Perceptron simple y multicapa

Felipe Oliver (58439)
Juan Bensadon (57193)

Sistemas de Inteligencia Artificial: TP3

Perceptrón simple lineal

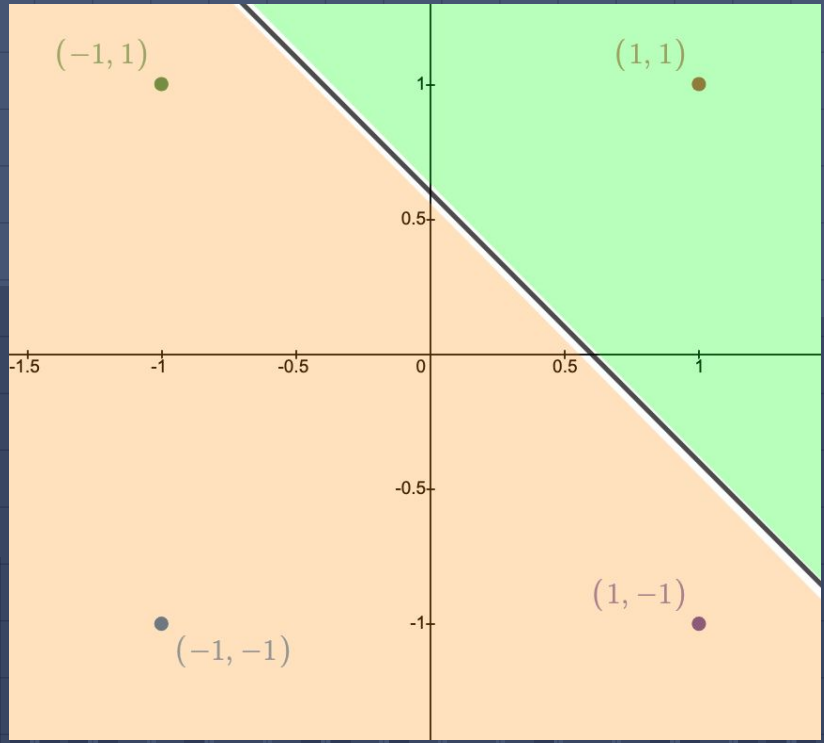
1

Función 1: AND

Entradas separables (clasificables) por una recta de acuerdo a su salida esperada.

Resultado:

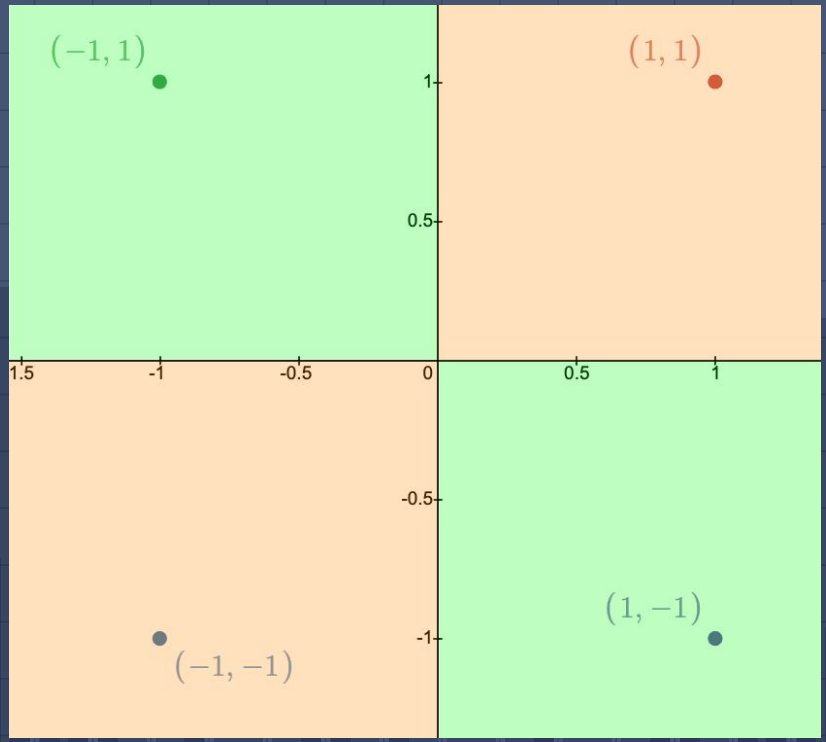
```
=== AND FUNCTION ===
X=[[-1  1]
 [ 1 -1]
 [-1 -1]
 [ 1  1]]
Y=[-1 -1 -1  1]
Training...
Resulting weights: [0.02 0.02]
Perceptron tests (all should be true):
True
True
True
True
```



