Instituto Politécnico Nacional

Escuela Superior de Cómputo

Neural Network

3CM1

Práctica 4: Red ADALINE

Hernández López Ángel Zait

2014080682

Ciclo escolar: 2019/2

29 de abril de 2019

Introducción.

Dentro de las redes neuronales, como hemos visto, existen dos tipos de modos, el modo regresor y el modo clasificador, el modo regresor, en resumen, hace un acercamiento a los valores que se tienen de un dataset, y el modo clasificador separa una serie de datos que se le da de inicio, un ejemplo es que dada una imagen, la red neuronal sepa dividir varios objetos, como entre que es una mesa y que es una silla.

Existen también varias funciones de transferencia, los cuales pueden ser lineales o no lineales, dependiendo el problema que uno desee resolver. En ocasiones es mejor tener una función lineal, ya que es más sencillo el cálculo del vector de salida de una RNA, ya que no usa tantos cálculos matemáticos.

Antes de entrar a saber que es una red ADALINE, debemos saber que es un perceptrón, ya que tienen un poco de relación. El perceptrón es un sistema de aprendizaje basado en ejemplos capaz de realizar tareas de clasificación, una de las características de clasificador del perceptrón es que la capa de salida es que la salida la codifica una función de salida en escalón, que transforma la suma ponderada de las entradas que es un valor real, en una salida binaria.

El detalle es que la regla de aprendizaje del perceptrón no permite producir salidas reales. Esta regla es fundamentalmente una regla de aprendizaje por refuerzo en la que se potencia las salidas correctas y no se tienen en cuenta las incorrectas. Por esto, la regla de aprendizaje del perceptrón y de la red ADALINE es diferente y en ADALINE se le agrega el factor de aprendizaje y a su vez, el valor del error no solamente es $e = t - a$ como en el perceptrón, ya que se le agrega otro dato que nos va a ayudar a que el error por época haga que se acerque poco a poco a cero y que al momento de usar el modo regresor, este se acerque poco a poco al dato cercano.

Marco teórico.

La red ADALINE (Adaptive Linear Neuron) fue propuesta en 1960 por Widrow y Hoff; esta red toma en cuenta la señal del error para el aprendizaje, que es la diferencia entre el valor deseado y el valor de salida de la red neuronal, y a su vez, se eleva al cuadrado la diferencia, por lo que la ecuación del error queda de la siguiente forma:

$$e = (t - a)^2 \dots (1)$$

- Arquitectura

A continuación, veremos la arquitectura de la red ADALINE (Figura 1), la cual tiene un parecido a la red del perceptrón simple, la diferencia es que ADALINE solo tiene una neurona, y el perceptrón, puede tener varias.

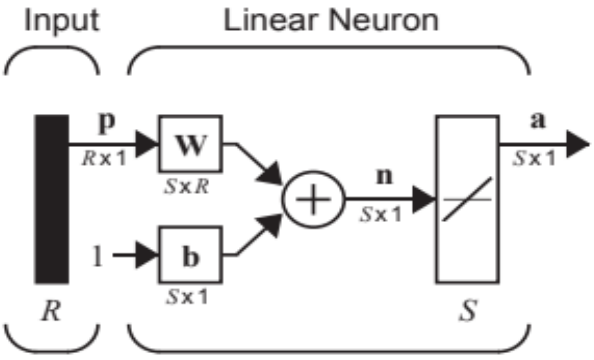

Figura 1. Arquitectura ADALINE

Otra característica de esta RNA es que la función de transferencia es línea, por lo que la función a utilizar aquí es 'Linear' y la ecuación de la salida de la neurona queda de la siguiente forma:

$$a = purelin(Wp + b) \dots (2)$$

- Regla de aprendizaje.

Esta regla actualiza automáticamente los valores de W y b usando una técnica llamada regla delta, que consiste en calcular las derivadas parciales de cada parámetro de la RNA con respecto a la señal del error. Para esta regla, se parte de la siguiente ecuación:

$$W_i(k+1) = W_i(k) - \Delta W_i \dots (3)$$

Para obtener de manera explícita una expresión para ΔW_i, observe que en la ecuación del error no aparece de forma explícita el término W_i, por lo que se usará la regla de la cadena para obtener la expresión deseada. El diseño de una regla de aprendizaje consiste en proponer una forma para el ΔW_i. En el caso de ADALINE, el ΔW_i Hace uso de la regla delta, por lo que tenemos lo siguiente:

$$\Delta W_i = -\alpha \frac{\partial e}{\partial W_i} \dots (4)$$

Donde α es el factor de aprendizaje, que toma valores de entre 0 y 1, este valor es importante ya que permite regular el tamaño del incremento o decremento que se aplicará a cada peso sináptico. Se busca evitar que estos valores diverjan u oscilen demasiado. Con ayuda de la regla de la cadena en la ecuación (4), podemos definir la regla de aprendizaje de la siguiente forma:

$$W(k+1) = W_i(k) - 2\alpha(t-a)p^T \dots (5)$$

$$b(k+1) = b(k) - 2\alpha(t-a) \dots (6)$$

Dado que ADALINE utiliza purelin como función de transferencia es adecuada para usarse como regresor o clasificador.

