Perceptrón Simple y Multicapa

Sistemas de Inteligencia Artificial 2C-2022
Grupo 1 - "Augusta Ada King"

Domingues Paula	60148
Donikian Gastón	60067
Pavan Matias	58296
Rodriguez Manuel	60258

"Any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated,' so that activity in one facilitates activity in the other"

–Donald O. Hebb

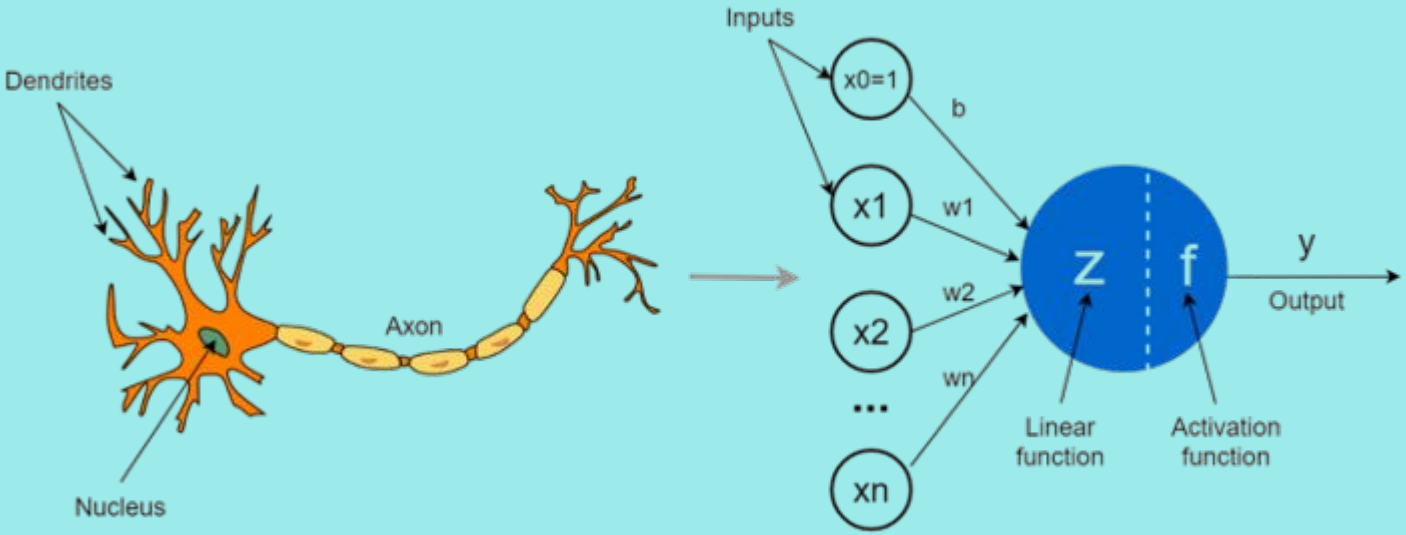
01 Perceptrón Simple

02 Perceptrón multicapa

01

Perceptrón Simple

Introducción



Ejercicio 1

Evaluación de perceptrón simple escalonado

Perceptrón Simple
Datos de entrada

AND

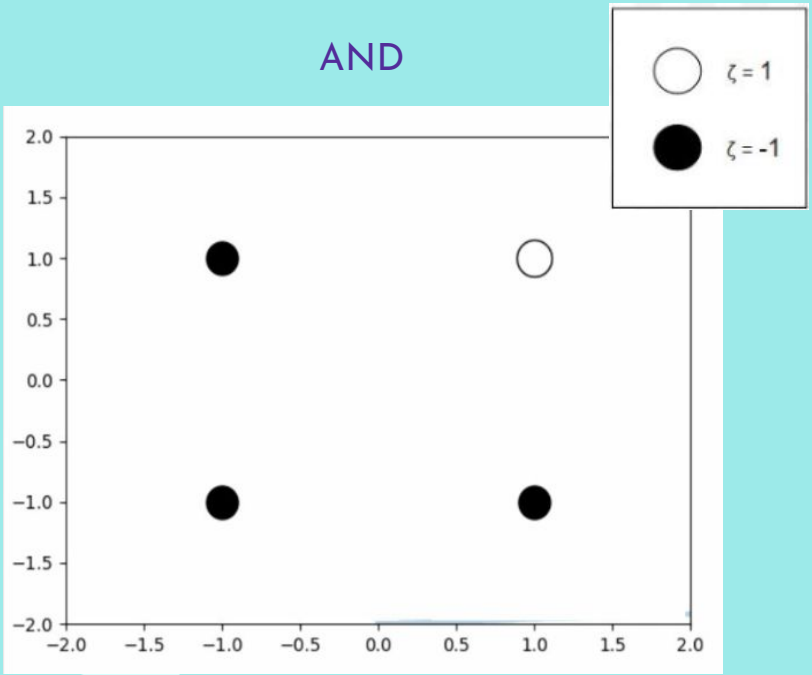
A	B	Esperado
-1	1	-1
1	-1	-1
1	1	1
-1	-1	-1

XOR

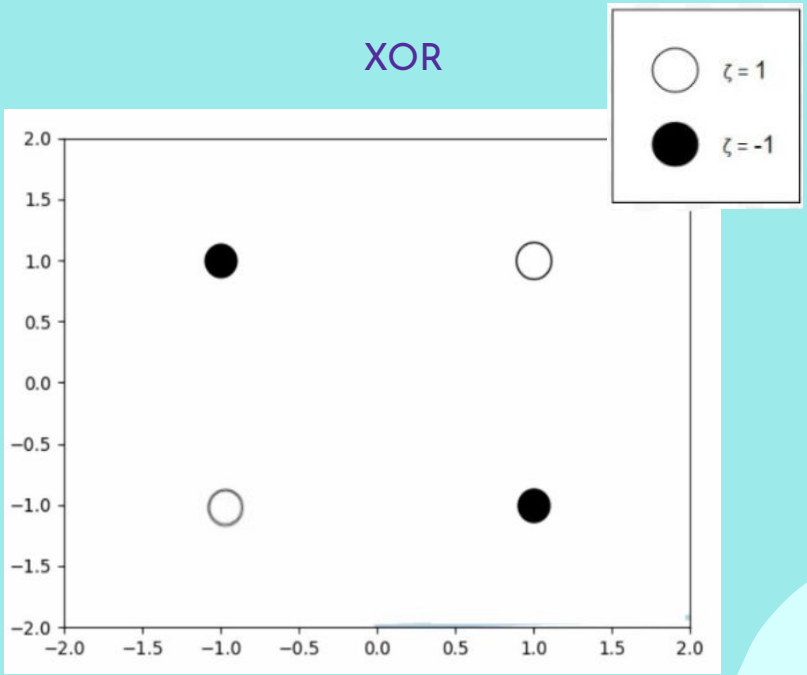
A	B	Esperado
-1	1	1
1	-1	1
1	1	-1
-1	-1	-1

Perceptrón Simple
Datos de entrada

AND

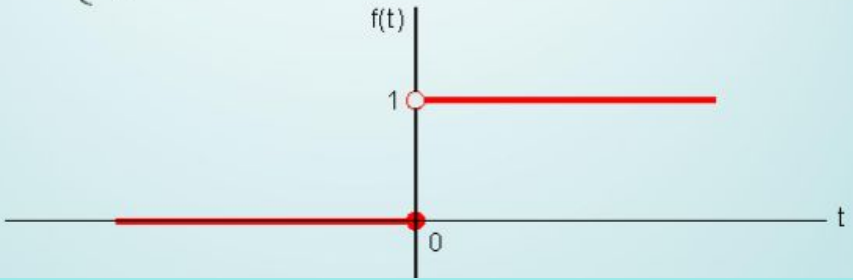


XOR



Perceptrón Simple
Función de activación - Escalón

$$U(t) = \begin{cases} 0, & t \leq 0 \\ 1, & t > 0 \end{cases}$$



$$O = U(\sum_{i=1}^{n} w_i \xi_i - \text{umbral})$$

Queremos que

O^u sea igual a ζ^u donde

ζ es el valor esperado para ξ

Perceptrón Simple
Aprendizaje del Perceptrón

$$\mathrm{w}_i^{nuevo} = \mathrm{w}_i^{viejo} + \Delta \mathrm{w}_i$$

$$\Delta \mathrm{w}_i = \eta(\zeta^u - O^u)\xi_u^i$$

η es la tasa de aprendizaje

$(\zeta^u - O^u)$ es el error asociado

ξ_u^i es la entrada asociada

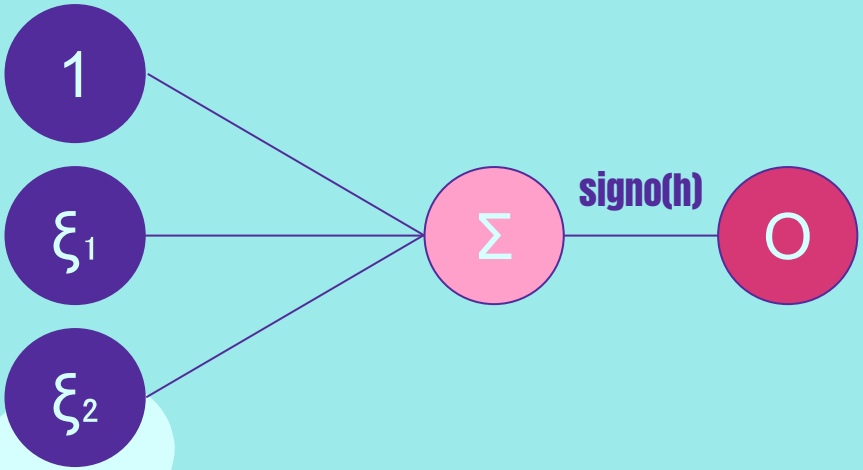
Queremos encontrar los w que minimicen:

