SIA - Trabajo práctico 3
Redes Neuronales

Grupo 4
Tomás Dallas
Tomás Dorado

Implementación

- Implementación en Python
- Se utilizaron las librerías:
  - numpy: para manejo de arrays, generación de randoms y multiplicaciones vectoriales
  - matplotlib: para graficar
  - pandas: para parseo

Ejercicio 1

Implemente el algoritmo de perceptrón simple con función de activación escalón y utilícelo para aprender los siguientes problemas:

- Función lógica 'Y'
- Función lógica 'O exclusivo'

¿Qué puede decir acerca de los problemas que puede resolver el perceptrón simple escalón en relación a la resolución de los problemas que se le pidió que haga que el perceptrón aprenda?

Ejercicio 1 - Resultados

```
AND:
weights:  [-0.35614474  0.79859068  0.63865929]
input:  [-1 -1]
expected value:  -1
result value:  -1
input:  [-1  1]
expected value:  -1
result value:  -1
input:  [ 1 -1]
expected value:  -1
result value:  -1
input:  [1 1]
expected value:  1
result value:  1
```

```
OR:
weights:  [0.73510771 0.62289248 0.30726577]
input:  [-1  1]
expected value:  1
result value:  1
input:  [ 1 -1]
expected value:  1
result value:  1
input:  [-1 -1]
expected value:  -1
result value:  -1
input:  [1 1]
expected value:  1
result value:  1
```

Ejercicio 1 - Resultados

```
XOR:
weights:  [0.02454653 0.47608339 0.12782402]
input:  [-1  1]
expected value:  1
result value:  -1
input:  [ 1 -1]
expected value:  1
result value:  1
input:  [-1 -1]
expected value:  -1
result value:  -1
input:  [1 1]
expected value:  -1
result value:  1
```

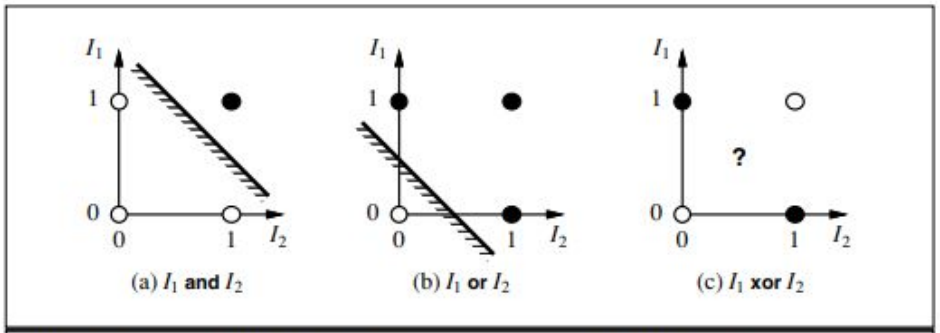


Figura 20.20 Separabilidad lineal en perceptrones umbral. Los puntos negros indican un punto del espacio de entrada donde el valor de la función es 1, y los puntos blancos indican un punto donde el valor es 0. El perceptrón devuelve 1 en la región del lado no sombreado de la línea. En (c), no existe una recta que clasifique correctamente las entradas.

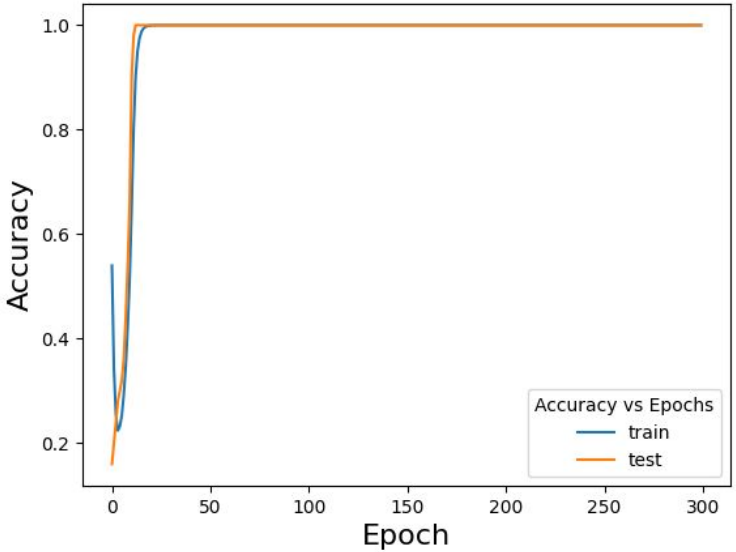
Ejercicio 2

Implemente el algoritmo de perceptrón simple no lineal y util´ıselo para aprender el problema especificado en el archivo TP3 − ej2 − Conjunto entrenamiento.xlsx

