Universidad Simón Bolívar
Departamento de Cómputo Científico

Tarea III: Adaline y Perceptrón Multicapas

Georvic Tur - 12-11402

15/10/2017

1. Los datos en reglin.mat corresponden a un problema de regresión lineal. Los datos contenidos en dicho archivo contienen un conjunto de 60 puntos para el entrenamiento, 10000 puntos para datos de prueba y los datos que corresponden al modelo real.

   A Entrene una red neuronal con n = [1, 2, 3, 4, 6, 8, 12, 20, 40] neuronas en la capa oculta y una neurona lineal en la capa de salida. Utilice el algoritmo de descenso de gradiente por 700 épocas.

   B Grafique el error cuadrático medio del entrenamiento y del conjunto de prueba en cada caso. Comente sus resultados. ¿Cuál es el mejor valor para n?

   C Grafique el error del conjunto de prueba así como las funciones entrenadas (relación entrada–salida) para $n = 2$, $n = 8$ y $n = 40$. Interprete sus resultados. ¿Es decreciente el error sobre el entrenamiento y la prueba?

   D Emplee una aproximación polinómica a este conjunto de datos mediante un Adaline. Cuál es el grado del polinomio que mejor interpola estos datos?

Solución:

a) Se ha implementado un perceptrón multicapas [1]. Tanto la capa de entrada como la capa de salida tienen una neurona. Su capa oculta tiene un número variable de neuronas, tal como lo requiere el problema. Para todos los casos se usó una tasa de aprendizaje de 0,1 y se usaron la tangente hiperbólica y la función identidad como funciones de activación. Si bien se usaron 700 épocas para el entrenamiento, se implementó un criterio de parada temprana si se detecta que el error de validación ha estado subiendo durante las últimas diez pruebas. A las capas de entrada y la oculta se les agregó sesgos. Todos los pesos se inicializan en 0,5.

b) A continuación se muestran algunos de los gráficos generados. Los demás gráficos están disponibles en el directorio comprimido entregado junto a este informe.

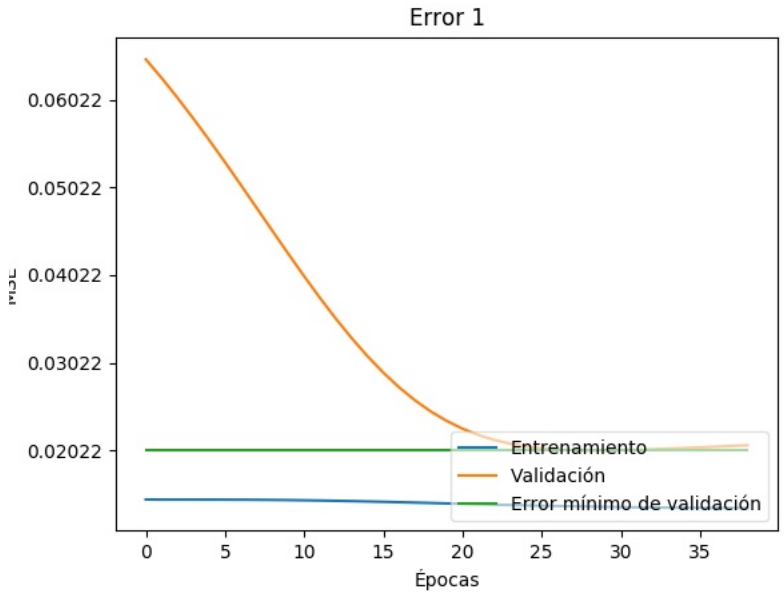

Figura 1: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 39 épocas usadas y una neurona en la capa oculta.