- Proceso de aprendizaje:

1. Dado un conjunto de entrenamiento. Se define la arquitectura y el modelo matemático de la RNA.
2. Se pide al usuario los siguientes valores
   i. epochmax: Se refiere al número máximo de épocas a realizar.
   ii. eepoch: Es un valor pequeño al cual se desea que llegue el valor de la señal del error ($e \to 0$).
   iii. El valor del factor del aprendizaje, α.
3. Se inicializa aleatoriamente entre -1 y 1 los valores W y b.
4. Se inicia una época de aprendizaje, en la cual se propaga hacia adelante cada uno de los datos del conjunto de entrenamiento. Se toma el primer dato, se aplica como entrada a la RNA y se obtiene el valor de salida (a), después se calcula el error y se aplican las reglas de aprendizaje.
5. Una vez terminada la época en el paso 4, se calcula el error de época con la siguiente expresión:

   $$Eepoch = \frac{1}{N}\sum_{j=1}^{N} e_j \dots (7)$$

   Es decir, un promedio de los errores por cada época pasada.
6. Se verifica si se cumple alguno de los siguientes criterios de finalización, si no se regresa al paso 4:
   i. Eepoch=0. Todos los datos estén bien clasificados.
   ii. Cuando Eepoch<eepoch (Valor dado por el usuario, Te^-3). Se hace una última época para verificar que efectivamente todos los datos están bien clasificados.
   iii. eepochmax<epocas. Solo se usa para detener la ejecución del programa.

Y con estos pasos, podemos desarrollar una red ADALINE como función de transferencia ‘Linear’ con una sola neurona.

Diagrama de flujo.

Ahora veremos el diagrama de flujo de la red ADALINE que se describen en la diagrama 1, 2 y 3. Primero que nada se pide al usuario varios valores, los cuales son el nombre del archivo donde se encuentra el dataset junto con la extensión, además se le pide el número de épocas máximas que el programa irá desarrollando, en seguida el error mínimo que queremos y por último el factor de aprendizaje 'alpha', como ADALINE puede ser de modo regresor y clasificador, le pedimos al usuario que ingrese un '1' si quiere clasificar o '2' si lo quiere usar en modo regresor. Después de eso, el programa calcula y obtienen el valor de p, t y s, dependiendo la opción elegida, como se muestra en el diagrama 1, y en seguida, crea los valores de w y b; una vez creadas este crea los documentos donde se estarán guardando los valores de los pesos, bias y error, y los guarda.

Después de eso (diagrama 2), empezamos con el aprendizaje donde inicializamos los valores i, épocas y flag, que nos ayudarán a saber cuándo parar el aprendizaje. Después entra a un siclo, donde se obtiene el primer valor de p, para poder ingresarlo a la función de transferencia; una vez obtenido el valor de la salida, se calcula el error, que en este caso está expresada como $e = t - a$, para después guardarla como el valor absoluto de él. Después se hace la regla de aprendizaje, y se guardan los nuevos valores de los pesos y bias; luego preguntamos si 'i' es igual a 'pu' donde 'pu' es la cantidad de 'p' que tenemos. Si esto es falso, seguirá con el recorrido del programa, pero si es verdadero, se calculará el valor de Eepoch, y se guardará, en seguida preguntaremos por los criterios de finalización, y de pendiendo de ello, actuará el programa.

La última fase, que se encuentra en el diagrama 3 es la impresión de los valores de los pesos sinápticos y del bias, y después se grafican los valores de los pesos y de los bias, es decir, la evolución de ellos, al igual del error por época. Si el modo seleccionado fue clasificador, este también grafica los puntos involucrados, el vector de pesos y la frontera de decisión y termina el programa.

