UNIVERSIDAD SIMÓN BOLÍVAR
Departamento de Cómputo Científico
CO-6612, Introducción a las redes neuronales
Tarea 2: Clasificadores lineales
María Victoria Jorge
11-10495

# 1 Clasificadores Lineales

En esta sección se presentarán los experimentos realizados para construir clasificadores lineales mediante aprendizaje con reforzamiento, un perceptrón de una capa y un adaline.

Para medir la calidad de todos los modelos se utilizará la técnica de validación cruzada. Además, se utilizará el porcentaje de datos bien clasificados como métrica de *performance*. Por último, para asegurarnos que los modelos no están aprendiendo a clasificar basándose en el orden en el que reciben los datos, en cada corrida estos se desordenarán aleatoriamente.

Como los datos presentados en el archivo lincloud3 tienen solo dos dimensiones, pueden graficarse e identificar fácilmente si son linealmente separables, para asegurarnos que los modelos utilizados a continuación serán capaces de clasificarlos. En la figura 1 se tienen todos los datos identificados según la clase a la que pertenecen, y se puede observar que es posible trazar una línea entre ambas clases para separarlas.

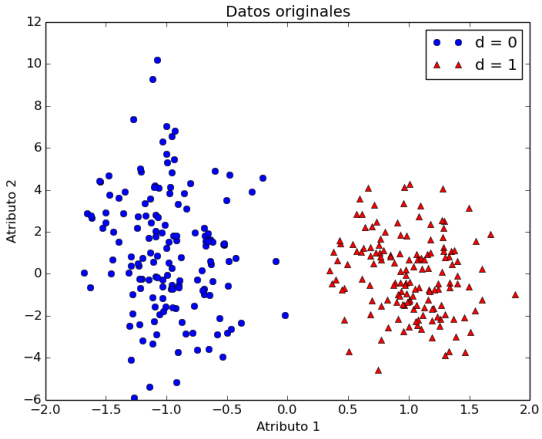



Figure 1: Datos originales

## 1.1 Aprendizaje con reforzamiento

Para estas pruebas se inicializaron los pesos de las dos neuronas en valores aleatorios entre -0.5 y 0.5, y se usaron esos mismos valores en todas las corridas. Como condición de parada para el entrenamiento se tomó que no existiera error

en la clasificación de los datos o que se superara un número máximo de épocas (1000 épocas).

En la Tabla 1 están los resultados obtenidos de la ejecución de la validación cruzada 5-fold usando la red basada en aprendizaje con reforzamiento. Se observa que para las tres tasas de aprendizaje utilizadas se obtuvieron resultados parecidos, clasificando correctamente la mayoría de los datos. El número de épocas indicado es el promedio de épocas utilizadas durante las 5 iteraciones de la validación cruzada.

| Total datos | $\alpha$ | % Correctos | % Incorrectos | Épocas |
|---|---|---|---|---|
| 300 | 0.1 | 99.6667 | 0.3333 | 4.8 |
| 300 | 0.01 | 99.6667 | 0.3333 | 2 |
| 300 | 0.001 | 99.3333 | 0.6667 | 4.2 |

Table 1: Exactitud aprendizaje con reforzamiento

## 1.2 Perceptrón

Para estas pruebas se inicializaron los pesos en valores aleatorios entre -0.5 y 0.5, y se usaron esos mismos valores en todas las corridas. Se utilizó una sola neurona y la función de activación sgn o umbral. Como condición de parada para el entrenamiento se tomó que no existiera error en la clasificación de los datos o que se superara un número máximo de épocas (1000 épocas). Además se cambió la respuesta deseada de los datos cuyo valor era 0, por -1, para mantener la simetría respecto al 0 y que ambas clases tuvieran el mismo peso.

En la Tabla 2 se pueden observar los resultados obtenidos luego de ejecutar validación cruzada con 5-fold para diferentes tasas de aprendizaje en una única neurona. Por cada iteración se utilizaron 240 datos para entrenar y 60 para probar el modelo. Luego de 5 iteraciones se probó en todos los experimentos con los 300 datos. El número de épocas indicado para cada tasa de aprendizaje es el promedio de los 5 entrenamientos realizados por tasa, con diferentes datos para entrenar en cada una.

