

TP 3: Perceptrones Simples y Multicapas

- Britos, Nicolás Ignacio - 59.529
- Griggio, Juan Gabriel - 59.092
- Roca, Agustín - 59.160

Introducción

Objetivos

- Comprender e implementar diferentes algoritmos de Perceptrón:
 - Simple
 - Lineal
 - No lineal
 - Multicapa
- Analizar los resultados



Desarrollo del trabajo

Tecnología utilizada

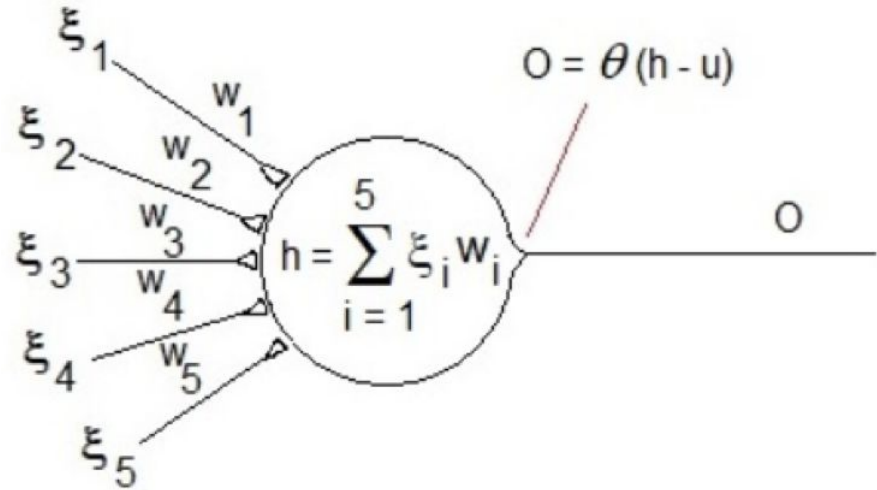


PYTHON



Modelo de neuronas

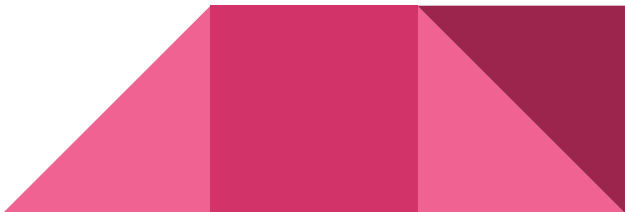
- ξ_1, \dots, ξ_5 : entradas
- w_1, \dots, w_5 : pesos
- θ : función de activación
- O : salida
- u : umbral



Aprendizaje

$$w_i^{nuevo} = w_i^{viejo} + \Delta w_i$$

$$\Delta w_i = \begin{cases} 2\eta \xi_i^\mu \zeta^\mu & \text{si } O^\mu \neq \zeta^\mu \\ 0 & \text{en otro caso} \end{cases}$$

$$\Delta w_i = \eta (\zeta^\mu - O^\mu) \xi_i^\mu$$


Perceptrón Simple

Algoritmo

```
i = 0
w = zeros(N+1, 1)
error = 1
error_min = p * 2
while error > 0 ∧ i < COTA
    Tomar un número  $i_x$  al azar entre 1 y p
    Calcular la excitación  $h = x[i_x].w$ 
    Calcular la activación  $O = \text{signo}(h)$ 
     $\Delta w = \eta * (y[i_x] - O).x[i_x]$ 
     $w = w + \Delta w$ 
    error = CalcularError(x, y, w, p)
    if error < error_min
        error_min = error
        w_min = w
    end
    i = i + 1
end
```

Problema a resolver

Implementar Perceptrón Simple con función de activación Escalón y que resuelva los siguientes problemas:

