Mendoza Hernandez Yair Alberto
3BM4
Primer departamental

P1. Compuerta OR red perceptron

Marco Teórico

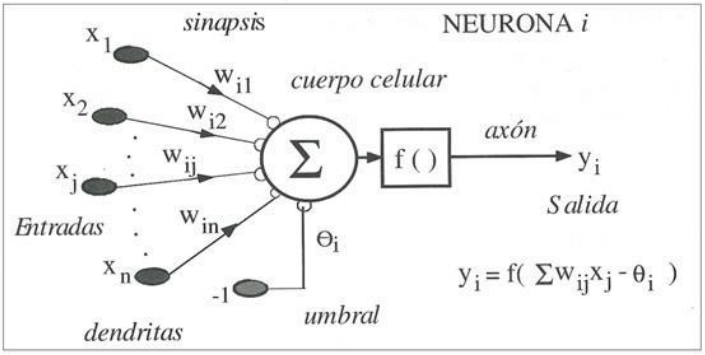
RED PERCEPTRON

El perceptrón (Perceptron en inglés) fue el primer modelo de Red Neuronal Artificial supervisada. Es la más simple de las Redes neuronales.

Fue creada por Rosenblatt en 1958 y su éxito incial se debió a que era capaz de aprender y reconocer patrones sencillos. Con el desarrollo del perceptrón, surge el área de las Redes Neuronales Artificiales dentro de la Inteligencia Artificial. Sin embargo, Marvin Minskyy Seymur Papert escriben el libro "Perceptrons", en el que se hace un análisis del Perceptrón mostrando sus flaquezas y decae el apoyo dado a la investigación de las Redes Neuorales Artificiales durante algunas décadas.

Las principales limitaciones del perceptrón son que sirve únicamente para problemas linealmente separables y que sean de dos clases. Hablando vulgarmente, esto quiere decir que el perceptrón sólo lo podemos usar cuando el problema sea distinguir entre una de dos posibles clases y, que trazando una línea, plano o hiperplano en un plano o hiperplano, se puedan separar perfectamente estas dos clases

.



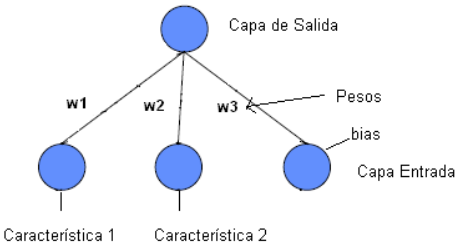
El perceptrón simple es un modelo neuronal unidireccional, compuesto por dos capas de neuronas, una de entrada y otra de salida. La operación de una red de este tipo, con n neuronas de entrada y m neuronas de salida, se puede expresar de la siguiente forma:

$$y_i(t) = f\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right), \forall i, 1 \le i \le m$$

el perceptrón es la única red neuronal que tiene un teorema de convergencia el cual establece que, si el problema es linealmente separable, el perceptrón encontrará la solución. Aunque no se sabe cuánto tiempo le llevara encontrar la solución y mucho menos si la solución encontrada será la óptima, se sabe que se tendrá una solución.

Arquitectura del Perceptron

La siguiente imagen ilustra la arquitectura del perceptron para patrones con sólo dos características:



Como se puede apreciar, el perceptron está formado por dos capas, una de entrada con un número de nodos determinado y una de salida con un sólo nodo el cuál se encuentra conectado a cada uno de los nodos de la capa de entrada mediante una conexión que está valuada con un peso (w1, w2 y w3). Existe un nodo extra llamado bias el cuál no tiene contacto con el exterior y su valor siempre es 1. Cabe hacer la aclaración que algunos autores no toman en cuenta la capa de entrada debido a que en ésta no se lleva cabo ningún procesamiento de la información, simplemente sirve como enlace con el exterior de la red neuronal y su única tarea es recibir los valores de entrada del exterior y pasárselos al nodo de la capa de salida. Para estos autores el perceptrón consta de una capa. Para este artículo, el perceptrón consta de dos capas: una de entrada y una de salida.

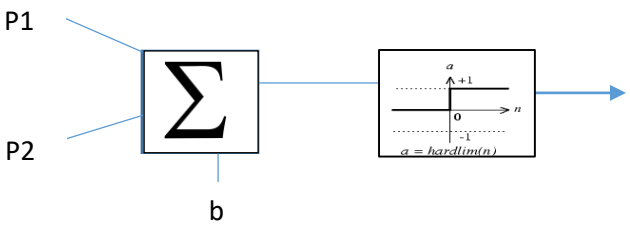
Para la primera capa, tendremos igual número de nodos que número de características en los patrones a analizar más el nodo del bias, o bien, dicho de otro modo, tendremos igual número de nodos que número de elementos en nuestro vector que representa al patrón más el nodo del bias. Es decir, si tenemos un patrón representado por el siguiente vector p1 = [a1,a2], entonces tendremos en nuestra primera capa dos nodos de entrada más el nodo del bias cuya salida siempre será 1. Los dos nodos de la capa de entrada relacionados con las características del patrón, deberán ser alimentados con los valores respectivos de los patrones que se estén usando.

