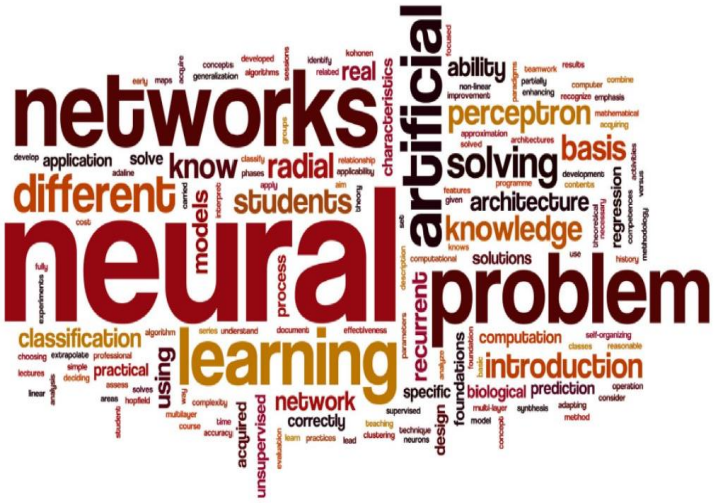
INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

NEURAL NETWORKS

PROF.: MARCO ANTONIO MORENO ARMENDÁRIZ
ALUMNO: ORTEGA VICTORIANO IVAN
No. DE LISTA: 29
GRUPO: 3CM2

"ENTRENAMIENTO DE UNA RED ADALINE"



INTRODUCCIÓN:

El adaline (de ADAptative LINear Element) es un tipo de red neuronal artificial desarrollada por el profesor Bernard Widrow y su alumno Ted Hoff en la Universidad de Stanford en 1960. [1] El modelo está basado en la Neurona de McCulloch-Pitts.

Con respecto al perceptrón el Adaline posee la ventaja de que su gráfica de error es un hiperparaboloide que posee o bien un único mínimo global, o bien una recta de infinitos mínimos, todos ellos globales. Esto evita la gran cantidad de problemas que da el perceptrón a la hora del entrenamiento debido a que su función de error (también llamada de coste) posee numerosos mínimos locales.

Algunas aplicaciones de la red Adaline son:

Asociación de patrones: se puede aplicar a este tipo de problemas siempre que los patrones sean linealmente separables.

En el campo del procesamiento de señales:

Filtros de ruido: Limpiar ruido de señales transmisoras de información.

Filtros adaptativos: Un adaline es capaz de predecir el valor de una señal en el instante t+1 si se conoce el valor de la misma en los p instantes anteriores (p es >0 y su valor depende del problema). El error de la predicción será mayor o menor según qué señal queramos predecir. Si la señal se corresponde a una serie temporal el Adaline, pasado un tiempo, será capaz de dar predicciones exactas.

Se pueden combinar múltiples Adalines formando lo que se denomina el Madaline.

MARCO TEÓRICO.

La estructura general de la red tipo Adaline puede visualizarse en la Fig. 1 [2]:

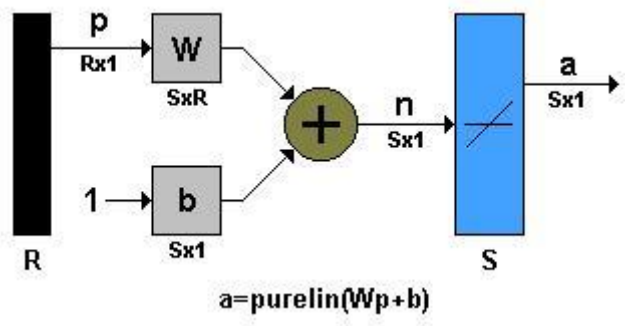


Figura 1: Estructura de una Red ADALINE

En donde:

p: Patrones de entrada

b: Umbrales de activación

a: Salida de la neurona

La salida de la red está dada por:

$$\boldsymbol{a} = pureline\,(\boldsymbol{W}\boldsymbol{p}^T + \boldsymbol{b}) = \boldsymbol{W}\boldsymbol{p}^T + \boldsymbol{b} \tag{2.8}$$

En similitud con el Perceptrón, el límite de la característica de decisión para la red Adaline se presenta cuando *n=0*, por lo tanto:

$$Wp^T+b=0 \tag{2.9}$$

En la siguiente figura, se específica la línea que separa en dos regiones el espacio de entrada, como se muestra en la siguiente figura:

