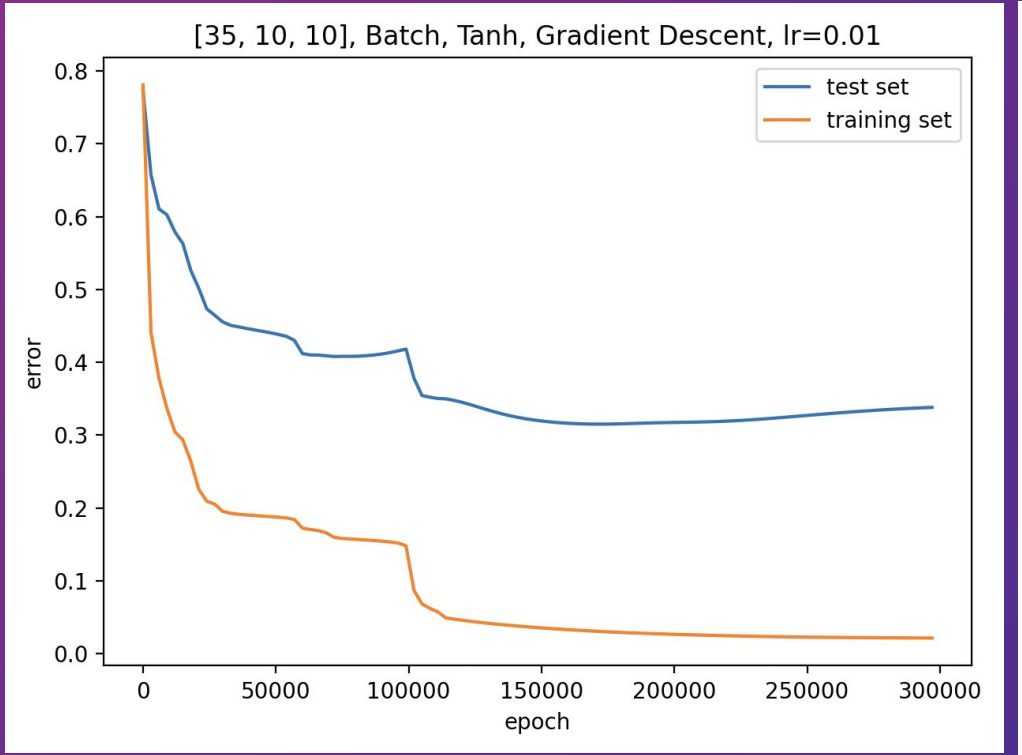
Testing Set (Noise intensity = 0.3)