Desarrollo y solución propuesta

la práctica consistido en desarrollar una red perceptron basada en una sola neurona para poder describir la respuesta de una compuerta OR, los valores apropiados para cada una de las combinaciones en una compuerta OR se presentan en la siguiente tabla

Entradas		Salida
0	0	0
0	1	1
1	0	1
1	1	1

Para poder encontrar estas salidas correspondientes a los puntos de prueba se utilizó una sola neurona con la siguiente configuración

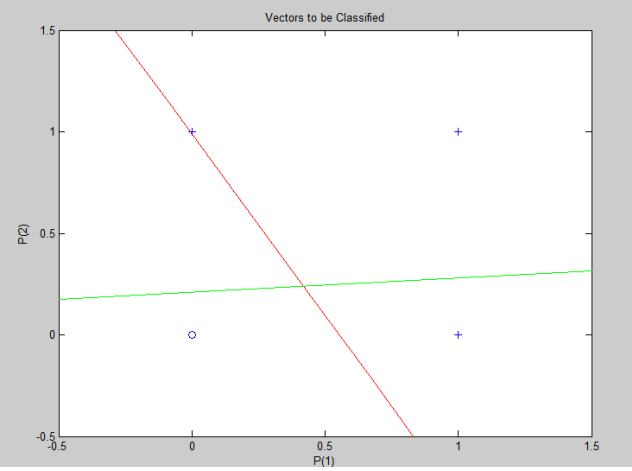


la cual discriminaría las entradas con relación a los target de salida en este caso la función que se utilizó para la neurona fue hardlim por lo que las salidas y los target quedaron de la misma manera, para encontrar las salidas correspondientes se utilizaron pesos y polarizaciones Random para el inicio, posteriormente se metió en un ciclo for la función de la neurona para que se hicieran iteraciones hasta que el error correspondiente de la salida de la neurona con los target fuera cero este error se actualiza junto con los pesos y la polarización, cuando se termina la primera iteración se vuelve a ejecutar el ciclo con el fin de corroborar que el error efectivamente en todos los target sea cero, de no ser así el ciclo volverá a iniciar hasta que efectivamente el error sea 0 para todas las salidas, una vez que la neurona termina el proceso se puede decir que se entrenó de manera satisfactoria y el comportamiento de esta neurona corresponde ya a el comportamiento de la compuerta OR para corroborar eso mediante el vector de pesos, se procedió a graficar la línea que pasa por las intersecciones correspondientes a las características de este modo esta linea representaría la frontera de decisión con lo cual del lado que marca el vector de peso quedaría una clase y por debajo podemos encontrar la otra (como se muestra en la gráfica 1)

Algoritmo

```
clear all; close all; clc;
P=[0 0 1 1;0 1 0 1];
T=[0 1 1 1];
 w=rand(2,1);
 b=rand;
 error=0;
 L=length(P)
 while error<L
     for i=1:L
     a=hardlim((w'*P(:,i))+b);
     e=T(i)-a;
     w=w+e*P(:,i);
     b=b+e;
         if e==0
             error=error+1;
         else
             error=0;
         end
     end
 end
     intp1=-b/w(1,:);
     intp2=-b/w(2,:);
     m=(intp2-0)/(0-intp1);
mw=-1/m;
l =linspace(-2,2,10);
lw=linspace(0,0.5,10);
f=(m*l)+intp2;
fw=mw*l+intp2;
plotpv(P,T)
hold on
plot(f,l,'r',fw,lw,'g')
```

Graficas



- frontera de decisión
  - vector de pesos

Grafica 1. Compuerta OR modelada con una red perceptron básica

Conclusiones

Como se puede ver en la práctica con una red perceptron que implica una sola neurona es posible separar de manera correcta dos clases distintas mediante la frontera de decisión generada por los pesos obtenidos de la neurona, es fácil demostrar los resultados de la compuerta OR mediante una gráfica de los puntos de prueba para las diferentes combinaciones junto con las fronteras de esta manera gráfica podemos ver más fácilmente el funcionamiento de la red perceptron

P2.Perceptron 7 puntos

Desarrollo y solución propuesta

En la práctica se desea clasificar 7 patrones distintos en 3 clases diferentes a partir de una red perceptron para resolver esta práctica se hizo uso de una red perceptron de dos neuronas debido a que al final se necesitaría clasificar en 3 clases y con una no sería suficiente, para el uso de esta red utilizamos la función de activación hardlim simétrica por lo que los target deberían ser definidos entre 1 y -1 se comenzó por desplegar los diferentes patrones en un solo vector posteriormente se definieron los target y se acomodaron en la misma disposición a su correspondencia en los patrones anteriores ,después se seguiría el mismo procedimiento de la práctica uno por lo cual se procedió a realizar iteraciones sucesivas sobre la ecuación que describe a la red perceptron (Ec. 2)con el fin de encontrar los pesos y polarizaciones correspondientes a las barreras de decisión que harían que se clasifiquen de manera correcta los patrones .

$$n=W^{T}*p^{+}b^{\wedge} \text{ Ecu. 2)}$$

$$a=hardlim(a)$$

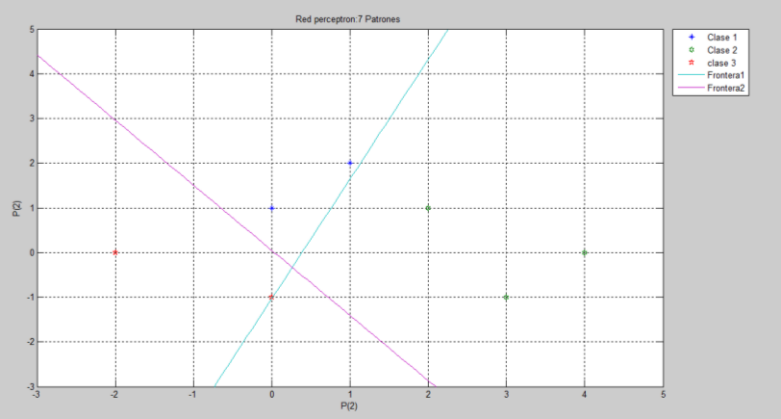
Estos pesos y polarizaciones se obtuvieron al evaluar cada uno de los putos de los patrones en la función correspondiente con lo cual se compararía el error de cada una de los resultados de la función de activación con los target con el fin de asegurar su correcta clasificación