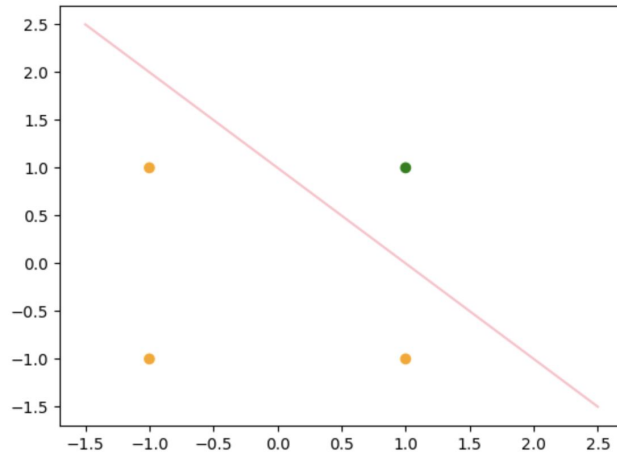
- Entrada: $X = \{ \{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\} \}$
- Salida AND: $Y = \{-1, -1, -1, 1\}$

- Entrada: $X = \{ \{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\} \}$
- Salida XOR: $Y = \{1, 1, -1, -1\}$



Resultado

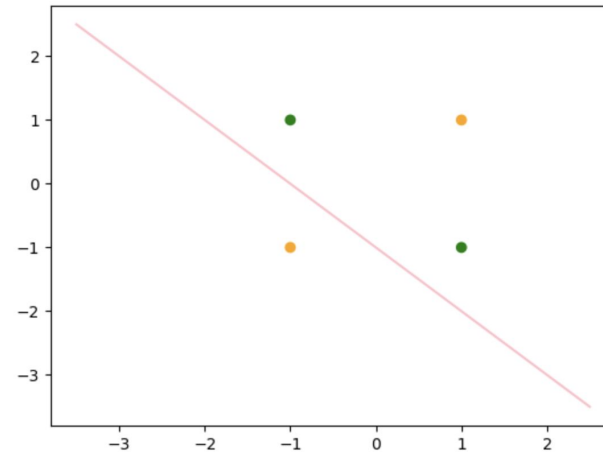
AND: Linealmente Separable



$\mu=0.001 \Rightarrow \text{err}=0$ en 20 épocas

$\mu=0.01 \Rightarrow \text{err}=0$ en 12 épocas

XOR: No Linealmente Separable



No llega a $\text{err}=0$

Conclusión del Perceptrón Simple con función de activación Escalón

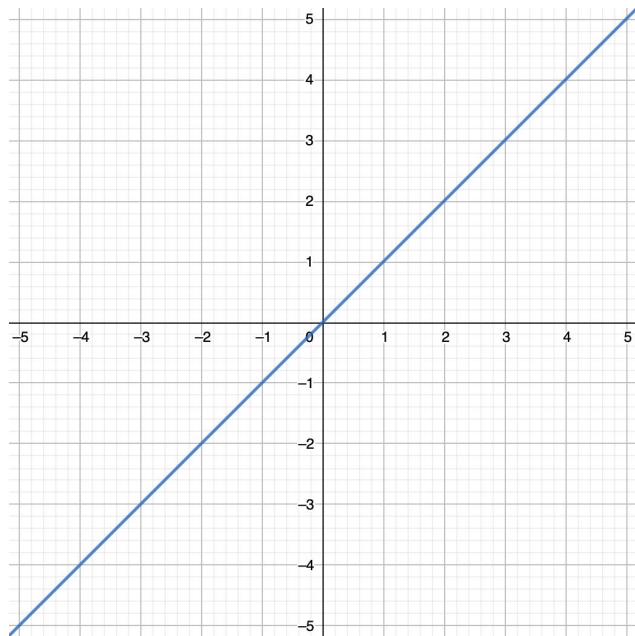
- Resuelve los problemas linealmente separables
- Tarda más o menos épocas según el factor de aprendizaje