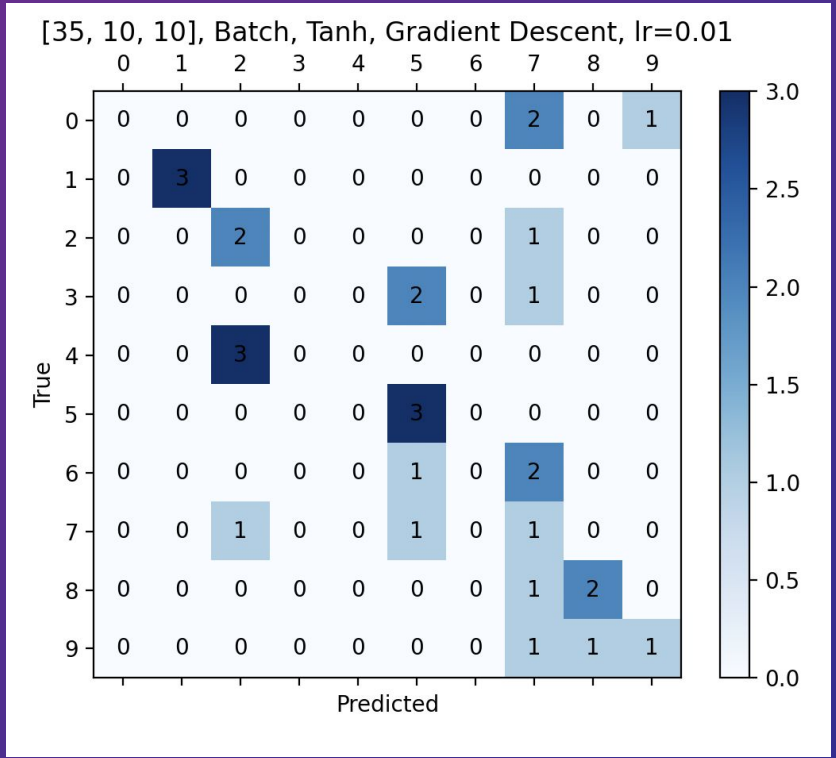


Grad. desc: Error vs Epoch

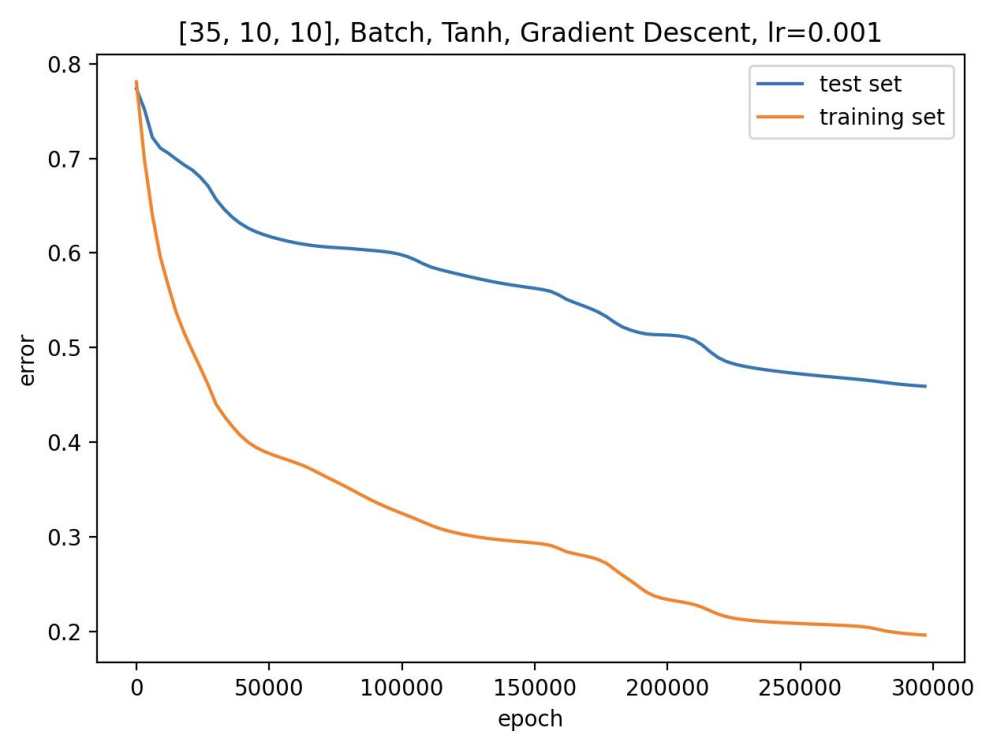


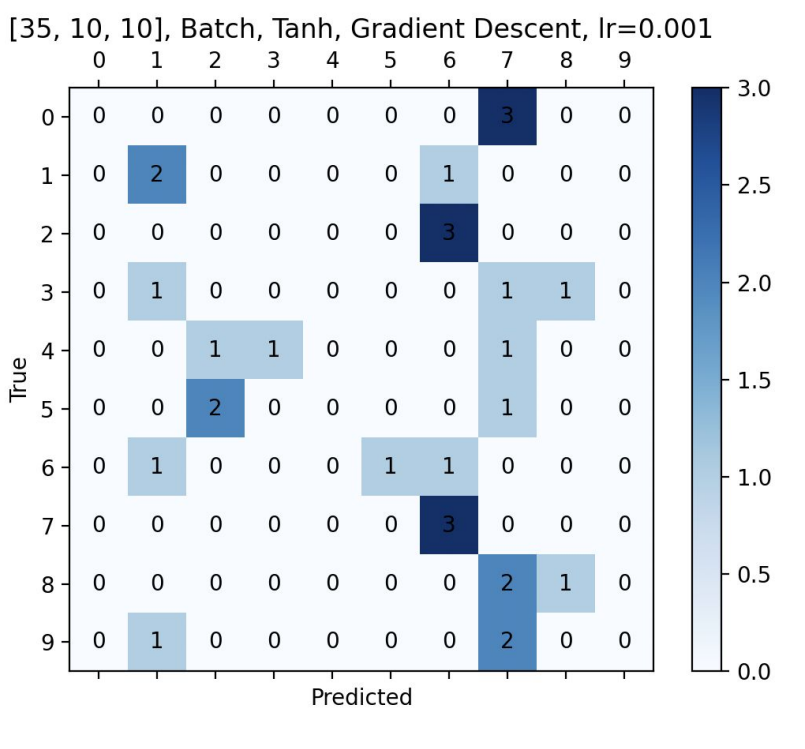