Algoritmo

```
clear all; close all; clc;
P=[0 1 2 3 4 -2 0; 1 2 1 -1 0 0 -1];
t=[-1 -1 1 1 1 -1 -1;
    -1 -1 -1 -1 -1 1 1];
w=rand(2,2);
b=rand(2,1);
 band=0;
 while band<length(P(1,:))
     for i=1:length(P(1,:))
     a=hardlims((w'*P(:,i))+b);
     e=t(:,i)-a;
     w=w+(e*P(:,i)')';
     b=b+e;
     if e==[0;0]
         band=band+1;
     else
         band=0;
     end

     end
 end
  intp11=-b(1,:)/w(1,1);
  intp12=-b(1,:)/w(1,2);
  intp21=-b(2,:)/w(2,1);
  intp22=-b(2,:)/w(2,2);
  m1=(intp12-0)/(0-intp11);
  m1w=-1/m1;
  m2=(intp22-0)/(0-intp21);
```

```
  m2w=-1/m2;
  l=linspace(0,6,100);

 f1w=m1w*l+intp12;
 f2w=m2w*l+intp22;

plot(P(1,1:2),P(2,1:2),'*',P(1,3:5),P(2,3:5),'h',P(1,6:7),P(2,6:7),'p',li
nspace(-4,3,100),(w(1,1)*linspace(-4,3,100)+b(1,1))/-w(2,1),linspace(-
4,3,100),(w(1,2)*linspace(-4,3,100)+b(2,1))/-w(2,2))
 grid on
 hold on
 xlim([-3,5])
 ylim([-3,5])
%  plot(f1w,l,'.-c',f2w,l,'--m') se utilizo para verificar el vector de
%  pesos
```

Graficas



Grafica 2. Clasificación de los 7 patrones y barreras de decisión generadas mediante los pesos generados

Conclusiones

En esta práctica se vio la utilización de la red perceptron solo que esta vez ocupando una neurona de mas con lo cual nos fue posible clasificar de manera correcta más de 2 clases podemos ver también como al ser pocos puntos es más fácil que se entrene de manera correcta las neuronas, es bueno recalcar que en este caso se varió la función de activación de las neuronas por lo cual los target fueron definidos de diferente manera a como se obtuvieron en la práctica anterior

P3.Clasificación Iris Red perceptron

Desarrollo y solución propuesta

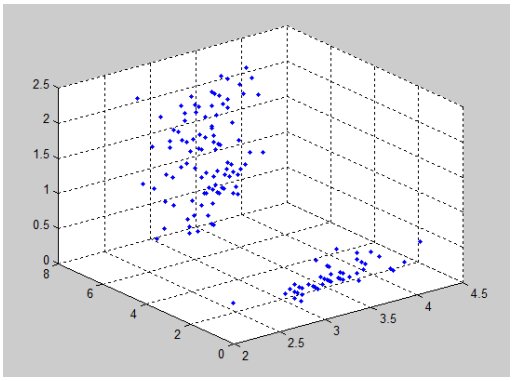
En esta práctica se necesitaba clasificar correctamente una base de datos correspondiente a diferentes tipo de iris los cuales estaban clasificados mediante 3 características , estas describían los tipo de iris como iris cetosa , iris-versicolor e iris viginica era necesario clasificar todas estos datos mediante una red perceptron por lo cual se utilizó nuevamente una red basada en 2 neuronas debido a que se tendrían tres clases distintas, los target en esta ocasión se eligieron aleatoriamente para cada clase con la condición que entre ellos fueras distintos para las 3 clases de esta forma la neurona debería asignar correctamente los valores de las clases ,una vez definidos los patrones y los target se metieron estos datos a la neurona caracterizada por una función de activación hardlims para generar los valores de salida con los que nuevamente generaríamos los pesos y polarizaciones, en este caso al tener tres características de entrada el número de pesos y polarizaciones aumentarían para cada rama que conecta a las características con la función de activación esto genero la necesidad de incluir un nuevo eje con lo cual ahora se tendría un mayor número de intersección y por lo tanto una gráfica en donde las barreras de decisión corresponderían ahora a planos de decisión para diferir una clase de otra esta grafica se muestra en (grafica. 3) por otra parte para esta red también se obtuvo el porcentaje de efectividad a partir de la comparación de los target con los resultados de la neurona haciendo que una clasificación de todos los datos fuera el 100% de efectividad caso que no se pudo obtener sin embargo si se obtuvo un alto grado de efectividad