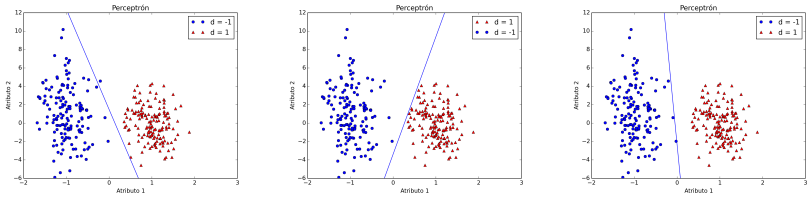
| Total datos | $\alpha$ | % Correctos | % Incorrectos | Épocas |
|---|---|---|---|---|
| 300 | 0.1 | 99.6666 | 0.3334 | 3 |
| 300 | 0.01 | 99.6666 | 0.3334 | 3.4 |
| 300 | 0.001 | 99.3333 | 0.6667 | 15.2 |

Table 2: Exactitud del perceptrón

Con estos resultados puede observarse que al disminuir la tasa de aprendizaje el perceptrón necesita más épocas para cumplir con la cota de clasificar correctamente todos los datos de entrenamiento. Sin embargo, parece que al aumentar la cantidad de épocas para estos datos, el perceptrón tiene más prob-

lemas para clasificar los datos nuevos.

Esto puede apreciarse de forma más clara en las gráficas de la figura 2, donde se muestra para cada tasa de aprendizaje utilizada el peor hiperplano generado de todas las iteraciones de la validación cruzada. Podemos observar que en las tres figuras el hiperplano está muy cerca de las dos fronteras, lo que puede originar errores de clasificación cuando esos datos no se encuentran dentro del conjunto de entrenamiento, como sucedió en las gráficas 2b y 2c. Con esto se puede concluir que el modelo tiene problemas para generalizar.




(a) $\alpha = 0.1$ (b) $\alpha = 0.01$ (c) $\alpha = 0.001$

Figure 2: Modelos de perceptrón obtenidos en los tres experimentos

## 1.3 Adaline

Para estas pruebas se inicializaron los pesos de la neurona en valores aleatorios entre -0.5 y 0.5, manteniéndolos fijos durante todas las iteraciones de la validación cruzada de 5-fold. Como condición de parada para el algoritmo de entrenamiento se usó un máximo de 5000 épocas. Además, se modificó el archivo que contiene los datos para que las clases sean -1 y 1, en vez de 0 y 1 como estaban originalmente.

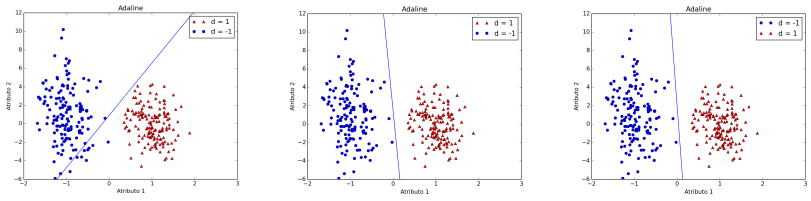
En la Tabla 3 se encuentran los resultados obtenidos para diferentes tasas de aprendizaje. Se puede observar que con tasas de aprendizaje muy pequeñas el Adaline converge mejor a la solución esperada, clasificando correctamente todos los datos.

| Total datos | $\alpha$ | % Correctos | % Incorrectos | Épocas |
|---|---|---|---|---|
| 300 | 0.1 | 98 | 2 | 5000 |
| 300 | 0.01 | 100 | 0 | 5000 |
| 300 | 0.001 | 100 | 0 | 5000 |

Table 3: Exactitud del Adaline

Se realizaron pruebas con otras tasas de aprendizaje pero se presentaron solo las más significativas. El valor más alto con el que se obtuvieron buenos resultados fue $\alpha = 0.1$. Con $\alpha$ más grandes, 5000 épocas no son suficientes para que el Adaline genere porcentajes tan altos de aciertos, debido a que se le dificulta la convergencia.