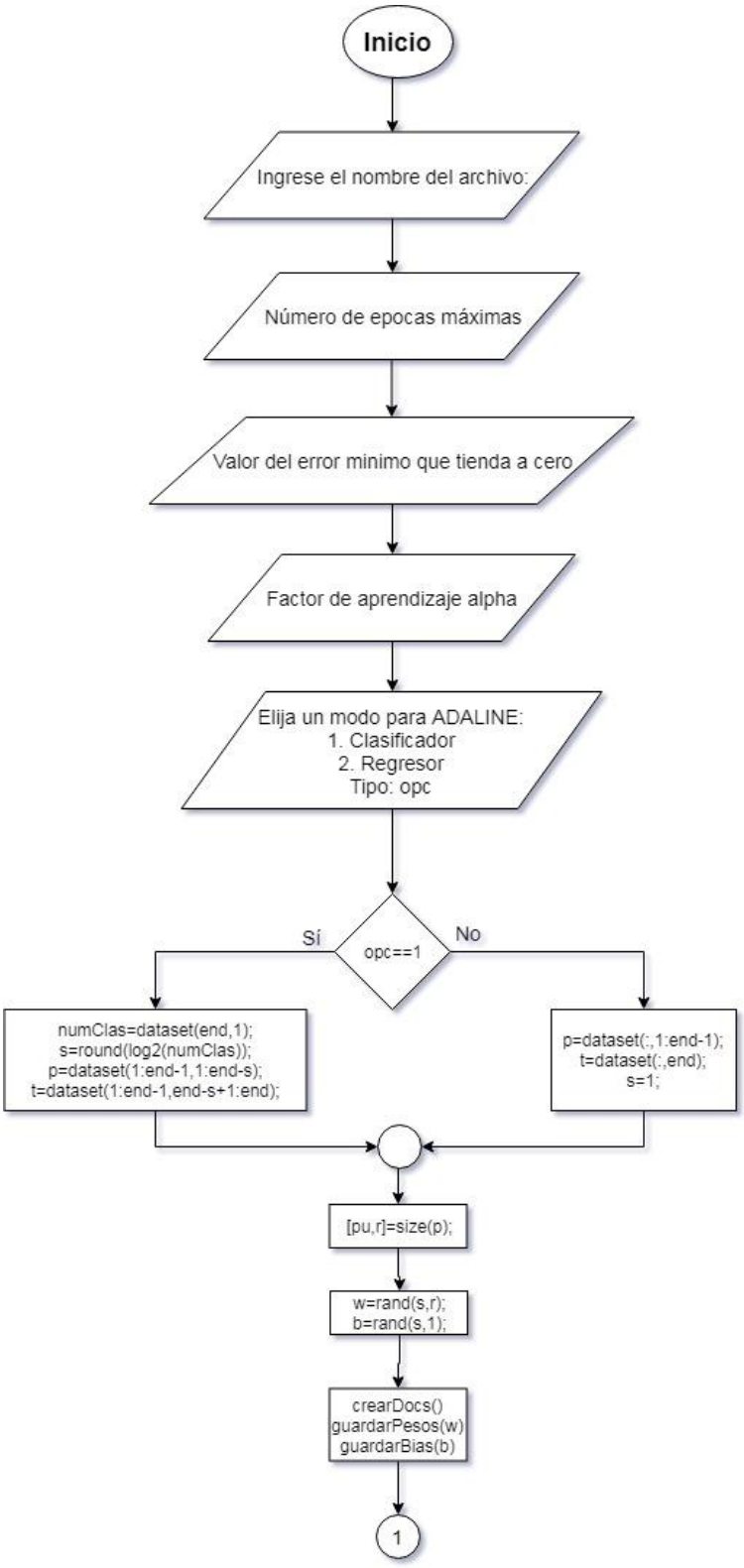

Diagrama 1. Ingreso de datos

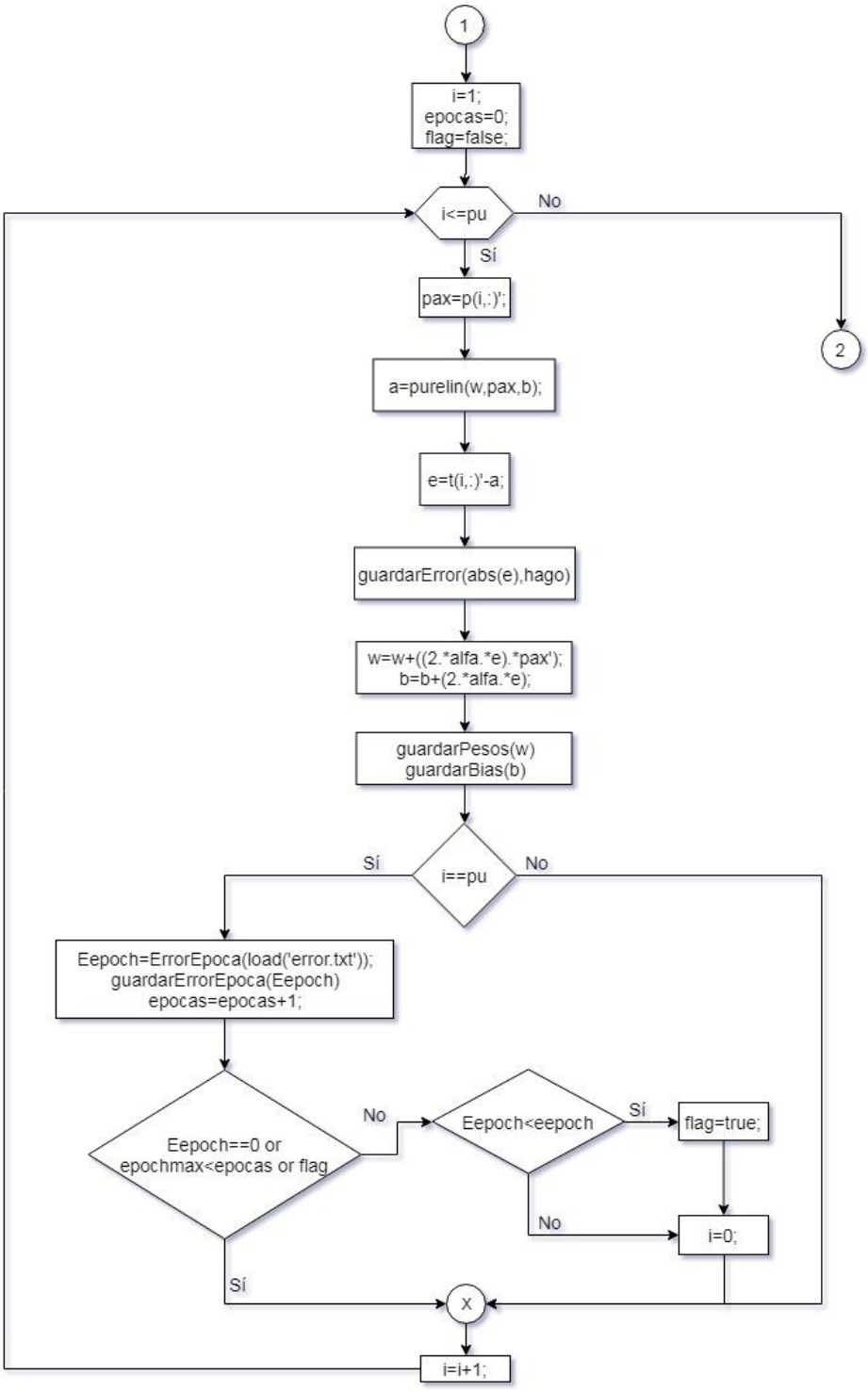

Diagrama 2. Aprendizaje

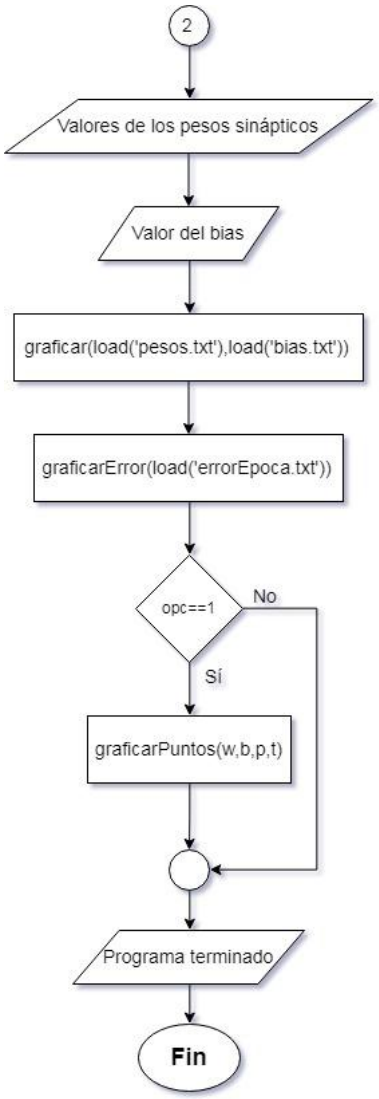

Diagrama 3. Graficado

Experimentos

Como notamos, ADALINE tiene modo regresor y clasificador, a continuación se presentaran experimentos realizados con el modo regresor con un dataset de 7 bits, es decir, el código binario del número 0 al 128 en decimal, por lo que quedó de la siguiente manera:

```
Nombre del archivo donde se encuentra el dataset: sieteBits.txt
Numero de epocas máxima: 100
Valor del error minimo que tienda a cero: 0.01
Factor de aprendizaje alpha: 0.01