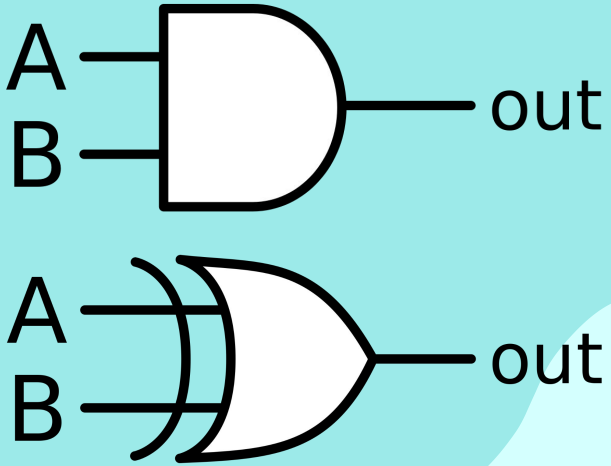
$$E(w) = \frac{1}{p}\sum_{u=1}^{p}(\zeta^u - O^u)^2$$

Función de error del perceptron

Perceptrón Simple

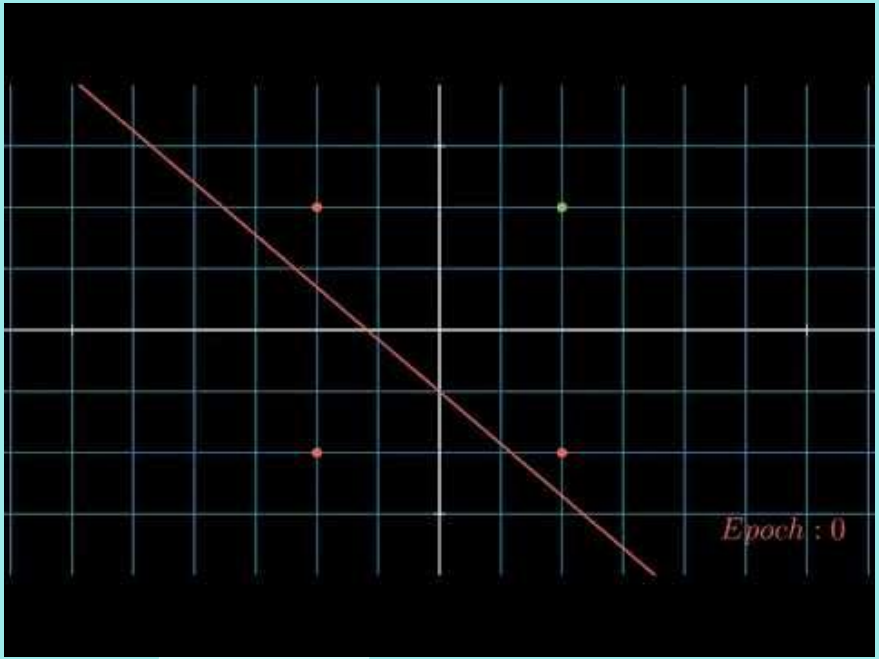
Perceptrón escalón contra dataset AND y XOR



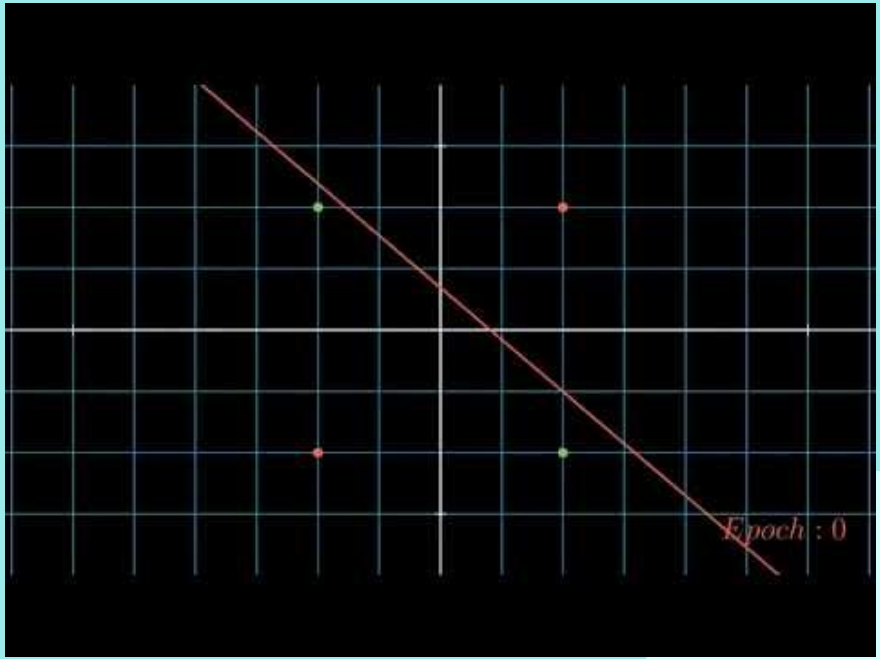


Perceptrón Simple
Resultados

AND



XOR



Parámetros: n = 0.01, F = SIGNO

¿Qué puede decir acerca de los problemas que puede resolver el perceptrón simple escalón en relación a los problemas planteados en la consigna?

Ejercicio 2

Evaluación de perceptrón simple lineal y no lineal

Perceptrón Simple
Datos de entrada

$$\xi = (\xi_1, \xi_2, \xi_3) =$$

1.200	-0.800	0.000
1.200	0.000	-0.800
1.200	-0.800	1.000
0.000	1.200	-0.800
7.900	1.000	0.000
0.400	0.000	2.700
0.000	0.400	2.700
-1.300	3.230	3.000
0.400	2.700	0.000
0.400	2.700	2.000
-1.300	0.000	3.230
0.000	-1.300	3.230
7.900	1.000	-2.000
1.800	0.000	1.600
0.000	-2.000	2.000
-0.500	0.600	0.000
0.000	1.800	1.600
-2.000	2.000	0.000
-0.500	0.600	2.500
7.900	0.000	1.000
-1.300	3.230	0.000
0.000	7.900	1.000
-2.000	0.000	2.000
-2.000	2.000	-1.000
0.000	-0.500	0.600
1.800	1.600	1.300
-0.500	0.000	0.600
1.800	1.600	0.000

$$\zeta =$$

21.755
7.176
43.045
2.875
26.503
68.568
61.301
23.183
2.820
17.654
72.512
88.184
4.653
49.000
76.852
7.871
18.543
2.660
51.000
64.107
1.480
0.320
40.131
0.995
24.974
21.417
18.243
6.914

Perceptrón Simple
Problema de estandarización y escalaje

Inputs se escalarizan o estandarizan según la función de activación.

- Signo, Linear → Estandarización
- TANH → Escalarización en (-1,-1)
- EXP, RELU → Escalarización en (0,1)

Los outputs siempre se estandarizan para evitar perder la informacion que dan los outliers

Perceptrón Simple
Funciones de activación

LINEAL

IDENTIDAD

$$f(x) = x$$

NO LINEAL

TANH

$$tanh(x) = \frac{sinh(x)}{cosh(x)}$$

RELU

$$f(x) = max(0, x)$$

EXP

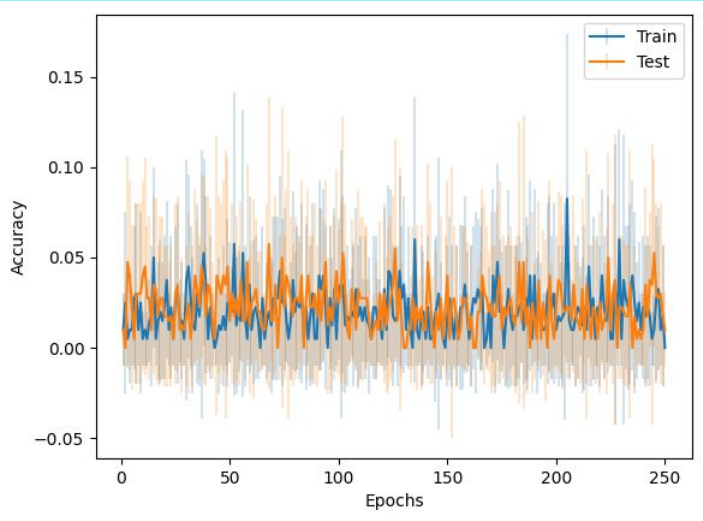
$$f(x) = \frac{1}{1 + e^{-x}}$$

Perceptrón lineal
Resultados

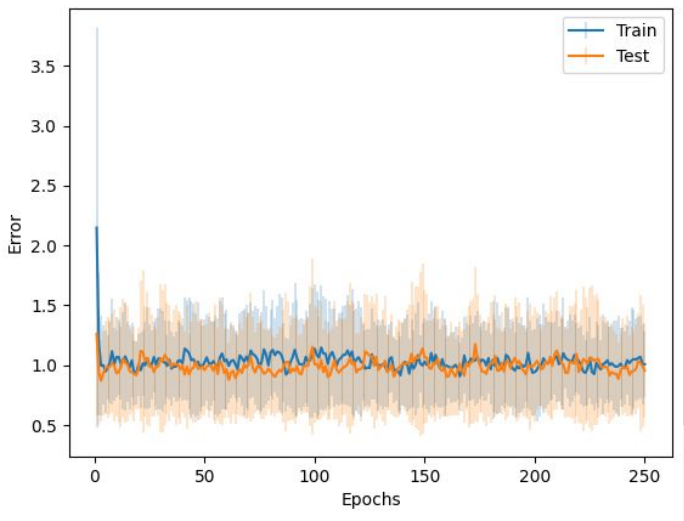
$$f(x) = x$$

Parámetros:
n = 0.05
Epochs = 250
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Accuracy