Grad. desc, lr=0.01



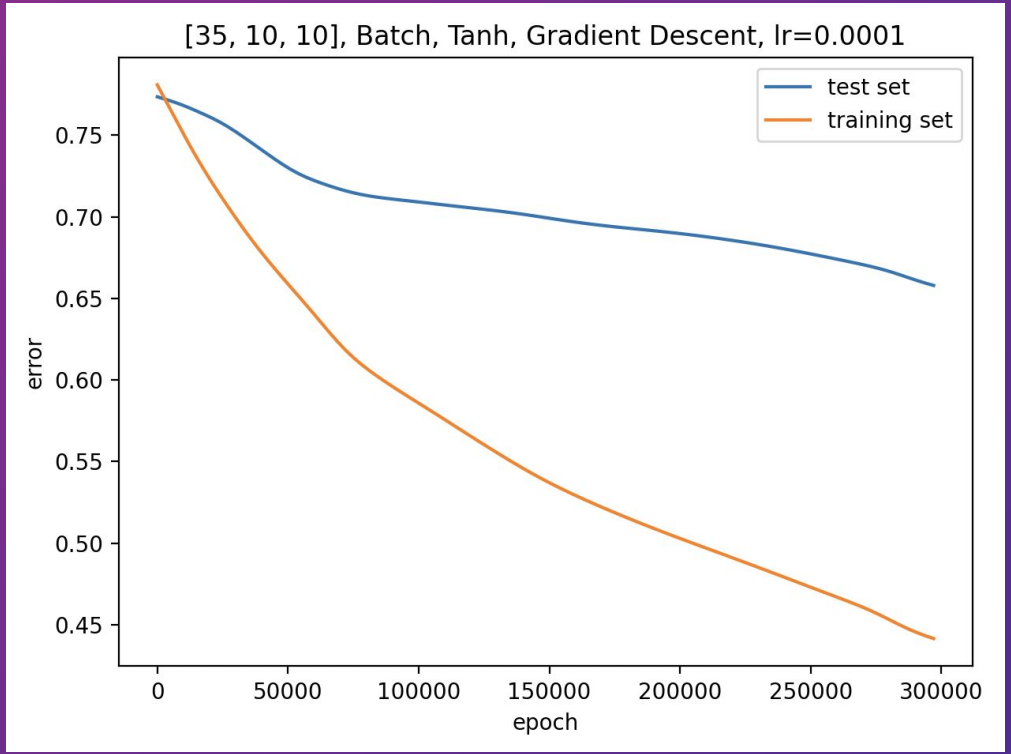


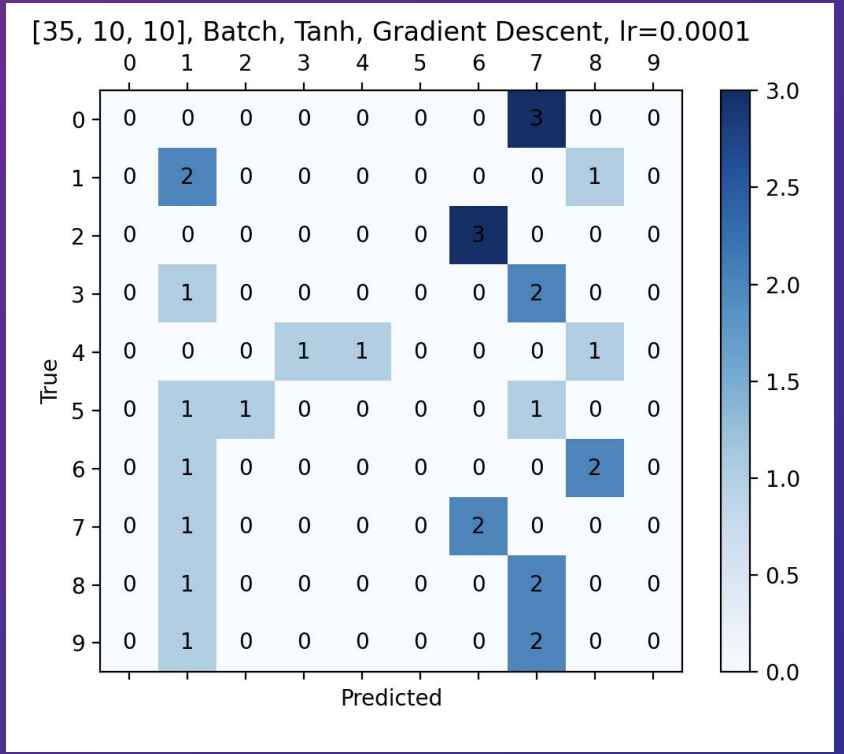
Grad. desc, lr=0.001