Función 2: XOR

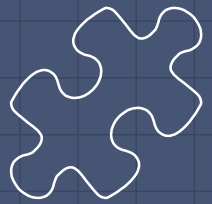
Las entradas no son separables. La modificación de los pesos en base a algunas entradas hace que el perceptrón inevitablemente falle al predecir el resultado de otras:

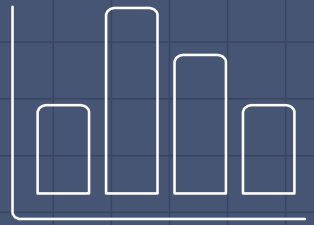
```
=== XOR FUNCTION ===
X=[[-1  1]
 [ 1 -1]
 [-1 -1]
 [ 1  1]]
Y=[ 1  1 -1 -1]
Training...
Resulting weights: [0. 0.]
Perceptron tests (all should be true):
False
False
True
True
```



Conclusión 1

Es importante analizar el problema y los datos antes de utilizarlos, para saber exactamente qué tipo de red utilizar, y si tenemos una red de tipo adecuado.





Perceptrón simple no lineal

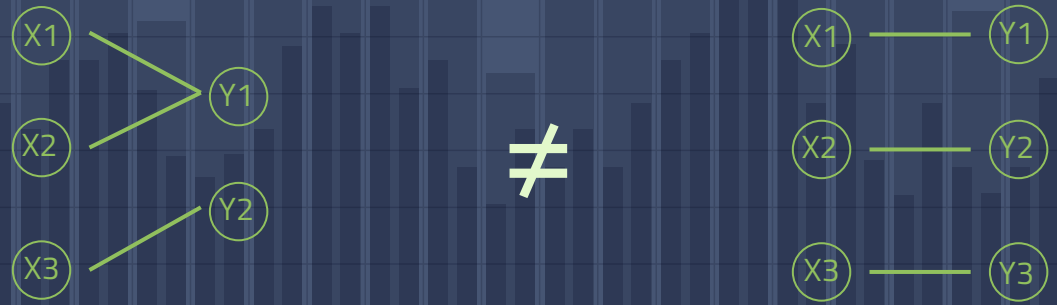
2

¿Perceptrón lineal?

-La salida de la función provista son números de punto flotante.
-Nuestro perceptrón simple utiliza la función escalón como función de activación clasificando entradas en dos categorías

⇩

No podemos utilizarlo para aprender la función provista (sus resultados son un valor específico correspondiente al de la entrada, no una clasificación binaria)



Perceptrón no lineal

-Podemos obtener un valor correspondiente a cada entrada que aproxime el resultado de la función original.

-Usamos la función de activación Logística (los valores del archivo están comprendidos entre 0 y 100).

-Normalizamos primero los valores para verse comprendidos entre 0 y 1.

Generalización

1) Todo el set de datos

-Threshold: 200

-Learning rate: 0,01

-Función de costo: $\sum (pred(x) - y)^2/n$ dio un valor de 0,0055

Generalización

2) Mitad el set de datos (100 elementos training, 100 validation)

-Threshold: 200

-Learning rate: 0,01

-Función de costo sobre training set: $\sum (pred(x) - y)^2/n$ dio un valor de 0.0044

-Función de costo sobre validation set: $\sum (pred(x) - y)^2/n$ dio un valor de 0,0061

Buscando el training set óptimo

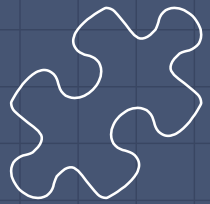
```python
def find_optimal_training_group(x, y):
  current_min_error = 1000
  min_i, min_j = 0, 0
  for i in range(len(y)):
    for j in range(i, len(y)):
      train_x = np.array(x[i:j+1])
      train_y = np.array(y[i:j+1])

      validate_x = x[:i]
      validate_x.extend(x[j+1:])
      validate_x = np.array(validate_x)

      validate_y = y[:i]
      validate_y.extend(y[j+1:])
      validate_y = np.array(validate_y)

      train_y = normalize_arr(train_y)
      validate_y = normalize_arr(validate_y)
      perceptron = NonLinearPerceptron(no_of_inputs=3, threshold=200)
      perceptron.train(train_x, train_y)
      error = perceptron.cost_function(validate_x, validate_y)
      # print(error + " " + str(i) + " - " + str(j))
      print(error)
      print(i)
      print(j)
      if error < current_min_error:
        current_min_error = error
        min_i, min_j = i, j
  print("Minimum error was " + str(current_min_error) + " for training set indexes " + str(min_i) + ", " + str(min_j))
```