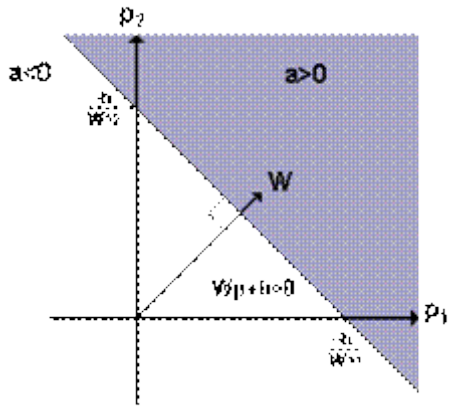


Figura 2: Característica de Decisión de una Red Tipo Adaline

La salida de la neurona es mayor que cero en el área gris, en el área blanca la salida es menor que cero. Como se mencionó anteriormente, la red Adaline puede clasificar correctamente patrones linealmente separables en dos categorías. [2]

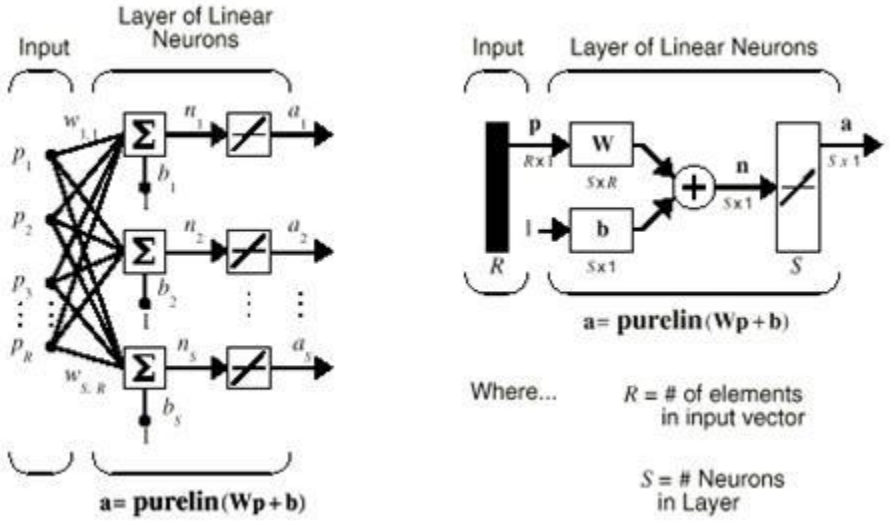


Figura 3.

La arquitectura neuronal mostrada en la Fig. 3 tiene una capa de *s* neuronas conectadas a *R* entradas a través de una matriz de pesos *W*. [2]

Esta red es con frecuencia llamada MADALINE o Múltiples ADALINE. A la derecha de la Fig. 3 define un vector de salida *a* de longitud *S*.

La regla de Widrow – Hoff puede entrenar solamente una capa de redes lineales. Esto no es tanto una desventaja, ya que una red de una sola capa es tan capaz como una red de múltiples capas. Para cada red lineal multicapa, existe una red lineal de una sola capa. [2]

Resultados.

El script realizado en Matlab corresponde a la simulación del entrenamiento de una Red Adaline usando matrices, for loops, y sentencias condicionales (if). El problema planteado es entrenar una red Adaline que pueda decodificar binario a decimal, para ello se plantearon las siguientes matrices y valores iniciales:

$$W = [\,0.84 \quad 0.39 \quad 0.78\,]$$

$$b = [\,0\,]$$

$$itmax = 10$$

$$e_{it} = 0.01$$

$$\alpha = 0.3$$

Y el conjunto de entrenamiento que se muestra en la Tabla 1:

	P1	P2	P3	t
d1	0	0	0	0
d2	0	0	1	1
d3	0	1	0	2
d4	0	1	1	3
d5	1	0	0	4
d6	1	0	1	5
d7	1	1	0	6
d8	1	1	1	7

Tabla 1. Conjunto de Entrenamiento

Donde N = 8.

Nota: La matriz de pesos W puede inicializarse con valores cualesquiera en sus entradas siempre y cuando: $0 < W_{ij} < 1$.

Por cada iteración se propagaron hacia adelante los elementos del conjunto de entrenamiento hasta que el error de iteración E_{it} sea cero (que es el caso ideal) o sea menor al valor de e_{it}.

Utilizando el script de Matlab nos pide los valores de entrada $itmax$, α y e_{it}.

Utilizamos los establecidos en clase:

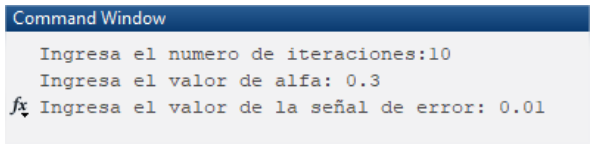


Figura 4. Ejecución del script de Matlab estableciendo valores.