Algoritmo

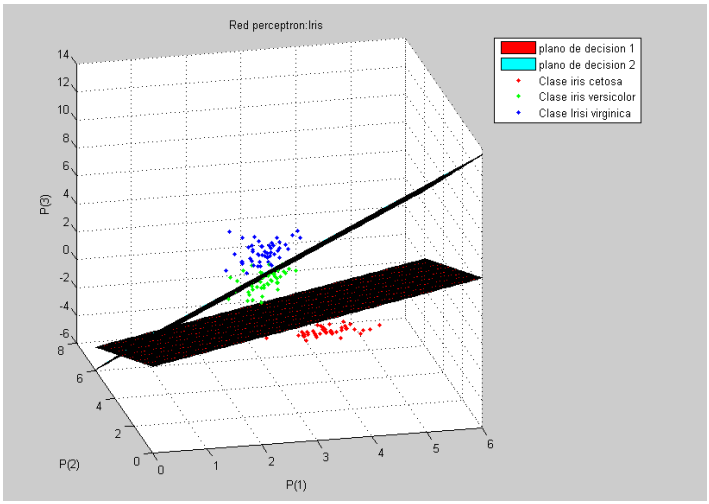
```
close all; clear all; clc;
% Datos = importdata('IrisDataBase.dat')
% Adquisición de datos
dat = fopen('IrisDataBase.dat');
IrisDataBase = textscan(dat, '%f %f %f %f %s');
fclose(dat);
Datos(:,1)=IrisDataBase{2};
Datos(:,2)=IrisDataBase{3};
Datos(:,3)=IrisDataBase{4};
figure(1)
plot3(Datos(:,1),Datos(:,2),Datos(:,3),'.')
grid on;
DatosS=Datos(1:50,:);%Datos Iris-setosa
DatosVS=Datos(51:100,:); %Datos Iris-versicolour
DatosV=Datos(101:150,:);%Datos Iris-virginica
%%
P=Datos';
T=[zeros(50,1)',zeros(50,1)',ones(50,1)';zeros(50,1)',ones(50,1)',ones(50
,1)'];
W=rand(2,3);
b=rand(2,1);
epocas=input('numero de epocas: ');

for x=1:epocas
    for y=1:length(P(1,:))
      a=hardlim((W*P(:,y))+b);
      e=T(:,y)-a;
      W=W+(e*P(:,y)'); % Modificación de pesos
```

```
      b=b+e; % Modificación del valor de b
    end
end
    for k=1:2
        for y=1:length(P)
             CA(k,y)=hardlim((W(k,:)*P(:,y))+b(k));

        end
    end
Comp=CA==T;
G1=find(Comp(1,:)==1);
G2=find(Comp(2,:)==1);
G=[length(G1),length(G2)];
efectividad=(min(G)*100)/150;
intp11=-b(1,:)/W(1,1);
intp12=-b(1,:)/W(1,2);
intp13=-b(1,:)/W(1,3);
intp21=-b(2,:)/W(2,1);
intp22=-b(2,:)/W(2,2);
intp23=-b(2,:)/W(2,3);
%%
syms x y z x2 y2 z2
A1=[intp11,0,0];B1=[0,intp12,0];C1=[0,0,intp13];
A2=[intp21,0,0];B2=[0,intp22,0];C2=[0,0,intp23];
u=B1-A1; v=C1-A1;
u2=B2-A2;v2=C2-A2;
p=[x-intp11 y-0 z-0;u(1) u(2) u(3) ;v(1) v(2) v(3)];
p2=[x2-intp21 y2-0 z2-0; u2(1) u2(2) u2(3);v2(1) v2(2) v2(3)];
EC1=det(p);
EC2=det(p2);
R=solve(EC1,'z');
R2=solve(EC2,'z2');
[x,y]=meshgrid(0:.1:6);
[x2,y2]=meshgrid(0:.1:6);
figure(2)
hold on
surf(x,y,(161*x)/214 - (49*y)/65 + 1127/5000,'facecolor','r')
surf(x2,y2,(4972541871005141*x2)/2709053458594275 -
((9945083742010282*y2)/7054714458587993)+2 +
4972541871005141/9007199254740992,'facecolor','c')
plot3(DatosS(:,1),DatosS(:,2),DatosS(:,3),'.r')
plot3(DatosVS(:,1),DatosVS(:,2),DatosVS(:,3),'.g')
plot3(DatosV(:,1),DatosV(:,2),DatosV(:,3),'.b')
grid on;
hold off
disp(['el desempeño es de: ',num2str(efectividad),'%'])
```

Graficas



Grafica 3 muestras obtenidas de diferentes tipos de iris



Grafica 4 clasificación de 3 tipo de Iris mediante una red perceptron

Conclusiones

Para la clasificación de muestras con más de dos características el espacio en donde se pueden observar las clasificaciones cambia a un espacio de 3 dimensiones por lo cual las berreras de decisión se convierten en planos, podemos además observar como en esta práctica se pudo obtener un nivel de eficiencia superior al 80% con un numero de épocas arriba de 100 debido a que hay datos en dos clases que se pueden confundir es por lo cual el sistema no tiene un 100% de efectividad, esta combinación es debido a que las características arrojan valores muy parecido entre ellas por lo demás podemos ver como a pesar de ello se pueden clasificar de una buena manera las muestras paras las clases especificadas

P4.Compuerta OR red ADALINE

MARCO TEÓRICO

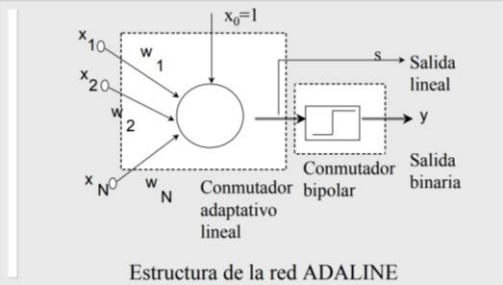
RED ADALINE

La Red Adaline fue desarrollada en el 1960 por Bernard Widrow y su estudiante Marcian Hoff de la universidad de Stanford.
ADALINE proviene de Adaptive Lineal Element (Elemento Lineal Adaptativo), pero antes de que se le diera este nombre esta red sufrió un cambio ya que primeramente se llamaba Adaptive Lineal Neuron (Neurona Linear Adaptiva), dicho cambio se dio por que la Red Adaline es un dispositivo que consta de un único elemento de procesamiento, como tal no es técnicamente considerada una red neuronal.

Las Redes ADALINE son redes muy similares al Perceptrón con la diferencia de que su función de activación es lineal en lugar de ser un limitador fuerte como es el caso del Perceptrón, estas presentan la misma limitación del Perceptrón respecto al tipo de problemas que pueden resolver, ya que ambas redes solo pueden resolver problemas linealmente separables.
Son redes de aprendizaje supervisado que usan la regla de Widrow – Hoff para dicho aprendizaje o también denominada regla Delta. El algoritmo que estas usan es el LMS (Least Mean Square) siendo este más eficiente que la regla de aprendizaje del Perceptrón puesto que minimiza el error medio cuadrático.
Adaline está limitada a una única neurona de salida, un vector x como su entrada y un número real y como su salida.

ARQUITECTURA

Como se aprecia en la (Imagen 1), la Red Adaline está formada por un elemento denominado Combinador Adaptativo Lineal (ALC).
La salida lineal obtenida del ALC es aplicada a un Conmutador Bipolar.
El Umbral se representa a través de una conexión ficticia de peso W0



Estructura de la red ADALINE

ALGORITMO DE APRENDIZAJE:

1.- Inicializar los pesos en forma aleatoria
2.- Introducir un patrón de entrada
3.- Calcular la salida (y), compararla con la deseada (d) y obtener la diferencia (dp - yp)
4.- Multiplicar el resultado del paso anterior por la entrada correspondiente a cada uno de los pesos y ponderarla por la tasa de aprendizaje.
5.- Actualizar los pesos, sumando al valor antiguo la cantidad obtenida en el paso anterior
6.- Si no se ha cumplido el criterio de parada, regresar al paso 2, si se ha acabado todos los patrones repetir el algoritmo.