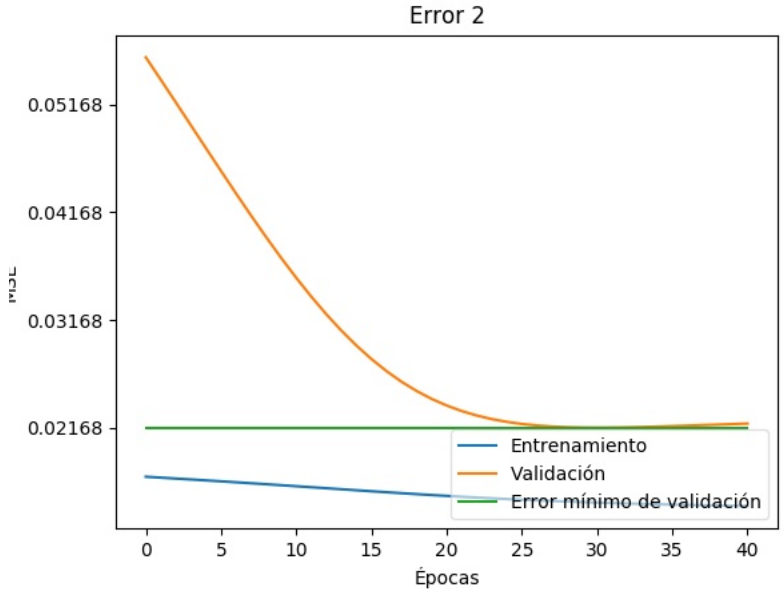

Figura 2: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 41 épocas usadas y dos neuronas en la capa oculta.

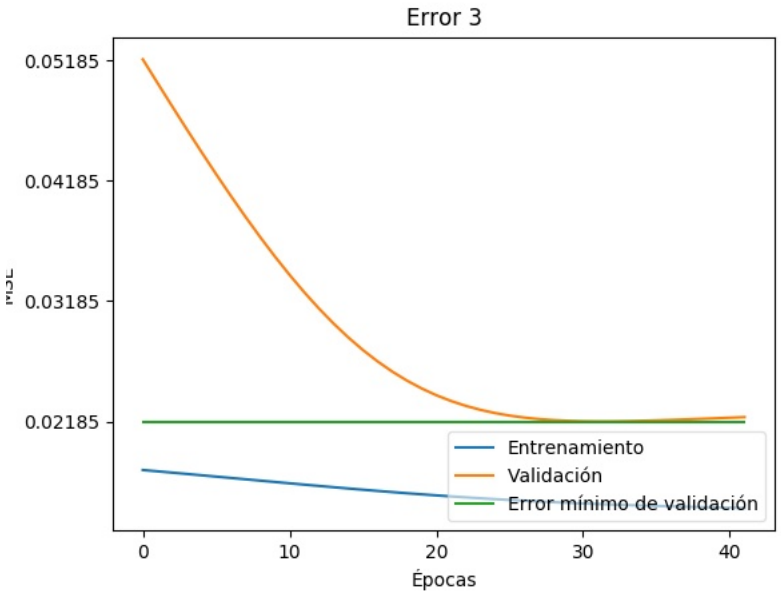

Figura 3: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 42 épocas usadas y tres neuronas en la capa oculta.

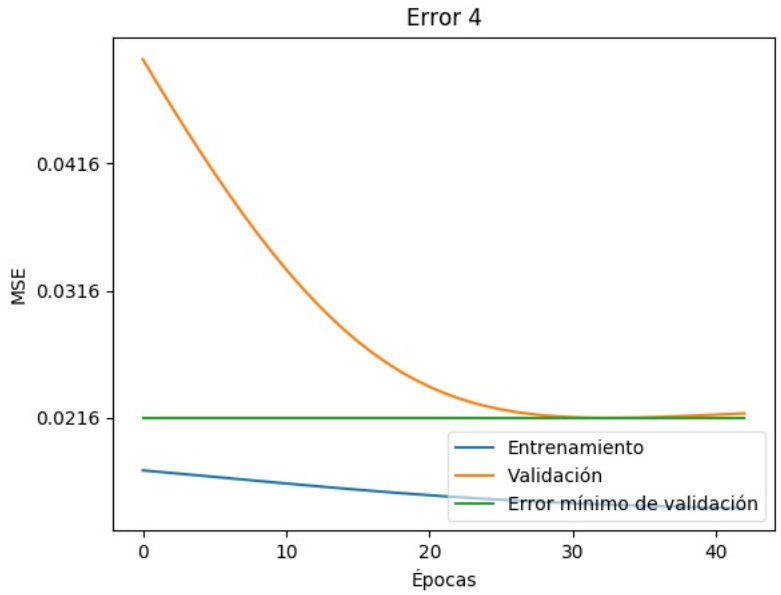

Figura 4: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 43 épocas usadas y cuatro neuronas en la capa oculta.