Al igual que para el Perceptrón, se graficaron los hiperplanos generados en los experimentos para el Adaline. En la figura 3 se encuentran los peores de cada validación cruzada para cada una de las tasas de aprendizaje utilizadas. Puede observarse en la figura 3a que ya la tasa de aprendizaje 0.1 puede ser muy alta dependiendo de la diversidad del conjunto de entrenamiento. En ese caso los valores más cercanos a la frontera no se encontraban en dicho conjunto lo que ocasionó una mala generalización del modelo.




(a) $\alpha = 0.1$ (b) $\alpha = 0.01$ (c) $\alpha = 0.001$

Figure 3: Modelos de Adaline obtenidos en los tres experimentos

# 2 Interpolador

El algoritmo de Adaline puede ser utilizado para iterpolación de funciones si se calcula la salida de la siguiente forma $y = a + b_1x + b_2x^2 + ... + b_kx^k$, siendo k el grado del polinomio que se usará para interpolar, 'a' el sesgo, y cada $b_i$ los $w_i$ del Adaline.

Sabiendo esto debemos conocer cómo se comportan los datos que tenemos en p3data para saber el grado del polinomio con el que aproximaremos la función que los originó. Con la figura 4 se puede apreciar que los datos parecen ser originados por una función de grado impar, aún cuando tengan algo de ruido. Es por esto que se realizarán pruebas con polinomios de grado 3 y 5 con diferentes tasas de aprendizaje para determinar cuál verdaderamente aproxima mejor estos datos.

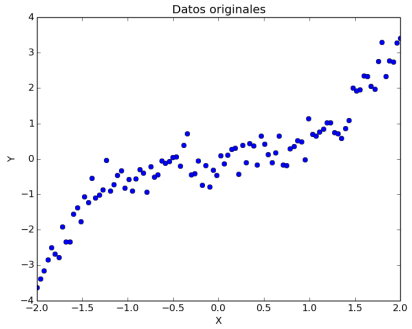
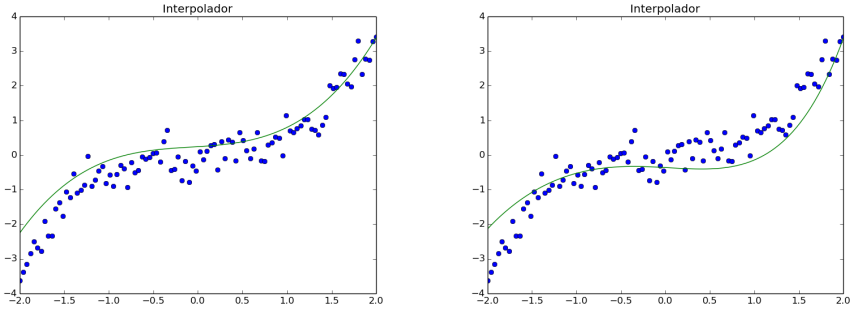



Figure 4: Datos originales a interpolar

Usando una tasa de aprendizaje de 0.01 se obtuvieron los resultados pre-

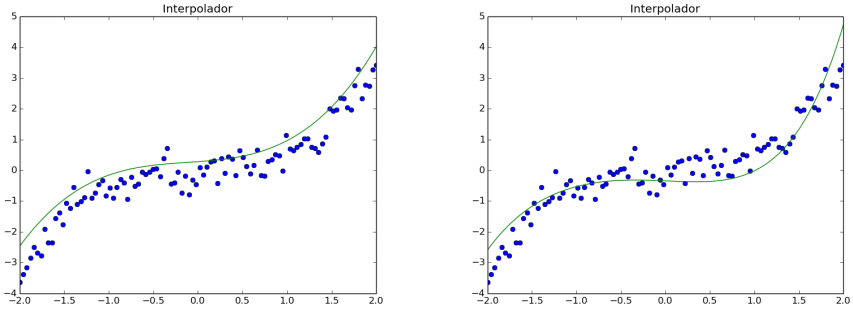
sentes en la figura 5. Allí podemos observar las aproximaciones realizadas con los polinomios de grado 3 en 5a y de grado 5 en 5b.