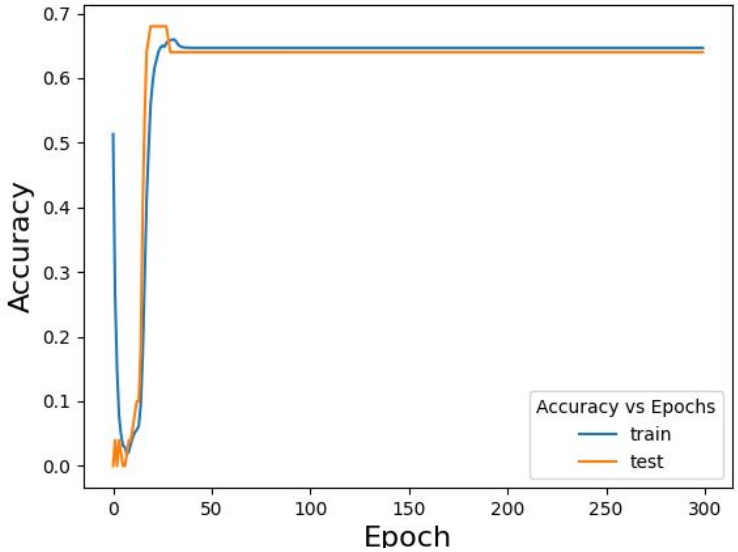
- Evalúe la capacidad de generalización del perceptrón utilizando, de los datos en el archivo, un subconjunto de ellos para entrenar y otro subconjunto para testear.
- ¿Cómo podría escoger el mejor conjunto de entrenamiento?
- ¿Cómo podría evaluar la máxima capacidad de generalización del perceptrón para este conjunto de datos?

