Jesus Wahrman 15-11540

Tarea Adaline

2)

Para probar si clasifico bien lo que haremos es decir que el máximo output del conjunto de neuronas es 1 y el resto es 0, y asi comparamos con el output esperado. Para entrenar no se hará esto.

Primero quisimos comparar usando las mismas tasas que usamos para la tarea del perceptrón, usamos igualmente una condición de parada de 50 épocas y pesos iniciales aleatorios y pequeños, los resultados fueron:

0.1	0.01	0.001
0.32%	70.45%	82.55%

Por lo que podemos ver que con 0.001 es bastante mejor que el perceptrón que alcanzaba un máximo de 75%. Aunque con 0.1 da muy mal, esto debido a lo que vimos en clase, si el salto es muy alto afecta al algoritmo. Ahora probaremos con tasas más cercanas a 0.001: 0.005, 0.0005 y 0.0001.

0.005	0.0001	0.0005
77.72%	85.28%	84.05%

Podemos ver que con los menores que 0.001, mejoró, hasta logró llegar a 85%, que ya es más de 10% mejor que el perceptrón. Ahora probaremos con menores que 0.0001: 0.00001, 0.00003 y 0.00005.

0.00001	0.00003	0.00005
85.81%	85.92%	85.70%

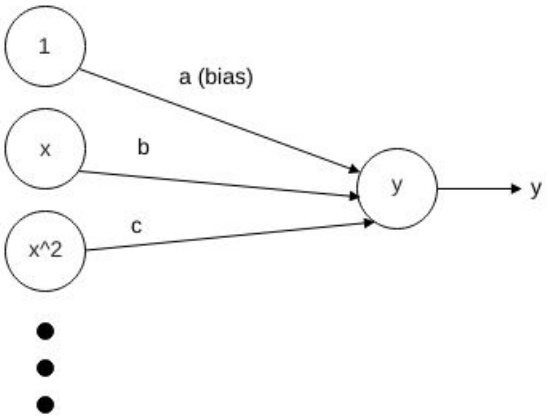
Con estos 3 nos dio porcentajes mejores y muy parecidos, así que para la condición de parada que elegimos a partir de estos valores ya no se nota tanto la diferencia, pero podemos concluir que el Adaline es mucho mejor que el Perceptrón para clasificar los números, con una mejora de más del 10%. No probaremos con tasas más pequeñas ya que se puede ver que no afecta tanto en el porcentaje a partir de este punto y también es muy probable que caigamos en errores de precisión a la hora de hacer cálculos.

Elegimos 50 épocas para hacer una comparación más justa con el perceptrón y lo mantuvimos ya que parece ser una buena cantidad de iteraciones. No colocamos una condición de parada que dependa del error ya que no estábamos seguros cual sería un valor aceptable para el error en este problema. Y finalmente usamos varios valores para la cota de aprendizaje y nos movíamos a donde parecía mejorar hasta llegar a un punto en el que ya no variaba casi.

3)
Para esto la idea fue simular un polinomio de grado n de la forma:

$f(x) = a + bx + cx^2 + dx^3 + \ldots$

De la siguiente forma:



Como el número de datos de entrada es pequeño, podemos tener un número de épocas grande y no afectará el rendimiento, lo que podría ocurrir es “overfitting” que es que nuestro modelo se adapte mucho a estos datos y no funcione bien con otros, pero como estos son los únicos datos que manejamos no nos preocupamos tanto por esto. Por lo tanto decidimos usar 10000 épocas. Con una tasa de aprendizaje de 0.000001 ya que al ser tantas épocas nos permite tener una tasa baja. Probaremos con varios grados:

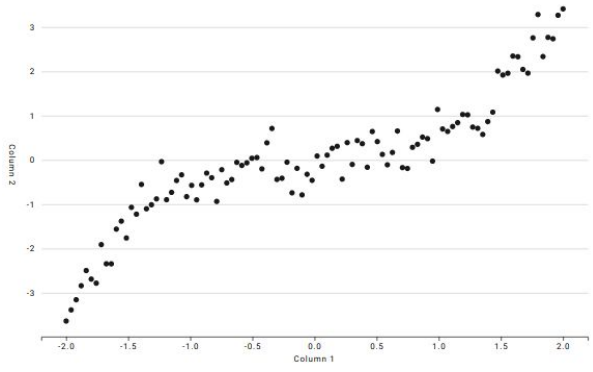
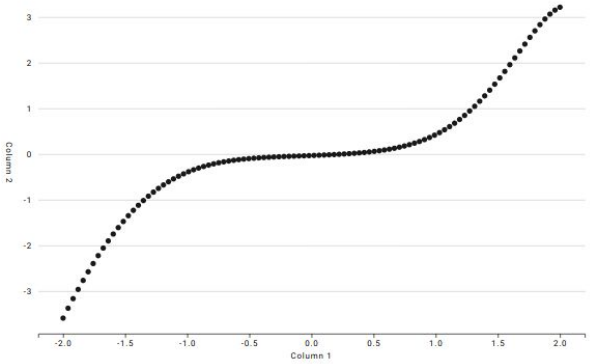
Grado del polinomio	2	3	4	5	6
Error cuadratico medio	24.103143	5.551966	5.638839	6.483037	5.966129

Grado del polinomio	7	8	9	10
Error cuadratico medio	5.515211	5.417606	5.979288	6.058539

No se colocaron polinomios más altos ya que a partir del grado 11 ya daban resultados que no tenían sentido, cosa que se puede deber a errores de precisión o que los datos de entrada son muy grandes para el modelo del adaline.