Elija un modo para ADALINE:
      1. Clasificador
      2. Regresor
Tipo: 2
Valores de los pesos sinapticos:
   63.9873   31.9935   15.9951    7.9958    3.9962    1.9964    0.9965
```

```
Valor del bias:
    0.0326
Programa terminado
```

Como podemos notar, debemos tener un valor muy pequeño para el error y para el factor de aprendizaje para que pueda hacer convergencia, y se noten los cambios de los pesos, bias y el error (Figura 2 y 3 respectivamente).

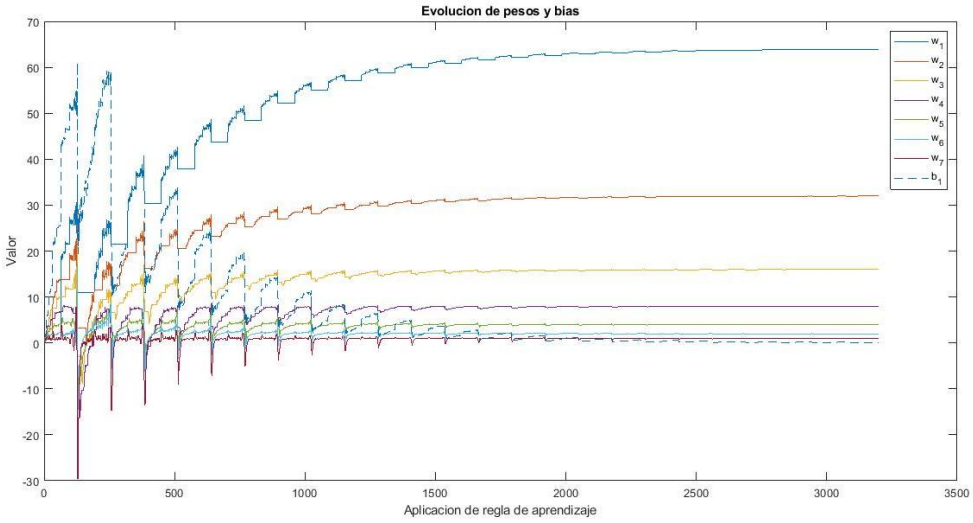

Figura 2. Evolución de los pesos y bias de 7 bits.

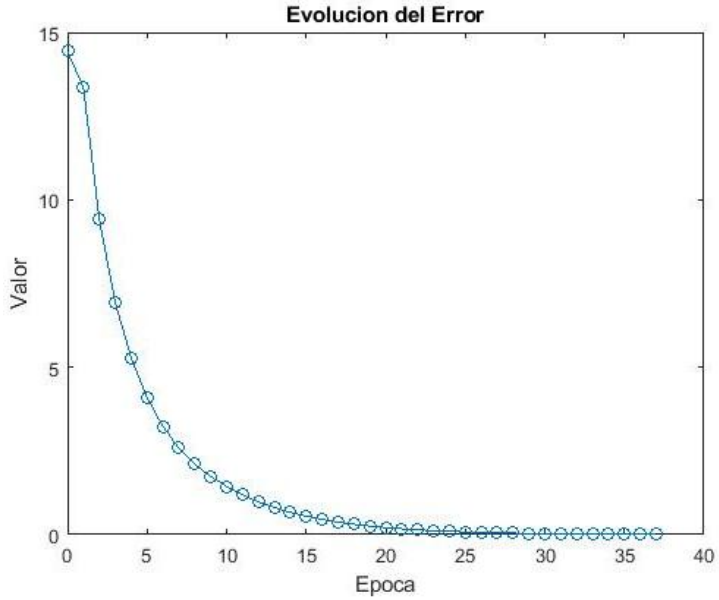

Figura 3. Evolución del error de 7 bits.

