Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental

P2.1. Compuerta OR red perceptron aplicada en Hardware

Marco Teórico

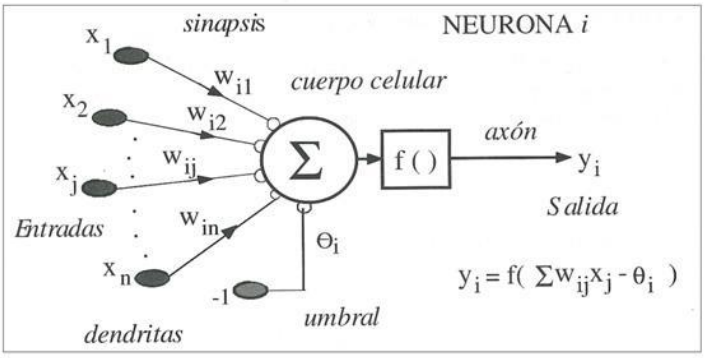
RED PERCEPTRON

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:

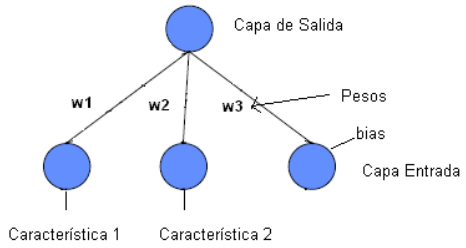
$$y_i(t) = f(\sum_{j=1}^{n} w_{ij} x_j - \theta_i), \forall i, 1 \leq i \leq m$$

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.

Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental

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 propuesta de solución

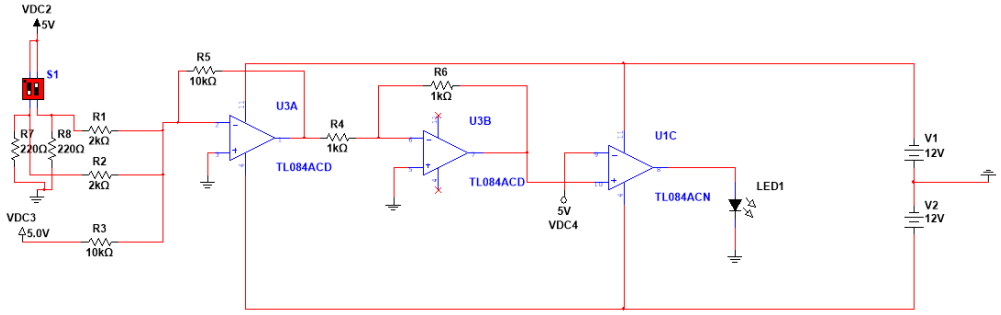
Se desea implementar un red perceptron a un sistema electrónico para simular el comportamiento de una compuerta OR para ello se propone emular electrónicamente la ecuación que describe a la red perceptron por lo tanto se utilizaran los elementos que permiten realizar operaciones matemáticas y lógicas 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

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

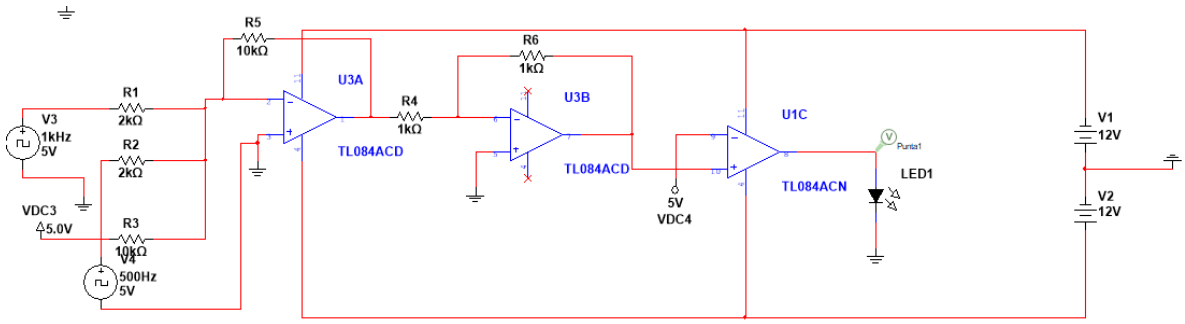
entonces para el ingreso de los patrones se propone utilizar una serie de interruptores que generen una tensión basada en divisores de voltajes que lleguen a un sumador lo cual represente el cuerpo de la neurona en donde convergen todos los patrones la polarización se dejara fija utilizando un valor 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

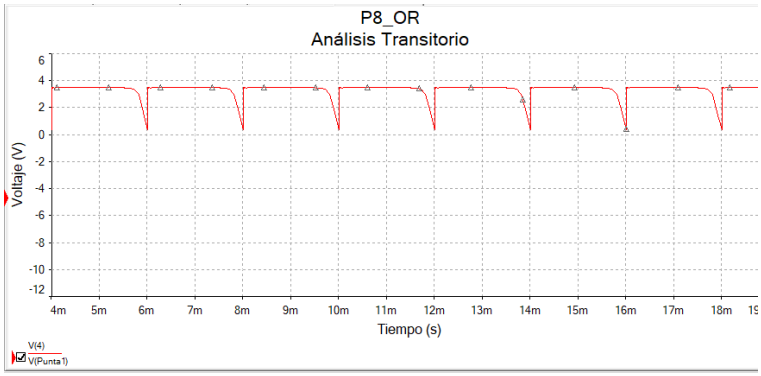


Circuito 1 circuito propuesto para representar la red ADALINE mediante hardware



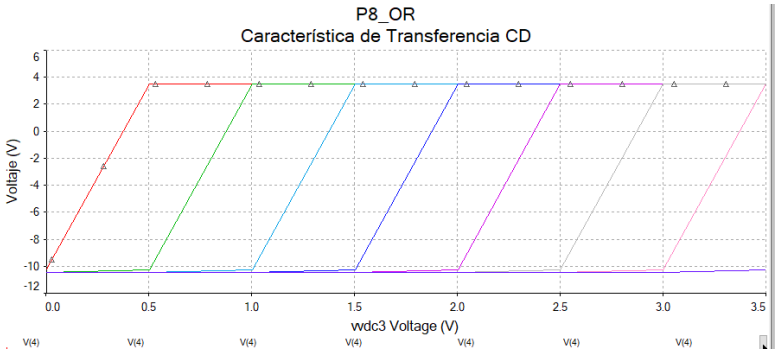
Circuito 2 circuito propuesto para realizar los análisis transitorio y de CD

Graficas

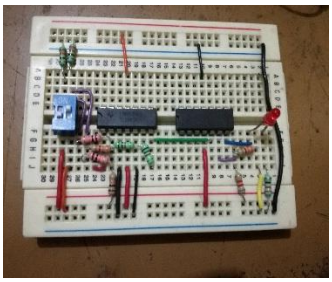


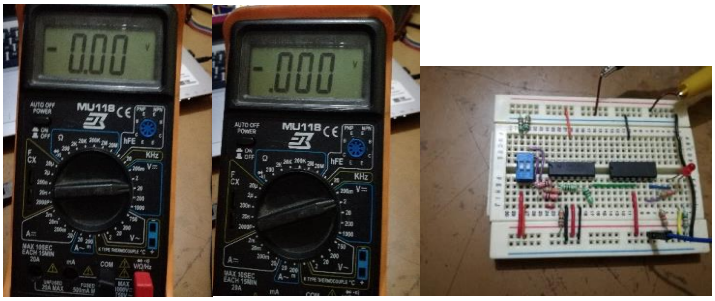
Grafica 9 análisis transitorio

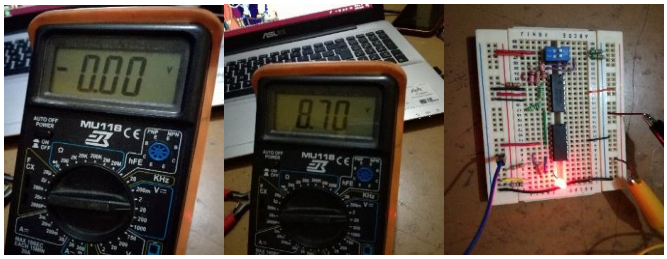
Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental

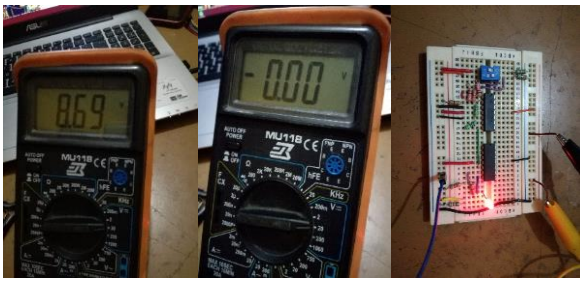


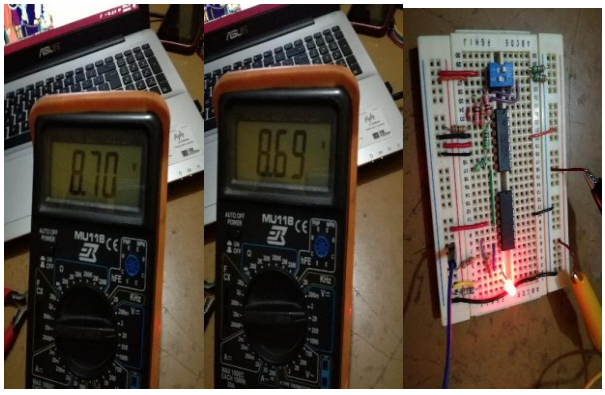
Grafica 10 análisis en CD











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

P2.3. 1 Regla de Hebbs

Desarrollo y solución propuesta

Para esta práctica se desea realizar una clasificación mediante un red asociativa basada en la regla de Hebbs en donde nuestra clasificación se hará basada en emoticones con diferentes estados de ánimo estos emoticones están plasmados en imágenes digitales por lo cual haremos el tratamiento de la imagen para poder descomponerla y obtener una señal con la cual entrenar al algoritmo , los estados de ánimo a reconocer son feliz, enojado y serio después de obtener cada una de las imágenes que usaremos como patrón se les realizo un preprocesamiento definiendo en primer lugar un tamaño fijo para las 3 imágenes de 640x480 pixels posteriormente a eso se trabajaron en escala de grises por lo cual nuestras imágenes quedaron unidimensionales posterior a eso realizamos una segmentación de la imagen dejando únicamente visible la parte de interés que eran los rasgos propios de cada emoticón que nos definían el estado de ánimo así todo lo que estuviera por encima de 118 de intensidad de color seria eliminado posterior a eso se realizaron distintas operaciones morfológicas con el fin de quitar ruido de la imagen y obtener lo más definido posible los rasgos principales

Una vez realizado l procesamiento de la imagen se procedió a realizar un barrido de la imagen por columna con el fin de encontrar la firma que describe la imagen segmentada de esta forma obtendríamos ya un vector en lugar de una matriz y dicho vector contendría las características de cada imagen basados en los pixeles que describen a la imagen además se normalizaron los resultados para tener vectores congruentes estos vectores serian entonces lo que se meterían a la regla de Hebbs para realizar la red asociativa y así definir qué target se le asignaría a cada patrón

Cabe recalcar que a estos patrones primero se orthonormalizaron como lo decide la regla de Hebbs una vez que ya están orthonormalizados se calculan los pesos para cada patrón basados en los target asignados así para cada uno de las emociones hay un target característico

Después de tener los pesos generados por el entrenamiento simplemente se procede a ingresar una nueva imagen diferente a las imágenes patrón se le hace el mismo

preprocesamiento y a esta solo se le normaliza posteriormente se evalúa en los pesos generados anteriormente de tal forma que nos arrojara una respuesta y dependiendo de la cercanía que tenga con los target propuestos será entonces la pertenencia a la emoción correspondiente en este caso se ingresó una imagen que refleja una sonrisa y al ser el resultado de la evaluación muy cerca a [1,0] entonces se le asigna la emoción "Esta Feliz"

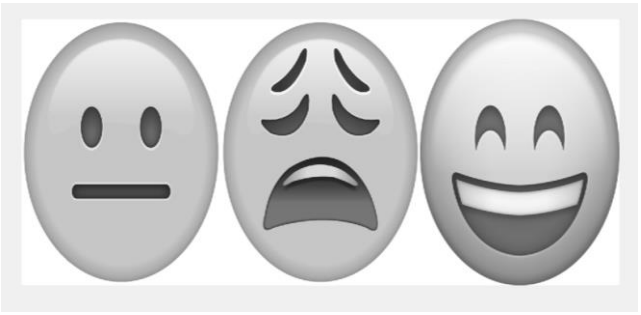
Algoritmo

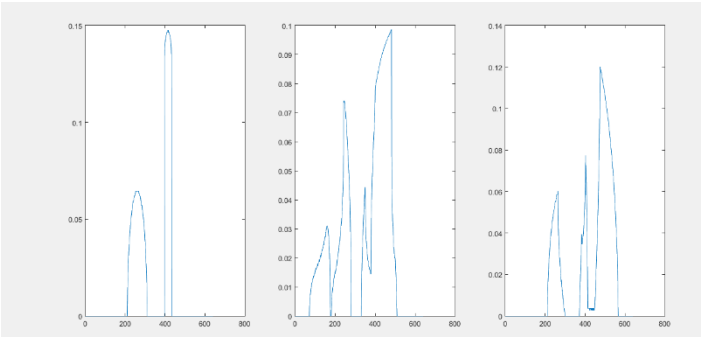
```
clc, clear all , close all ;
%REGLA DE HEBBS
I1=imresize(rgb2gray(imread('Eserio.png')),[640,480]);
I2=imresize(rgb2gray(imread('Etriste.png')),[640,480]);
I3=imresize(rgb2gray(imread('Efeliz.png')),[640,480]);
% figure
% subplot(1,3,1),imhist(I1);
% subplot(1,3,2),imhist(I2);
% subplot(1,3,3),imhist(I3);
figure
imshow([I1,I2,I3])
bin1=I1<118;
bin2=I2<118;
bin3=I3<118;
bin3=imerode(bin3,ones(6,6));
bin3=bwareaopen(bin3,200);
figure
imshow([bin1,bin2,bin3])
p1=sum(bin1,2);
p1=p1/norm(p1);
p2=sum(bin2,2);
p2=p2/norm(p2);
p3=sum(bin3,2);
p3=p3/norm(p3);
figure
subplot(1,3,1),plot(p1);
subplot(1,3,2),plot(p2);
subplot(1,3,3),plot(p3);
%%
%Regla de Hebbs
P=[p1,p2,p3];
Porth=orth(P);
T=[-1 1 1;-1,-1 1];
W=T*P';
%%
%introduccion de la imagen de prueba
IPr=imread('Eprueba.png');
Ip=imresize(rgb2gray(IPr),[640,480]);
% figure
%  imhist(Ip);
binp=Ip<118;
binp=imerode(binp,ones(6,6));
binp=bwareaopen(binp,200);
figure
imshow(binp)
Pp=sum(binp,2);
Ppn=Pp/norm(Pp);
```

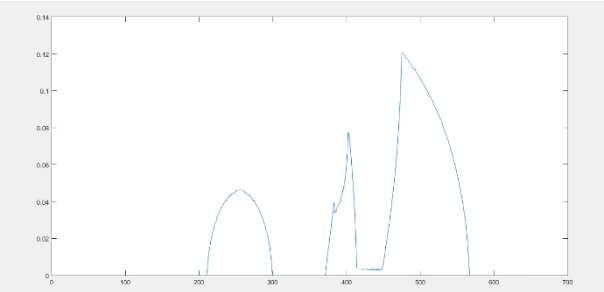
Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental

```
figure
plot(Ppn)
%%
%comprobacion de la imagen patron
a=W*Ppn;
figure
imshow(IPr)
    if a(1)<0 & a(2)<0
    clc
    disp('Estas serio')
    elseif a(1)>0 && a(2)<0
    clc
    disp('Estas triste ')
    elseif a(1)>0 & a(2)>0
    clc
    disp('Estas Feliz......')
    else
    disp('No se reconoce')
    end
```

Graficas







```
Estas Feliz......
>> a

a =

    1.2059
    0.1646

>>
```

Conclusiones

Como conclusiones se puede decir que las redes asociativas parecen funcionar como funciona la clasificación humana es decir la red asociara siempre a lo que se vea más parecido es decir al patrón de entrenamiento sin importar si son exactamente iguales al patrón podemos ver como a pesar de que el resultado de la nueva imagen no es exactamente el target propuesta si es un aproximado muy cercano y por lo tanto se le asigna a la emoción que se propuso funcionando de manera correcta para este caso especifico