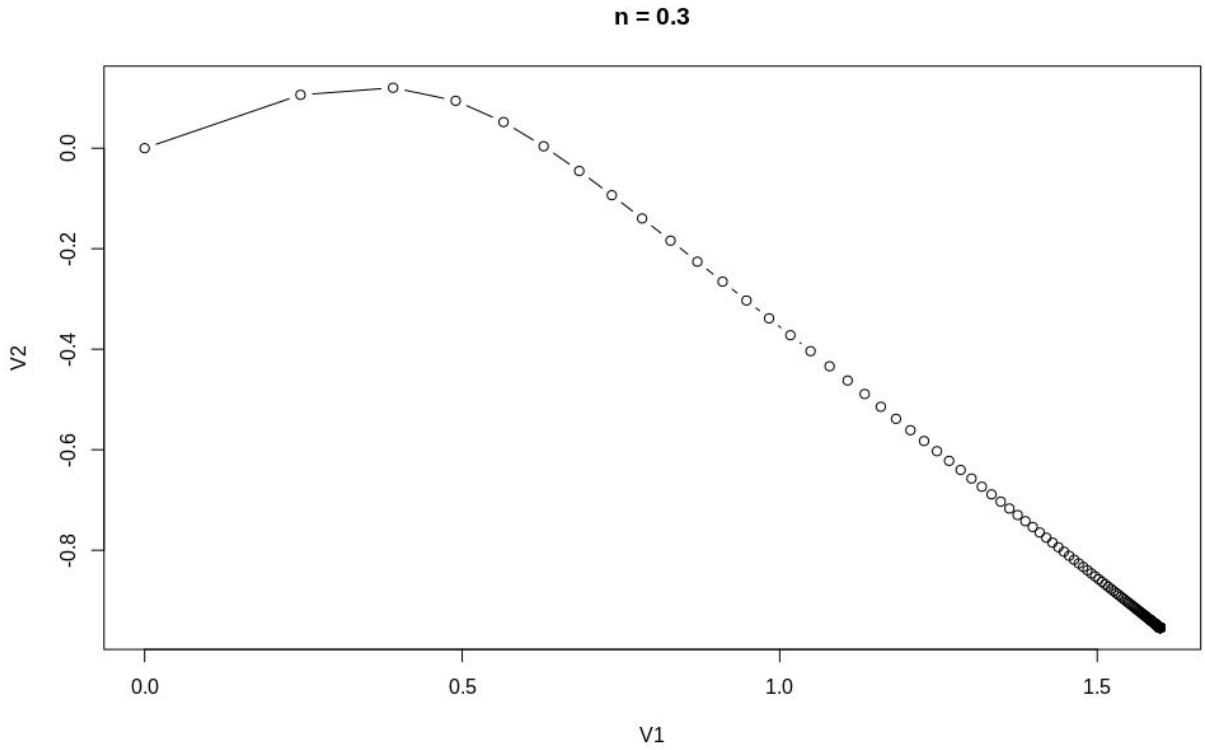
La idea original era hacer un polinomio de grado mayor pero al ver que a partir del 11 los resultados no tenían mucho sentido se decidió dejarlo hasta 10 y tomar esos valores del error cuadrático medio.