Ejercicio 2 - Resultados

eta=0.1, train n = 150, test n = 50



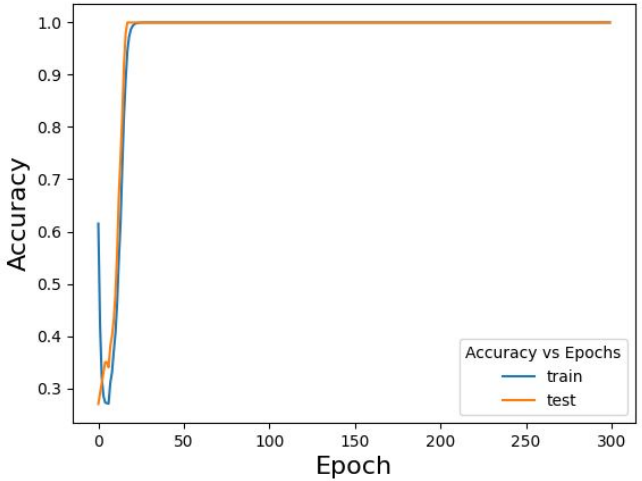
delta = 0.01



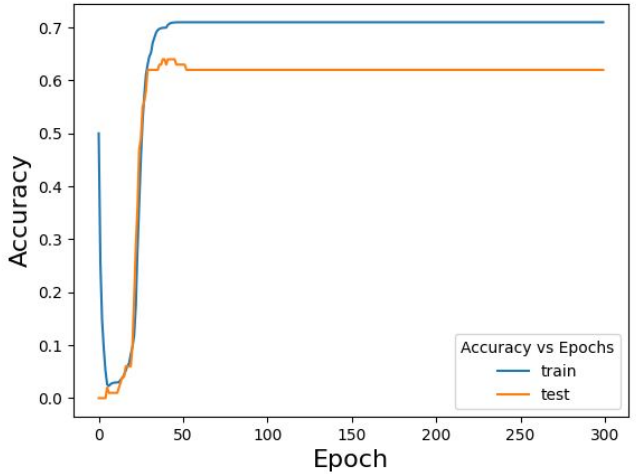
delta = 0.001

Ejercicio 2 - Resultados

eta=0.1, train n = 100, test n = 100



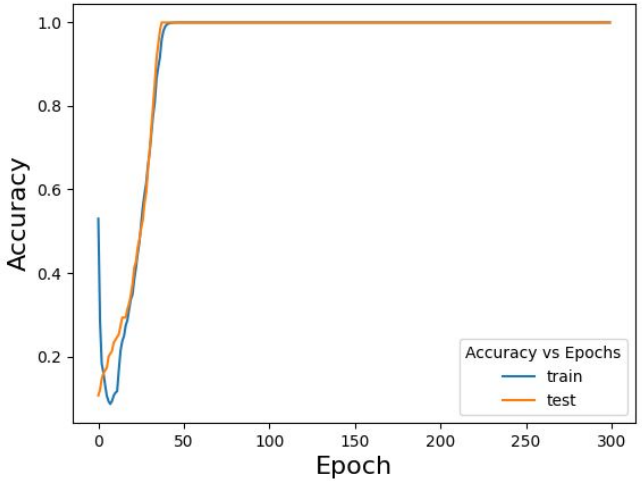
delta = 0.01



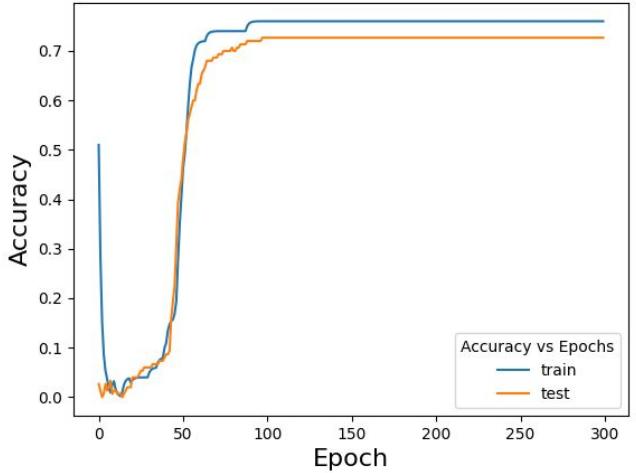
delta = 0.001

Ejercicio 2 - Resultados

eta=0.1, train n = 50, test n = 150



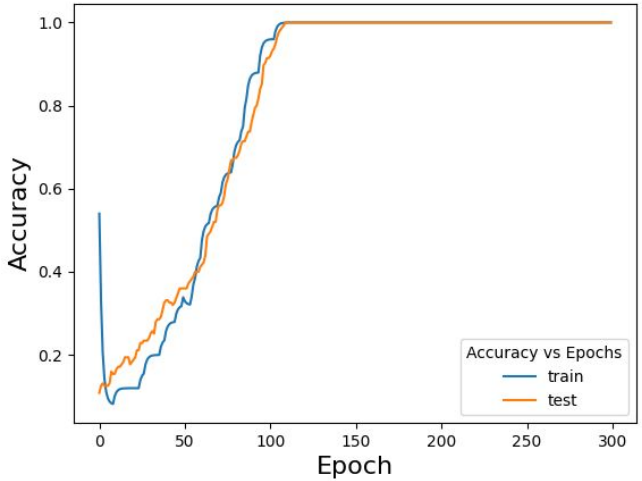
delta = 0.01



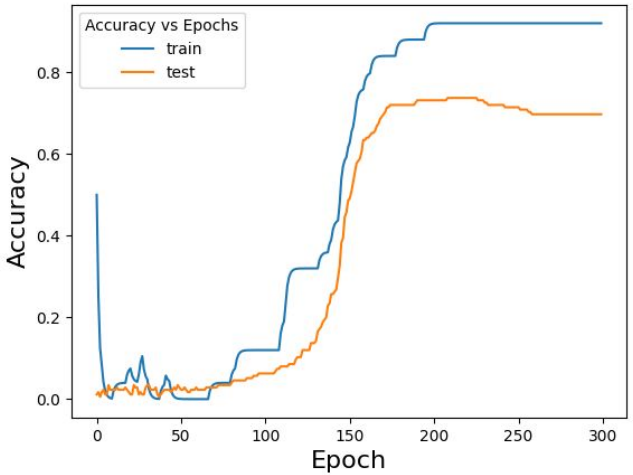
delta = 0.001

Ejercicio 2 - Resultados

eta=0.1, train n = 25, test n = 175



delta = 0.01



delta = 0.001

Ejercicio 2 - Resultados

- Dependiendo de cuán preciso se quiera que sea puede tener una buena generalización, como pudimos ver en los gráficos.
- Para escoger el mejor conjunto de entrenamiento habría que probar con muchas combinaciones del subconjunto de entrenamiento y realizar los gráficos de *accuracy* para ver cual es el mejor.
- También para evaluar la máxima capacidad de generalización habría que hacer lo mismo.

Ejercicio 3

Implemente un perceptrón multicapa y utilícelo para aprender los siguientes problemas:

1. Función lógica ‘O exclusivo’
2. Discriminar si el número es par, con entradas dadas por el conjunto de números decimales del 0 al 9 (archivo numeros.dat) representados por imágenes de 5 x 7 pixeles. Entrene con un subconjunto de los dígitos y utilice el resto para testear a la red ¿Qué podría decir acerca de la capacidad para generalizar de la red?