Desarrollo y solución propuesta

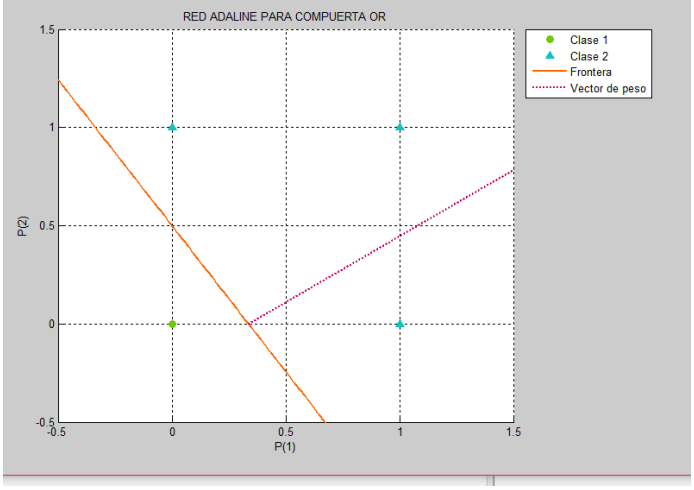
En esta práctica se presenta el mismo problema que en la práctica uno el cual es el simular el comportamiento de una compuerta OR solo que en esta ocasión se hizo utilizando una red ADALINE para realizar esta red se utilizaron los mismos patrones de entrenamiento que en la tabla 1 en esta ocasión para que la red funcionara correctamente se cambiaron los target de manera que estos fueran de forma bipolar teniendo -1 y 1 además de esto la salidas que generaría la función de activación purelin dejaran de ser lineales es decir no variaran solo en signo si no que estas salida tendrán una magnitud decimal representativa así que una vez que se obtuvieron los valores generados por la ecuación se actualizan los pesos y las polarización pero en esta ocasión se utilizara un parámetro alpha, este parámetro nos ayuda a saber que tan "bien " aprende la neurona, una vez que se actualizan los pesos se hacen tantas iteraciones como sean necesarias hasta obtener valores contantes en más de 1 iteración de este modo quedaran clasificados de manera correcta los patrones.

Algoritmo

```
clc, clear all, close all;
%Red ADALINE
%utilice el algoritmo ADALINE para clasificar correctamente los
%siguientes patrones
P=[0,0,1,1;0,1,0,1];
T=[-1,1,1,1];
W=rand(2,1);
b=rand;
R=(((1/4)*P)*P');
Lamax=max(eig(R));
alpha=1/(4*Lamax)*.99;
epocas=input('numero de epocas: ');
R2=0;
for x=1:epocas
    for y=1:length(P)
 a=purelin(W'*P(:,y)+b); % Salida de la red
 e=T(y)-a; % Error
 W=W+e*alpha*P(:,y); % Modificación de pesos
 b=b+e*alpha; % Modificación del valor de b
    end
end
intp1=-b/W(1,:);
intp2=-b/W(2,:);
m=(intp2-0)/(0-intp1);
mw=-1/m;
l =linspace(-1,1.5,80);
lw=linspace(0,1.5,20);
f=(m*l)+intp2;
fw=mw*lw+intp2;
%Graficar Frontera y vector de pesos
hold on
axis([-0.5,1.5,-0.5,1.5])
plot(P(1,1),P(2,1),'o','MarkeredgeColor',[113/255,200/255,8/255],'MarkerF
aceColor',[113/255,200/255,8/255])
plot(P(1,2:4),P(2,2:4),'^','MarkeredgeColor',[12/255,196/255,192/255],'Ma
rkerFaceColor',[12/255,196/255,192/255])
```

```
plot(f,l,'m','Linewidth',2,'color',[255/255,102/255,0/255])
plot(fw,lw,'g:','Linewidth',2,'color',[203/255,5/255,104/255]),title('RED
ADALINE PARA COMPUERTA OR')
xlabel('P(1)'),ylabel('P(2)'),legend('Clase 1','Clase
2','Frontera','Vector de peso','Location','northeastoutside')
grid on
hold off
```

Graficas



Grafica 5 clasificación de los patrones que definen el comportamiento de una compuerta OR mediante una red ADALINE

Conclusiones

Se puede observar como mediante el uso de una red ADALINE la separación entre clases se puede ver más clara esto debido a que las fronteras quedan equidistantes de los puntos patrón ,manejar una red ADALINE es prácticamente lo mismo que usar un perceptron solo que en la red ADALINE se puede hacer más flexible la clasificación debido a que los cambios en los pesos no son tan bruscos ,el vector que asocia la frontera de decisión se puede adecuar en muchas más direcciones que en el caso de perceptron aparte se puede observar que en esta red es necesario el uso de target bipolares