Grad. desc, lr=0.0001





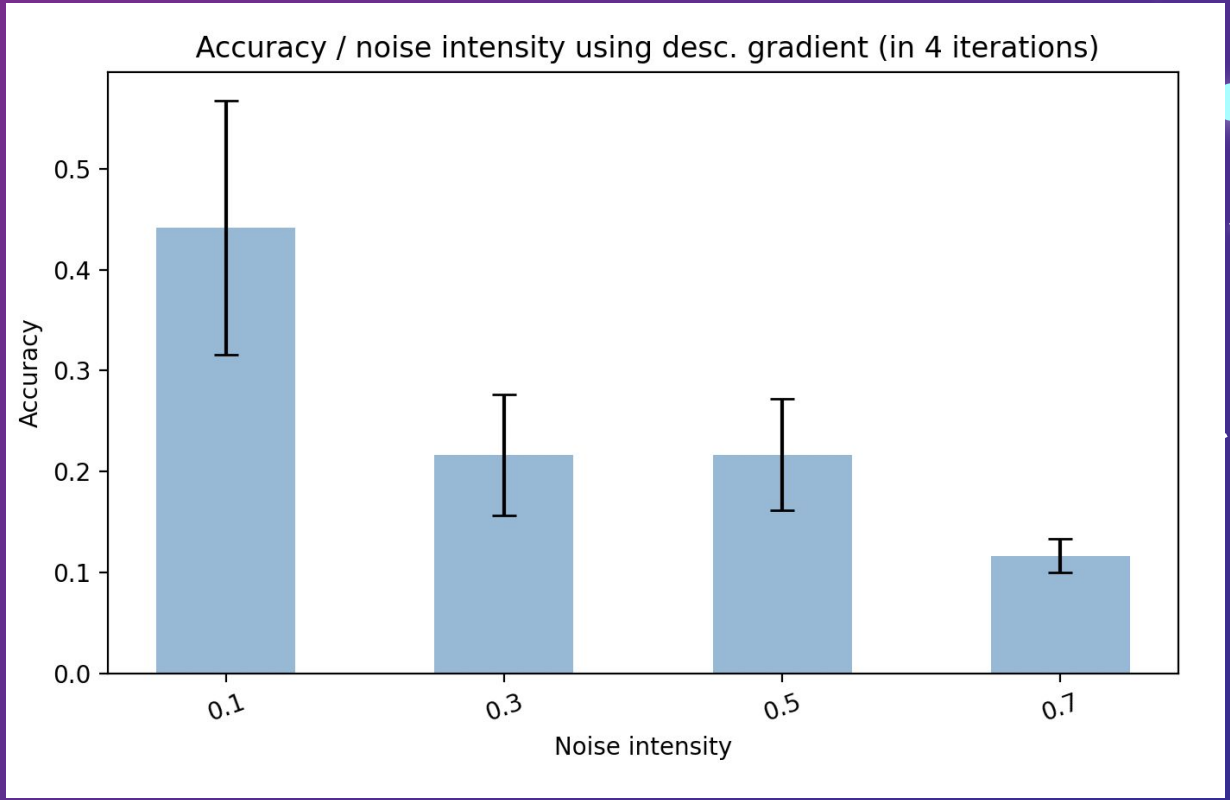
Conjunto de Entrenamiento:
Imágenes de los dígitos del 0 al 9 sin ruido