P2.3. 2 Regla de Hebbs Seudoinversa

Desarrollo y solución propuesta

Esta es una variante del algoritmo anterior en donde a la imágenes se le realizó el mismo procesamiento que anteriormente y se desea clasificar las mismas tres emociones con la única diferencia de que en esta ocasión se usó en la regla de Hebbs la matriz seudoinversa de los patrones encontrados en cada emoción esto se hizo para eliminar la orthonomralizacion y poder comparar cuál de las dos formas de la regla de Hebbs era la que daba mejor resultados por lo cual este algoritmo nos sirve para poder hacer esa comparación ya que como podemos observar el resultado que arroja esa solución con la matriz

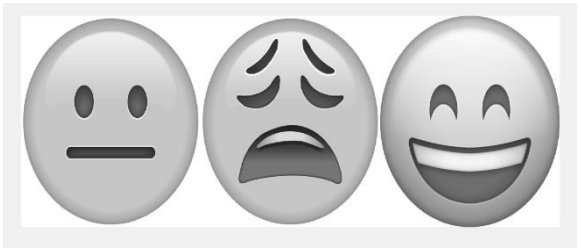
seudoinversa se aproxima más a los valores propuestos en los target y además este algoritmo hace que no importe si la imagen con la que se va a probar esta normalizada o no entonces tenemos que este algoritmo genera de igual manera una matriz de pesos con lo cual la red asociara a cada una de las expresiones de los emoticones según el parecido de los target con los valores generados en la evaluación

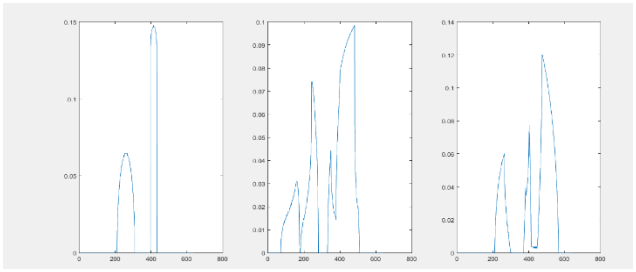
Algoritmo

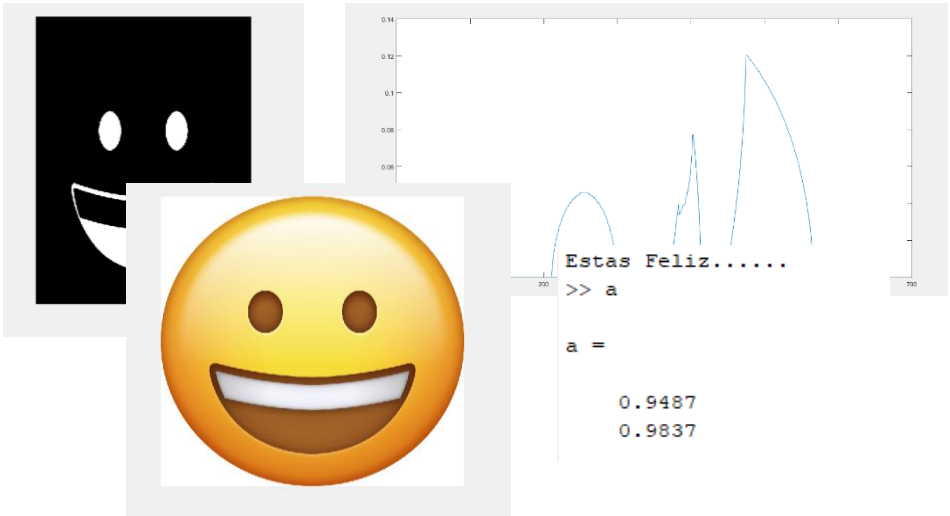
```
clc, clear all , close all ;
%REGLA DE HEBBS SEUDOINVERSA
I1=imresize(rgb2gray(imread('Eserio.png')),[640,480]);
I2=imresize(rgb2gray(imread('Etriste.png')),[640,480]);
I3=imresize(rgb2gray(imread('Efeliz.png')),[640,480]);
figure
imshow([I1,I2,I3])
bin1=I1<118;
bin2=I2<118;
bin3=I3<118;
bin3=imerode(bin3,ones(6,6));
bin3=bwareaopen(bin3,200);
figure
imshow([bin1,bin2,bin3])
p1=sum(bin1,2);
p2=sum(bin2,2);
p3=sum(bin3,2);
figure
subplot(1,3,1),plot(p1);
subplot(1,3,2),plot(p2);
subplot(1,3,3),plot(p3);
%%
%Regla de Hebbs
P=[p1,p2,p3];
% Porth=orth(P);
P=inv(P'*P)*P';
T=[-1 1 1;-1,-1 1];
W=T*P;
%%
%introduccion de la imagen de prueba
IPr=imread('Eprueba.png');
Ip=imresize(rgb2gray(IPr),[640,480]);
% figure
%  imhist(Ip);
binp=Ip<118;
binp=imerode(binp,ones(6,6));
binp=bwareaopen(binp,200);
figure
imshow(binp)
Pp=sum(binp,2);
% Ppn=Pp/norm(Pp);
figure
plot(Pp)
%%
%comprobacion de la imagen patron
a=W*Pp;
figure
```

```
imshow(IPr)
    if a(1)<0 & a(2)<0
    clc
    disp('Estas serio')
    elseif a(1)>0 && a(2)<0
    clc
    disp('Estas triste ')
    elseif a(1)>0 & a(2)>0
    clc
    disp('Estas Feliz......')
    else
    disp('No se reconoce')
    end
```

Graficas







Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental

Conclusiones

Se observa claramente que la regla de Hebbs por seudoinversa en mucha más efectiva por los resultados que genera la evaluación que es más parecida a los target propuestos y debido a que evita un tratamiento a cada imagen que queramos probar en nuestra red

P2.4 .1Regla Instar 20 muestras

Desarrollo y solución propuesta

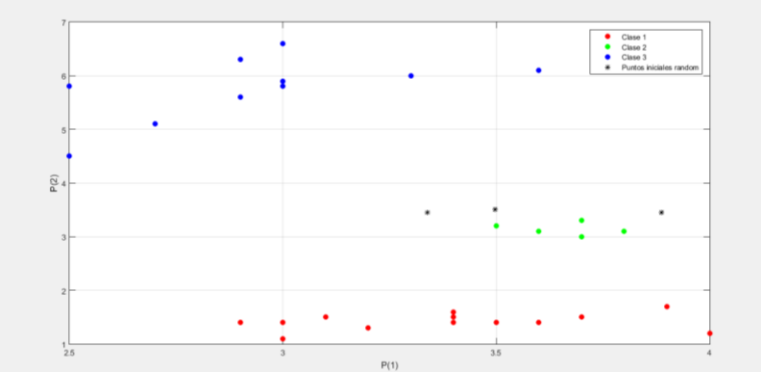
En esta práctica se desea encontrar los centros de cada una de las clases generadas por un distribución de datos de dos características con un total de 30 datos estas distribuciones se pueden observar pero en realidad no se sabe cuál es el centro característico de cada una así que se usa la regla instar basada en las redes por competencia de la forma ganador toma todo con la cual se desea encontrar los centros representativos para ello primero se generaron centros iniciales Random con los cuales comenzaríamos a calcular distancia euclidiana con cada uno de los puntos de nuestra distribución de este modo se actualizarían los pesos según sea la distancia más corta de estos centros iniciales con cada uno de los datos de modo que entre más se acerque se actualizara el centro de clase se disminuirá el valor de alfa y generaremos un nuevo centro esto se hará de manera iterativa hasta que el factor de aprendizaje sea mínimo de tal suerte que al final cuando no se pueden actualizar más los pesos y los puntos centros estos últimos serán finalmente los que representen a cada clase cuando el valor de alfa sea casi cero