P5.ADALINE 7 puntos

Desarrollo y solución propuesta

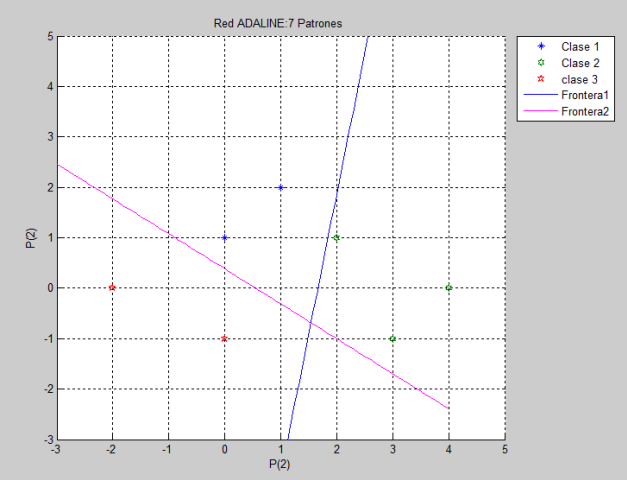
En esta práctica se procedió a clasificar 7 patrones propuestos con sus respectivos target, para la solución del problema se utilizó una red ADALINE mediante la cual se usarían 2 neurona para clasificar los 7 puntos, organizados en 3 clases, cada clase se separaría correctamente a partir de la aplicación de una red , posterior a organizar patrones y target se procedió a calcular el factor alpha de manera automática utilizando el número de patrones en el universo de trabajo y siguiendo el procedimiento descrito en clase y así obtener el factor de aprendizaje luego de esto los patrones se metieron en las neuronas para obtener su valores de salida los cuales nos dirían su correspondencia con cada clase, a partir de la comparación de estas salidas con los target tomando en cuenta que la red ADALINE organiza números decimales se compara el error que se encuentra con respecto a los target antes mencionados, de esta forma este error y el signo que los acompaña ayuda encontrar los pesos y polarizaciones que reflejan la correcta clasificación, para este caso el sistema se detendría hasta que el número de épocas propuesto se acabara, se propusieron un numero de épocas menos que en la red perceptron esto porque se sabe que la red ADALINE se adapta más rápido que perceptron una vez obtenidos estos pesos y polarizaciones se procedió a graficarlas en el espacio de trabajo para así corroborar el correcto funcionamiento del algoritmo

Algoritmo

```
clear all, close all, clc;
%Clasificacion de 7 muestras a partir de una red ADALINE
P=[0 1 2 3 4 -2 0; 1 2 1 -1 0 0 -1];
t=[-1 -1 1 1 1 -1 -1;-1 -1 -1 -1 -1 1 1];
R=(((1/length(P(1,:)))*P)*P');
Lamax=max(eig(R));
alpha=1/(4*Lamax)*.99;
w=rand(2,2);
b=rand(2,1);
 epocas=input('numero de epocas: ');
 for j=1:epocas
     for i=1:length(P(1,:))
     a=purelin((w'*P(:,i))+b);
     e=t(:,i)-a;
     w=w+(e*alpha*P(:,i)')';
     b=b+e*alpha;
     end
 end
  intp11=-b(1,:)/w(1,1);
  intp12=-b(1,:)/w(1,2);
  intp21=-b(2,:)/w(2,1);
  intp22=-b(2,:)/w(2,2);
  m1=(intp12-0)/(0-intp11);
  m1w=-1/m1;
  m2=(intp22-0)/(0-intp21);
  m2w=-1/m2;
  L=linspace(-4,2,15);
  l=linspace(0,2,15);
 f1w=m1w*L+intp21;
 f2w=m2w*l+intp22;
```

```
grid on
hold on
plot(P(1,1:2),P(2,1:2),'*',P(1,3:5),P(2,3:5),'h',P(1,6:7),P(2,6:7),'p')
plot(linspace(-4,4,100),(w(1,1)*linspace(-4,4,100)+b(1,1))/-w(2,1))
plot(linspace(-4,4,100),(w(1,2)*linspace(-4,4,100)+b(2,1))/-w(2,2))
xlim([-3,5])
ylim([-3,5])
```

Graficas



Grafica 6 clasificación de los 7 patrones de entrenamiento a partir de una red ADALINE

Conclusiones

Podemos observar como la clasificación se comporta de mejor manera con la red ADALINE, la separación de las clases se puede ver más clara y consistente en un menor número de épocas que con la red perceptron a pesar de que son los mismos datos, al ser la red ADALINE la clasificación es más flexible y al tener el factor de aprendizaje adaptado a la red esta puede aprender de mejor manera

P6.Clasificacion iris red ADALINE

Desarrollo y solución propuesta

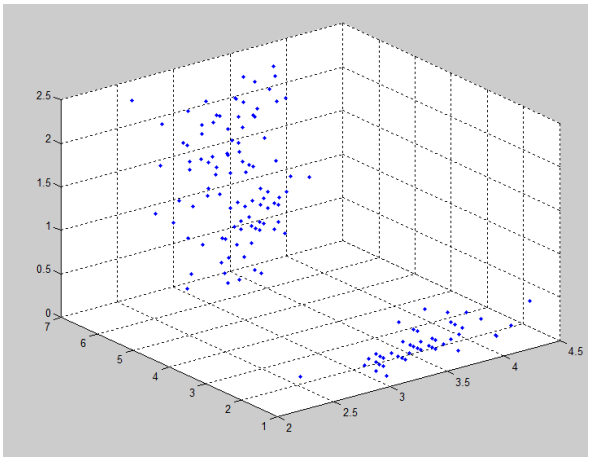
Mediante una red ADALINE se propone hacer la clasificación de una base de datos que describen los tipos de iris, esta clasificación se hace utilizando las 3 características principales que definen a cada uno de estos tipos de iris de modo que estos datos generaran nuestro universo con el cual trabajaremos asignando a cada uno de estos patrones un target bipolar y por ende al ser una red ADALINE generaremos un factor de aprendizaje basado en estos patrones, una vez que obtenemos el factor de aprendizaje no queda mas que aplicar la red ADALINE bajo la función de activación purelin para generar nuestros pesos y polarizaciones representantes de la clasificación, este aprendizaje se hace mediante un numero de épocas las cuales serán menos que las puesta en la red perceptron con el fin de comprobar que esta red funciona mejor y responde en menor tiempo debido al factor de aprendizaje cada uno de los pesos que se obtengan se actualizarán basados en el error que se genere con respecto a los target, de este modo los pesos actualizados hasta el fin de las épocas nos darán las intersecciones que definen los planos de decisión, en este caso son planos debido al número de características, estos planos pasaran a ser graficados al final junto con las muestras para corroborar que la clasificación es correcta, aparte de la parte grafica se usara una matriz de confusión para conocer la efectividad del sistema