Conjunto de Testeo: 30 Imágenes de los dígitos del 0 al 9 con ruido

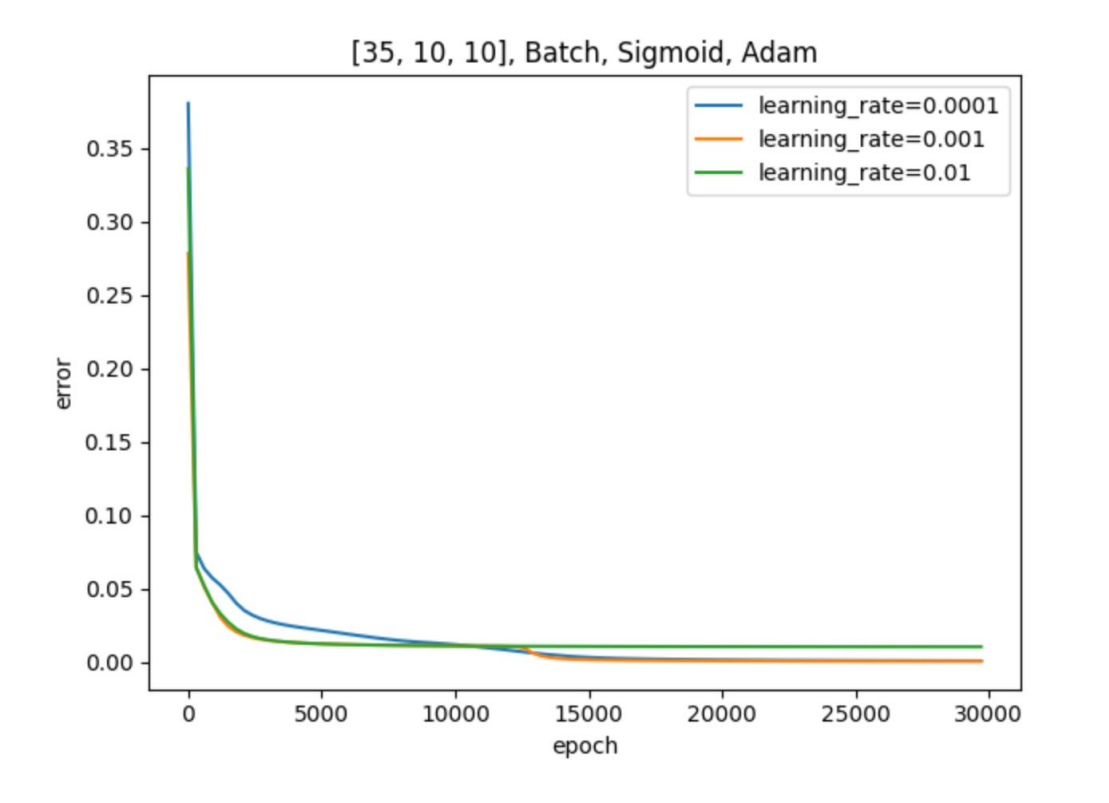
Layers: [10]