(a) Polinomio grado 3 (b) Polinomio grado 5

Figure 5: Interpoladores con $\alpha = 0.01$

Los resultados producto de los mismos polinomios pero con una tasa de aprendizaje para el Adaline de 0.001 se presentan en la figura 6a.




(a) Polinomio grado 3 (b) Polinomio grado 5

Figure 6: Interpoladores con $\alpha = 0.001$

El polinomio de grado 3 parece ser el que mejor moldea la forma de los datos, sin embargo parece estar desplazado hacia arriba por la presencia de datos que contienen ruido. Por otro lado, aunque el polinomio de grado 5 tiene curvas más pronunciadas, parece estar más cerca del centro de los datos, teniendo su mejor aproximación con $\alpha = 0.001$.

# 3 O-exclusivo

Se demostrará que las clases del O-exclusivo (XOR) no son linealmente separables por reducción al absurdo.

Supongamos entonces que las clases del XOR son linealmente separables. Entonces existe un $\vec{w} = [w_0, w_1, w_2]^t$ que permite generar el hiperplano $\vec{w}^t\vec{x} = 0$ para un perceptrón de una neurona. Esta es la frontera que divide a las dos clases del XOR. Los datos que estén por debajo de esa frontera pertenecen a la clase 0, y los que están por encima pertenecen a la clase 1.

Por lo tanto se generan las siguientes ecuaciones:

$$0 * w_1 + 0 * w_2 + w_0 <= 0 \iff w_0 <= 0 \tag{1}$$

$$0 * w_1 + 1 * w_2 + w_0 > 0 \iff w_0 > -w_2 \tag{2}$$

$$1 * w_1 + 0 * w_2 + w_0 > 0 \iff w_0 > -w_1 \tag{3}$$

$$1 * w_1 + 1 * w_2 + w_0 <= 0 \iff w_0 <= -w_1 - w_2 \tag{4}$$

De (1) obtenemos que $w_0$ es un número no positivo. Usando esta información en (2) y (3) tenemos que $w_1$ y $w_2$ son números positivos.

Luego, sabiendo que $w_1$ es positivo, podemos asegurar que $-w_2 > -w_2 - w_1$. Usando esta cota inferior en (2) se sigue cumpliendo que $w_0 > -w_2 - w_1$. Esto entra en contradicción con (4) que se obtuvo directamente del perceptrón. Por lo tanto la suposición que se hizo inicialmente sobre la separabilidad lineal de las clases del XOR es errónea.

# 4 Descenso del gradiente

Considerando la función de costo

$$E(w) = \frac{1}{2}\sigma^2 - r^t w + \frac{1}{2} w^t R w$$

sabemos que el vector de pesos tiene dos componentes $w_1$ y $w_2$. Si desarrollamos la función de costo para dejarlo en función de estas componentes nos queda:

$$E(w) = \frac{1}{2}\sigma 2 - \begin{bmatrix} 0.8182 & 0.354 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 & 0.8182 \\ 0.8182 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$= \frac{1}{2}\sigma^2 - (0.8182 w_1 + 0.354 w_2) + \frac{1}{2} \begin{bmatrix} w_1 + 0.3182 w_2 & 0.8182 w_1 + w_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$= \frac{1}{2}\sigma^2 - 0.8182 w_1 - 0.354 w_2 + \frac{1}{2}(w_1^2 + 1.6364 w_1 w_2 + w_2^2)$$

Luego, para encontrar el valor óptimo de w para el cual E(w) es mínimo debemos calcular el gradiente de la función de costo e igualarlo a cero para determinar los valores que deben tener $w_1$ y $w_2$.