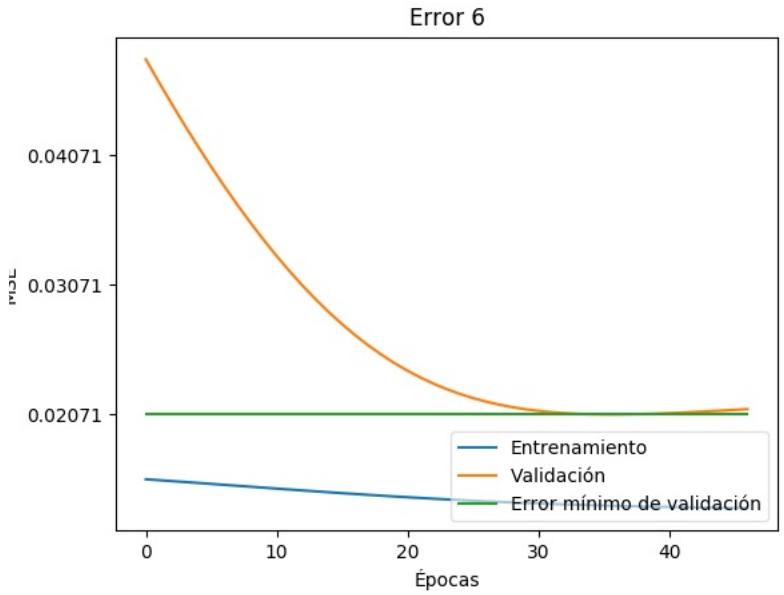

Figura 5: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 47 épocas usadas y seis neuronas en la capa oculta.

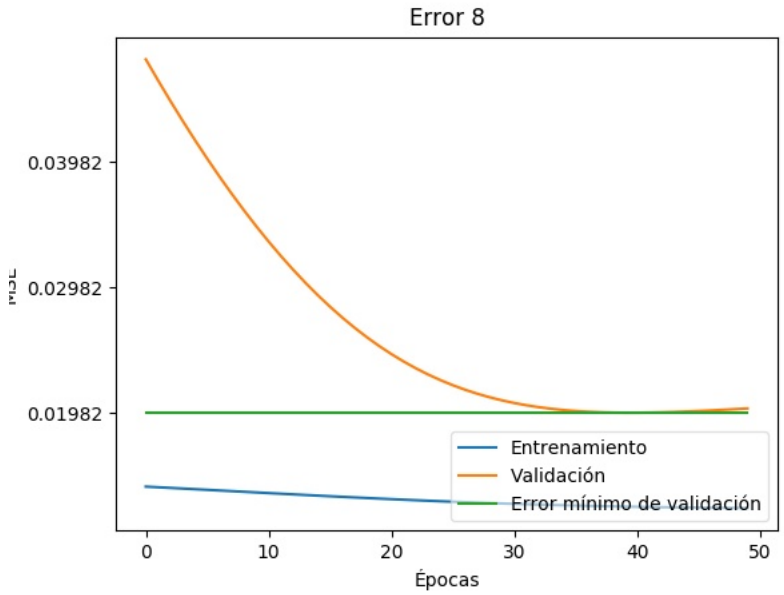

Figura 6: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 50 épocas usadas y ocho neuronas en la capa oculta.

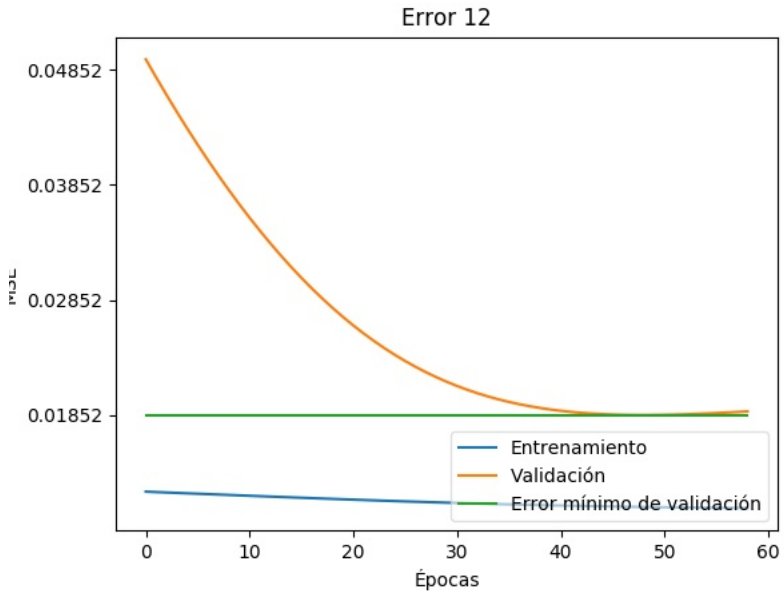

Figura 7: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 59 épocas usadas y 12 neuronas en la capa oculta.