Pero si nosotros tenemos un valor muy grande para el factor de aprendizaje, este no converge y puede que no sé llegue a ningún resultado, como se muestran en la figura 4 y 5.

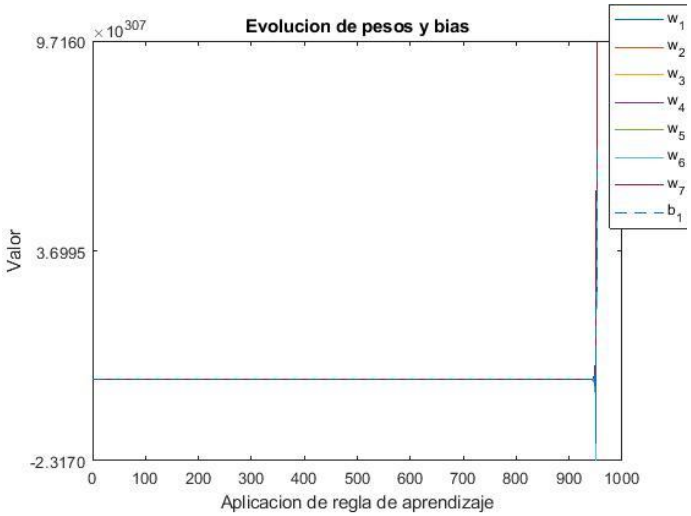

Figura 4. Evolución de los pesos y bias de 7 bits erróneo.

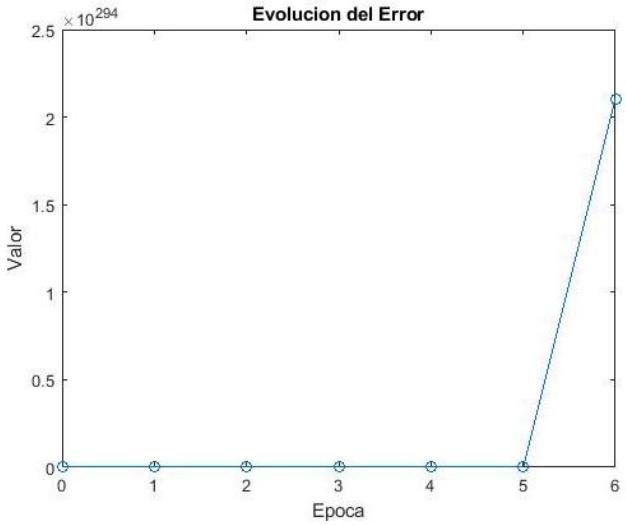

Figura 5. Evolución del error de 7 bits erróneo.

Ahora intentaremos con otro dataset, en este caso será de 14 bits, que puede tomar más tiempo y la computadora puede que tarde más en hacer el proceso.

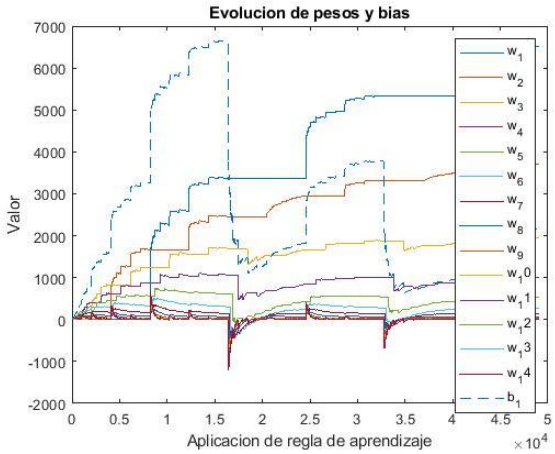

Figura 6. Evolución de los pesos y bias de 14 bits erróneo.

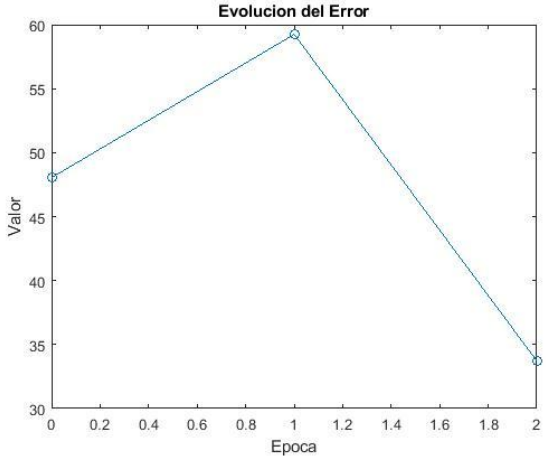

Figura 7. Evolución del error de 14 bits erróneo.

Como la es muy tardado y son demasiados datos en los cuales la regla de aprendizaje se va haciendo más y más grande también, y que con tantas operaciones y tantos datos se hace más lento y puede que marque un error, en este caso, solamente se hizo con dos épocas (Figura 7) con la respectiva evolución en pesos y bias, que como lo notamos en la figura 6, con solo una época se hacen miles de cálculos y realización de la regla de aprendizaje.