Algoritmo

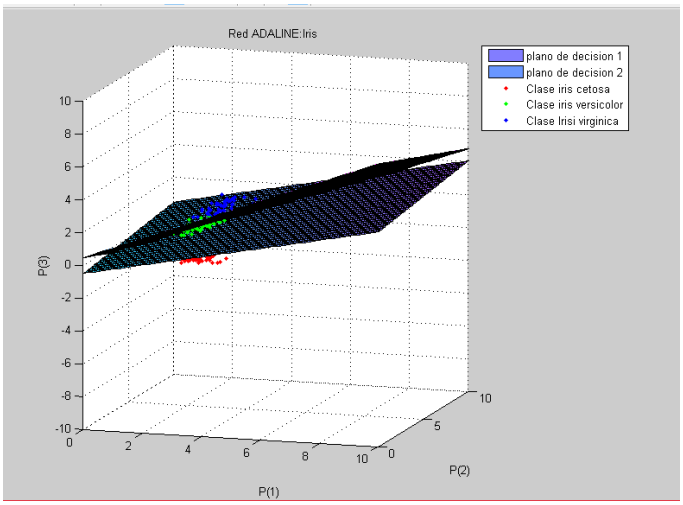
```
clear all,close all,clc;

dat = fopen('IrisDataBase.dat');
IrisDataBase = textscan(dat, '%f %f %f %f %s');
fclose(dat);
Datos(:,1)=IrisDataBase{2};
Datos(:,2)=IrisDataBase{3};
Datos(:,3)=IrisDataBase{4};
figure(1)
plot3(Datos(:,1),Datos(:,2),Datos(:,3),'.')
grid on;
DatosS=Datos(1:50,:);%Datos Iris-setosa
DatosVS=Datos(51:100,:); %Datos Iris-versicolour
DatosV=Datos(101:150,:);%Datos Iris-virginica
%%
P=Datos';
T=[ones(50,1)'*-1,ones(50,1)'*-1,ones(50,1)';
    ones(50,1)'*-1,ones(50,1)',ones(50,1)'];
T1=length(find(T(1,:)<0));
T2=length(find(T(2,:)>0));
W=rand(2,3);
b=rand(2,1);
R=(((1/length(P(1,:)))*P)*P');
Lamax=max(eig(R));
alpha=1/(4*Lamax)*.99;
epocas=input('numero de epocas: ');

for x=1:epocas
    for y=1:length(P(1,:))
      a=hardlims((W*P(:,y))+b);
      e=T(:,y)-a;
```

```
      W=W+(e*alpha*P(:,y)'); % Modificación de pesos
      b=b+e*alpha; % Modificación del valor de b
    end
end
    for k=1:2
        for y=1:length(P(1,:))
             CA(k,y)=purelin((W(k,:)*P(:,y))+b(k));

        end
    end
    G1=length(find(CA(1,:)<0));
    G2=length(find(CA(2,:)>0));
   verdadero1=T1;
   verdadero2=T2;
   falso1=T1-G1;
   falso2=T2-G2;
   P=[verdadero1 falso1
   falso2 verdadero2];
   d=(G1+G2)/(T1+T2);
   disp(sprintf('Desempeño=%5.2f%%',d*100))
intp11=-b(1,:)/W(1,1);
intp12=-b(1,:)/W(1,2);
intp13=-b(1,:)/W(1,3);
intp21=-b(2,:)/W(2,1);
intp22=-b(2,:)/W(2,2);
intp23=-b(2,:)/W(2,3);
%%
%Graficar los planos
syms x y z x2 y2 z2
A1=[intp11,0,0];B1=[0,intp12,0];C1=[0,0,intp13];
A2=[intp21,0,0];B2=[0,intp22,0];C2=[0,0,intp23];
u=B1-A1; v=C1-A1;
u2=B2-A2;v2=C2-A2;
p=[x-intp11 y-0 z-0;u(1) u(2) u(3) ;v(1) v(2) v(3)];
p2=[x2-intp21 y2-0 z2-0;
    u2(1) u2(2) u2(3);
    v2(1) v2(2) v2(3)];
EC1=det(p);
EC2=det(p2);
Z=vpa(solve(EC1,'z'),6);
Z2=vpa(solve(EC2,'z2'),6);
[x,y]=meshgrid(0:0.1:6);
[x2,y2]=meshgrid(0:0.1:6);
figure(2)
hold on
% surf(x,y,0.592683*x - 0.23321*y + 0.699018,'facecolor','r')
% surf(x2,y2,0.23653*x2 - 0.663633*y2 + 0.901827,'facecolor','c')
ezsurf(Z,[0 10 0 10])
colormap cool
ezsurf(Z2,[0 10], [0 10])
plot3(DatosS(:,1),DatosS(:,2),DatosS(:,3),'.r')
plot3(DatosVS(:,1),DatosVS(:,2),DatosVS(:,3),'.g')
plot3(DatosV(:,1),DatosV(:,2),DatosV(:,3),'.b')
grid on;
hold off
```

Graficas



Grafica 7 muestras obtenidas de diferentes tipos de iris



Grafica 8 clasificación de los datos de iris mediante una red ADALINE

Conclusiones

La red ADALINE funciona mejor que la red perceptron a pesar de tener un mayor número de muestras o tener más características, la red se adapta de muy buena manera a aprender los patrones de entrenamiento en esta práctica pudimos observar además como a pesar de que la clasificación tuvo un rendimiento por debajo del 100% aun así tuvo uno mayor que la red perceptron con las mismas épocas de entrenamiento, nuevamente es importante hacer mención que el factor de aprendizaje es muy importante para hacer más flexible la clasificación en este tipo de redes

P7. Red perceptron implementado en hardware (compuerta OR)