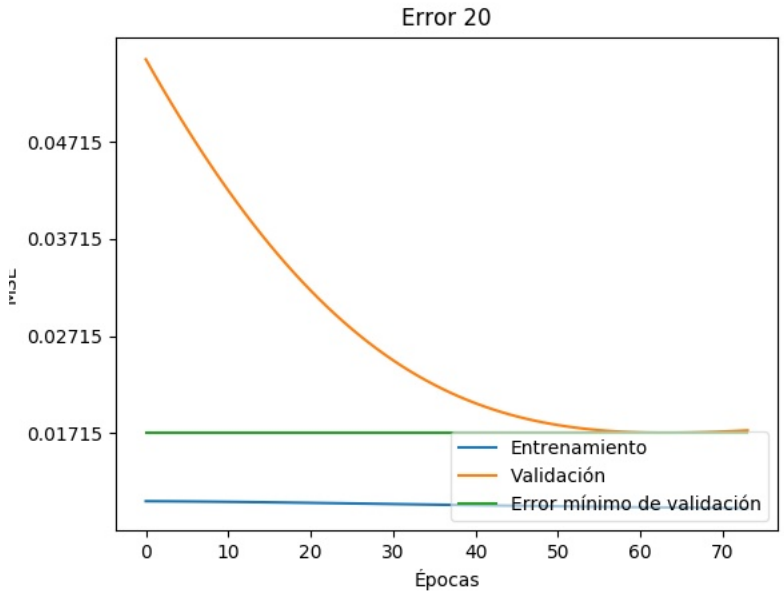

Figura 8: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 74 épocas usadas y 20 neuronas en la capa oculta.

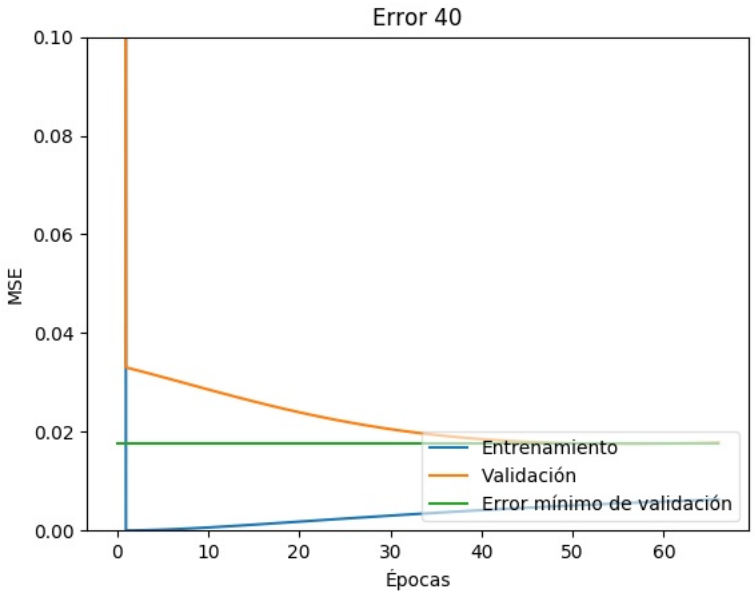

Figura 9: Error cuadrático medio del MLP con tasa de aprendizaje 0.1, tangente hiperbólica, 67 épocas usadas y 40 neuronas en la capa oculta.

De todas las gráficas obtenidas, el mejor error de validación se obtiene cuando la capa oculta tiene 20 neuronas.

c) A continuación se muestran los gráficos solicitados (los gráficos de error ya se han mostrado arriba):

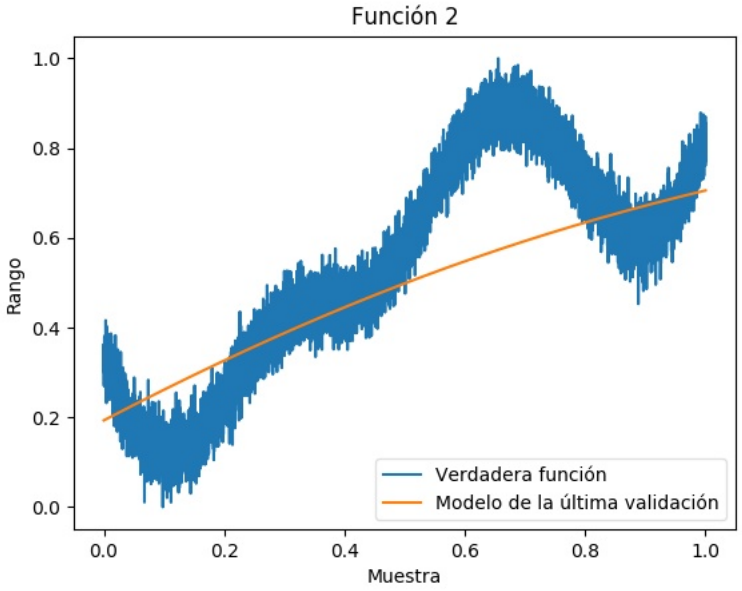

Figura 10: Datos de validación y patrón aprendido con una tasa de aprendizaje de 0.1, 41 épocas usadas, tangente hiperbólica y dos neuronas en la capa oculta. El error está en la figura 2

.