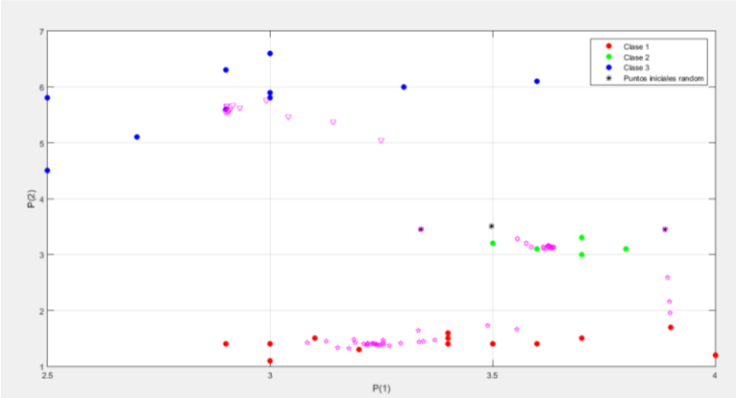
Algoritmo

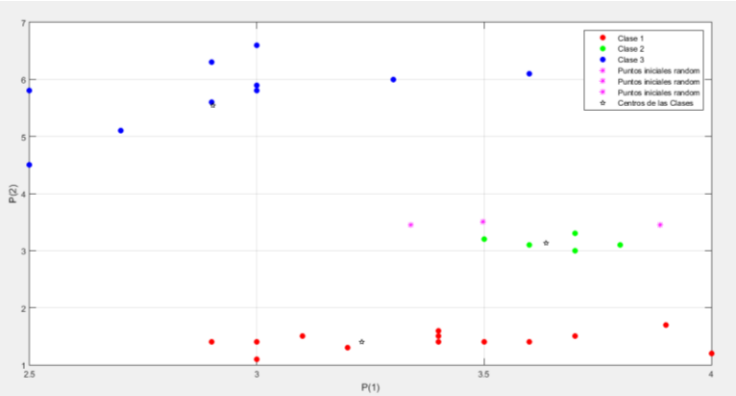
```
clc,clear all,close all
dat = fopen('Datacomp.dat');
Datacomp = textscan(dat, '%f %f %s');
fclose(dat);
Datos(:,1)=Datacomp{1};
Datos(:,2)=Datacomp{2};
DatosS=Datos(1:15,:);
DatosVS=Datos(16:20,:);
DatosV=Datos(21:30,:);
figure(1)
plot(DatosS(:,1),DatosS(:,2),'ro','markerFaceColor','r')
hold on
plot(DatosVS(:,1),DatosVS(:,2),'go','markerFaceColor','g')
plot(DatosV(:,1),DatosV(:,2),'bo','markerFaceColor','b')

grid on;
xy=Datos';
cm=sum(xy,2)/length(xy(1,:));
w=rand(2,3);
w(1,:)=w(1,:)+cm(1);
```

```
w(2,:)=w(2,:)+cm(2);
alph=.5;
plot(w(1,1),w(2,1),'*k',w(1,2),w(2,2),'*k',w(1,3),w(2,3),'*k')
xlabel('P(1)'),ylabel('P(2)'),legend('Clase 1','Clase 2','Clase
3','Puntos iniciales random')
C(1,:)=w(1,:);
C(2,:)=w(2,:);
pause;
while alph>.01
    for i=1:30
  numr=round(rand(1)*29)+1;
  a=max(compet((sum(w.*w)'-(2*w'*xy(:,numr))+(xy(:,numr)'*xy(:,numr)))*-
1).*[1:3]');
  w(:,a)=w(:,a)+alph*(xy(:,numr)-w(:,a)); %modifico los pesos  de la
neurona ganadora
  alph=alph-.01; %decremento el valor de alpha
  plot(w(1,1),w(2,1),'pm',w(1,2),w(2,2),'hm',w(1,3),w(2,3),'vm') %grafico
el movimiento que van teniendo los pesos
  pause(.06) %doy tiempo para que se despliegue lo anterior
    end
end
figure
plot(DatosS(:,1),DatosS(:,2),'ro','markerFaceColor','r')
hold on
plot(DatosVS(:,1),DatosVS(:,2),'go','markerFaceColor','g')
plot(DatosV(:,1),DatosV(:,2),'bo','markerFaceColor','b')
grid on;
plot(C(1,1),C(2,1),'*m',C(1,2),C(2,2),'*m',C(1,3),C(2,3),'*m')
plot(w(1,1),w(2,1),'pk',w(1,2),w(2,2),'pk',w(1,3),w(2,3),'pk')
xlabel('P(1)'),ylabel('P(2)'),legend('Clase 1','Clase 2','Clase 3',...
'Puntos iniciales random','Puntos iniciales random','Puntos iniciales
random','Centros de las Clases ')
```

Graficas







Conclusiones

En esta práctica se observó que efectivamente a partir de puntos aleatorios es posible con la competencia ganador toma todo generar centros que caractericen cada clase presente sin embargo es importante recalcar que para espacios más grandes en donde la distribución no es tan uniforme el tener pesos iniciales muy aleatorios puede hacer que no se actualicen los centros y dar una aproximación errónea a lo que en realidad podría representar cada centro por otro lado en eser caso no se dio la situación y funciono de manera correcta el algoritmo a pesar de que los valores iniciales fueran meramente aleatorios

P2.4.2 Regla instar iris

Desarrollo y solución propuesta

En esta caso se desea realizar lo mismo que en la práctica anterior solamente que en este caso se realizan algunas modificaciones para empezar tenemos una distribución diferente a que ahora se desea en contra los centros de las clases de una distribución que contiene tres características esta distribución es la ya conocida distribución del iris que nos da las características de tres tipos distintos de iris por lo cual entonces tendremos que encontrar 3 centros nuevamente pero ahora en un espacio tridimensional por decirlo de alguna manera se procede de la misma forma que la distribución anterior se entrena bajo una red basada en competencia ganador toma todo se proponen valores iniciales de centros de donde el

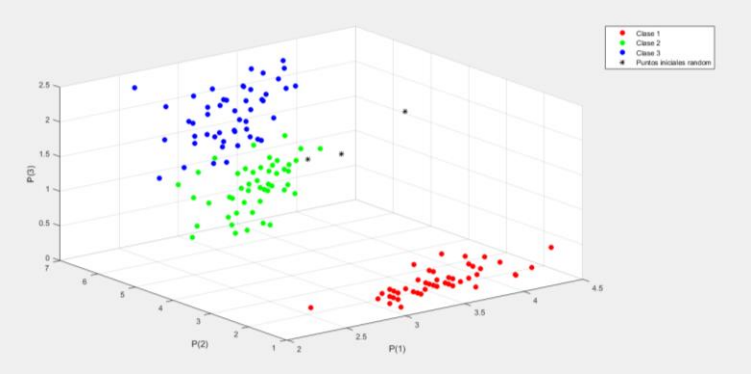
algoritmo partirá a recorrer los centros según se encuentren más cerca de cada uno de los datos de las muestras de tal forma que nuevamente se recurre a la distancia euclidiana y se ponen a competir a las 3 clases la que encuentre más cerca gana y por lo tanto es la que se actualiza hasta que el valor de alfa se encuentre muy bajo los pesos no dejaran de actualizarse así al final obtendremos los centros característicos de cada una de las clases