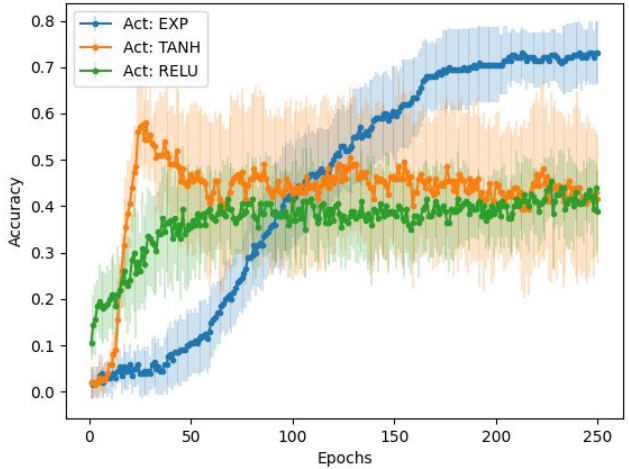
Error



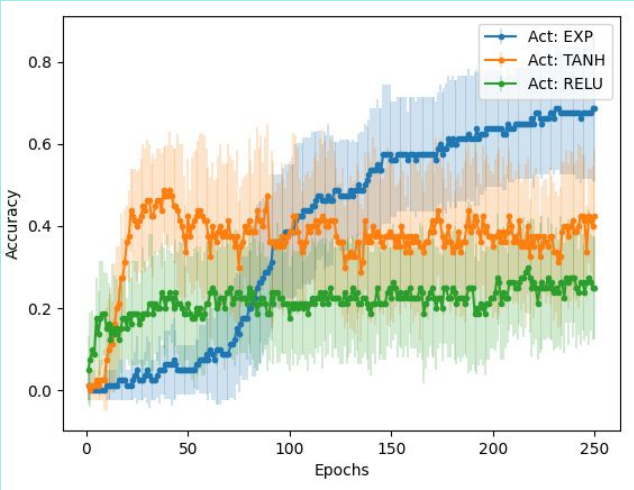
Perceptrón no lineal
Resultados variando la función de activación

Parámetros:
n = 0.05
Epochs = 250
Beta = 1
Training online
Porcentaje de entrenamiento: 70%
Prediction Threshold = 0.01

Training accuracy



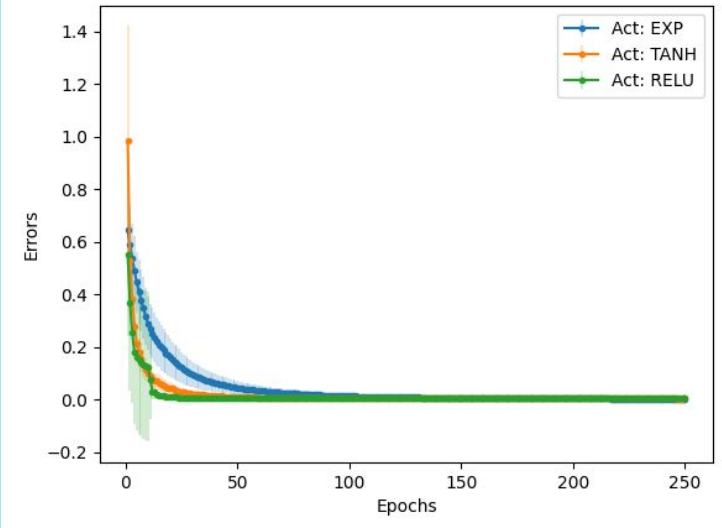
Test accuracy



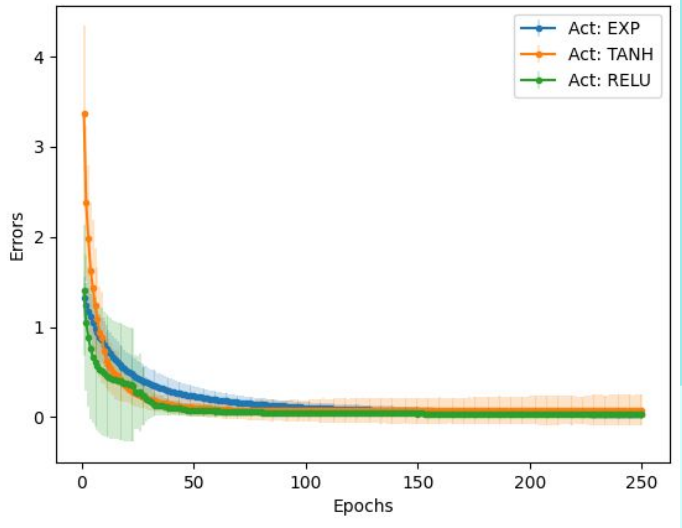
Perceptrón no lineal
Resultados variando la función de activación

Parámetros:
n = 0.05
Epochs = 250
Beta = 1.0
Training online
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Training error



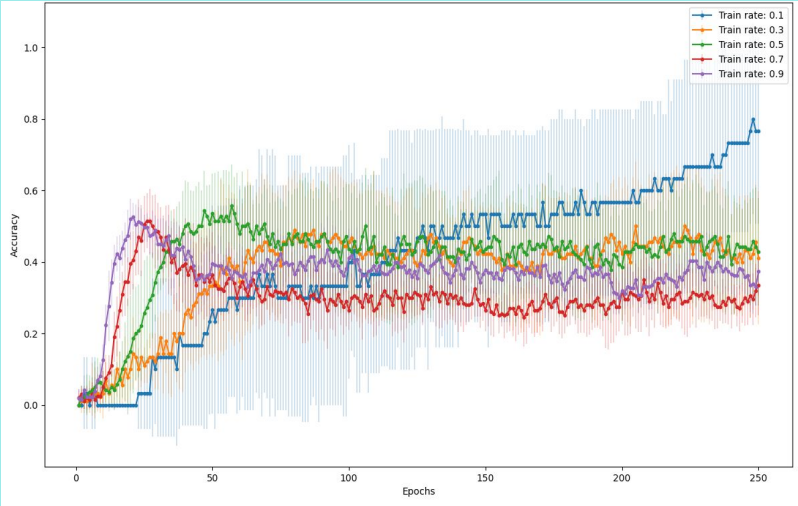
Test error



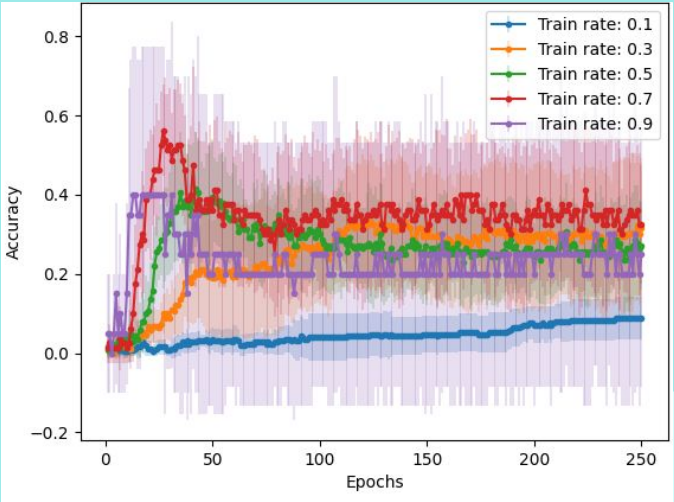
Perceptrón Simple
Resultados variando porcentaje de división

Parámetros:
n = 0.05
Epochs = 250
Beta = 1.0
Training online
F = TANH
Prediction threshold = 0.1

Training accuracy



Test accuracy

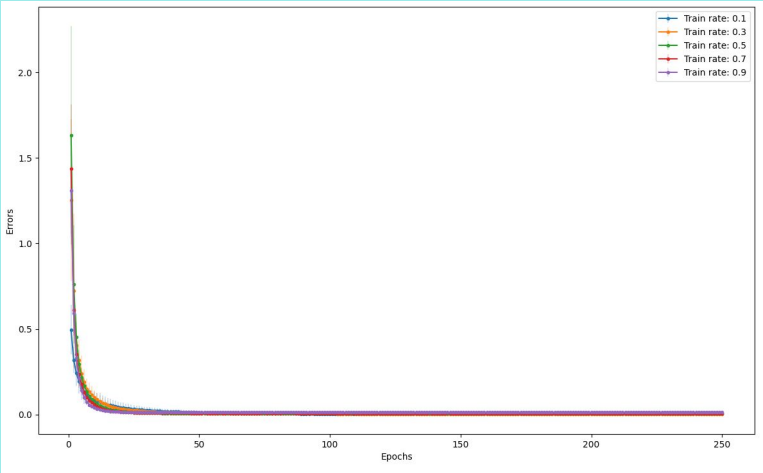


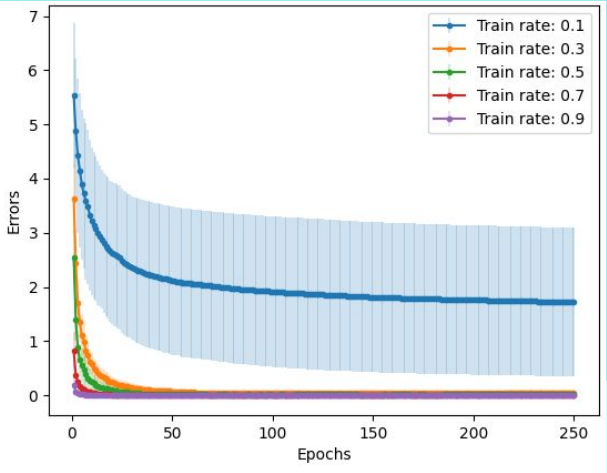
Perceptrón Simple
Resultados variando porcentaje de división

Training error

Test error

Parámetros:
n = 0.05
Epochs = 250
Beta = 1.0
Training online
F = TANH
Prediction threshold = 0.1