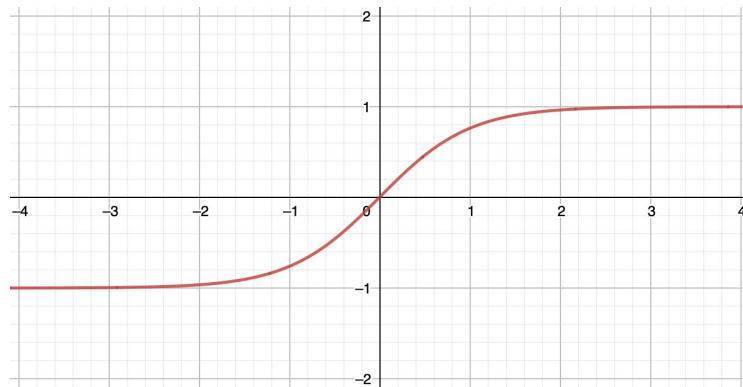
Perceptrón Simple Lineal y No Lineal

Perceptrón Simple Lineal vs No Lineal

Lineal: $y = x$



No lineal: $y = \tanh(x)$



Perceptrón Simple Lineal

- Resuelve los problemas no linealmente separables que el Perceptrón Simple con función de activación Escalón no resolvía
- La salida pertenece a los reales
- La convergencia depende fuertemente de η
- Ajuste de peso: método del gradiente descendiente

$$\begin{aligned}\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = \\ &= \eta \sum_{\mu=1}^p (\zeta^\mu - O^\mu) \xi_i^\mu\end{aligned}$$



Perceptrón Simple No Lineal

- Generalización del Perceptrón Simple Lineal
- Ajuste de peso: método del gradiente descendiente, aparece la derivada de la función de activación por regla de la cadena

$$\Delta w_i = \eta \sum_{\mu=1}^p (\zeta^\mu - g(h^\mu)) g'(h^\mu) \xi_i^\mu$$

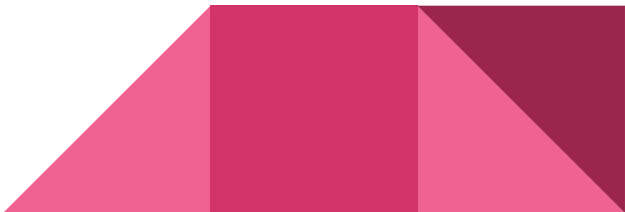
$$h^\mu = \sum_{i=0}^N w_i \xi_i^\mu$$

Si $g(h) = \tanh(\beta h)$ entonces $g'(h) = \beta(1 - g^2(h))$

Si $g(h) = \frac{1}{1 + \exp^{-2\beta h}}$ entonces $g'(h) = 2\beta g(h)(1 - g(h))$



Problema a resolver

- Implementar los algoritmos de Perceptrón Simple Lineal y Perceptrón Simple No Lineal
 - Dado un dataset, evaluar la capacidad de aprender la función de las muestras y generalización
 - ¿Cómo podría escoger el mejor conjunto de entrenamiento?
 - ¿Cómo podría evaluar la máxima capacidad de generalización del Perceptrón para este conjunto de datos?
- 

Resultado del Perceptrón Simple Lineal

% de población para entrenamiento	Error promedio de entrenamiento	Precisión promedio de entrenamiento	Precisión promedio del test	Épocas
80	1796	0.362	0.2999	500
60	4387	0.349	0.2811	500
40	6301	0.327	0.2703	500
20	9002	0.311	0.2501	500



Resultado del Perceptrón Simple No Lineal

% de población para entrenamiento	Error promedio de entrenamiento	Precisión promedio de entrenamiento	Precisión promedio del test	Épocas
80	0.32	1	1	500
60	0.699	1	1	500
40	0.892	1	1	500
20	1.463	1	1	500


Respondiendo las preguntas...