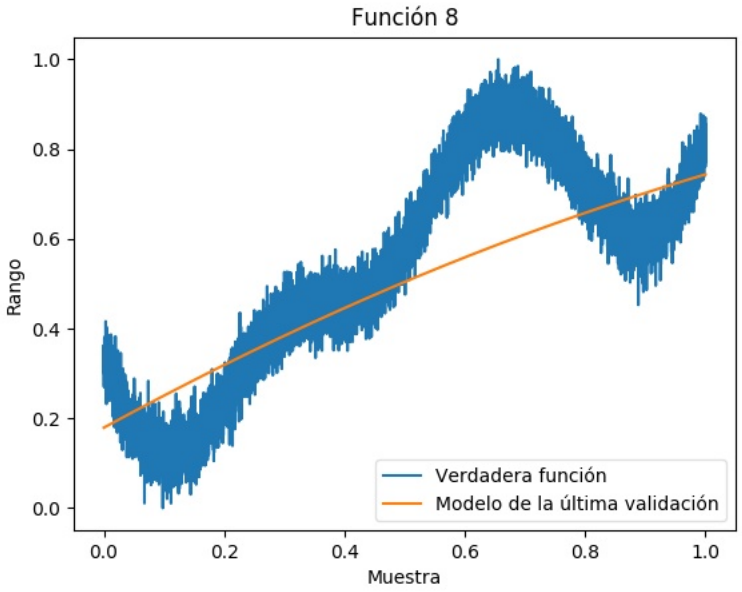

Figura 11: Datos de validación y patrón aprendido con una tasa de aprendizaje de 0.1, 50 épocas usadas, tangente hiperbólica y ocho neuronas en la capa oculta. El error está en la figura 6.

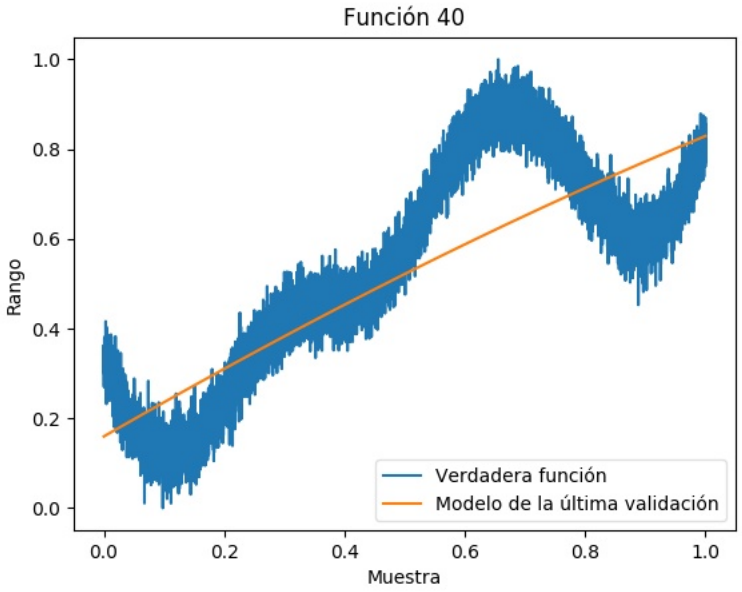

Figura 12: Datos de validación y patrón aprendido con una tasa de aprendizaje de 0.1, 67 épocas usadas, tangente hiperbólica y 40 neuronas en la capa oculta. El error está en la figura 9.