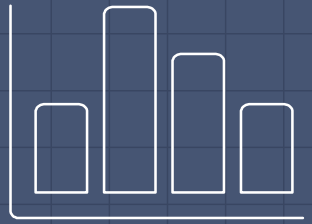
```
Minimum error was 0.0019421857498223094 for training set indexes 16, 199
```

Conclusión 2

Este tipo de red tiene una buena capacidad para generalizar, y esto nos permite, eligiendo un buen set de entrenamiento, predecir con precisión el resultado de la función sobre otras entradas.

Un training set más amplio no necesariamente será mejor.



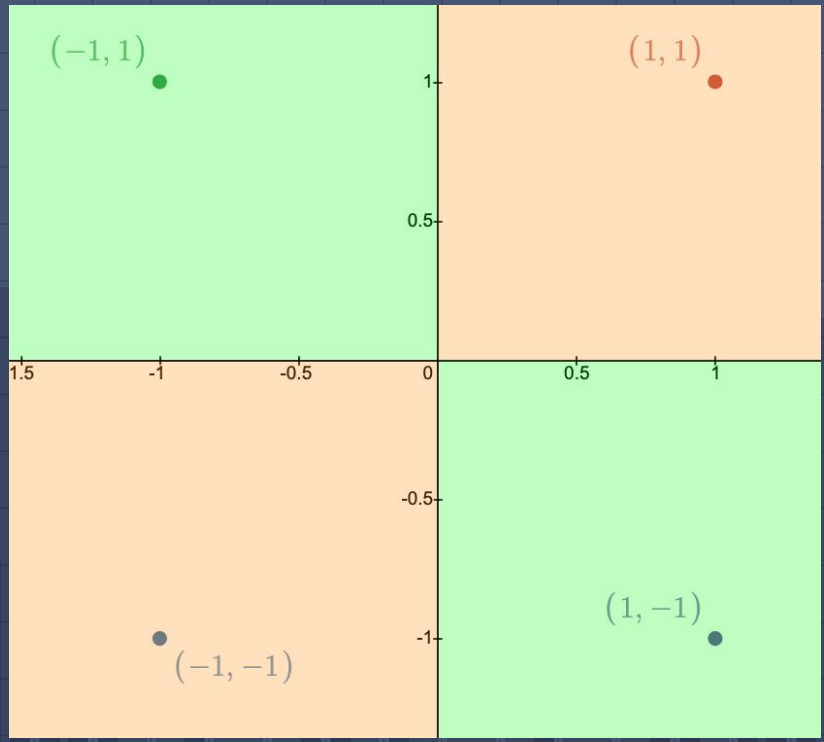


Perceptrón multicapa

3

Función 1: XOR

Las entradas no son separables, pero el perceptrón multicapa tiene la capacidad de separar más allá de funciones convencionales, por lo cual obtenemos el resultado deseado.



Resultados ej3.1

Resultados esperados

```
esperado para el [-1;1]: 1
resultado: [[0.98990252]]
esperado para el [1;-1]: 1
resultado: [[0.99165293]]
esperado para el [-1;-1]: -1
resultado: [[-0.99610596]]
esperado para el [1;1]: -1
resultado: [[-0.98864042]]
[[ 1.09133109  1.15401603]
 [ 1.28758507  1.35446256]
 [-0.58352894  1.40450733]
 [-0.33134648  1.56068898]
 [ 1.7675488  -1.04061325]
 [ 1.30161897  1.36455083]]
[[ 0.8194464   1.38863859 -1.64194307 -1.61022789 -3.00382094  1.37475104]]
error final:
-9.997117418763276e-09
```

Matriz(nodos de entrada con nodos ocultos) de los pesos finales

Matriz(nodos ocultos con nodos de salida) de los pesos finales

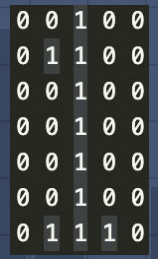
Error final

Función 2: Paridad visual

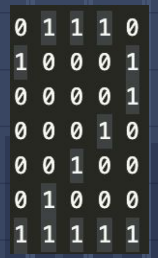
Se trata de un problema más complejo. Primero entrenamos la red con todos los elementos, y obtuvimos resultados correctos.