Posteriormente el script empieza a hacer las iteraciones necesarias hasta llegar a uno de los criterios de finalización. Para este ejemplo, se muestra que concluye el aprendizaje en la cuarta iteración como se muestra en la figura 5:

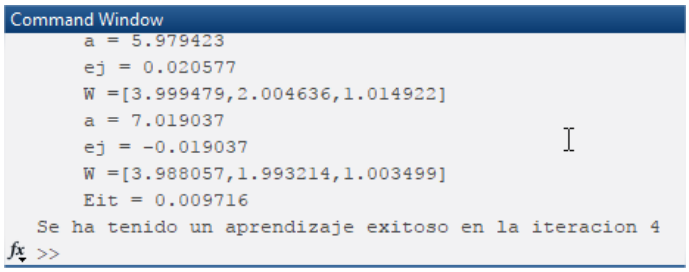


Figura 5. Resultados.

Posteriormente grafica el cambio de los valores de las entradas de la matriz W y E_{it} con respecto a la propagación de cada uno de los elementos del conjunto de entrenamiento en cada iteración como se muestra en la figura 6:

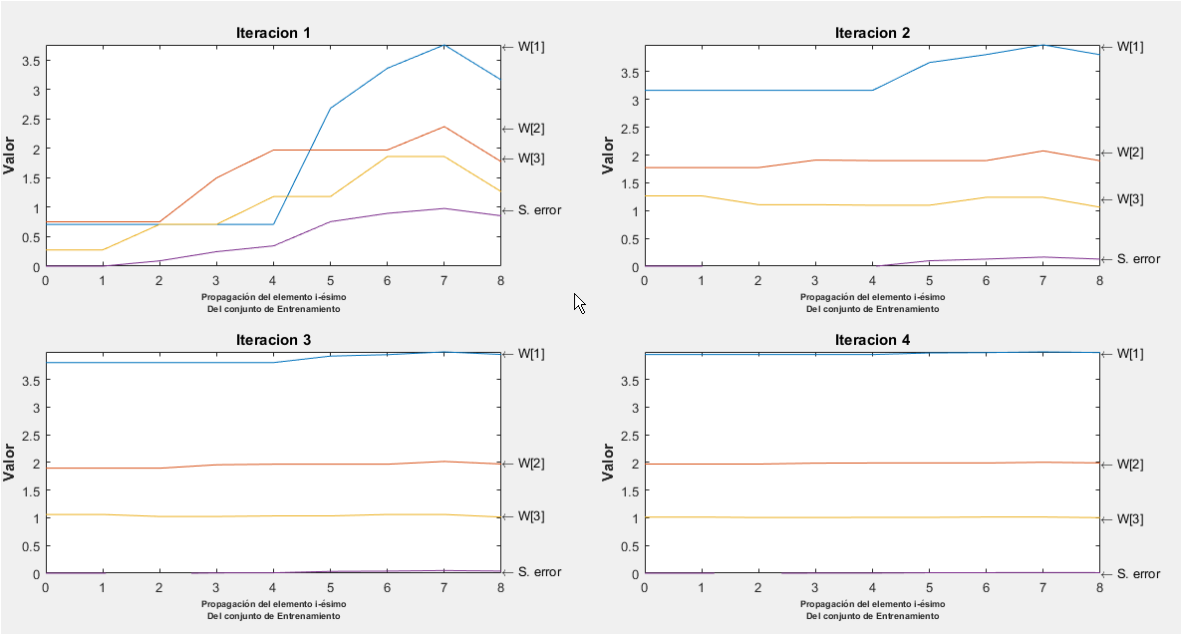


Figura 5. Grafica de los resultados.

Vemos que para la última iteración la señal del error converge cada vez más a cero.

CONCLUSIONES.

En este programa considero que lo más difícil fue la graficación, ya que tuve que buscar información sobre como poder etiquetar las curvas en la gráfica y hacer los subplots. Además, fue de ayuda para comprender mejor el funcionamiento de la red Adaline, desarrollando más pruebas, con los mismos valores de entrada a excepción del número de iteraciones que se aumentaron y estableciendo el valor de $e_{it} = 0$, la red tiene un aprendizaje exitoso con un error de 0 en la iteración 27, mientras que si establecemos $\alpha = 0.5$, la red tiene un error de 0 en la segunda iteración. Ahí pude observar como el factor de aprendizaje puede hacer que el entrenamiento sea o más rápido o más lento, aunque su valor dependerá mucho del problema a resolver.