Al comparar los gráficos de error, observamos que el error mínimo de validación es decre-

ciente. Sin embargo, para $n = 40$, éste aumenta ligeramente con respecto a $n = 20$. Para una misma gráfica de errores, el error de entrenamiento siempre es decreciente, mientras que el error de validación es decreciente hasta cierto punto. Como se implementó el criterio de parada temprada para el error de validación, se puede asumir que para cada gráfica de error, si hubiera corrido el algoritmo durante más épocas, éste error hubiera aumentado.

d) Para este apartado, se construyó un *ADALINE* que realiza interpolación polinómica. Se usaron 700 épocas con un criterio de parada temprana. También se usó una tasa de aprendizaje de 0,01. A continuación sólo se muestra el gráfico del polinomio obtenido con menor error (los demás se encuentran en la carpeta):

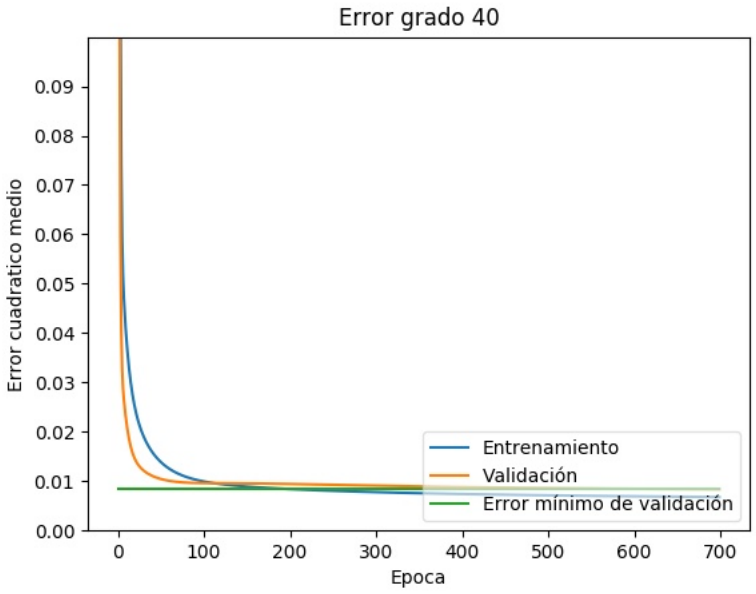

Figura 13: Error de entrenamiento y validación con una tasa de aprendizaje de 0.01, 700 épocas usadas, función logística y 40 neuronas en la primera capa.

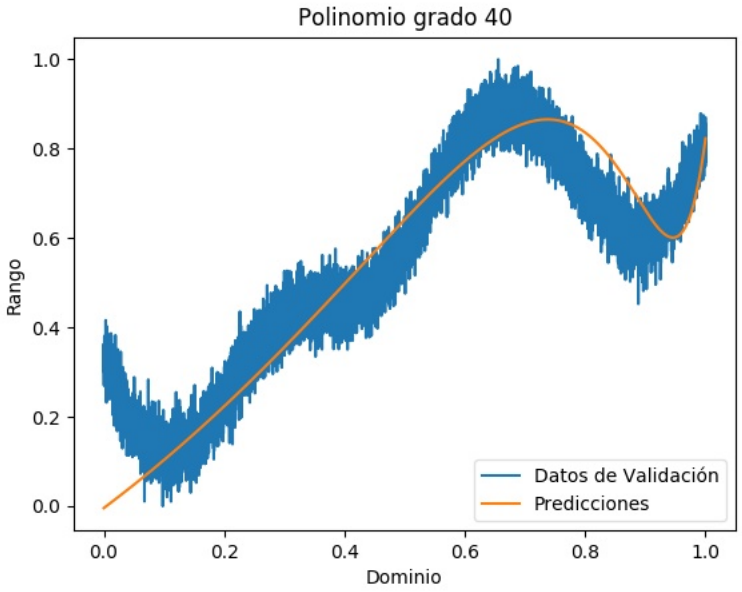

Figura 14: Datos de validación y patrón aprendido con una tasa de aprendizaje de 0.01, 700 épocas usadas, función logística y 40 neuronas en la primera capa.

2. Los datos rabbit.mat muestran la dependencia del peso de la retinas de ciertos conejos australianos en función de su edad. Se tiene que un modelo no lineal encontrado usando mínimos cuadrados sobre los datos, está descrita por la ecuación:

$$Y = 233{,}846(1 - exp(-0{,}00604x)) + error \tag{1}$$

Diseñe e implemente su propio perceptrón multicapas que aproxime los datos (no utilice el toolbox de Matlab). Justifique su respuesta (criterio de parada, arquitectura de red, tasas de aprendizaje, algoritmo de aprendizaje, etc.) y aproximación usando todos los recursos disponibles para argumentar su respuesta.