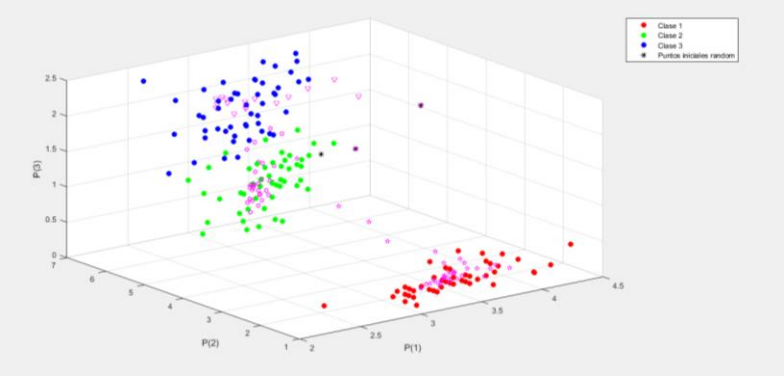
Algoritmo

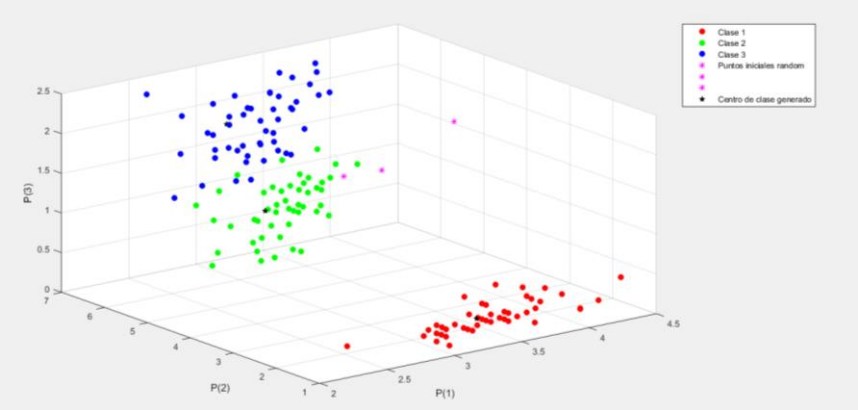
```
clc,clear all,close all
dat = fopen('IrisDataBase.dat');
IrisDataBase=textscan(dat, '%f %f %f %f %s');
fclose(dat);
Datos(:,1)=IrisDataBase{2};
Datos(:,2)=IrisDataBase{3};
Datos(:,3)=IrisDataBase{4};
DatosS=Datos(1:50,:);
DatosVS=Datos(51:100,:);
DatosV=Datos(101:150,:);

figure
plot3(DatosS(:,1),DatosS(:,2),DatosS(:,3),'ro','markerFaceColor','r')
hold on
plot3(DatosVS(:,1),DatosVS(:,2),DatosVS(:,3),'go','markerFaceColor','g')
plot3(DatosV(:,1),DatosV(:,2),DatosV(:,3),'bo','markerFaceColor','b')
grid on;
xy=Datos';
cm=sum(xy,2)/length(xy(1,:));
w=rand(3,3);
w(1,:)=w(1,:)+cm(1);
w(2,:)=w(2,:)+cm(2);
w(3,:)=w(3,:)+cm(3);
alph=.5;
plot3(w(1,1),w(2,1),w(3,1),'*k',w(1,2),w(2,2),w(3,2),'*k',w(1,3),w(2,3),w
(3,3),'*k')
xlabel('P(1)'),ylabel('P(2)'),zlabel('P(3)')...
    ,legend('Clase 1','Clase 2','Clase 3','Puntos iniciales random')
C(1,:)=w(1,:);
C(2,:)=w(2,:);
C(3,:)=w(3,:);
pause;
while alph>.1
  numr=round(rand(1)*149)+1;
  a=max(compet((sum(w.*w)'-(2*w'*xy(:,numr))+(xy(:,numr)'*xy(:,numr)))*-
1).*[1:3]');
  w(:,a)=w(:,a)+alph*(xy(:,numr)-w(:,a)); %modifico los pesos  de la
neurona ganadora
  alph=alph-.005; %decremento el valor de alpha

plot3(w(1,1),w(2,1),w(3,1),'pm',w(1,2),w(2,2),w(3,2),'hm',w(1,3),w(2,3),w
(3,3),'vm') %grafico el movimiento que van teniendo los pesos
  pause(.06) %doy tiempo para que se despliegue lo anterior
```

```
end
figure
plot3(DatosS(:,1),DatosS(:,2),DatosS(:,3),'ro','markerFaceColor','r')
hold on
plot3(DatosVS(:,1),DatosVS(:,2),DatosVS(:,3),'go','markerFaceColor','g')
plot3(DatosV(:,1),DatosV(:,2),DatosV(:,3),'bo','markerFaceColor','b')
grid on;
plot3(C(1,1),C(2,1),C(3,1),'*m',C(1,2),C(2,2),C(3,2),'*m',C(1,3),C(2,3),C
(3,3),'*m')
plot3(w(1,1),w(2,1),w(3,1),'pk',w(1,2),w(2,2),w(3,2),'pk',w(1,3),w(2,3),w
(3,3),'pk','markerFaceColor','k')
xlabel('P(1)'),ylabel('P(2)'),zlabel('P(3)')...
 ,legend('Clase 1','Clase 2','Clase 3','Puntos iniciales
random','','','Centro de clase generado')
```

Graficas







Conclusiones

Se podría decir que en esta práctica lo único que hay que agregar es que el algoritmo de la regla instar se puede aplicar a cualquier tipo de distribución de tal suerte que los centros de clases de datos se pueden encontrar siempre y cuando podamos hacer que la actualización de pasos y centros se haga de manera correcta en este caso nuestra distribución se puede ver de manera visual cada uno de los conjunto pero en el caso en el que se desee encontrar los pesos de una distribución más homogénea es posible que el algoritmo tienda a generar ciertos errores

P2.5.1Kohonen 30 muestras

Desarrollo y solución propuesta

En esta práctica se pretende encontrar la solución para encontrar los centros descriptivos de la distribución que consta de 30 muestras para ello se usa una red competitiva del tipo ganador comparte todo se generan pesos semialatorios muy cercanos a donde está la distribución de datos y se generan pesos iniciales por donde comenzaremos a buscar la distancia más corta de un punto en la distribución con respecto a los centros iniciales luego de este se actualizarán los pesos según la distancia más corta solo que en esta ocasión al ser una red ganador comparte se usara una función gaussiana en la que se evaluara la distancia y a su vez esta no arrojara un valor de la razón de aprendizaje que cada una de los centros debe de tener y esta misma ira disminuyendo según se hacer que el centro propuesto al centro real de tal manera que la campana se cerrara más y los valores que queden dentro al final serán lo que más se acerquen al centro real así obtendremos la aproximación más certera de cuáles son los centros que representan a cada uno de los conjuntos

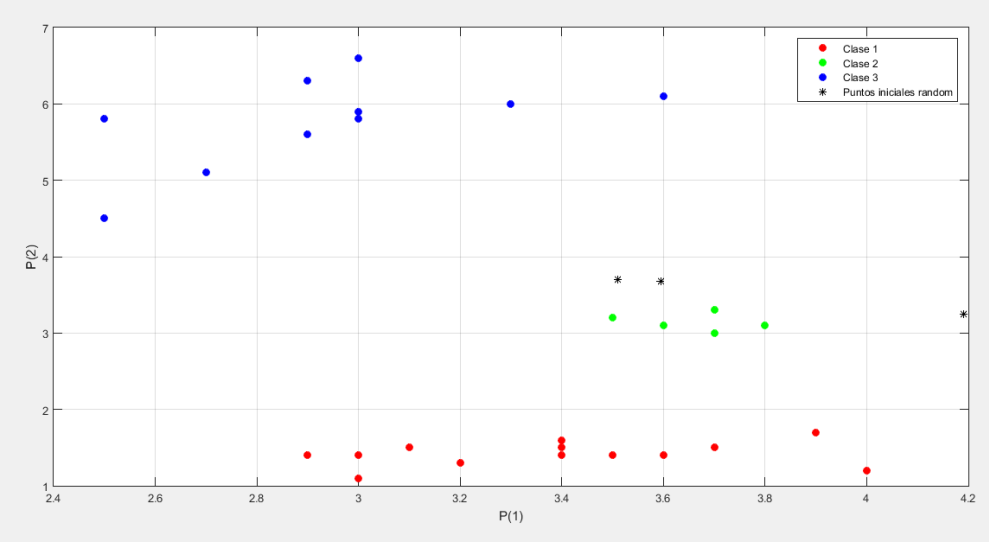
Algoritmo

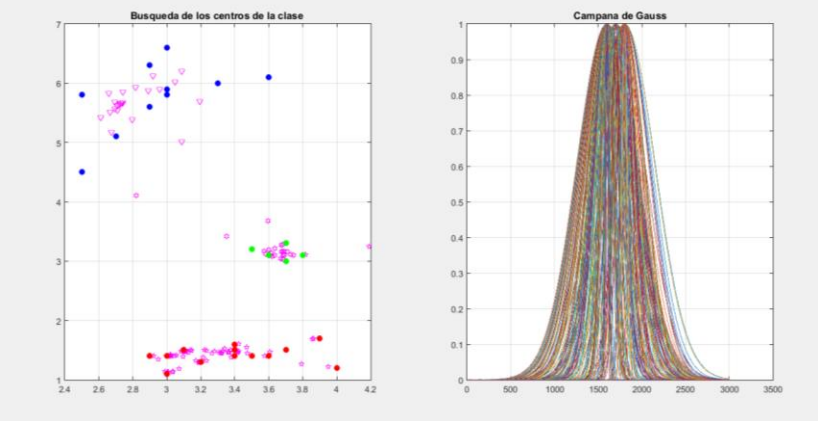
```
clc,clear all,close all
dat = fopen('Datacomp.dat');
```