REFERENCIAS

[1] «Laboratorio del Departamento de Informática,» [En línea]. Available: http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/perceptron-simple.html. [Último acceso: 9 Octubre 2017].

[2] I. H. F. Gutiérrez, «Polilibro Redes Neuronales Artificiales 1,» [En línea]. Available: http://www.hugo-inc.com/RNA/Unidad%202/2.2.1.html. [Último acceso: 10 Octubre 2017].

ANEXO: CÓDIGO FUENTE.

RedADALINE.m

```
clc
clear
iteraciones = input('Ingresa el numero de iteraciones:');
alpha = input('Ingresa el valor de alfa: ');
eit = input('Ingresa el valor de la señal de error: ');
W = rand(1,3); %Asignamos valores aleatorios a la matriz de pesos
%W = [0.84 0.39 0.78];
d = [           % Conjunto de entrenamiento
    0 0 0;
    0 0 1;
    0 1 0;
    0 1 1;
    1 0 0;
    1 0 1;
    1 1 0;
    1 1 1
    ];
W_1 = zeros(1,9);   % Vector para guardar los valores de cambio de W[1]
W_2 = zeros(1,9);   % Vector para guardar los valores de cambio de W[2]
W_3 = zeros(1,9);   % Vector para guardar los valores de cambio de W[3]
ErrorDeIteracion = zeros(1,9);   % Vector para guardar los valores de cambio de la señal de error
t = 0.0;    % Iniciamos t en 0
Eit = 0.0;  % Error de iteracion
figure
rango = 0:8;    % Rango para el plot
for i = 1:iteraciones   % Loop sobre el numero maximo de iteraciones
    fprintf('Inicia la iteracion %d\n',i);
    for j = 1:8     % Loop sobre los elementos del cto. de entrenamiento
        W_1(j) = W(1);  % Asignamos los valores de cambio de las entradas
        W_2(j) = W(2);  % de W y la señal de error para el ploteo
        W_3(j) = W(3);
        ErrorDeIteracion(j) = Eit;
        %fprintf('\tSe propaga d%d hacia adelante\n',j);
        a = purelin(W*d(j,:)');
        ej = t - a;
        t = t+1;
        fprintf('\ta = %f\n',a);
        fprintf('\tej = %f\n',ej);
        if ej ~= 0.0 % Si hubo cambios en la señal del error del termino j
            W = W+(2*alpha*ej*d(j,:)); % Se ajustan los valores de la matriz de pesos
        end
        fprintf('\tW =[%f,%f,%f]\n',W(1),W(2),W(3));
        Eit = Eit+(1/8)*ej; % Actualizamos el error de iteracion
    end
    W_1(j+1) = W(1);  % Asignamos los valores de cambio de las entradas
    W_2(j+1) = W(2);  % de W y la señal de error para el ploteo
    W_3(j+1) = W(3);
    ErrorDeIteracion(j+1) = Eit;
    fprintf('\tEit = %f\n',Eit);
    subplot(2,ceil(iteraciones/2),i);   % Creamos un subplot por iteracion
    plot(rango,W_1);
    text(8,W_1(8),'\leftarrow W[1]');
    titulo = strcat('Iteracion',{' '},num2str(i));
    title(titulo);
    hold on
    plot(rango,W_2);
    text(8,W_2(8),'\leftarrow W[2]');
    plot(rango,W_3);
    text(8,W_3(8),'\leftarrow W[3]');
    plot(rango,ErrorDeIteracion);
    text(8,ErrorDeIteracion(8),'\leftarrow S. error');
    axis([0 8 0 inf])
    hold off
    xlabel({'Propagación del elemento i-ésimo','Del conjunto de Entrenamiento'},'FontSize',7,'FontWeight','bold')
    ylabel('Valor','FontWeight','bold')
    if Eit == 0 % Si el error de iteracion es cero (caso ideal)
        fprintf('Se ha tenido un aprendizaje exitoso en la iteracion %d\n',i);
        break   % Termina el aprendizaje de la red
    end
    if Eit < eit % Si el error de iteracion es menor al valor establecido de la señal del error eit
        fprintf('Se ha tenido un aprendizaje exitoso en la iteracion %d\n',i);
        break    % Termina el aprendizaje de la red 2/3 de probabilidad de asegurar un aprendizaje adecuado
    end
    Eit = 0;    % Se reinician el valor del error de iteracion y t
    t = 0;
end
clear
```