Método de Optimización:
Gradiente Descendente

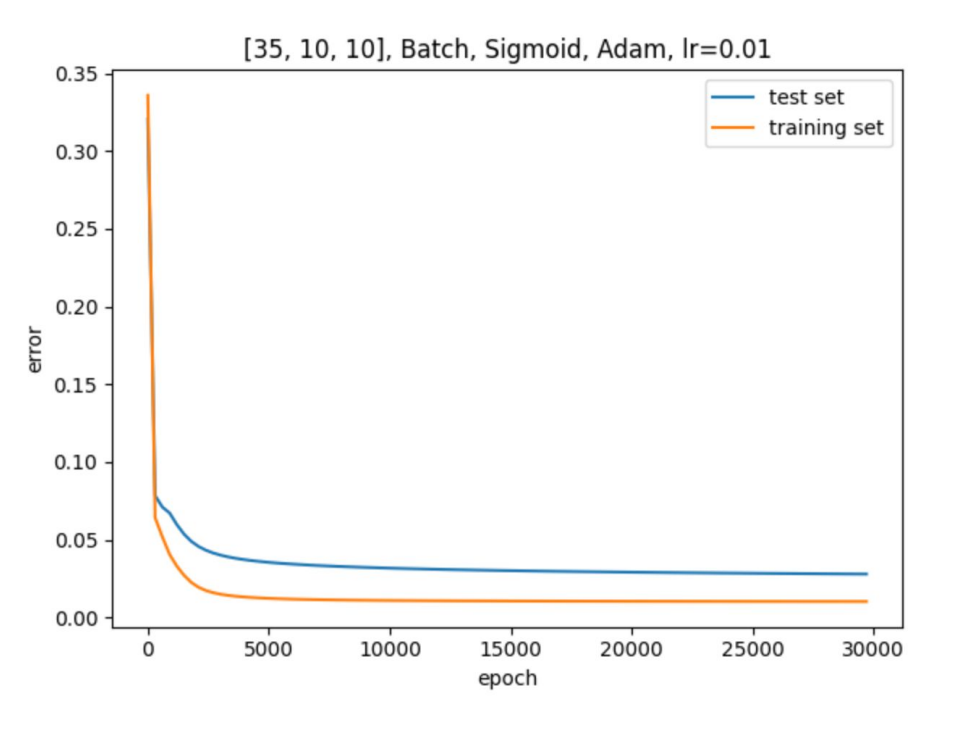
Learning Rate: 0.01

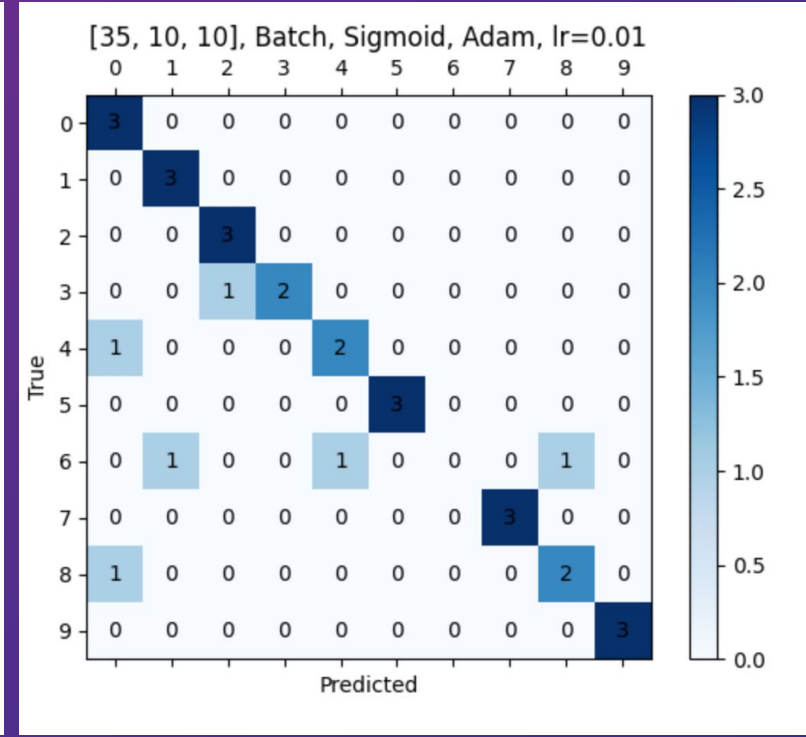


Adam: Error vs Epoch

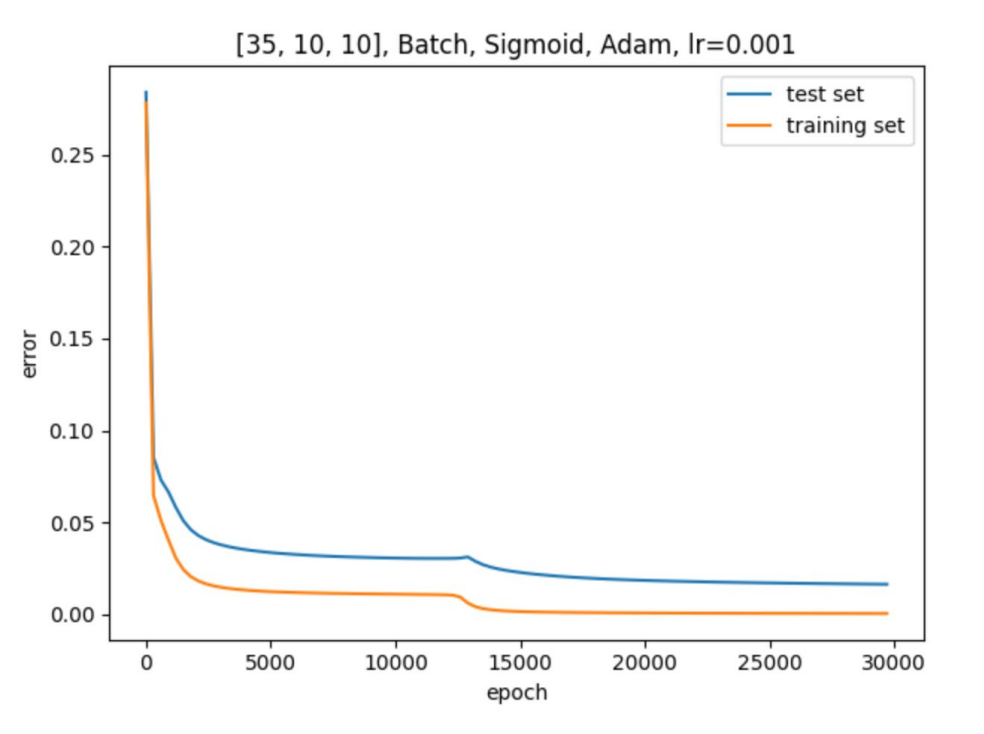


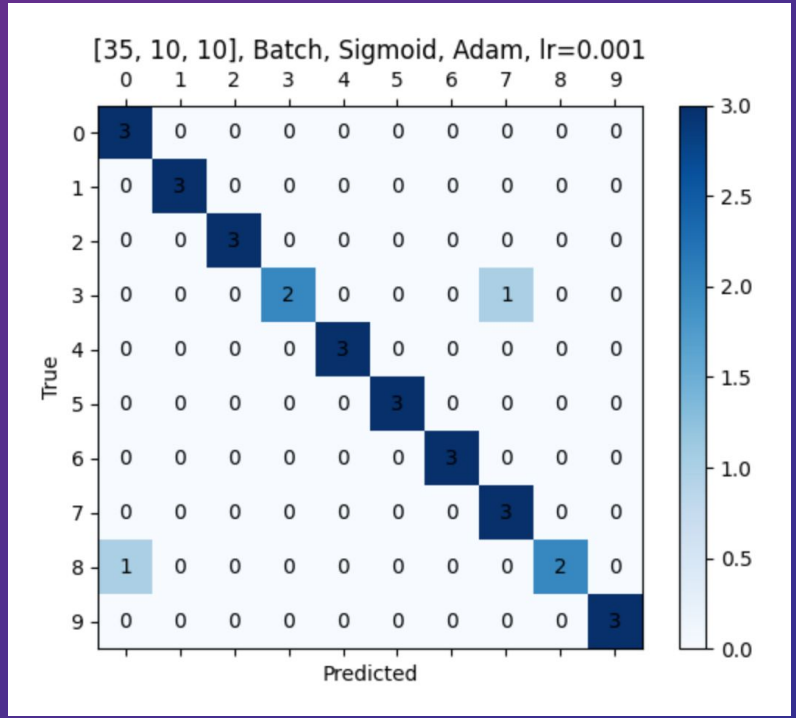
ADAM, lr=0.01