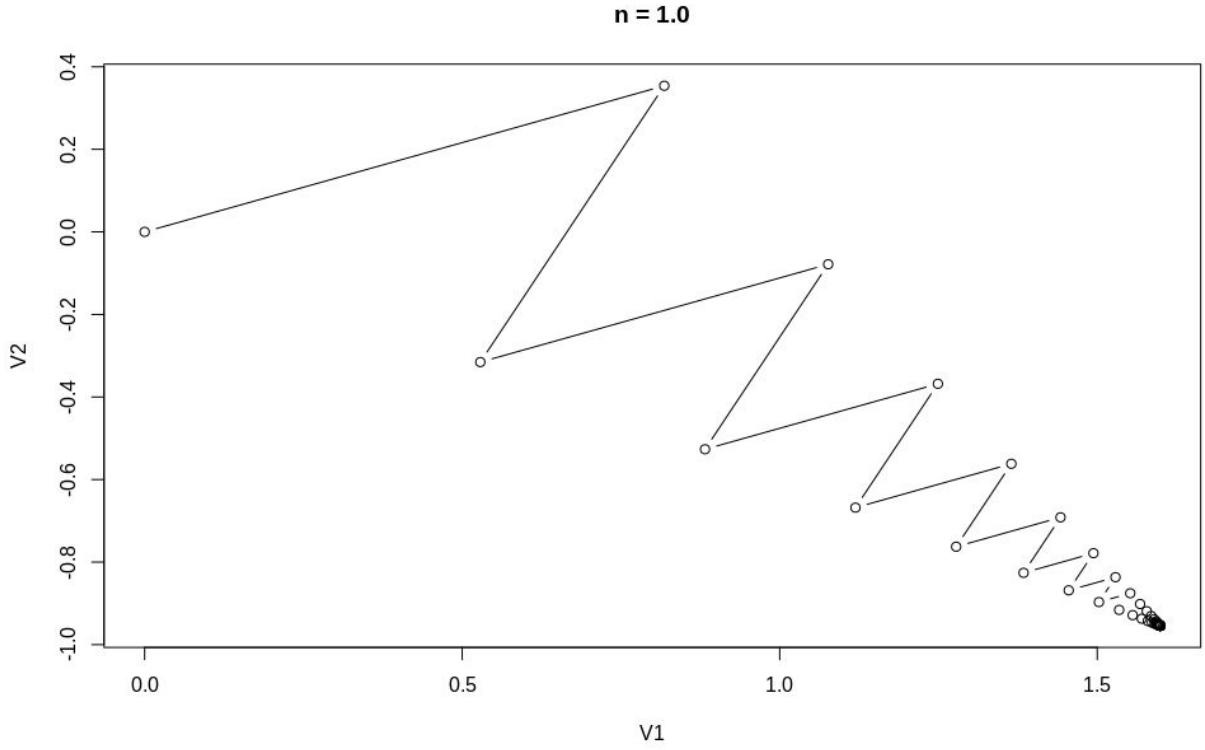
Finalmente para ver que tan bueno es el interpolador, graficamos los (x,y) dados y los (x,y') que da nuestro interpolador, tomamos el de grado 8 ya que es el que menor error dio.

Puntos dados	Puntos obtenidos
	

Podemos ver que el interpolador con grado 8 hizo un buen trabajo, ya que la gráfica obtenida se parece mucho a los puntos dados.

4)
b)


n = 0.3
V2
0.0
-0.2
-0.4
-0.6
-0.8
0.0
0.5
1.0
1.5
V1


n = 1.0
V2
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
0.0
0.5
1.0
1.5
V1

Tarea 3

4 a)

$$\mathcal{E}(w) = \frac{1}{2}\nabla^2 - r^T w + \frac{1}{2} w^T R w$$

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad r = \begin{bmatrix} 0.8182 \\ 0.354 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0.8182 \\ 0.8182 & 1 \end{bmatrix}$$

Multiplicando las matrices

$$\mathcal{E}(w) = \frac{1}{2}\nabla^2 - (0.8182)w_1 - (0.354)w_2 + \frac{w_1^2}{2} + (0.8182)w_1 w_2 + \frac{w_2^2}{2}$$

Queremos que:

$$\nabla\mathcal{E}(w) = \begin{bmatrix} \frac{\partial \mathcal{E}(w)}{\partial w_1} & \frac{\partial \mathcal{E}(w)}{\partial w_2} \end{bmatrix} = [0 \; 0]$$

$$\frac{\partial \mathcal{E}(w)}{\partial w_1} = -0.8182 + w_1 + (0.8182)w_2$$

$$\frac{\partial \mathcal{E}(w)}{\partial w_2} = -0.354 + (0.8182)w_1 + w_2$$

$$\begin{cases} -0.8182 + w_1 + (0.8182)w_2 = 0 \\ -0.354 + (0.8182)w_1 + w_2 = 0 \end{cases}$$

Resolviendo el sist. llegamos a:

$$w_1 = 1.598$$
$$w_2 = -0.954$$

$$\therefore w = \begin{bmatrix} 1.598 \\ -0.954 \end{bmatrix}$$

b)

$$\nabla\mathcal{E}(w) = \begin{bmatrix} -0.8182 + w_1 + (0.8182)w_2 \\ -0.354 + (0.8182)w_1 + w_2 \end{bmatrix}$$

i) $\eta = 0.3$

$$w_{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w_{(1)} = w_{(0)} - \eta\nabla\mathcal{E}(w_{(0)}) = \begin{bmatrix} 0.24546 \\ 0.1062 \end{bmatrix}$$

$$w_{(2)} = w_{(1)} - \eta\nabla\mathcal{E}(w_{(1)}) = \begin{bmatrix} 0.391214 \\ 0.120289 \end{bmatrix}$$

$$w_{(3)} = w_{(2)} - \eta\nabla\mathcal{E}(w_{(2)}) = \begin{bmatrix} 0.489784 \\ 0.094375 \end{bmatrix}$$

$$w_{(4)} = w_{(3)} - \eta\nabla\mathcal{E}(w_{(3)}) = \begin{bmatrix} 0.565144 \\ 0.052040 \end{bmatrix}$$