Solución:

Para resolver este problema, se usó el algoritmo de perceptrón multicapa implementado para la primera pregunta [1]. Sin embargo, en este caso se diseñó una arquitectura que consiste de una capa inicial con una neurona, dos capas ocultas con dos neuronas todas y una capa de salida con una neurona. Ambas capas ocultas tienen una función de activación logística y la capa final ha de tener una lineal por tratarse de un problema de regresión. Para el entrenamiento se usó una tasa de aprendizaje de 0.1 y se necesitaron 1000 épocas. Los datos fueron escalados al intervalo comprendido entre 0 y 1. Se usó el algoritmo de descenso

del gradiente. Para obtener un conjunto de validación se tomó una muestra aleatoria del 20 % de los datos originales.

El patrón aprendido por el modelo diseñado para esta pregunta se puede encontrar en la figura 15. Simplemente se fue tanteando hasta encontrar una arquitectura que aproximara bien los datos.

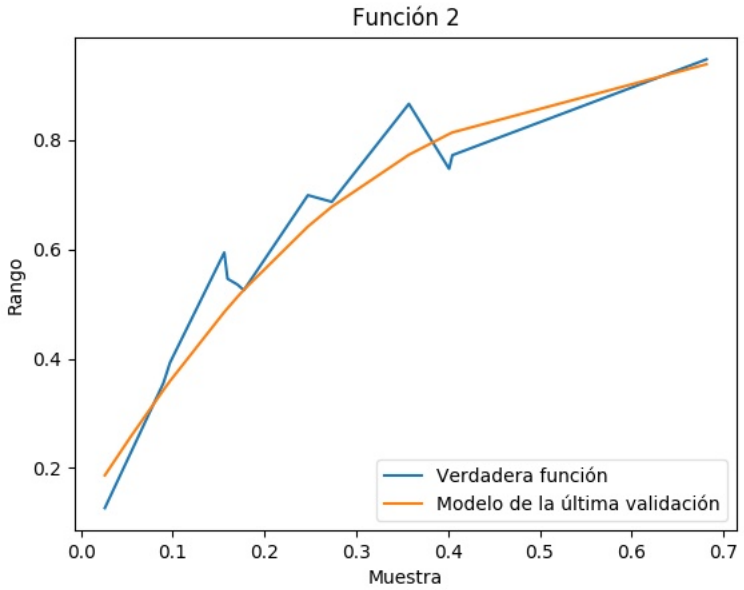

Figura 15: Datos normalizados (muestra aleatoria del 20 %) en azul y en naranja el patrón aprendido por el perceptrón multicapa. Se usaron dos capas ocultas de dos neuronas y con funciones logísticas cada una. La capa final es de una neurona con una función lineal. Tasa de aprendizaje igual a 0.1 y entrenada por 1000 épocas.

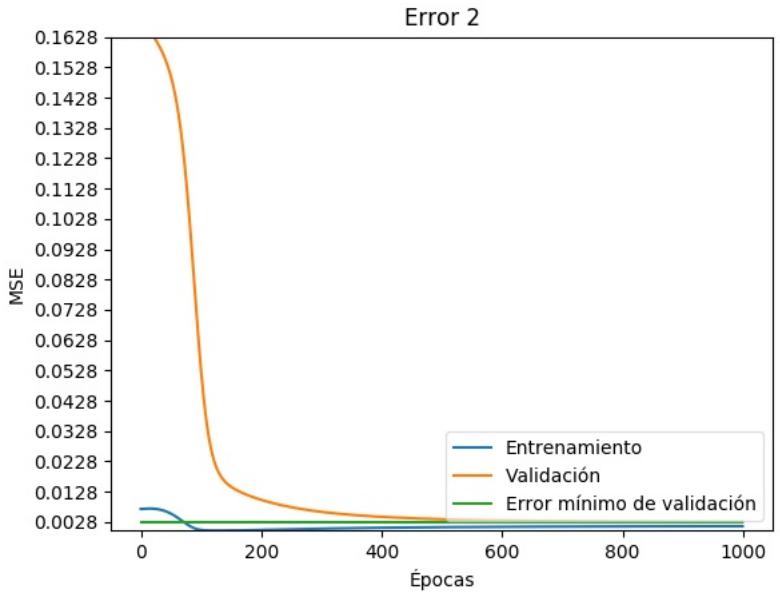

Figura 16: Error de entrenamiento y validación de la figura 15

3. Sean $x \in \mathbb{R}^d$ los patrones, $w \in \mathbb{R}^d$ el vector de pesos, t la respuesta deseada (target), $\lambda > 0$ una constante de regularización y m el número total de datos. Sea $R(w)$ la función de costos regularizada.

$$R(w) = \frac{1}{m} \sum_{i=1}^{m} l(t_i - \langle w, x_i \rangle) + \frac{\lambda}{2} \|w\|^2 \tag{2}$$

con

$$l(\xi) = \left\{ \begin{array}{ll} \frac{1}{2}\xi^2, & \text{for } |\xi| < 1 \\ |\xi| - \frac{1}{2}, & \text{en caso contrario} \end{array} \right\}$$

A Calcule el gradiente (en lote) de $R(w)$ con respecto a w.