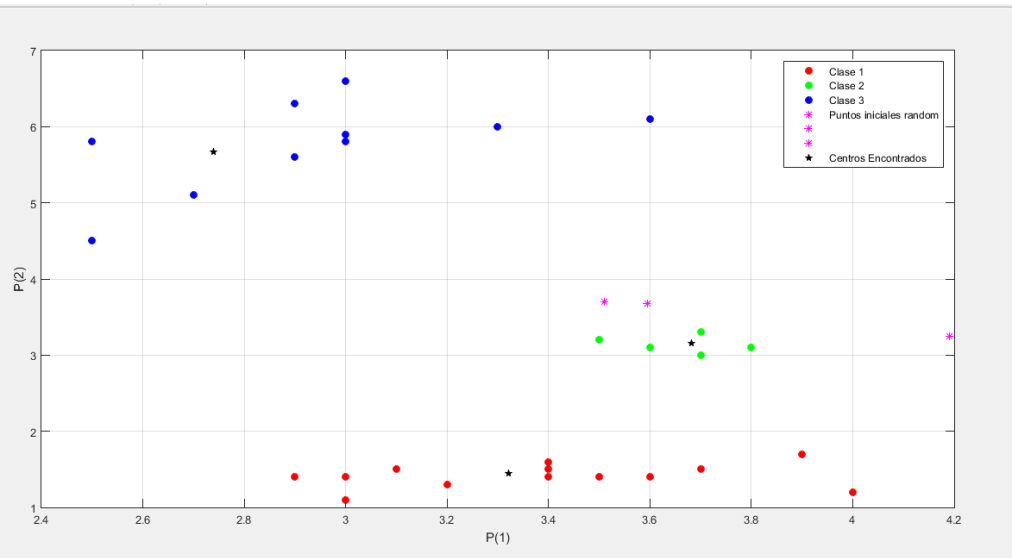
```
Datacomp = textscan(dat, '%f %f %s');
fclose(dat);
Datos(:,1)=Datacomp{1};
Datos(:,2)=Datacomp{2};
DatosS=Datos(1:15,:);
DatosVS=Datos(16:20,:);
DatosV=Datos(21:30,:);
figure(1)
plot(DatosS(:,1),DatosS(:,2),'ro','markerFaceColor','r')
hold on
plot(DatosVS(:,1),DatosVS(:,2),'go','markerFaceColor','g')
plot(DatosV(:,1),DatosV(:,2),'bo','markerFaceColor','b')
grid on;
xy=Datos';
cm=sum(xy,2)/length(xy(1,:));
w=rand(2,3);
w(1,:)=w(1,:)+cm(1);
w(2,:)=w(2,:)+cm(2);
alph=.5;
plot(w(1,1),w(2,1),'*k',w(1,2),w(2,2),'*k',w(1,3),w(2,3),'*k')
xlabel('P(1)'),ylabel('P(2)')...
 ,legend('Clase 1','Clase 2','Clase 3','Puntos iniciales random')
C(1,:)=w(1,:);
C(2,:)=w(2,:);
s=-15:0.01:15;
pause;
sigm=3.5;
pause();
while sigm>.1
  numr=round(rand(1)*29)+1;
  a=max(compet((sum(w.*w)'-(2*w'*xy(:,numr))+(xy(:,numr)'*xy(:,numr)))*-
1).*[1:3]');
  multp=exp((-1/2)*(a/sigm).^2);
  B=exp((-1/2)*((s-a)/sigm).^2);
  w(:,a)=w(:,a)+multp*(xy(:,numr)-w(:,a)); %modifico los pesos  de la
neurona ganadora
  sigm=sigm-.025;
  hold on

subplot(1,2,1),plot(w(1,1),w(2,1),'pm',w(1,2),w(2,2),'hm',w(1,3),w(2,3),'
vm') %grafico el movimiento que van teniendo los pesos
  hold on
  subplot(1,2,1),
plot(DatosS(:,1),DatosS(:,2),'ro','markerFaceColor','r')

subplot(1,2,1),plot(DatosVS(:,1),DatosVS(:,2),'go','markerFaceColor','g')
  subplot(1,2,1),plot(DatosV(:,1),DatosV(:,2),'bo','markerFaceColor','b')
  title('Busqueda de los centros de la clase ')
  grid on
  subplot(1,2,2),plot(B),title('Campana de Gauss')
  grid  on
  pause(.06) %doy tiempo para que se despliegue lo anterior
end
figure
plot(DatosS(:,1),DatosS(:,2),'ro','markerFaceColor','r')
hold on
```

```
plot(DatosVS(:,1),DatosVS(:,2),'go','markerFaceColor','g')
plot(DatosV(:,1),DatosV(:,2),'bo','markerFaceColor','b')
grid on;
plot(C(1,1),C(2,1),'*m',C(1,2),C(2,2),'*m',C(1,3),C(2,3),'*m')
plot(w(1,1),w(2,1),'pk',w(1,2),w(2,2),'pk',w(1,3),w(2,3),'pk','markerFace
Color','k')
xlabel('P(1)'),ylabel('P(2)')...
 ,legend('Clase 1','Clase 2','Clase 3','Puntos iniciales
random','','','Centros Encontrados')
```

Graficas







Conclusiones

La red basadas en competencia comparte todo no resultan de los más eficientes presentan algunos problemas para estos casos principalmente debido a que la actualización de las clases se pueden hacer de manera incorrectas si hay clases que se acerquen en algunos puntos de manera muy próxima por otro lado se puede ver como la principal diferencia con la regla instar es como se hace la actualización de los pesos mencionados anteriormente ya que en la regla instar se hace uno a uno y en kohonen se hacen todos a la vez

P2.5.2 Kohonen iris

Desarrollo y solución propuesta

De igual manera que en la práctica anterior en esta ocasión se desea encontrar los centros de las clases presentes en la distribución de datos mediante una red por competencia del tipo ganador reparte todo básicamente en esta práctica se hace lo mismo que en la anterior solamente se le agrega una característica a la distribución de datos por lo demás el entrenamiento es el mismo se proponen un valor de sigma con el cual comenzara la campana gaussiana a evaluar las distancia generadas por los centros iniciales y las distribuciones de modo que en esta ocasión la actualización se hará de manera general actualizando al mismo tiempo para todas las clases y el valor de beta nos dirá la razón de aprendizaje a la que cada clase va de sus centro teniendo entonces una 3 valores distintos para estas razones de aprendizaje que irán disminuyendo según la distribución pueda entrar en la función gaussiana así al final hasta que valor de sigma de la campana sea lo suficientemente pequeña el algoritmo dejara de actualizar los pesos

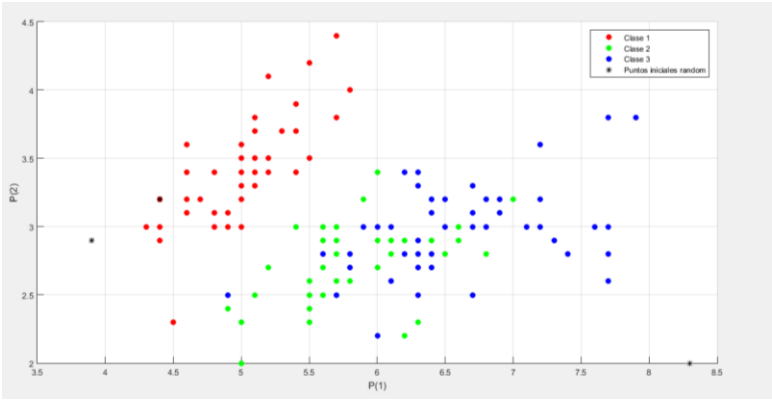
Algoritmo

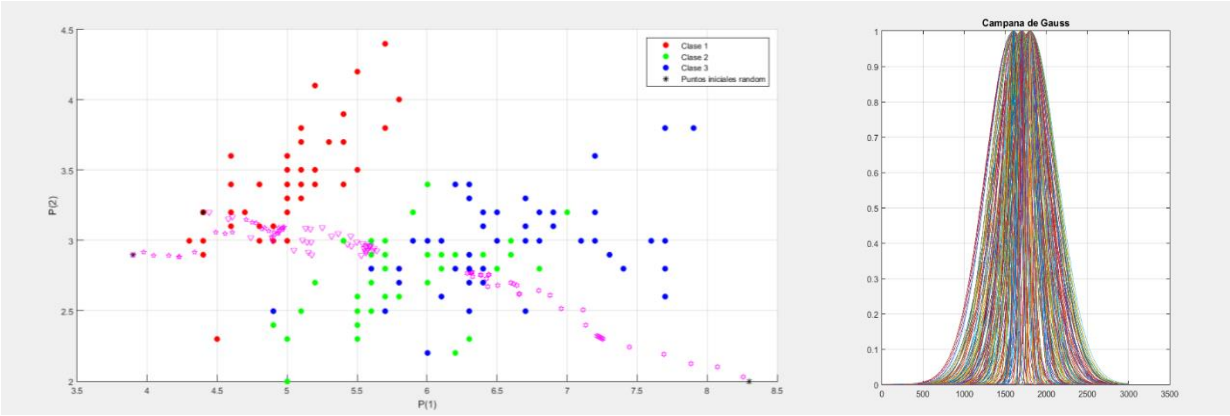
```
clear all;close all;clc;
dat = fopen('IrisDataBase.dat');
IrisDataBase = textscan(dat, '%f %f %f %f %s');
```