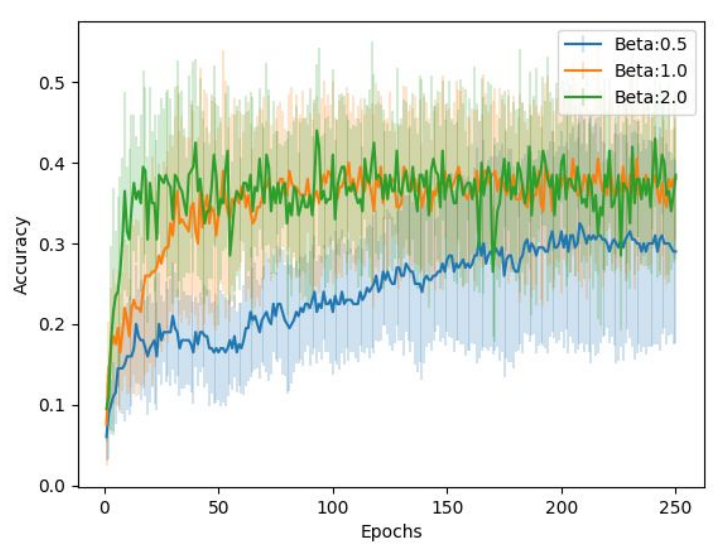




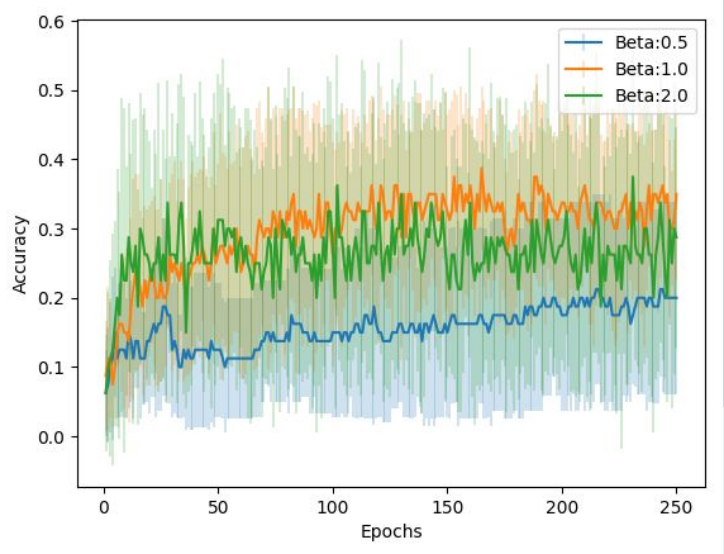
Perceptrón Simple
Resultados variando Beta (con RELU como F)

Parámetros:
n = 0.05
Epochs = 250
F = RELU
Training online
N = 10
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Training accuracy



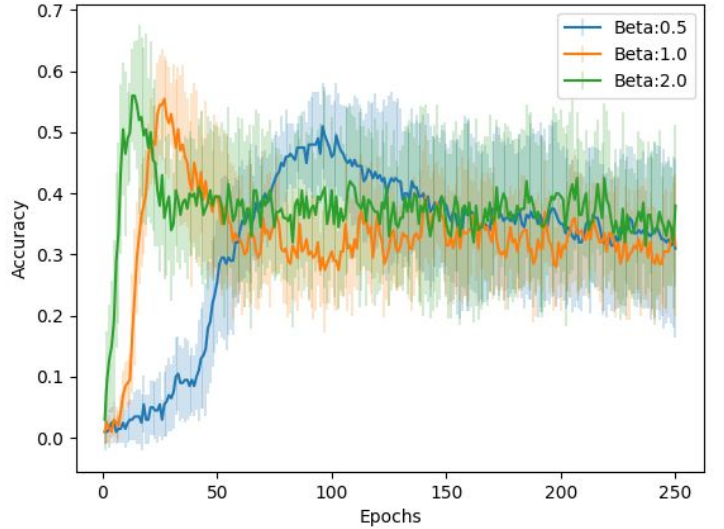
Test accuracy



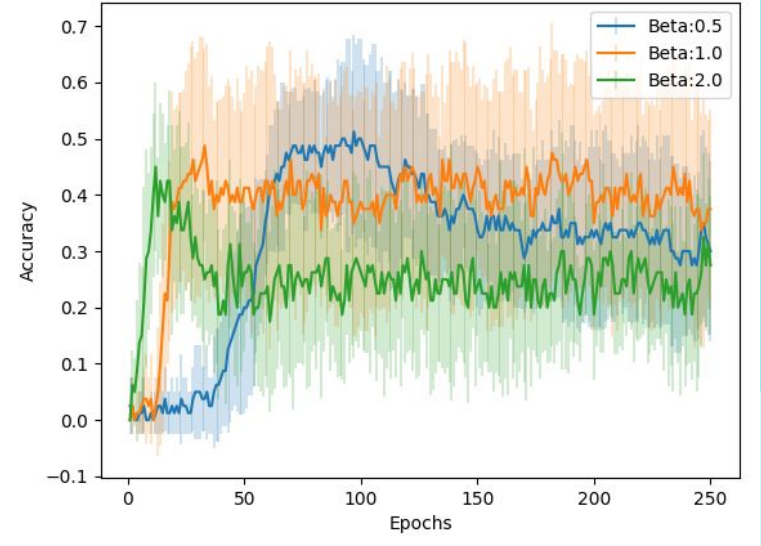
Perceptrón Simple
Resultados variando Beta (con tanh como F)

Parámetros:
n = 0.05
Epochs = 250
F = TANH
Training online
N = 10
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Training accuracy



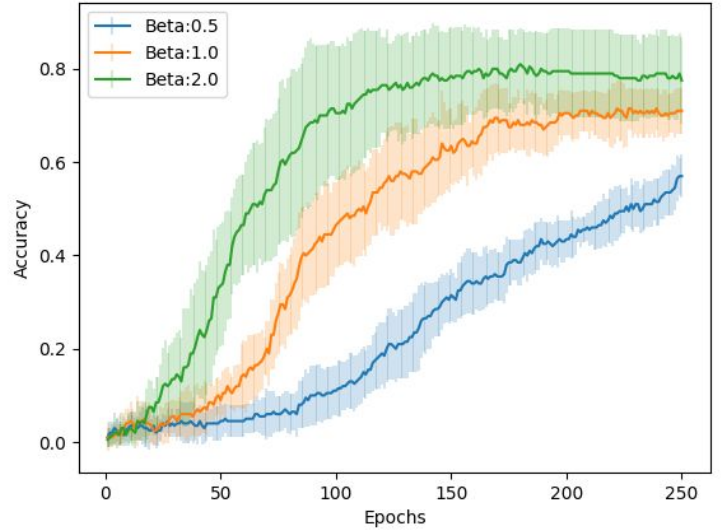
Test accuracy



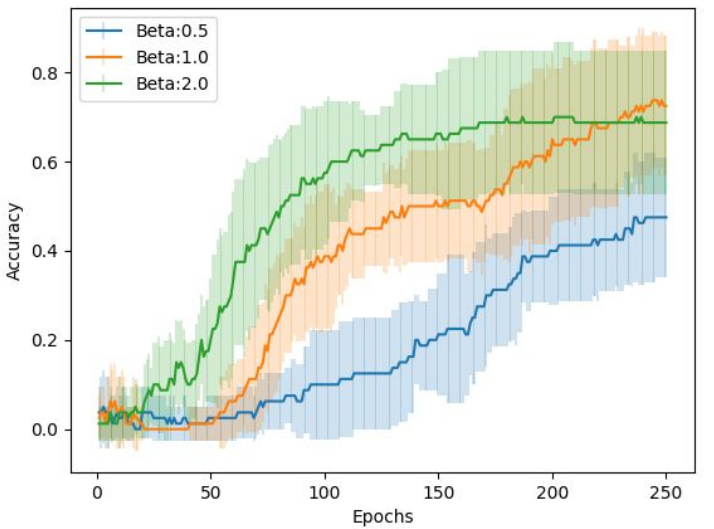
Perceptrón Simple
Resultados variando Beta (con exponencial como F)

Parámetros:
n = 0.05
Epochs = 250
F = EXP
Training online
N = 10
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Training accuracy



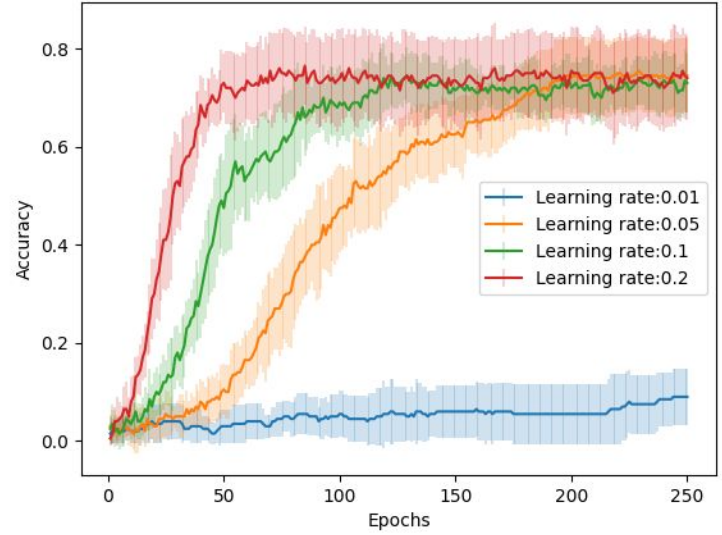
Test accuracy



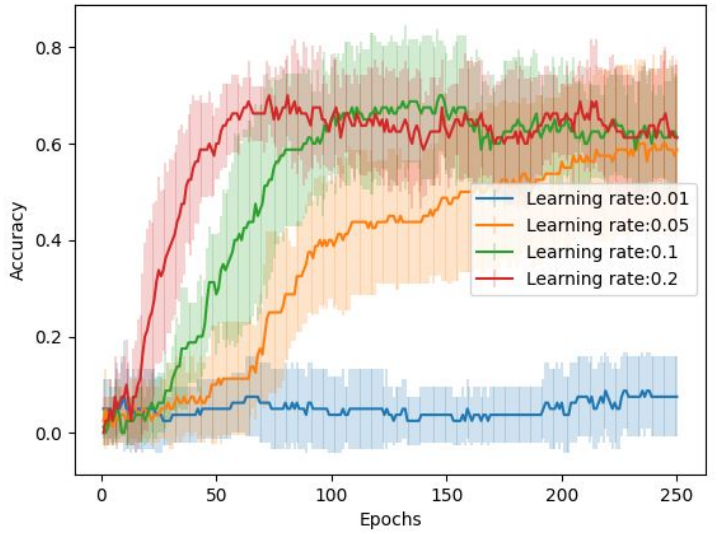
Perceptrón Simple
Resultados variando learning rate