$$\frac{\partial E(w)}{\partial w_1} = -0.8182 + w_1 + \frac{16364}{2} w_2 = 0 \tag{5}$$

$$\frac{\partial E(w)}{\partial w_2} = -0.354 + w_2 + \frac{1.6364}{2} w_1 = 0 \tag{6}$$

Despejando $w_1$ de (5) obtenemos,

$$w_1 = 0.8182(1 - w_2) \tag{7}$$

Luego, sustituyendo en (6):

$$-0.354 + w_2 + 0.8182 * 0.8182(1 - w_2) = 0$$

De donde obtenemos que $w_2 = -0.9543$ y sustituyendo en (7) obtenemos que $w_1 = 1.599$.

Ahora calcularemos el valor óptimo a través del algoritmo del Descenso del Gradiente. Para ambos experimentos se utilizó una precisión de 0.0001.

- $\eta = 0.3$
  En la figura 7 se encuentra la trayectoria de la evolución de los valores de $w_1$ durante la ejecución del algoritmo. Los puntos indican los valores que tomó esta componente durante todas las iteraciones y la línea indica la trayectoria realizada.

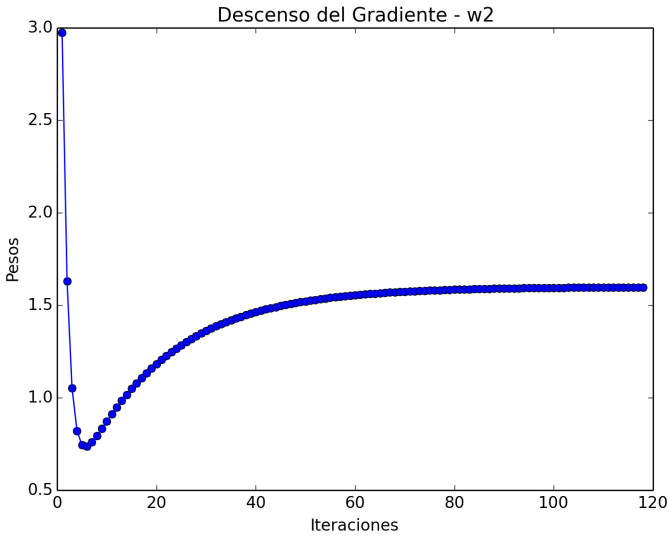



Figure 7: Trayectoria de $w_1$

En la figura 8 se encuentran los valores que tomó $w_2$ durante las iteraciones. Los triángulos indican los valores específicos que tomó esta componente en cada iteración y la línea indica la trayectoria realizada.

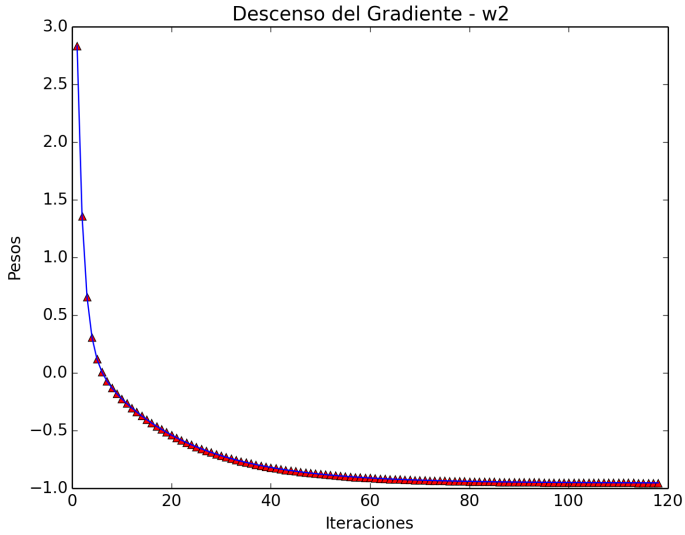



Figure 8: Trayectoria de $w_2$

Al final el algoritmo con 0.0001 de precisión se obtuvo que el valor óptimo de w, donde E(w) es mínimo es $w = \begin{bmatrix} 1.5973 & -0.9526 \end{bmatrix}$, valores bastante cercanos a los que se calcularon directamente con el gradiente de la función.