- **¿Cómo podría escoger el mejor conjunto de entrenamiento?**

Mayor porcentaje de la población para entrenamiento => menor error y mayor precisión

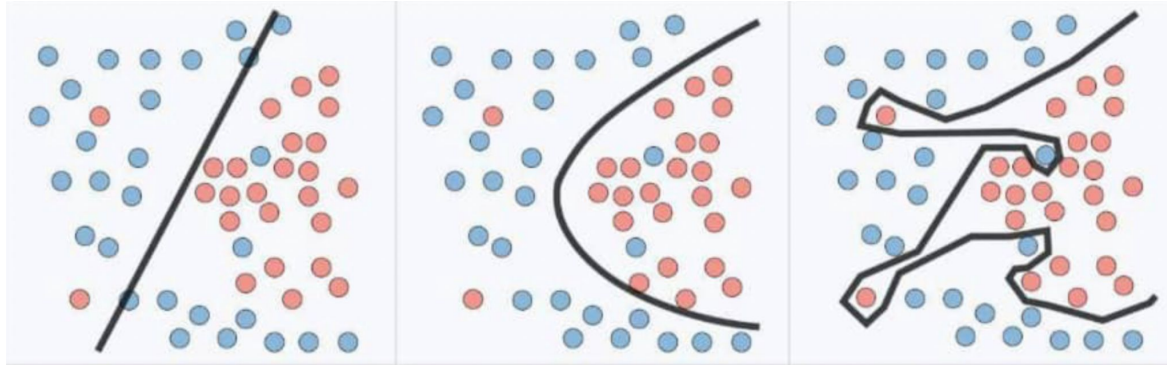
- **¿Cómo podría evaluar la máxima capacidad de generalización del Perceptrón para este conjunto de datos?**

Encontrando una relación óptima entre % de entrenamiento y de prueba

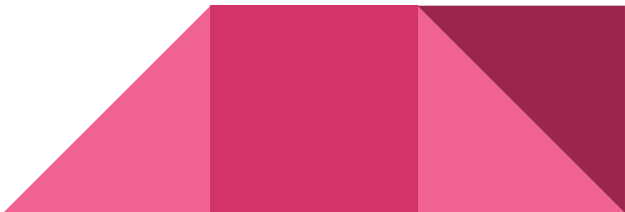


¡Ojo! Overfitting

Underfitting - Perfecto - Overfitting

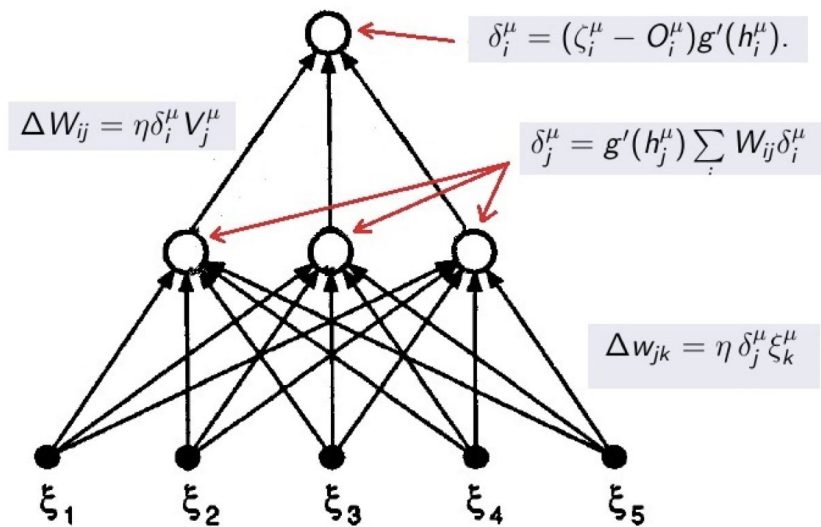


Conclusión del Perceptrón Simple Lineal y No Lineal

- A mayor cantidad de ejemplares para entrenamiento, mayor precisión y menor error
 - El entrenamiento tiene que ser equilibrado: ni overfitting ni underfitting
 - El no lineal nos da un mejor resultado si el conjunto de datos no es linealmente separable
- 