B Escriba el algoritmo de descenso de gradiente estocástico para minimizar $R(w)$.

Solución:

a) El gradiente en lote para la función de costos es el siguiente:

Como $\frac{\partial \lambda \|w\|^2}{\partial w_j} = \lambda w_j$ y por la regla de la cadena, se tiene lo siguiente:

$$\frac{\partial R(\vec{w})}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} l'(t_i - \langle w, x_i \rangle))(-x_j) + \lambda w_j$$

donde

$$l'(\xi) = \left\{ \begin{array}{ll} \xi, & \text{for } |\xi| < 1 \\ sign(\xi), & \text{en caso contrario} \end{array} \right\}$$

Podemos demostrar la diferenciabilidad por partes de la función $l(\xi)$.

Por la definición de l y debido a que h tiende a cero desde la derecha, se tiene que:

$$\lim_{h \to 0^+} \frac{l(-1+h) - l(-1)}{h} = \lim_{h \to 0^+} \frac{\frac{1}{2}(-1+h)^2 - (|-1| - \frac{1}{2})}{h}$$

Desarrollando esta expresión, llegamos a :

$$\lim_{h \to 0^+} \frac{\frac{1}{2} - h + \frac{1}{2}h^2 - 1 + \frac{1}{2}}{h} = \lim_{h \to 0^+} \frac{-h + \frac{1}{2}h^2}{h}$$

Por lo que el límite es:

$$\lim_{h \to 0^+} -1 + \frac{1}{2}h^2 = -1$$

Ahora veamos el límite por la izquierda.

Por la definición de l y debido a que h tiende a cero desde la izquierda, se tiene que:

$$\lim_{h \to 0^-} \frac{l(-1+h) - l(-1)}{h} = \lim_{h \to 0^-} \frac{(|-1+h| - \frac{1}{2}) - (|-1| - \frac{1}{2})}{h}$$

Si simplificamos y tomamos en cuenta que $-1+h$ es negativo debido a que h tiende a cero por la izquierda, tenemos que:

$$\lim_{h \to 0^-} \frac{|-1+h| - 1}{h} = \lim_{h \to 0^-} \frac{1-h-1}{h}$$

Finalmente tenemos que el límite es:

$$\lim_{h \to 0^-} \frac{1-h-1}{h} = -1$$

Como se verificó que ambos límites tienden al mismo valor, se tiene que la función $l(\xi)$ es diferenciable para $\xi = -1$.

La verificación de diferenciabilidad por partes para $\xi = 1$ es similar y dicho límite es igual a 1 cuando h tiende a cero por la izquierda y cuando tiende a cero por la derecha.

b) Se obtuvo la siguiente instancia del algoritmo de descenso del gradiente estocástico [2] para esta función de pérdida. En lugar de sumar los errores para todas las instancias de entrada, éste algoritmo sólo toma una instancia a la vez. Por eso no se muestra una sumatoria como en el caso del gradiente por lotes.

Algorithm 1 Instancia del Algoritmo de Descenso de Gradiente Estocástico

procedure DGE
  $\vec{w} := \vec{w_0}$
  $\eta := \eta_0$
  $\lambda := \lambda_0$
  repeat
    for Cada instancia $\vec{x_i}$ con su salida t_i **do**
      if $|t_i - \langle \vec{w}, \vec{x_i} \rangle| < 1$ **then**
        e := $t_i - \langle \vec{w}, \vec{x_i} \rangle$
      else
        e := $sign(t_i - \langle \vec{w}, \vec{x_i} \rangle)$
      for Cada componente j de $\vec{w}$ **do**
        $\frac{\partial E}{\partial w_j} := \lambda w_j - x_{i,j} e$
      $\vec{w} := \vec{w} - \eta \nabla E(\vec{w})$
  until Condición de Parada

Referencias

[1] Simon Haykin, *Neural networks and learning machines, 3rd ed.* Upper Saddle River: Pearson Education, 2009. pp. 129-141.

[2] Christopher M. Bishop, *Pattern Recognition and Machine Learning, 2nd ed.* Singapore: Springer, 2006. pp. 240-241.