Parámetros:
Epochs = 250
Beta = 1.0
F = EXP
Training online
N = 10
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Training accuracy



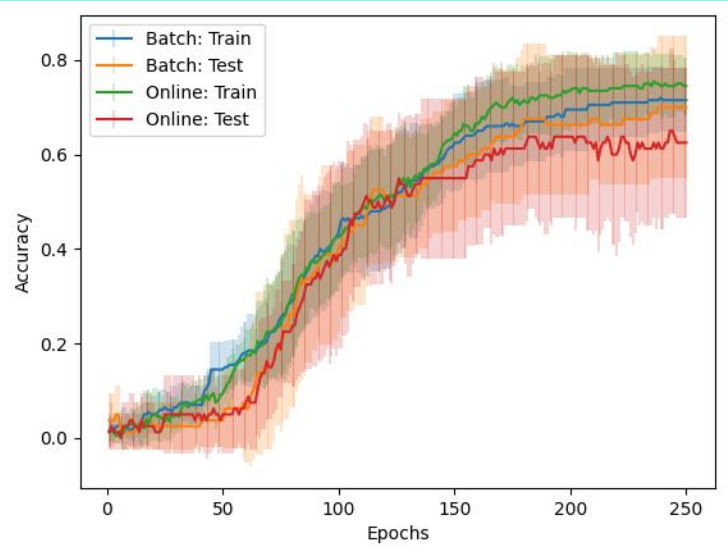
Test accuracy



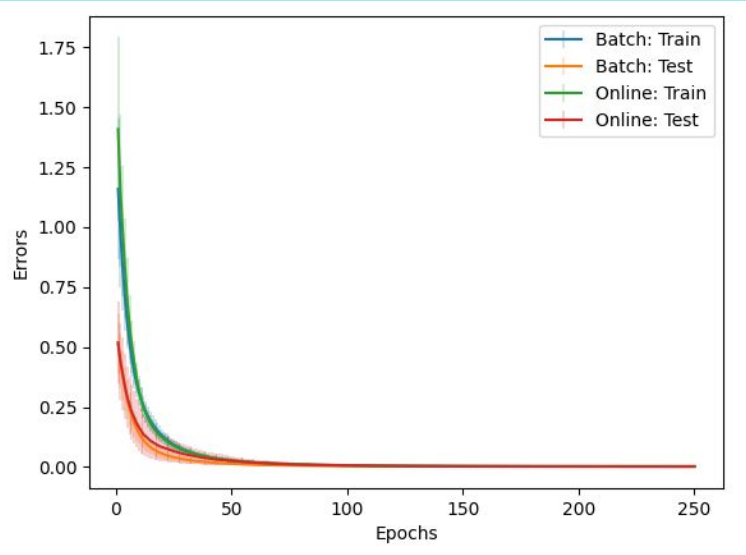
Perceptrón Simple
Resultados de Batch vs. Online

Parámetros:
n = 0.05
Epochs = 250
Beta = 1.0
F = EXP
N = 100
Prediction threshold = 0.01
Porcentaje de entrenamiento: 70%

Accuracy

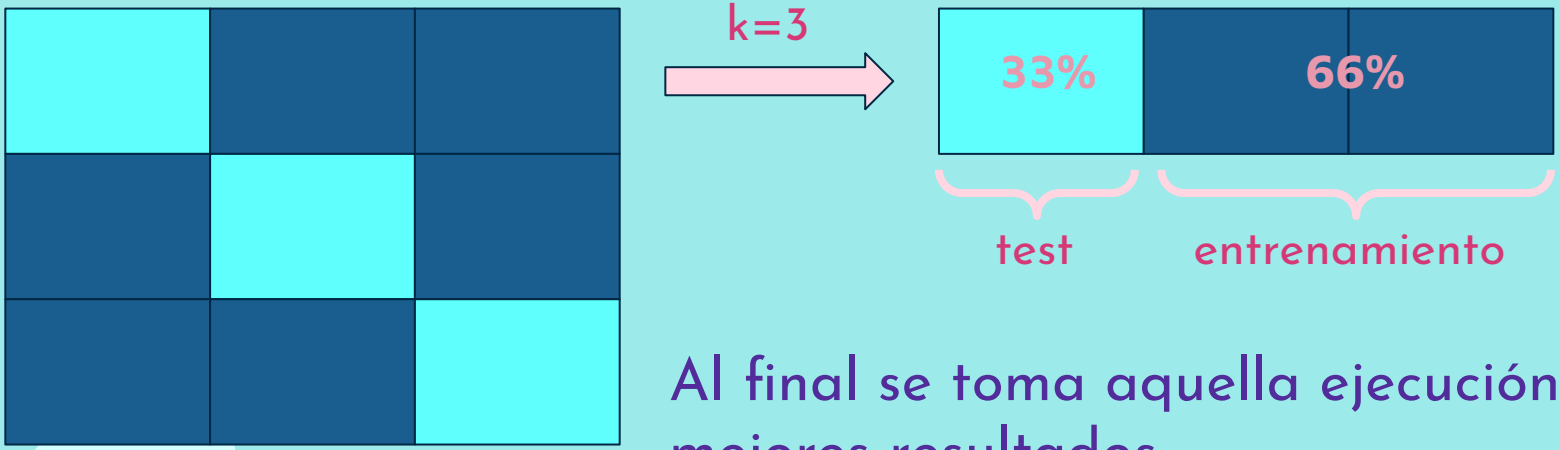


Error



Validación cruzada
Definición

Se parte el dataset en K partes, se ejecuta K veces el entrenamiento variando el dataset de entrenamiento y el de testeo.

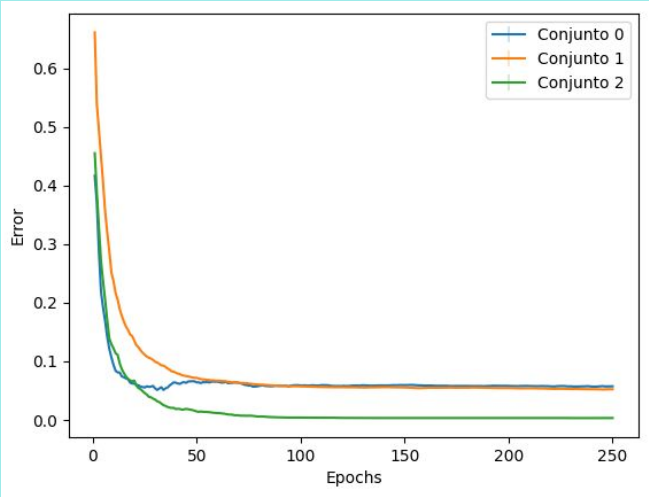


Al final se toma aquella ejecución con mejores resultados.

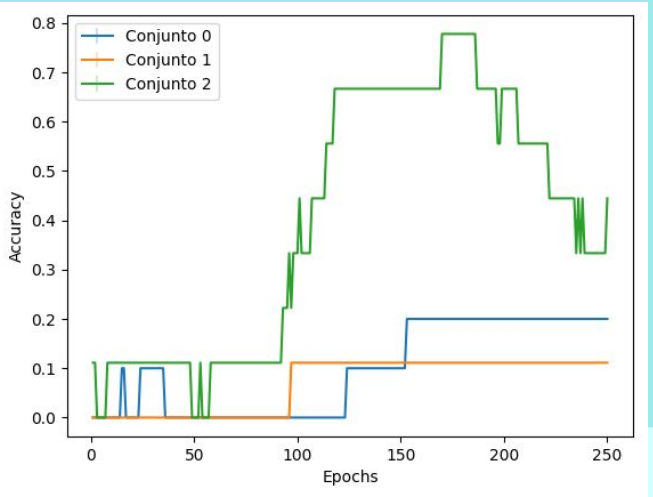
Validación cruzada
Resultados

Parámetros:
K = 5
n = 0.05
Epochs = 250
Beta = 1
F = EXP
Training online
Prediction threshold = 0.01

Train accuracy



Test accuracy

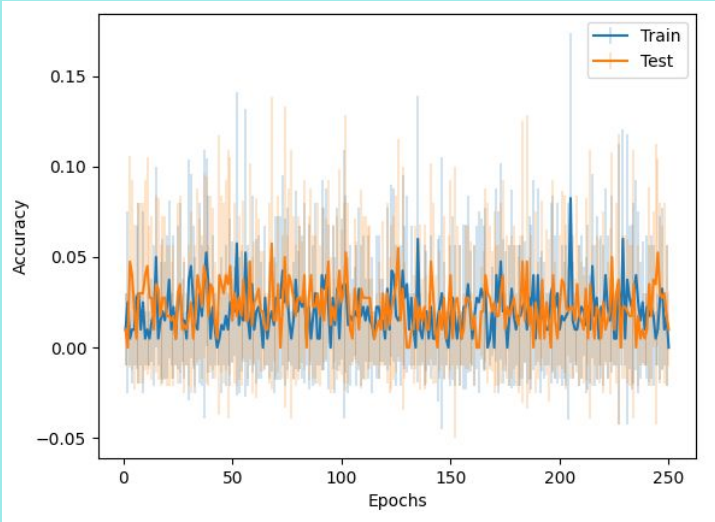


Vemos que tomar el conjunto 2 como testeo fue lo que dio la mejor performance.