Para el modo clasificador, se hizo la prueba con dos clases, el cual la primer clase la conforman tres puntos y la clase dos solamente un punto.

```
Nombre del archivo donde se encuentra el dataset: dosClases.txt
Numero de epocas máxima: 100
Valor del error minimo que tienda a cero: 0.1
Factor de aprendizaje alpha: 0.01
Elija un modo para ADALINE:
      1. Clasificador
      2. Regresor
Tipo: 1
Valores de los pesos sinapticos:
```

```
   0.7561    0.7550
Valor del bias:
   -1.1943
Programa terminado
```

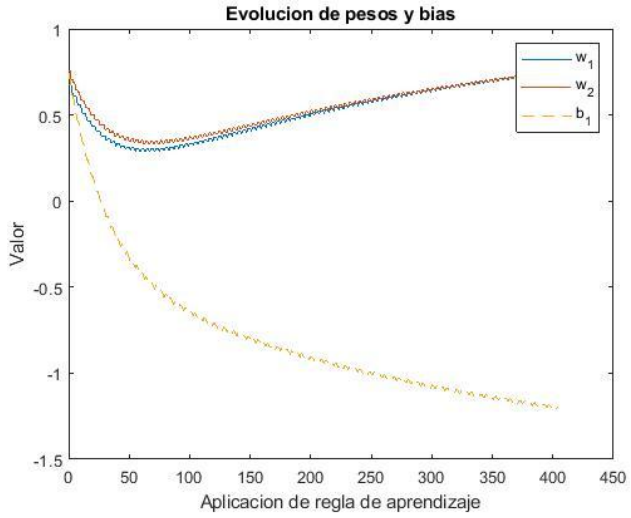

Figura 8. Evolución de pesos y bias de dos clases.

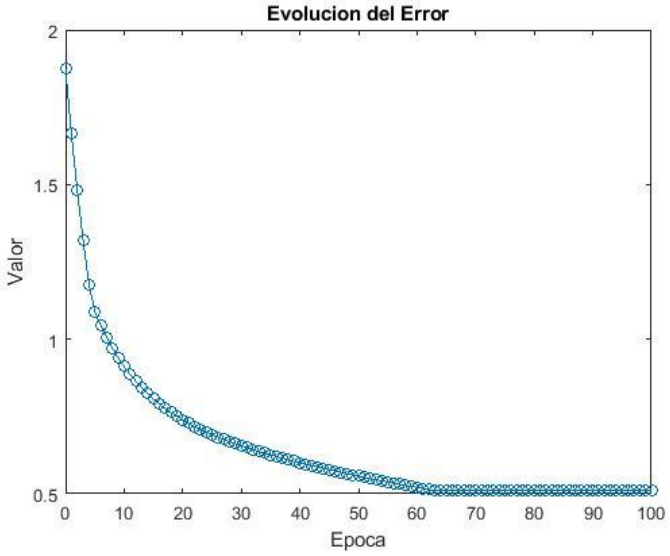

Figura 9. Evolución de error de dos clases.

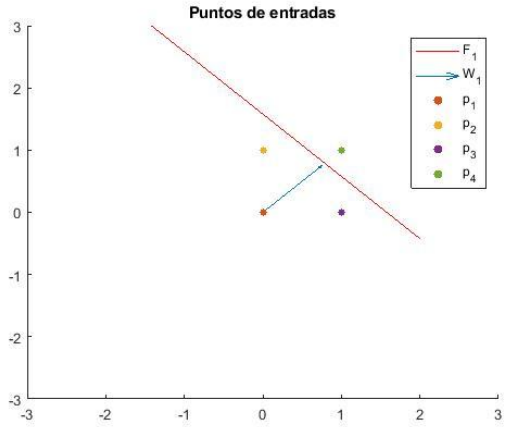

Figura 10. Puntos de dos clases.

Como notamos en las figuras 8 y 9, a pesar de que no convergió el programa, los valores de pesos y bias pueden clasificar los puntos, pero con un error con valor muy alto, ya que a pesar de que se haga la regla de aprendizaje, este llega un punto en el que se queda constante. A su vez en la figura 10 podemos observar los puntos y la frontera de decisión.

Ahora se probará al modo clasificador pero con cuatro clases, y cada clase tendrá dos puntos, entonces el experimento queda de la siguiente forma:

```
Nombre del archivo donde se encuentra el dataset: cuatroClases.txt
Numero de epocas máxima: 100
Valor del error minimo que tienda a cero: 0.1
Factor de aprendizaje alpha: 0.01
Elija un modo para ADALINE:
      1. Clasificador
      2. Regresor
Tipo: 1
Valores de los pesos sinapticos:
   -0.2917   -0.0193
    0.0903   -0.3295
Valor del bias:
    0.5076
    0.5902
Programa terminado
```

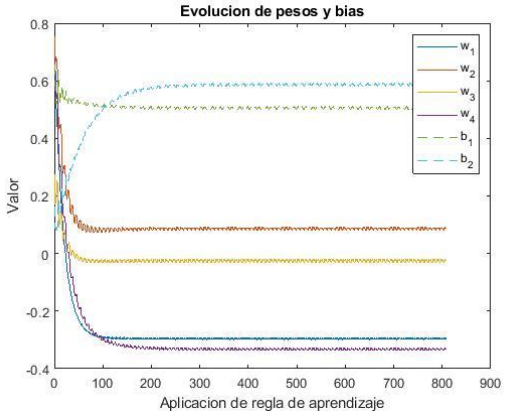

Figura 11. Evolución de pesos y bias de cuatro clases.