Al entrenar la red con menos elementos, el problema no se generalizó bien.

1 (par)



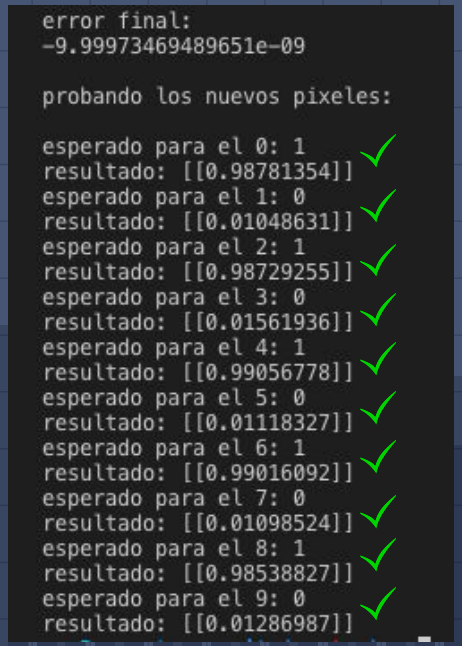
0 (impar)



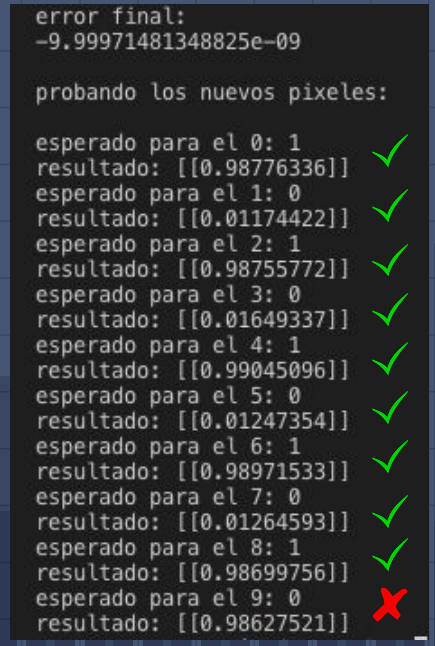
1 (par)

Función 2: Resultados

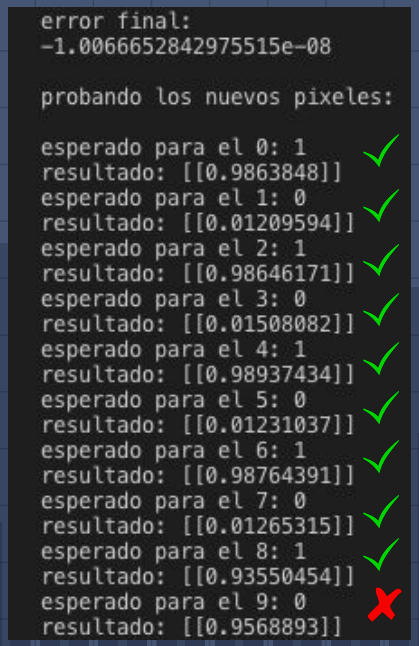
10 números de entrenamiento (0 al 9)



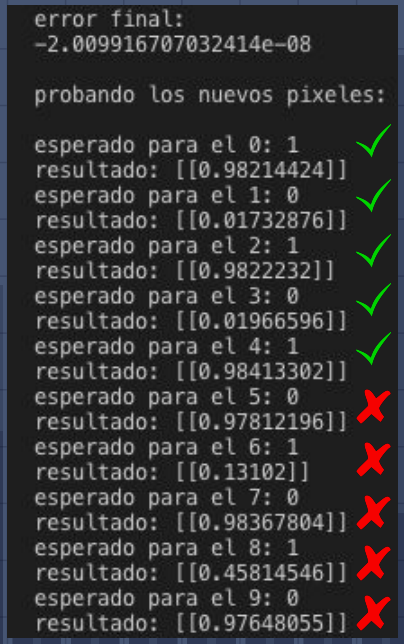
9 números de entrenamiento (0 al 8)



8 números de entrenamiento (0 al 7)



4 números de entrenamiento (0 al 3)



Capacidad de Generalización

Ninguno de los dos problemas planteados nos permiten evaluar correctamente la capacidad de generalización del perceptrón multicapa.

Las funciones están definidas de forma no inferible a lo largo de todo su dominio.

Gracias!