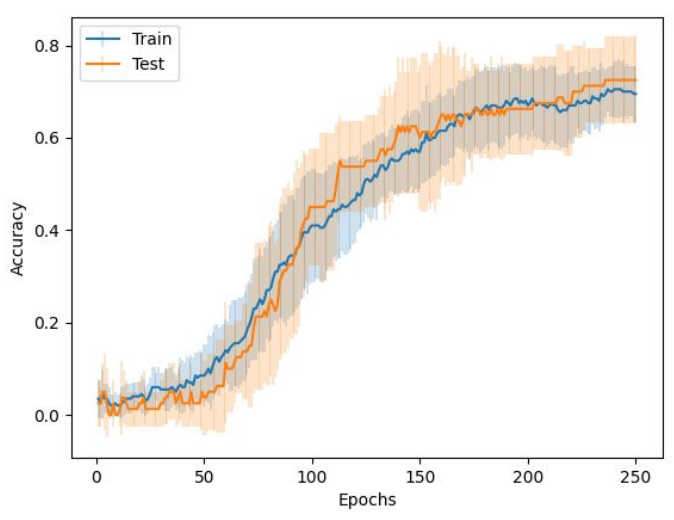
Perceptrón lineal vs no lineal
Comparación de resultados

Parámetros:
n = 0.05
Epochs = 250
Beta = 1
F = EXP
Training online
Train data
Prediction threshold = 0.01
Porcentaje de entrenamiento: 70%

Lineal



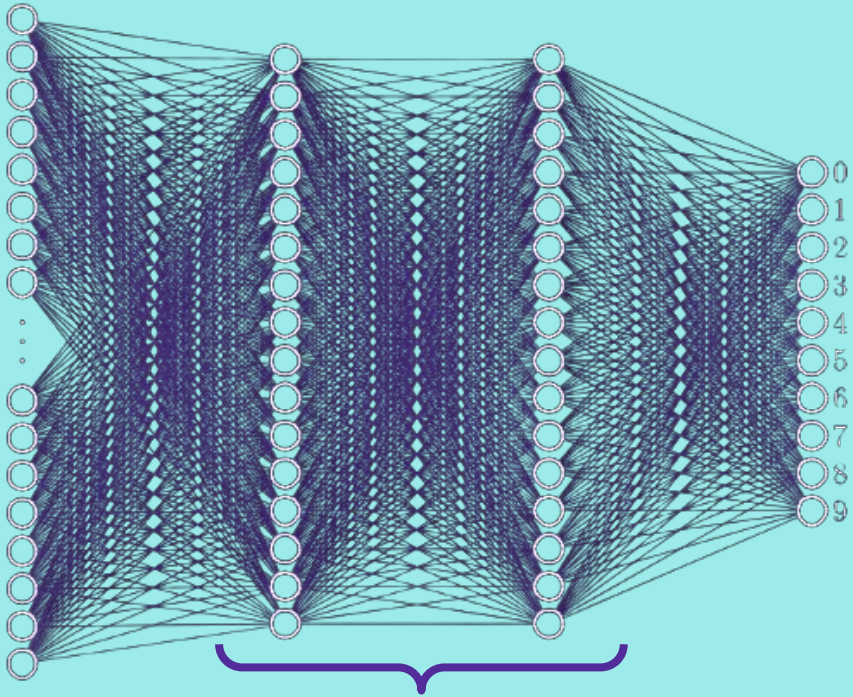
No Lineal



02

Perceptrón Multicapa

Perceptrón Multicapa
Arquitectura



Capa de entrada Capas ocultas Capa de salida

Ejercicio 3.1

"O exclusivo"

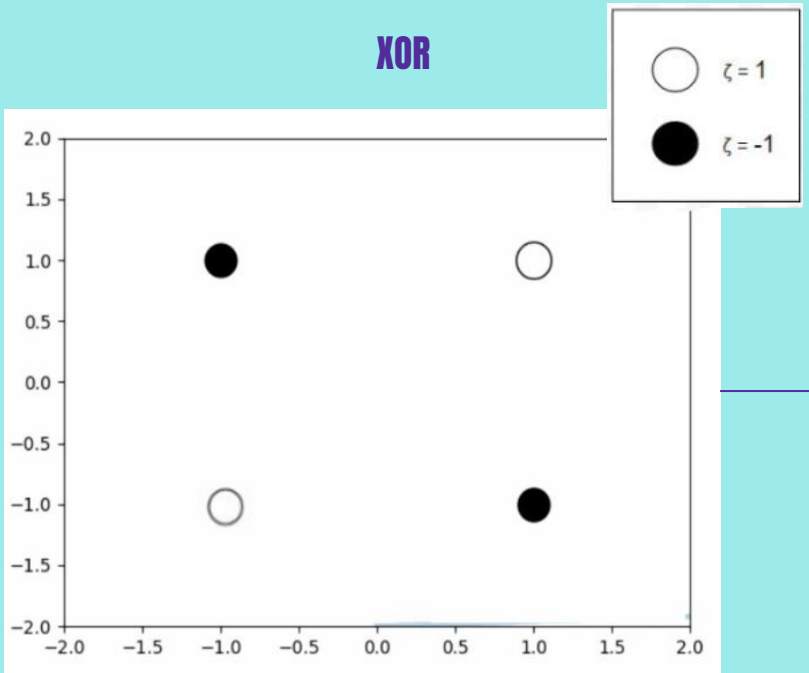
Ejercicio 3.1
Datos de entrada

Funcion logica "O exclusivo" (XOR)

A	B	Esperado
-1	1	1
1	-1	1
1	1	-1
-1	-1	-1

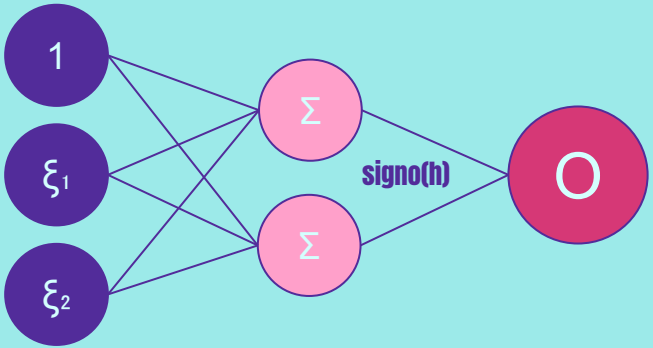
Problema con Perceptrón Simple

XOR

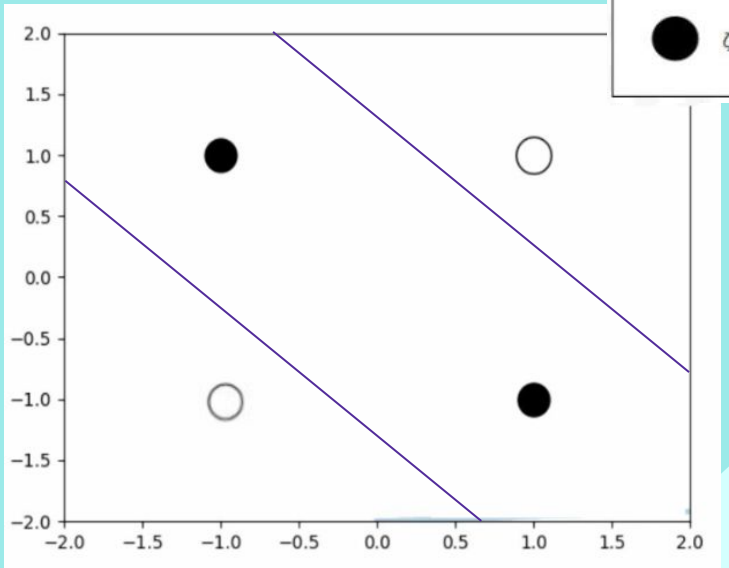


No es linealmente separable
(una única recta no sirve para clasificar los puntos)

Composición de funciones

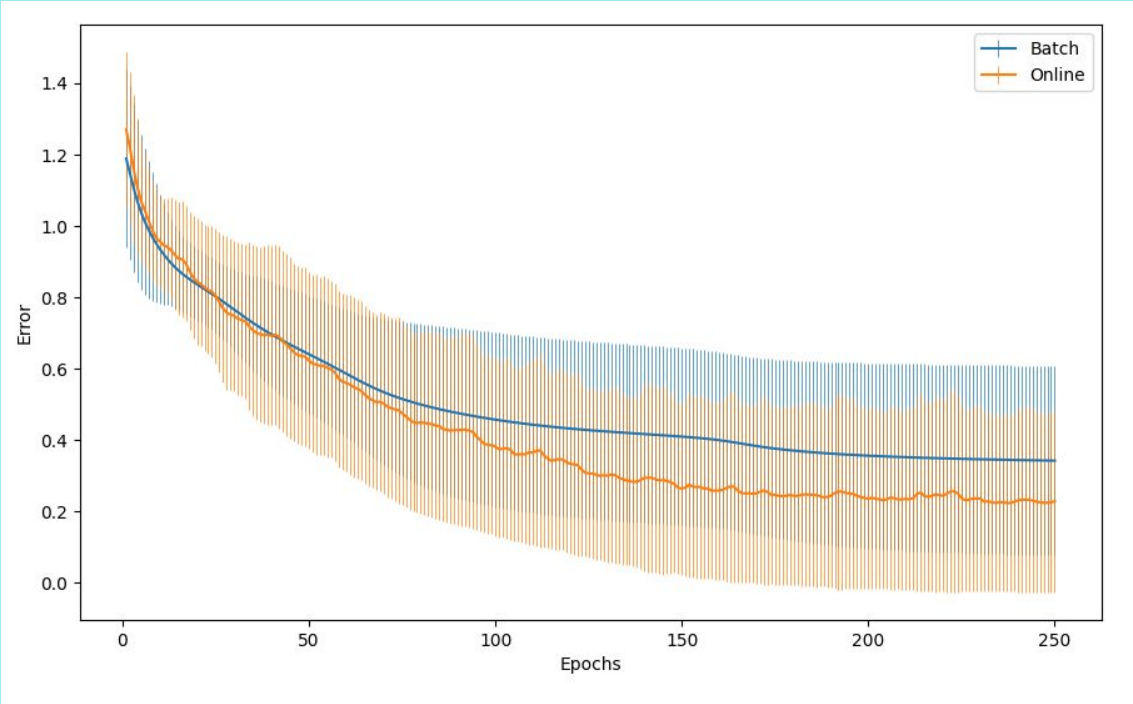


XOR



Perceptron multicapa
Resultados Batch vs Online

Parámetros:
n = 0.05
Tamaño = [2, 1]
Epochs = 1000
Beta = 1
F = SIGN
Training online
Prediction
threshold = 0.1
Sin division entre test y train