[35, 10, 10], Batch, Sigmoid, Adam, lr=0.01
test set
training set
error
epoch
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5000
10000
15000
20000
25000
30000

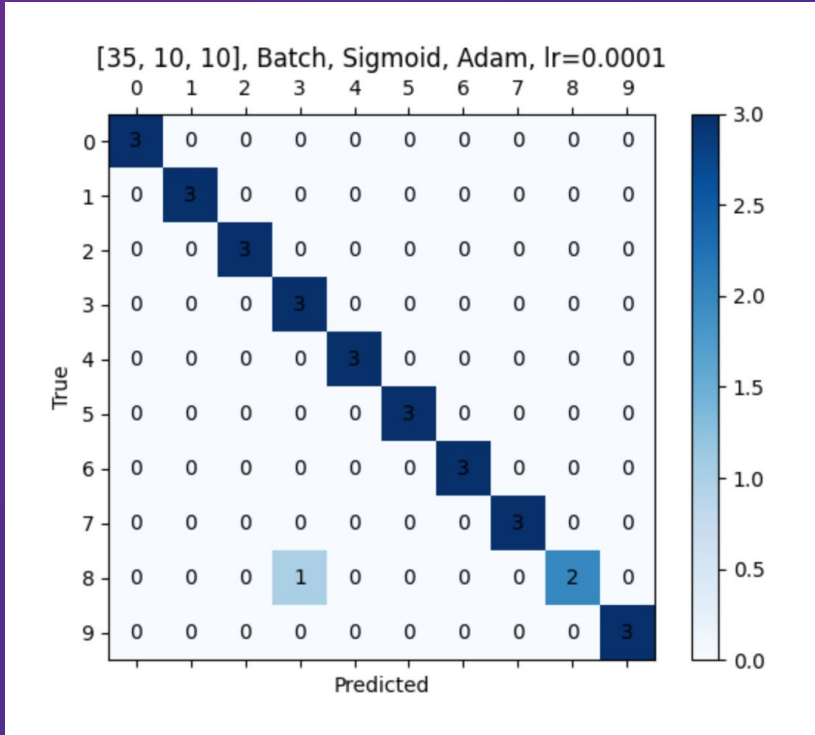

[35, 10, 10], Batch, Sigmoid, Adam, lr=0.01
True
Predicted
0 1 2 3 4 5 6 7 8 9
3.0
2.5
2.0
1.5
1.0
0.5
0.0