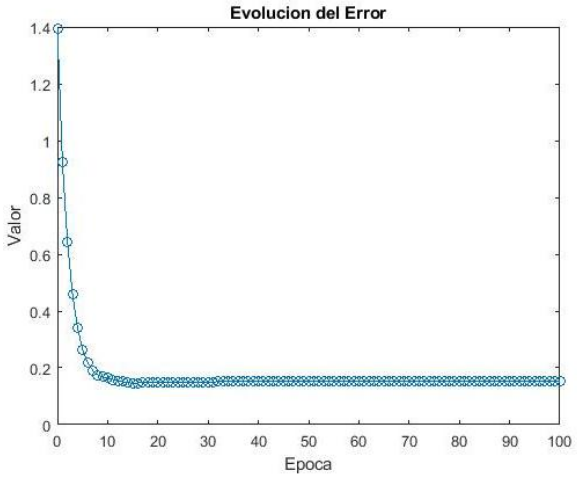

Figura 12. Evolución de error de cuatro clases.

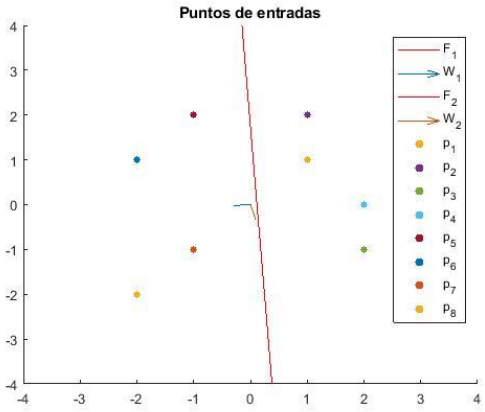

Figura 13. Puntos de cuatro clases.

Como en el ejemplo anterior, llega un momento en el que el error (Figura 12) es constante, al igual que la evolución de pesos y bias (Figura 11) con los puntos y las fronteras de decisión (Figura 13).

Discusión

Para el desarrollo de la práctica para poder tener mejor control del error se decidió que se tomara el valor absoluto, ya que puede que con el valor de Eepoch puede que se haga más grande de lo que queremos y que converja mucho más lento. Además, de que para tener una clasificación de n clases, también debemos ver que se debe aumentar el número de neuronas, ya que una de las características de esta RNA es que cuente con una sola neurona. La ventaja de esta RNA es que podemos usar el modo regresor y el modo clasificador con la misma función de transferencia que es 'Linear', ya que el valor que de la salida es el valor de 'n'.

Otra de las cuestiones es que puede que tarde demasiado en converger, ya que si queremos que el error se acerque demasiado al valor de cero, debemos saber un buen valor para el factor de aprendizaje porque si es que le damos un valor muy grande, puede que el error en lugar de hacerse más pequeño, puede que aumente, por eso es muy importante este valor.

Conclusión

La red ADALINE es una RNA que puede actuar de modo regresor y clasificador, dependiendo e problema que le ingresemos, y que tiene una regla de aprendizaje muy diferente a lo que es el perceptrón de múltiples entradas, ya que este tiene que tomar un factor de aprendizaje, el cual si es mucho más grande, puede que no llegue la convergencia del programa y el error en lugar de disminuir, crezca por los valores que va obteniendo por la suma. También se debe tener mucho cuidado con el valor del error, porque mientras más pequeño sea, más preciso es la aproximación, pero puede que tarde más tiempo y necesitemos más recursos para lograr eso, por eso es que se busca que tienda a cero y no sea el valor de cero.

Referencias.

T. Hagan, Martin (2014). Neural Network Design. Estados Unidos: eBook.

Isasi Viñuela, P; Galván León, I. M. (2004). Redes Neuronales Artificiales. Un enfoque práctico. Madrid: PEARSON.

Anexos.

- **P04_ADALINE_2014080682.m**

```
% Red ADALINE
% Hernández López Ángel Zait
clc; clear;

dtst=input('Nombre del archivo donde se encuentra el dataset: ','s');
dataset=load(dtst);
epochmax=input('Numero de epocas máxima: ');
eepoch=input('Valor del error minimo que tienda a cero: ');
alfa=input('Factor de aprendizaje alpha: ');
```

```
fprintf('Elija un modo para ADALINE:\n\t1. Clasificador\n\t2.
Regresor\n');
opc=input('Tipo: ');

% Calculo de s y obtención de t y p
if opc==1
    numClas=dataset(end,1);
    s=round(log2(numClas));
    p=dataset(1:end-1,1:end-s);
    t=dataset(1:end-1,end-s+1:end);
elseif opc==2
    p=dataset(:,1:end-1);
    t=dataset(:,end);
    s=1;
end

% Creación de los pesos y el bias
[pu,r]=size(p);
w=rand(s,r);
b=rand(s,1);

% Creación de documentos
crearDocs()
guardarPesos(w)
guardarBias(b)

% Inicio de las epocas
i=1;
epocas=0;
flag=false;
while i<=pu
    % Identificar si sobreescribira o creará un nuevo documento
    if i==1
        hago='w';
    else
        hago='a';
    end

    pax=p(i,:)';

    a=purelin(w,pax,b);    % Función purelin

    % Obtención del error
    e=t(i,:)'-a;
    guardarError(abs(e),hago)

    % Regla de aprendizaje
    w=w+((2.*alfa.*e).*pax');
    b=b+(2.*alfa.*e);

    % Guardado de los pesos y bias
    guardarPesos(w)
    guardarBias(b)

    if i==pu
```

```
        Eepoch=ErrorEpoca(load('error.txt'));   % Calculo de Eepoch
        guardarErrorEpoca(Eepoch)   % Guardado de Eepoch por epoca
        epocas=epocas+1;
        if (Eepoch==0 || epochmax<epocas || flag)   % Criterios de
finalización
            break
        elseif Eepoch<eepoch
            i=0; flag=true;
        else
            i=0;
        end
    end

    i=i+1;
end

disp('Valores de los pesos sinapticos:')
disp(w)

disp('Valor del bias:')
disp(b)

clf;
figure(1)
graficar(load('pesos.txt'),load('bias.txt'))    % Grafica de pesos y bias

figure(2)
graficarError(load('errorEpoca.txt'))   % Grafica de error por epoca

if opc==1
    figure(3)
    graficarPuntos(w,b,p,t);    % Grafica para los puntos
end

disp('Programa terminado')
```