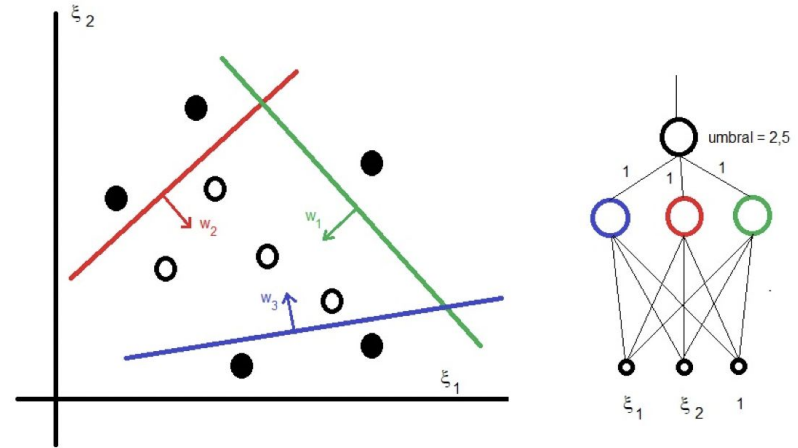
Perceptrón Multicapa

Perceptrón Multicapa



Perceptrón Multicapa

- Conjuntos de datos No Linealmente Separables
- Varios perceptrones simples nos generan varios hiperplanos



Error

$$E(w) = \frac{1}{2} \sum_{\mu, i} (\zeta_i^\mu - O_i^\mu)^2,$$

$$E(w) = \frac{1}{2} \sum_{\mu, i} (\zeta_i^\mu - g(\sum_j W_{ij} g(\sum_k w_{jk} \zeta_k^\mu)))^2.$$

Retropropagación del error

- El error de cada unidad de la capa de salida se propaga hacia las capas inferiores
- A partir de esto, se actualizan los pesos de las conexiones
- La actualización de pesos entre capas depende de los errores de las capas superiores

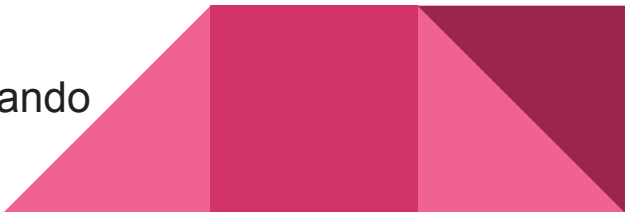


Aprendizaje incremental y en lotes

Incremental:

- Entrada presentada al azar
- Propagar hasta obtener activación de la salida
- Retropropagar el error
- Actualizar los pesos

Por lotes:

- Entrada presentada sin importar el orden
 - Propagar hasta obtener activación de la salida
 - Retropropagar el error
 - Se acumula el Δ para cada conexión
 - Al presentar todas las entradas, se actualizan los pesos usando los acumulados
- 

Algoritmo

1. Inicializar el conjunto de pesos en valores 'pequeños' al azar.
2. Tomar un ejemplo ξ^μ al azar del conjunto de entrenamiento y aplicarlo a la capa 0:
 $V_k^0 = \xi_k^\mu$ para todo k .
3. Propagar la entrada hasta a capa de salida
 $V_i^m = g(h_i^m) = g(\sum_j w_{ij}^m V_j^{m-1})$ para todo m desde 1 hasta M .
4. Calcular δ para la capa de salida
 $\delta_i^M = g'(h_i^M)(\zeta_i^\mu - V_i^M)$
5. Retropropagar δ_i^M
 $\delta_i^{m-1} = g'(h_i^{m-1}) \sum_j w_{ji}^m \delta_j^m$ para todo m entre M y 2
6. Actualizar los pesos de las conexiones de acuerdo
 $w_{ij}^{nuevo} = w_{ij}^{viejo} + \Delta w_{ij}^m$
donde $\Delta w_{ij}^m = \eta \delta_i^m V_j^{m-1}$
7. Calcular el *error*. Si *error* > *COTA*, ir a 2.