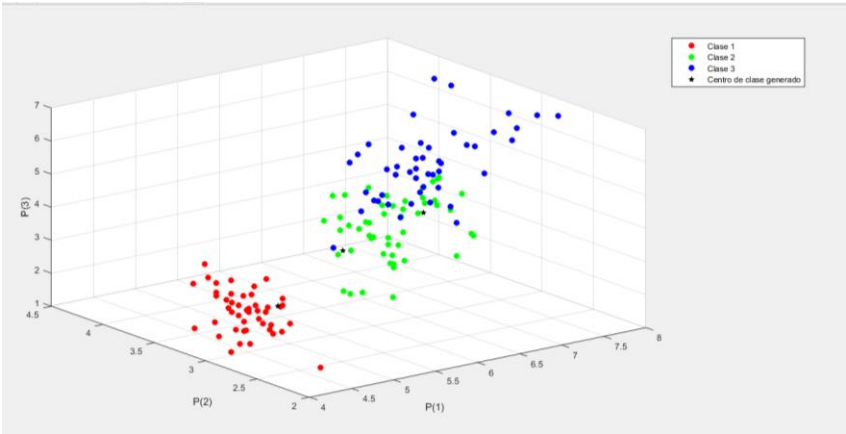
```
sigm=5; % inicializo el valor de sigma
fclose(dat);
Datos(:,1)=IrisDataBase{1};
Datos(:,2)=IrisDataBase{2};
Datos(:,3)=IrisDataBase{3};
DatosS=Datos(1:50,:);%Datos Iris-setosa
DatosVS=Datos(51:100,:); %Datos Iris-versicolour
DatosV=Datos(101:150,:);%Datos Iris-virginica
figure
hold on;
plot3(DatosS(:,1),DatosS(:,2),DatosS(:,3),'or','markerFaceColor','r')
plot3(DatosVS(:,1),DatosVS(:,2),DatosVS(:,3),'og','markerFaceColor','g')
plot3(DatosV(:,1),DatosV(:,2),DatosV(:,3),'ob','markerFaceColor','b')
grid on;
xy=Datos';% guardo la transpuesta de los datos para facilitar los
calculos
w=[3.9,8.3,4.4;
    2.9,2,3.2;
    1.5,3.8,3.4] ;%se inicializan los pesos
alph=.2; % se inicializa el valor de alpha
plot3(w(1,1),w(2,1),w(3,1),'*k',w(1,2),w(2,2),w(3,2),'*k',w(1,3),w(2,3),w
(3,3),'*k') % se grafican los pesos iniciales
xlabel('P(1)'),ylabel('P(2)'),zlabel('P(3)')...
 ,legend('Clase 1','Clase 2','Clase 3','Puntos iniciales random')
pause();
while alph>.01
   numr=round(rand(1)*(length(xy(1,:))-1))+1; %se selecciona al azar uno
de los datos
   a=(sum(w.*w)'-(2*w'*xy(:,numr))+(xy(:,numr)'*xy(:,numr))); %se
calculan las distancias de los pesos al punto seleccionado anteriormente
   Beta=[1;1;1]*exp((-1/2)*(a/sigm).^2)'; %se hace el calculo de la
distribucion normal
   w=w+(alph*Beta.*(([1;1;1]*xy(:,numr)')'-w)); %se modifican los pesos
segun los resultados anteriores
  B=Beta(1,:)
  alph=alph-.005; %se va decrementando alpha

plot3(w(1,1),w(2,1),w(3,1),'pm',w(1,2),w(2,2),w(3,2),'hm',w(1,3),w(2,3),w
(3,3),'vm') % se grafica el movimiento que tienen los pesos
  pause(.01) %se da tiempo para que se despliegue la animacion en
pantalla
  sigm=sigm-.025;% se decrementa el valor de sigma
end
figure
plot3(xy(1,1:50),xy(2,1:50),xy(3,1:50),'or','markerFaceColor','r')
hold on
plot3(xy(1,51:100),xy(2,51:100),xy(3,51:100),'og','markerFaceColor','g')
plot3(xy(1,101:150),xy(2,101:150),xy(3,101:150),'ob','markerFaceColor','b
')
plot3(w(1,1),w(2,1),w(3,1),'pk',w(1,2),w(2,2),w(3,2),'pk',w(1,3),w(2,3),w
(3,3),'pk','markerFaceColor','k')
grid on;
xlabel('P(1)'),ylabel('P(2)'),zlabel('P(3)')...
 ,legend('Clase 1','Clase 2','Clase 3','Centro de clase generado')
```

Graficas

Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental








```
    0.9969    0.0207    0.1951

B =

    0.0909    0.9958    0.9684

B =

    0.9352    0.0003    0.0183
```

Conclusiones

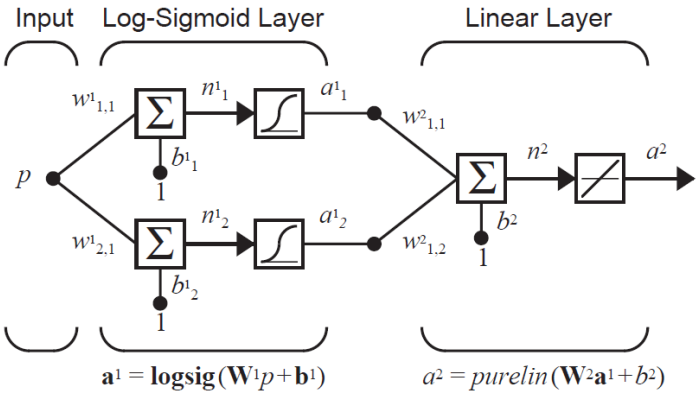
A pesar de que se utiliza una función gaussiana en lugar de solo disminuir un valor significativo de la alfa este algoritmo resulta un poco más complicado de aproximarse a los centros reales de cada uno de los conjuntos debido a que la misma función puede hacer que entren datos de más y entonces se desvié el centro que le pertenece a una clase a otra a pesar

de ello se pudo obtener una aproximación muy buena dejando claro en la evaluación final de las razones de aprendizaje para cada clase que al menos en 2 tienen muy cercano a valores de 0

P2.6.1 Backpropagation función

Desarrollo y solución propuesta

Se desea aproximar la función $1+\sin(\frac{\pi}{4}x)$ mediante una red neuronal basada en el algoritmo de retropropagacion para lo cual primero generamos los pesos y polarizaciones que están propuestos para la arquitectura siguiente



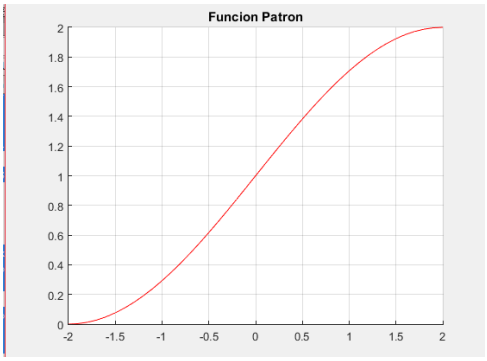
Estos pesos y polarizaciones nuevamente se hacen de manera aleatoria continuamos generando los patrones que deberá aprender la red los cuales son el barrido de números que se pueden evaluar en la función este barrido se hizo desde -2 hasta 2 con un incrementado 0.5 en cada dato luego proponemos nuestro factor de aprendizaje con un valor de alfa bajo de 0.08 iniciamos un vector en donde guardaremos el error cuadrático medio y se comienza la propagación del aprendizaje en donde actualizaremos los pesos y las polarizaciones a partir del error que genera con respecto al target que es el a misma función evaluada en el patrón que corresponde en el barrido aunado a esto se hace el uso de la sensibilidad basada en la derivada de la función tangente sigmoidea que es una función que es fácilmente derivable por lo cual las sensibilidad del algoritmo se dejan fijas y se puede entonces ahora actualizar de atrás hacia adelante los pesos y las polarizaciones el sistema se hace por 100 épocas y en cada época se guarda el error cuadrático con lo cual podremos observar el comportamiento de la red y como interactúa el aprendizaje de la función patrón finalmente se evalúan los puntos propuestos en los pesos generados por la red y obtenemos una respuesta muy similar a la función que se deseaba aprender