- **crearDocs.m**

```
function crearDocs()
    vo=fopen('errorEpoca.txt','w'); % Crea documento para guardar los
    fclose(vo);                     % errores por epoca

    vo=fopen('bias.txt','w');   % Crea documento para guardar los valores
    fclose(vo);                 % de bias

    vo=fopen('pesos.txt','w');  % Crea documento para guardar los valores
    fclose(vo);                 % de los pesos sinapticos
end
```

- **ErrorEpoca.m**

```
function [ep]=ErrorEpoca(e)
    % Calcula el error por epoca, el cual es un promedio
    ep=(1/numel(e))*(sum(e));
end
```

- **frontera.m**

```
function frontera(b,p1,p2,max)
    % Grafica la frontera de decición
    yd=-b/p1;
    m=(-p2)/(-p1);
    m=-1/m;
    xa=max+1;
    xb=-max-1;
    ya=(m*xa)+yd;
    yb=(m*xb)+yd;
    x=[xa xb];
    y=[ya yb];

    line(x,y,'Color','r')
end
```

- **graficar.m**

```
function graficar(w,b)
    [f,c]=size(w);
    [fa,ca]=size(b);
    objeto="";
    % Grafica los valores de los pesos sinapticos
    for i=1:c
        plot(0:f-1,w(:,i));
        objeto(i)=strcat('w_',string(i));
        hold on
    end

    n=c+1;
    % Grafica los valores de bias
    for i=1:ca
        plot(0:fa-1,b(:,i),'--');
        objeto(n)=strcat('b_',string(i));
        n=n+1;
        hold on
    end

    % Crea una barra donde dice que color es cada peso sinaptico y bias
    legend(objeto);
    title('Evolucion de pesos y bias')
    xlabel('Aplicacion de regla de aprendizaje')
    ylabel('Valor')
end
```

- **graficarError.m**

```
function graficarError(e)
    % Grafica el error por epoca
    plot(0:numel(e)-1,e,'-o');
```

```
    title('Evolucion del Error')
    xlabel('Epoca')
    ylabel('Valor')
end
```

- **graficarPuntos.m**

```
function graficarPuntos(w,b,p,t)
    [f,c]=size(p);
    [f1,c1]=size(w);
    colores=['k','b','g','r','c','m','y'];
    ese=1;

    m1=max(w);
    m2=max(p);
    maxi=max(m1,m2);
    maxi=max(maxi);
    texto=""; n=1;

    % Grafica los puntos involucrados en el dataset
    for i=1:f1
        frontera(b(i),w(i,1),w(i,end),maxi);    % Grafica la frontera
        texto(n)=strcat('F_',string(i));
        n=n+1;
        hold on
        quiver(0,0,w(i,1),w(i,end),0,'filled')
        texto(n)=strcat('W_',string(i));
        n=n+1;
        hold on
    end

    for i=1:f
%         if ese~=t(i,end)
%             ese=ese+1;
%         end
        scatter(p(i,1),p(i,end),25,'filled')
        texto(n)=strcat('p_',string(i));
        n=n+1;
        hold on
    end

    axis([-maxi-2 maxi+2 -maxi-2 maxi+2])
    title('Puntos de entradas')
    legend(texto)
end
```

- **guardarBias.m**

```
function guardarBias(b)
    % Inserta el nuevo valor del bias
    vo=fopen('bias.txt','a');
    for i=1:numel(b)
        fprintf(vo,'%f ',b(i));
    end
    fprintf(vo,'\n');
    fclose(vo);
end
```

- **guardarError.m**

```
function guardarError(e,h)
    % Crea e inserta los valores de los errores de una epoca
    vo=fopen('error.txt',h);
    fprintf(vo,'%f\n',e);
    fclose(vo);
end
```

- **guardarErrorEpoca.m**

```
function guardarErrorEpoca(e)
    % Guarda el valor del error de una epoca
    vo=fopen('errorEpoca.txt','a');
    fprintf(vo,'%f\n',e);
    fclose(vo);
end
```

- **guardarPesos.m**

```
function guardarPesos(w)
    % Guarda el valor de los pesos
    vo=fopen('pesos.txt','a');
    for i=1:numel(w)
        fprintf(vo,'%f ',w(i));
    end
    fprintf(vo,'\n');
    fclose(vo);
end
```

- **purelin.m**

```
function [a]=purelin(w,p,b)
    % Hace la función purelin
    a=(w*p)+b;
end
```