Ejercicio 3.2 y 3.3
Definición

- Valores de Entrada:
  - Dataset de 10 matrices de 5x7, representando un número.
  - Valores a evaluar.
- Valores de Salida:
  - 3.2 → Vector con 2 valores → [0,1] par, [1,0] impar
  - 3.3 → Vector de 10 valores con las probabilidades de cada uno → se toma el máximo
- Aplicación de ruido
  - Se itera todo el dataset aplicando de forma aleatoria entre 0 y 6 cambios en posiciones también aleatorias.
  - Se testea este nuevo dataset sobre una red entrenada con el dataset original.

0	0	1	0	0
0	1	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	1	1	1	0

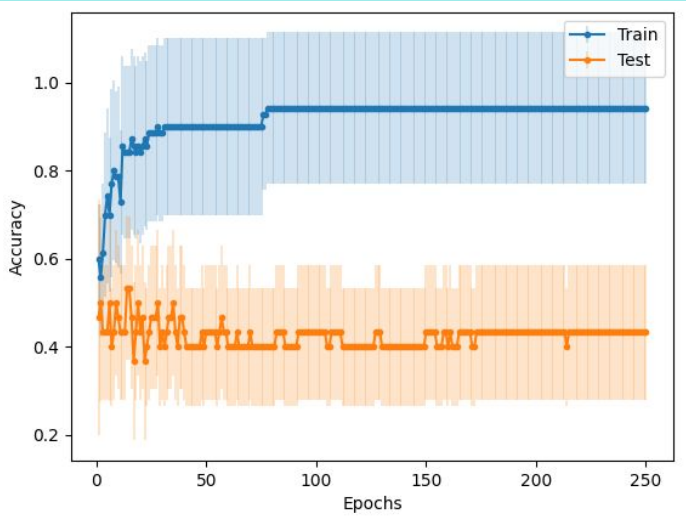
Ejercicio 3.2

Discriminar números pares

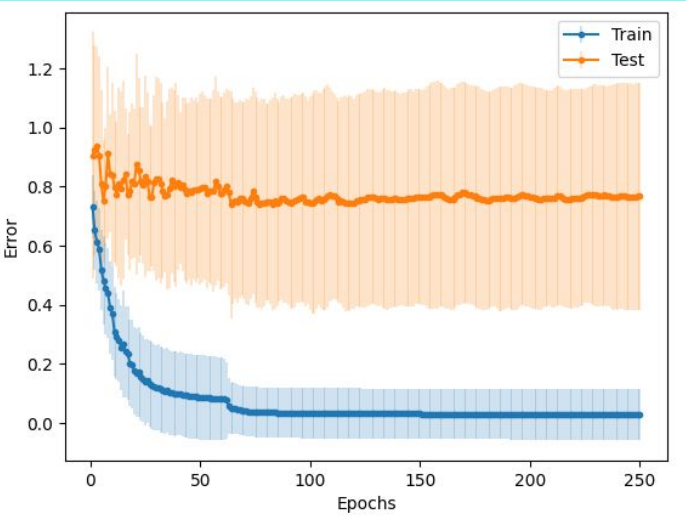
Perceptrón Multicapa
Resultados paridad

Parámetros:
n = 0.05
Tamaño = [5*7, 5*7, 2]
Epochs = 250
Beta = 1
F = EXP
Training online
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01
Momentum = 0.8

Accuracy



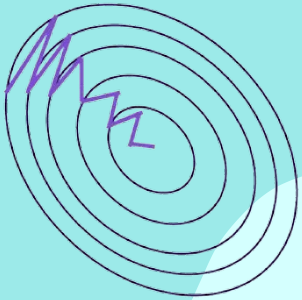
Error



Perceptrón Multicapa
Optimización: Momentum

- Optimización sobre método Estocástico Gradiente Descendente.
- Trata de solucionar casos de oscilación o divergencia en torno a un mínimo.
- Ayuda a acelerar las gradientes de los vectores en las direcciones correspondientes.
- Provoca mayor velocidad de convergencia.
- Utiliza hiper-parámetro **β** entre 0 y 1.

Stochastic Gradient Descent **withhout** Momentum

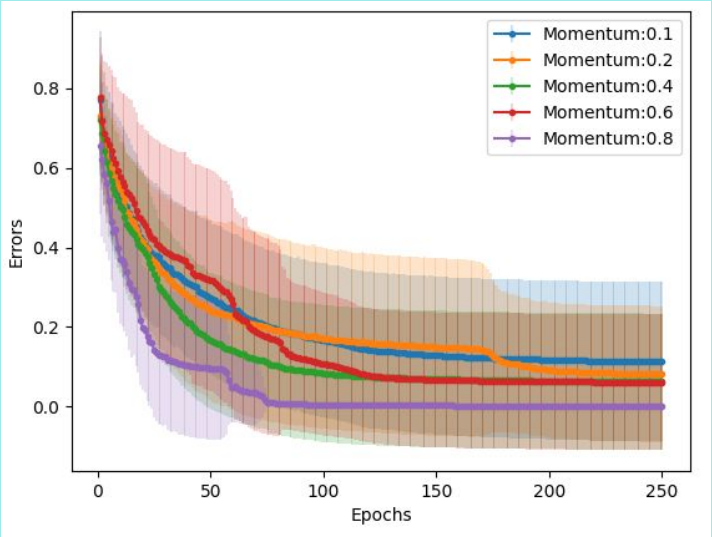


Stochastic Gradient Descent **with** Momentum

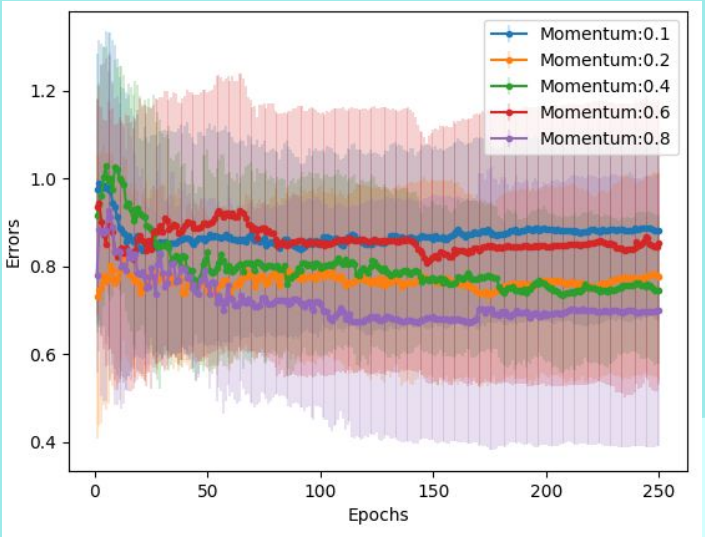
Perceptrón Multicapa
Resultados paridad
Analisis de momentum

Parámetros:
n = 0.05
Tamaño = [5*7, 5*7, 2]
Epochs = 250
Beta = 1
F = EXP
Training online
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01

Train error



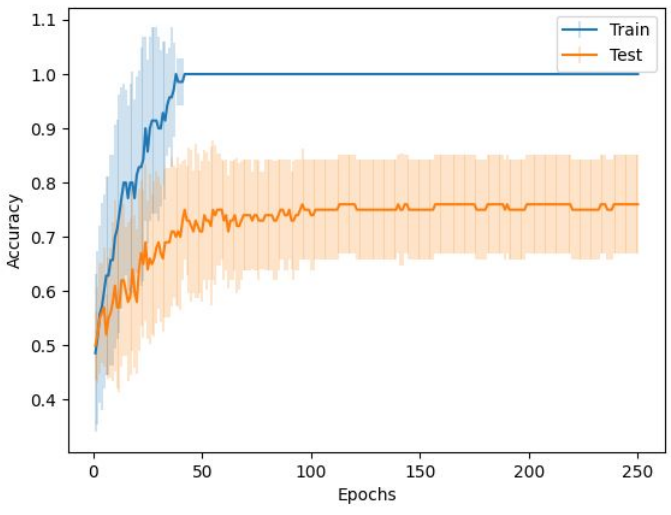
Test error



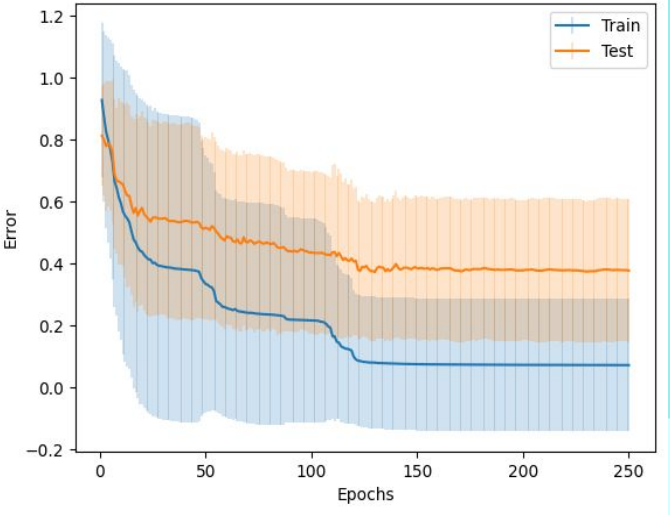
Perceptrón Multicapa
Resultados paridad (con ruido)