Ejercicio 3 - Resultados - Función lógica XOR

iterations = 100000
hidden layers = 5

```
[0 xor 1] is ~ [0.77875818]
[0 xor 0] is ~ [0.09566074]
[1 xor 0] is ~ [0.7787563]
[1 xor 1] is ~ [0.0956671]
```

iterations = 10000
hidden layers = 5

```
[0 xor 1] is ~ [0.50024248]
[0 xor 0] is ~ [0.49996717]
[1 xor 0] is ~ [0.49956341]
[1 xor 1] is ~ [0.49982967]
```

iterations = 100000
hidden layers = 2

```
[0 xor 1] is ~ [0.82367698]
[0 xor 0] is ~ [0.20876395]
[1 xor 0] is ~ [0.82329772]
[1 xor 1] is ~ [0.22203354]
```

iterations = 10000
hidden layers = 2

```
[0 xor 1] is ~ [0.81837723]
[0 xor 0] is ~ [0.20926814]
[1 xor 0] is ~ [0.50330304]
[1 xor 1] is ~ [0.50397375]
```

Ejercicio 3 - Resultados - Números pares

iterations = 100000
hidden layers = 1

```
0 is pair with probability [0.69574389]
1 is pair with probability [0.1687669]
2 is pair with probability [0.69554654]
3 is pair with probability [0.17092507]
4 is pair with probability [0.69615883]
5 is pair with probability [0.16868791]
6 is pair with probability [0.69610321]
7 is pair with probability [0.16866569]
8 is pair with probability [0.69498931]
9 is pair with probability [0.16943436]
```

iterations = 10000
hidden layers = 1

```
0 is pair with probability [0.69612815]
1 is pair with probability [0.16839579]
2 is pair with probability [0.6961096]
3 is pair with probability [0.16861751]
4 is pair with probability [0.6961708]
5 is pair with probability [0.16838981]
6 is pair with probability [0.69616685]
7 is pair with probability [0.16838861]
8 is pair with probability [0.69605094]
9 is pair with probability [0.16846603]
```

subconjunto training completo

Ejercicio 3 - Resultados - Números pares

iterations = 100000
hidden layers = 1

```
0 is even with probability [0.69611182]
1 is even with probability [0.16839204]
2 is even with probability [0.69612537]
3 is even with probability [0.16849521]
4 is even with probability [0.69616814]
5 is even with probability [0.1684007]
6 is even with probability [0.18164498]
7 is even with probability [0.68348962]
8 is even with probability [0.20397474]
9 is even with probability [0.64228864]
```

subconjunto training = {0, 1, 2, 3, 4, 5, 6}

iterations = 100000
hidden layers = 1

```
0 is even with probability [0.69611696]
1 is even with probability [0.16841293]
2 is even with probability [0.69610875]
3 is even with probability [0.16853346]
4 is even with probability [0.69617148]
5 is even with probability [0.16839125]
6 is even with probability [0.69613281]
7 is even with probability [0.16840856]
8 is even with probability [0.6812761]
9 is even with probability [0.67308712]
```

subconjunto training = {0, 1, 2, 3, 4, 5, 6, 7, 8}

Ejercicio 3 - Resultados - Números pares

iterations = 100000
hidden layers = 1

```
0 is even with probability [0.69616343]
1 is even with probability [0.16840839]
2 is even with probability [0.69615535]
3 is even with probability [0.69601209]
4 is even with probability [0.64057638]
5 is even with probability [0.69210832]
6 is even with probability [0.63599813]
7 is even with probability [0.69336504]
8 is even with probability [0.69614616]
9 is even with probability [0.69614489]
```

iterations = 1000000
hidden layers = 1

```
0 is even with probability [0.69617108]
1 is even with probability [0.16836135]
2 is even with probability [0.69617026]
3 is even with probability [0.69614647]
4 is even with probability [0.66174788]
5 is even with probability [0.69483616]
6 is even with probability [0.65258716]
7 is even with probability [0.69525651]
8 is even with probability [0.69616869]
9 is even with probability [0.69616785]
```

subconjunto training = {0, 1, 2, 3}

Conclusiones

- Perceptrón simple:
  - Ajustar valor de ETA junto con iteraciones
  - Tener en cuenta objetivo a estudiar (linealidad vs no linealidad)
- Perceptrón multicapa:
  - Ajustar número de iteraciones junto con cantidad de capas ocultas
- Utilizar otras funciones no lineales además de sigmoide
- Ajustar y probar parámetros de la función de activación a utilizar
- Diseño modularizado