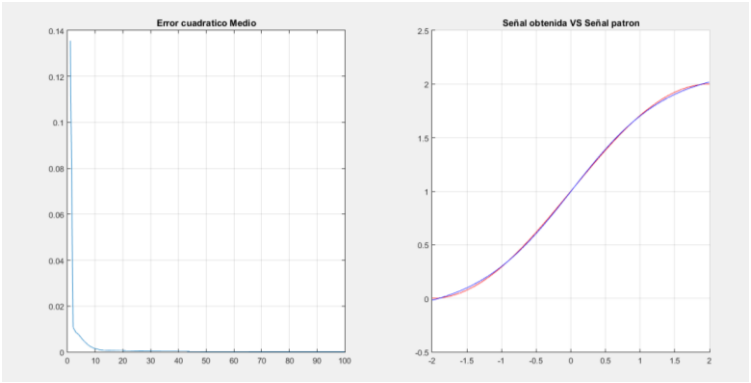
Algoritmo

```
clc,clear all,close all
w1=rand(2,1);
b1=rand(2,1);
w2=rand(2,1);
b2=rand;

patron=-2:.5:2;
```

```
patron=patron';
alpha=0.08;
for epoca=1:100
    LMS=0;
for i=1:length(patron)
    a0=patron(i);
    n1=w1*a0+b1;
    a1=tansig(n1);

    n2=w2'*a1+b2;
    a2=purelin(n2);

    % Calculamos el error.
    t=1+sin((pi/4)*patron(i));
    e=t-a2;
    % Calculamos la sensitividad.
    s2=-2*1*e;
    s1=diag(((1-a1).*a1))*w2*s2;
    % Actualizamos las w's y b's
    w2=w2-(alpha*s2*a1);
    b2=b2-(s2*alpha);

    w1=w1-(alpha*s1*a0);
    b1=b1-(s1*alpha);
     LMS= e^2 + LMS;
  end
 Emedio(epoca) = LMS/length(patron);

end
%entrenamiento
x = -2:0.1:2;
s = 1 + sin((pi/4)*x);
figure, hold on, plot(x,s,'r'), hold off,grid on,title('Funcion Patron')
figure, subplot(1,2,1), plot(Emedio),grid on,title('Error cuadratico
Medio')
for i=1:length(x)
    n1 = (w1*x(i))+b1;
    a1 = tansig(n1);
    a2(i) = (w2'*a1)+b2;
end
 subplot(1,2,2), hold on, plot(x,s,'r'),plot(x,a2,'b'), hold off,grid
on,title('Señal obtenida VS Señal patron')
```

Graficas





Conclusiones

P2.6.2 Backpropagation XOR

Desarrollo y solución propuesta

Se desea solucionar el problema de la descripción de la compuerta XOR que no se puede resolver de manera correcta con el algoritmo perceptron o Adaline debido a que este problema no implica una separabilidad lineal por ello se usa el algoritmo backpropagation con el cual se pretende mediante el uso de la función tangente sigmoidea encontrar la solución. dan los pesos iniciales se genera la matriz de patrones y se dan los valores que corresponden al target con el cual se puede ya utilizar el algoritmo y debido a que la función de activación es una función fácilmente derivable el sistema tiende a encontrar una solución mediante la actualización de pesos y polarizaciones realizando la correcta propagación del aprendizaje para esta red se toma igualmente el error cuadrático medio como medio para comprobar el correcto aprendizaje así al final además de ver gráficamente la correcta clasificación podemos observar el error tendiendo a cero en cada una de las épocas para al final de 1000 épocas ser 0 durante un periodo indefinido con lo cual obtendremos la certeza de que aprendió de correcta firmas así al evaluar los pesos en los patrones obtendremos la respuesta prácticamente igual a los target propuestos con lo cual finalmente tenemos descrito el comportamiento de la compuerta XOR

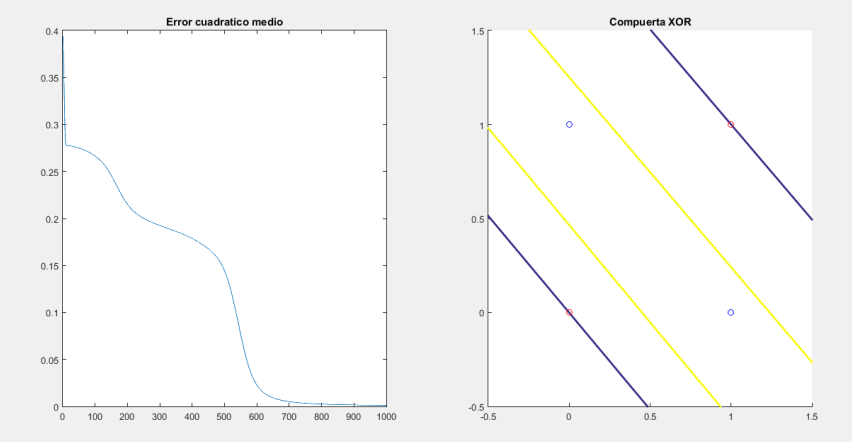
Algoritmo

```
clc,clear all,close all
% Patrones de aprendizaje y objetivos
```

```
P = [0 0 1 1;
     0 1 0 1];
T = [0 1 1 0];
Q = length(P);
% Valores iniciales
W1=rand(2,2);
b1=rand(2,1);
W2=rand(1,2);
b2=rand;
alfa = 0.05;
for Epocas = 1:1000
    sum = 0;
    for q = 1:Q
        % Propagación de la entrada hacia la salida
        a1 = tansig(W1*P(:,q) + b1);
        a2(q) = tansig(W2*a1 + b2);

        e = T(q)-a2(q);
        s2 = -2*(1-a2(q)^2)*e;
        s1 = diag(1-a1.^2)*W2'*s2;

        W2 = W2 - alfa*s2*a1';
        b2 = b2 - alfa*s2;
        W1 = W1 - alfa*s1*P(:,q)';
        b1 = b1 - alfa*s1;
        % Sumando el error cuadratico
        sum = e^2 + sum;
    end
    % Error cuadratico medio
    emedio(Epocas) = sum/Q;
end
figure, subplot(1,2,1), plot(emedio),title('Error cuadratico medio')

% Verificación de la respuesta de la multicapa
for q = 1:Q
    a(q) = tansig(W2*tansig(W1*P(:,q) + b1)+ b2);
end
a
% Frontera de decisión
u = linspace(-2, 2, 100);
v = linspace(-2, 2, 100);
for i = 1:length(u)
    for j = 1:length(v)
        z(i,j) = tansig(W2*tansig(W1*[u(i); v(j)] + b1) + b2);
    end
end
subplot(1,2,2), hold on, contour(u, v, z',[-0.9, 0, 0.9],'LineWidth', 2)
axis([-0.5 1.5 -0.5 1.5]), plot(P(1,[1,4]),P(2,[1,4]),'ro',
P(1,[2,3]),P(2,[2,3]),'bo'),title('Compuerta XOR')
```

Graficas

Mendoza Hernandez Yair Alberto
3BM4
Segundo departamental




```
a =

    0.0035    0.9528    0.9526    0.0002
```

Conclusiones

Mediante este algoritmo nos podemos dar cuenta que ya podemos solucionar problemas que nos son necesariamente problemas linealmente separables ya que el backpropagation nos da una mayor flexibilidad en cuanto a encontrar la correcta solución además de que el aprendizaje se hace de mejor manera debido a como estamos haciendo el entrenamiento en esta red así entonces una red de retro propagación en lo personal se hace una de las soluciones más efectivas a pesar de que implica usar una capa mas