- $\eta = 1.0$
  En las figura 9 se encuentra la trayectoria de la evolución de $w_1$ durante las iteraciones realizadas por el algoritmo. Los puntos indican el valor específico que tuvo $w_1$ en cada iteración, y la línea es la trayectoria que realizó en total.

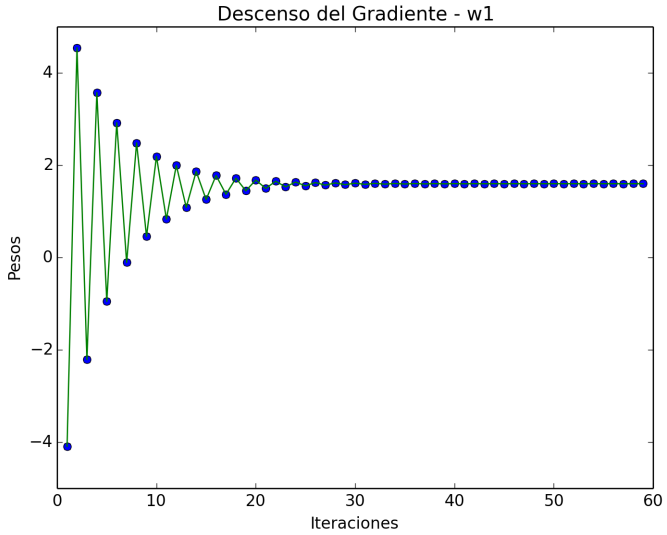



Figure 9: Trayectoria de $w_1$

Además, en la figura 10 se encuentran los valores por los que pasó $w_2$ durante las iteraciones, indicados por triángulos, junto a su trayectoria.

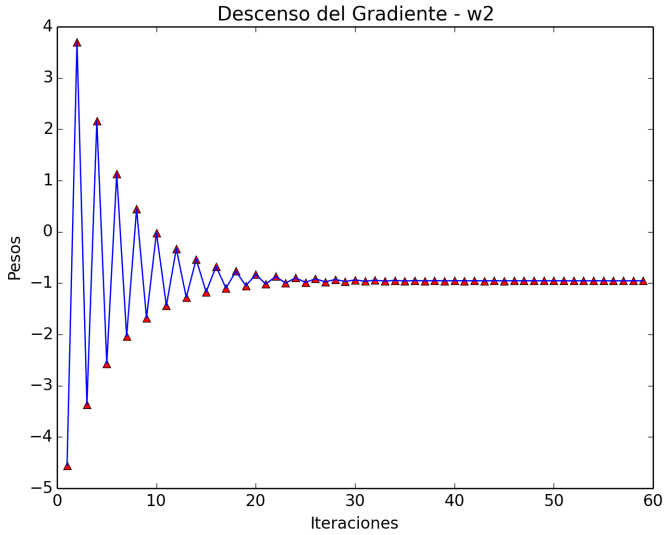



Figure 10: Trayectoria de $w_2$

Al finalizar el algoritmo con 0.0001 de precisión se obtuvo que el valor óptimo w para el cual E(w) es mínimo es $w = \begin{bmatrix} 1.5989 & -0.9543 \end{bmatrix}$ siendo una aproximación casi exacta a la calculada en la primera parte.

# 5 Detalles de implementación

Todos los experimentos de esta tarea fueron implementados en *Python 3*. Para el manejo de los arreglos de pesos y su generación aleatoria se usó la librería *Numpy* y para las gráficas la librería *Matplotlib*. La implementación de la validación cruzada fue realizada de forma manual, y los algoritmos del perceptrón y Adaline están basados en el código de los mismos implementado en *Matlab* que se encuentra en el Aula Virtual.