ADAM, lr=0.001


[35, 10, 10], Batch, Sigmoid, Adam, lr=0.001
test set
training set
error
epoch
0.25
0.20
0.15
0.10
0.05
0.00
0
5000
10000
15000
20000
25000
30000


[35, 10, 10], Batch, Sigmoid, Adam, lr=0.001
True
Predicted
0 1 2 3 4 5 6 7 8 9
3.0
2.5
2.0
1.5
1.0
0.5
0.0

ADAM, lr=0.0001



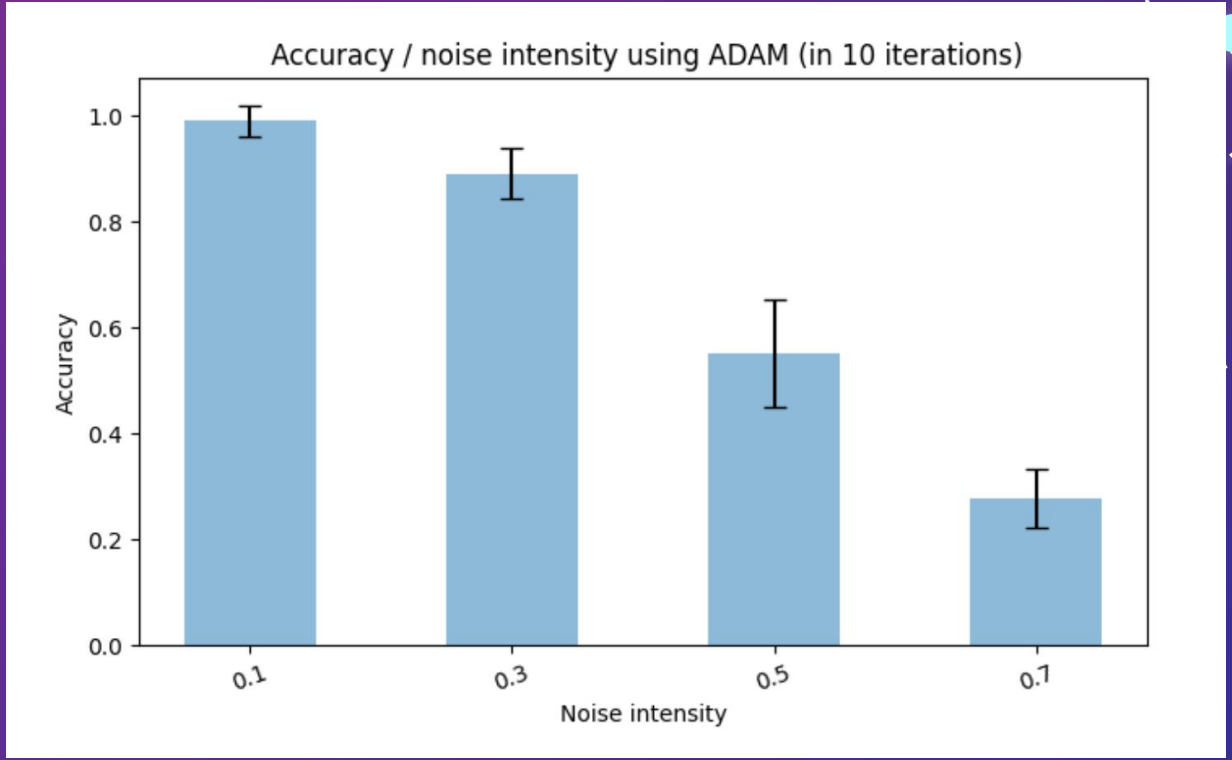
Conjunto de Entrenamiento:
Imágenes de los dígitos del 0 al 9 sin ruido

Conjunto de Testeo: 30 Imágenes de los dígitos del 0 al 9 con ruido

Layers: [10]

Método de Optimización:
ADAM

Learning Rate: 0.0001



Conclusiones

- El problema se puede resolver a través de un perceptrón multicapa.
- Este problema es generalizable, a diferencia de los anteriores: podemos observar que el error para el set de testeo disminuye a medida que se itera sobre las épocas.
- El método de optimización ADAM continúa convergiendo más rápido que el gradiente descendente y fue más preciso.

FIN