Problema a resolver

Implementar Perceptrón Multicapa y que resuelva el siguiente problema:

- Entrada: $X = \{ \{-1, 1\}, \{1, -1\}, \{-1, -1\}, \{1, 1\} \}$
- Salida XOR: $Y = \{1, 1, -1, -1\}$



Resultado

- Factor de Aprendizaje = 0.01
- Capas ocultas: 2 capas, una con 3 neuronas y otra con 2 neuronas
- Máximo de épocas = 5000

Salida esperada	Salida del Perceptrón	Error
1	0.939761	0.060239
1	0.942076	0.057924
-1	-0.920012	0.079988
-1	-0.919841	0.080159



Problema a resolver

Implementar Perceptrón Multicapa que determine si un número es par o impar

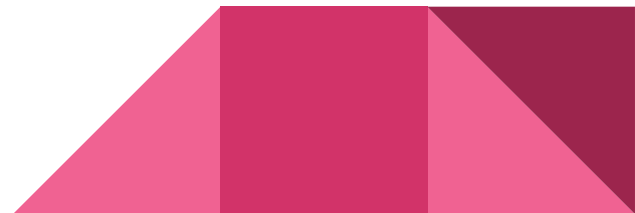
- Entrada: imágenes de 5x7 píxeles que representan números del 0 al 9
- Salida: par o impar
- ¿Qué podría decir acerca de la capacidad para generalizar de la red?



Resultado 1

- Capas: 5 nodos, 4 nodos y 1 nodo de salida
- Perceptrones Simples No Lineales
- Tamaño del set de entrenamiento: 3
- 500 épocas
- Factor de aprendizaje: 0.01
- Error: 0.005

	Precisión	Error
Entrenamiento	0.33	4
Prueba	0.66	2



Resultado 2

- Capas: 5 nodos, 4 nodos y 1 nodo de salida
- Perceptrones Simples No Lineales
- Tamaño del set de entrenamiento: 3
- 100 épocas
- Factor de aprendizaje: 0.01
- Error: 0.005

	Precisión	Error
Entrenamiento	1	0
Prueba	0.33	4



Problema a resolver

Construir un Perceptrón Multicapa con la misma entrada usada en el ejercicio anterior pero con 10 unidades de salidas de modo que cada salida representa a un dígito.

Una vez que la red haya aprendido, usar patrones de entrada correspondientes a los dígitos de entrenamiento pero con sus píxeles afectados por ruido y evaluar los resultados.



Resultados

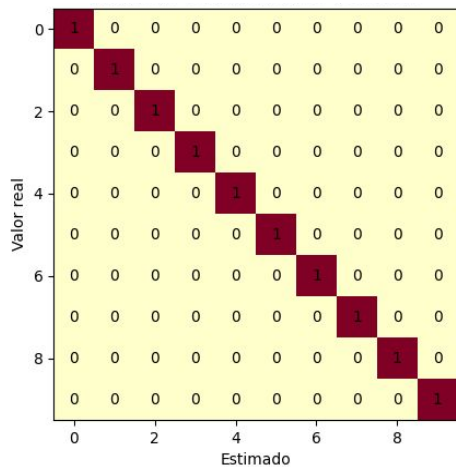
Todos los resultados poseen la estructura:

- Capas: 7 de entrada, 7, 10, 10 salida
- Perceptrones Multicapa
- Funcion de activacion tanh
- Factor de aprendizaje: 0.01
- Error: 0.005