Parámetros:
n = 0.05
Tamaño = [5*7, 5*7, 2]
Epochs = 250
Beta = 1
F = EXP
Training online
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01
Momentum = 0.8

Accuracy



Error



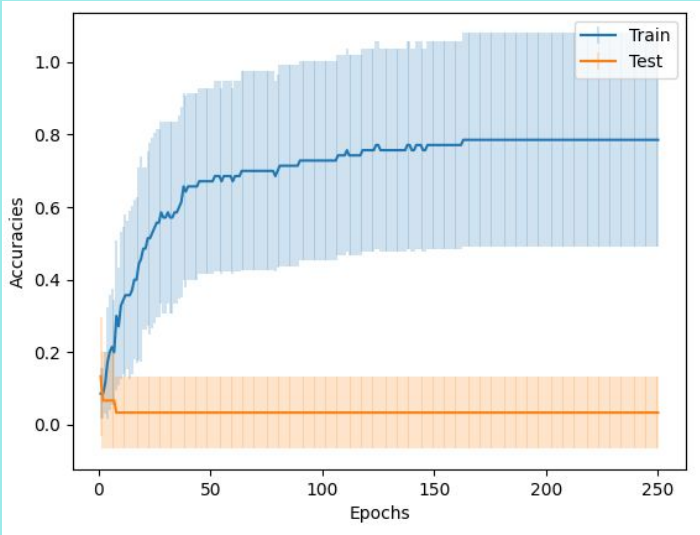
Ejercicio 3.3

Predecir digitos

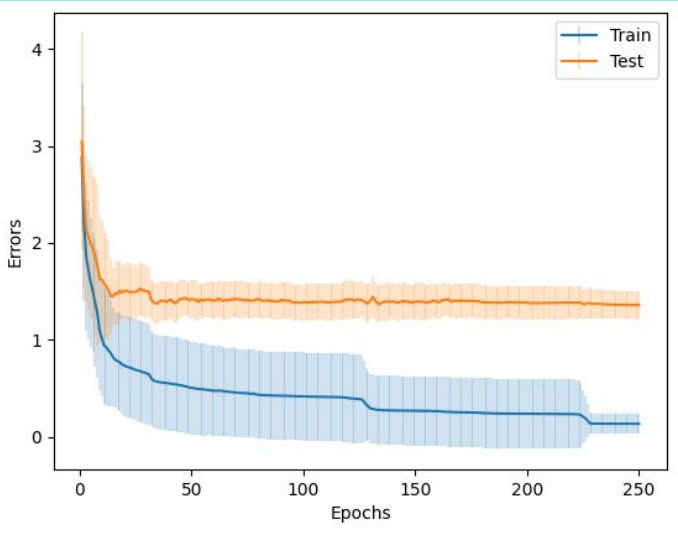
Perceptrón Multicapa
Resultados adivinar número

Parámetros:
n = 0.05
Tamaño = [5*7, 5*7, 10]
Epochs = 250
Beta = 1
F = EXP
Training online
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01
Momentum = 0.8

Accuracy



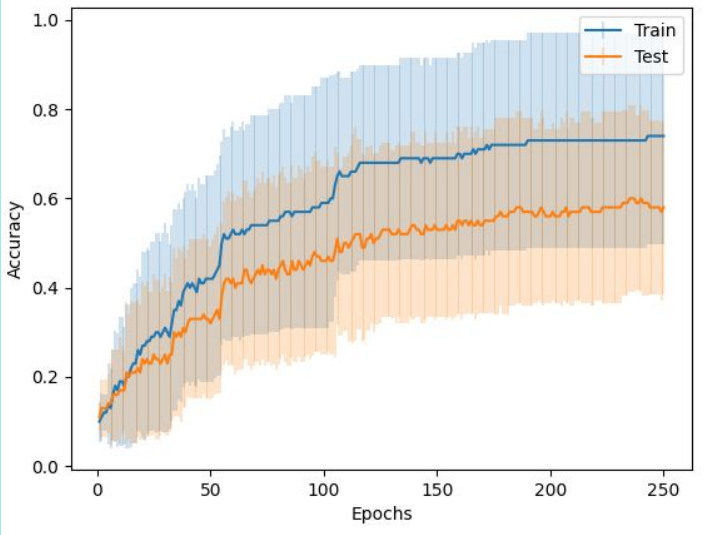
Error



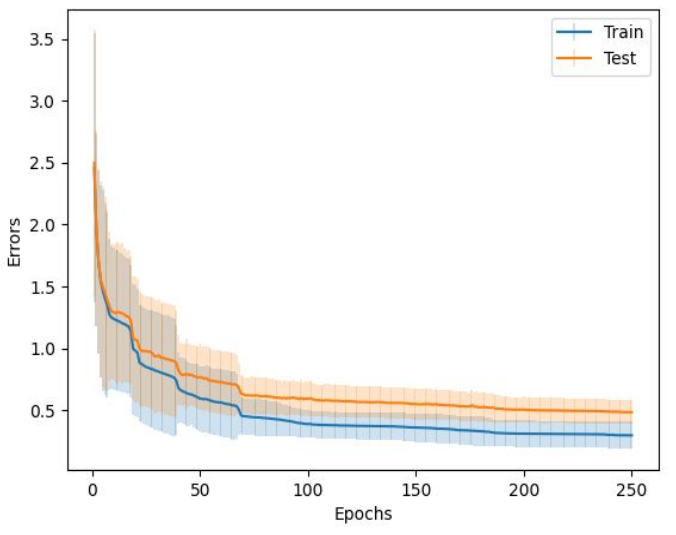
Perceptrón Multicapa
Resultados adivinar número (con ruido)

Parámetros:
n = 0.05
Tamaño = [5*7, 5*7, 10]
Epochs = 250
Beta = 1
F = EXP
Training online
Porcentaje de entrenamiento: 70%
Prediction threshold = 0.01
Momentum = 0.8

Accuracy



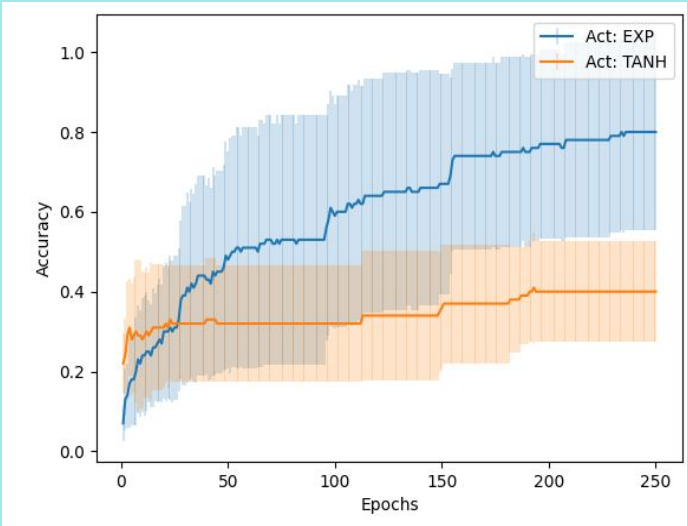
Error



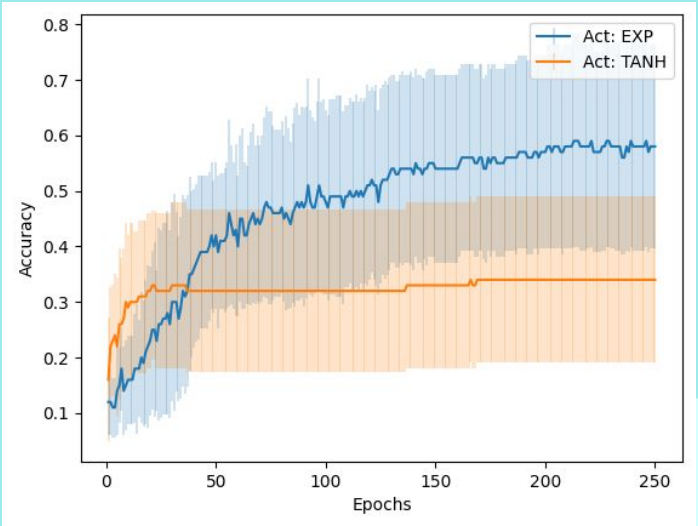
Perceptrón Multicapa
Resultados adivinar número (con ruido) variando F

Parámetros:
n = 0.05
Tamaño = [5*7, 5*7, 10]
Epochs = 250
Beta = 1
Training online
Porcentaje de entrenamiento: 100%
Prediction threshold = 0.01
Momentum = 0.8

Train accuracy



Test accuracy



Ahora...
veámoslo en acción!

Conclusiones

- Existen problemas que el perceptrón simple no puede resolver, por el otro lado una red multicapa es capaz de resolver muchísimos más y con mejor precisión, pero a mayor costo de computo.
- La elección del beta depende de la función de activación y del dataset.
- Utilizar una función lineal de activación en el multicapa no tiene sentido, pues el resultado es un perceptrón ya que es una combinación lineal
- La optimización de momentum permite que la convergencia hacia los mejores resultados sea mucho más posible, mejorando el funcionamiento de la red.
- El buen funcionamiento del modelo depende de la calidad del entrenamiento. El modelo es susceptible a problemas de overfitting o underfitting que no lo generalize correctamente. Por esto, se debe buscar un dataset lo más grande y realista posible.

¡GRACIAS!
¿PREGUNTAS?