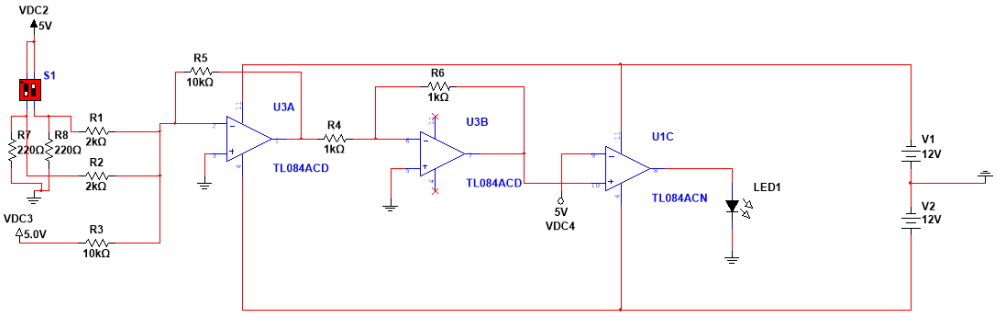
Desarrollo y propuesta de solución

Se desea implementar un red perceptron a un sistema electrónico para simular el comportamiento de una compuerta OR para ello se propone emular electrónicamente la ecuación que describe a la red perceptron por lo tanto se utilizaran los elementos que permiten realizar operaciones matemáticas y lógicas de manera analógica de modo que el dispositivo a utilizar serán amplificadores operaciones dispuestos en una configuración que emule las operaciones matemáticas descritas por la siguiente ecuación

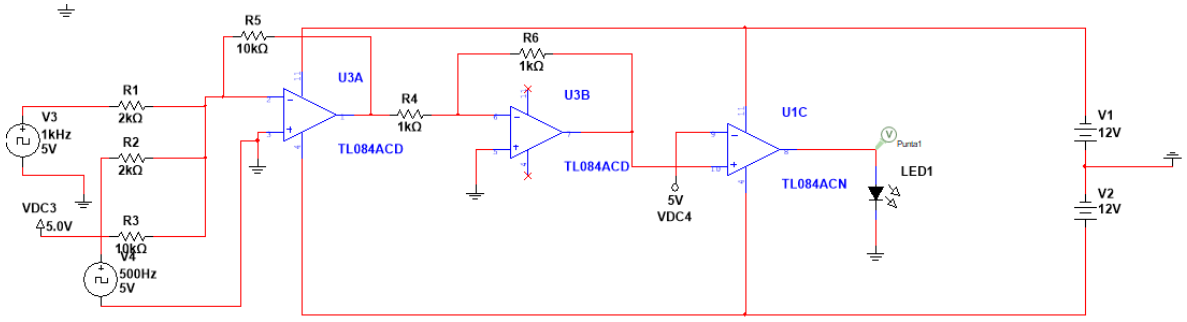
$$n=W^{T}*p^{+}b^{\wedge} \quad ec....3$$

entonces para el ingreso de los patrones se propone utilizar una serie de interruptores que generen una tensión basada en divisores de voltajes que lleguen a un sumador lo cual represente el cuerpo de la neurona en donde convergen todos los patrones la polarización se dejara fija utilizando un valor de voltaje invariante todas las tensiones que lleguen al sumador serán llevadas a un inversor con el fin de obtener el voltaje en valores aceptables una vez que tengamos la señal invertida la llevaremos a un comparador de tención que reflejara la función de activación de esta forma al encontrar una patrón que represente la clase correspondiente veremos la salida reflejada en un led y así comprobaremos el correcto funcionamiento del circuito

Simulación del circuito

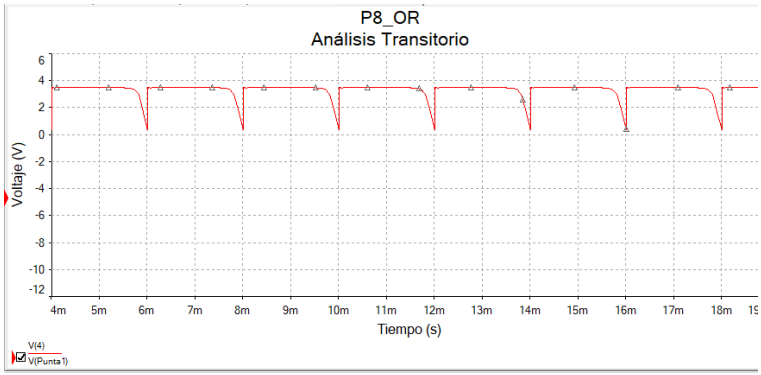


Circuito 1 circuito propuesto para representar la red ADALINE mediante hardware

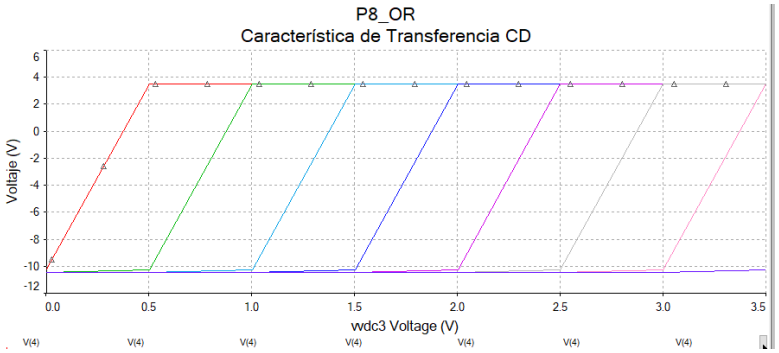


Circuito 2 circuito propuesto para realizar los análisis transitorio y de CD

Graficas



Grafica 9 análisis transitorio



Grafica 10 análisis en CD

Conclusiones

Se puede ver que una red neuronal se puede aplicar también en hardware y obtener el mismo comportamiento que en software en este circuito en la simulación pudimos corroborar como al introducir los patrones conocidos al final el led prende cuando reconoce la clase de salida en alto y se apaga cuando los patrones describen la clase de salida en bajo para esta práctica se pretendió que los divisores nos generaran un voltaje con el cual poder prender el led por lo cual se propuso que el voltaje para que se reflejara como un 1 fueran 3.5V con lo cual una alimentación basada en baterías podrían ser suficiente para que la suma de señales se haga de manera correcta y no ocasione una saturación del amplificador

REFERENCIAS

- Valls, J. 2007. Redes de Neuronas Perceptrón y Adaline.
- Torres, Luis .2008. Redes Neuronales Artificiales.
- http://www.varpa.org/~mgpenedo/cursos/scx/archivospdf/Tema3-0.pdf