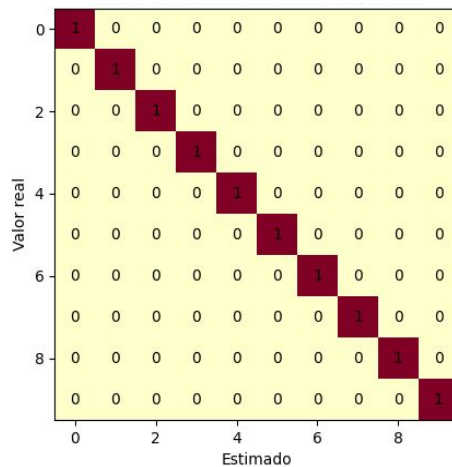


Resultado 1

Entrenamiento



Prueba

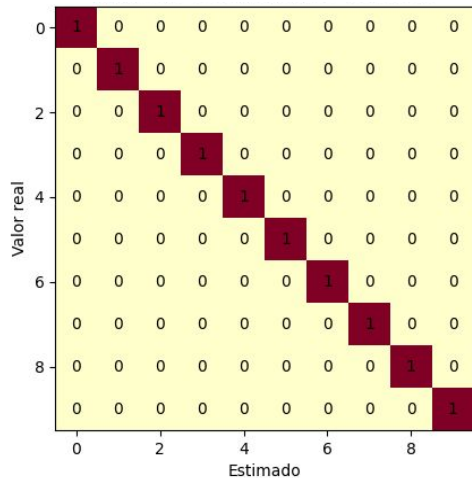


- Épocas: 500
- Ruido = 2%

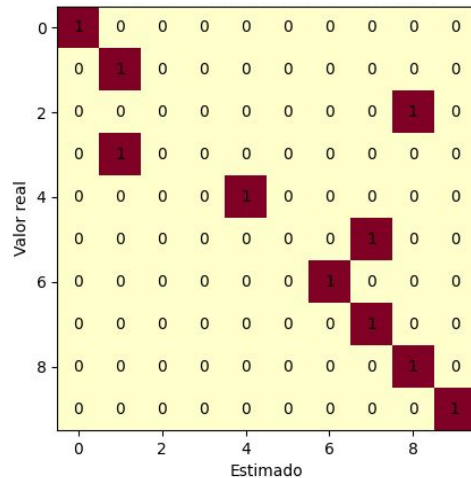


Resultado 2

Entrenamiento



Prueba



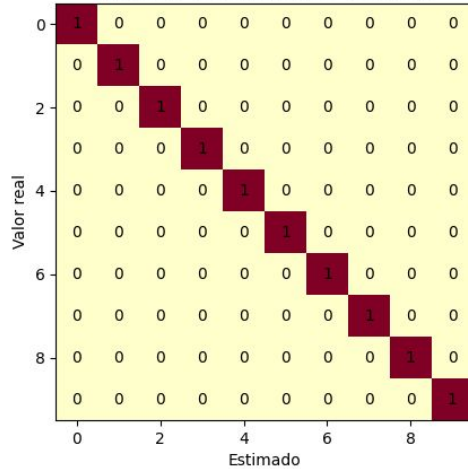
- Épocas: 500
- Ruido: 5%

0-16475-57-019

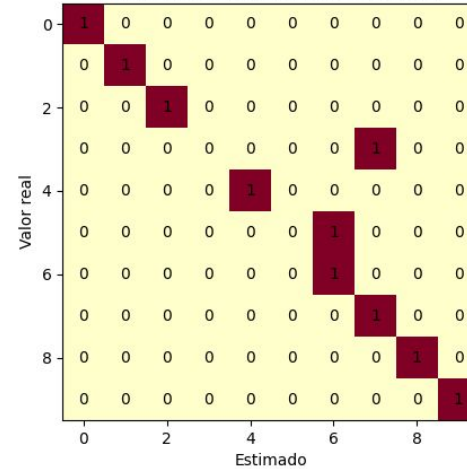
0-16475-58-000

Resultado 3

Entrenamiento



Prueba



- Capas: 10 de entrada, 10, 10 salida
- Épocas: 2000
- Ruido: 15%

0 1 2 3 4 5 6 7 8 9

Conclusiones

Conclusiones

- Los Perceptrones Simples sólo pueden clasificar problemas linealmente independientes.
- Los Perceptrones Multicapa pueden clasificar datos más complejos y en más categorías, pero la velocidad de procesamiento aumenta de manera casi exponencial.
- Más capas y/o más neuronas no necesariamente implican una mejor resolución del problema.
- El tipo de Perceptrón a utilizar dependen fuertemente del tipo de problema a resolver.





¡Muchas gracias!