**APROXIMACION DE UNA SUPERFICIE EN 3D MEDIANTE UNA RED NEURONAL BASADA EN LA REGLA INTAR VS BACKPROPAGATION**

Marco Teórico

Backpropagation

La propagación hacia atrás de errores o retropropagación (del inglés backpropagation) es un algoritmo deaprendizaje supervisado que se usa para entrenar redes neuronales artificiales. El algoritmo consiste en minimizarun error (comúnmente cuadrático) por medio de gradiente descendiente, por lo que la parte esencial del algoritmoes cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

Minimización del Error

Los algoritmos en Aprendizaje Automático pueden ser clasificados en dos categorías: supervisados y nosupervisados. Los algoritmos en aprendizaje supervisado son usados para construir "modelos" que generalmentepredicen un ciertos valores deseados. Para ello, los algoritmos supervisados requieren que se especifiquen losvalores de salida (output) u objetivo (target) que se asocian a ciertos valores de entrada (input). Ejemplos deobjetivos pueden ser valores que indican éxito/fallo, venta/no-venta, pérdida/ganancia, o bien ciertos atributosmulti-clase como cierta gama de colores o las letras del alfabeto. El conocer los valores de salida deseadospermite determinar la calidad de la aproximación del modelo obtenido por el algoritmo.

La especificación de los valores entrada/salida se realiza con un conjunto consistente en pares de vectores conentradas reales de la forma(\boldsymbol x,\boldsymbol t), conocido como conjunto de entrenamiento o conjunto de ejemplos. Losalgoritmos de aprendizaje generalmente calculan los parámetros \boldsymbol W de una función N(\boldsymbol x;\boldsymbol W) que permitenaproximar los valores de salida en el conjunto de entrenamiento.

Si (\boldsymbol x^{(q)},\boldsymbol t^{(q)}), q=1,\ldots,pson los elementos del conjunto de entrenamiento, la calidad de laaproximación en el ejemplo *q* se puede medir a través del error cuadrático:


E(\boldsymbol x^{(q)}; \boldsymbol W)=\frac{1}{2} \|N(\boldsymbol x^{(q)};\boldsymbol W) -\boldsymbol t^{(q)}\|^2
,

Donde \|\cdot\| es la norma euclidiana.

El error total es la suma de los errores de los ejemplos:

E(\boldsymbol W)=\sum_{q=1}^p E(\boldsymbol x^{(q)}; \boldsymbol W).

Un método general para minimizar el error es el actualizar los parámeros de manera iterativa. El valor nuevo de losparámetros se calcula al sumar un incremento \Delta\boldsymbol W al valor actual:


\boldsymbol W := \boldsymbol W + \Delta\boldsymbol W


El algoritmo se detiene cuando \boldsymbol W converge o bien cuando el error alcanza un mínimo valor deseado.

Si la función N(\boldsymbol x;\boldsymbol W) usada para aproximar los valores de salida es diferenciable respecto a los parámetros\boldsymbol W, podemos usar como algoritmo de aprendijaze el método de gradiende descendiente. En este caso, elincremento de los parámetros se expresa como


\Delta\boldsymbol W=-\gamma\frac{\partial E(\boldsymbol W)}{\partial \boldsymbol W},


Donde 0 < γ < 1 es un parámetro conocido como factor de aprendizaje.

Antes de continuar introduciremos un poco de notación. Definimos \bar{\boldsymbol v}=(v_1,\ldots,v_n,1)^T como el vector extendido del vector \boldsymbol v=(v_1,\ldots,v_n)^T. El par (\boldsymbol x, \boldsymbol t) representará a un elemento del conjunto deentrenamiento y una relación de entrada-salida, a menos que se indique otra cosa.

Desarrollo y solución propuesta

Se desea aproximar una función en tercera dimensión mediantes dos métodos distintos en el primero se usa la regla instar la cual se usa de modo ganador toma todo se mete como patrones las coordenadas en x y y transformando la superficie en vectores lineales para ello se le hace un reshape luego de eso se proponen unos puntos iniciales que abarquen la mayoría del espacio en donde se pretende mostrar la superficie luego se procede a ejecutar el algoritmo y se usa un total de 100 neuronas para el aprendizaje de esta función dando los resultados expuestos más abajo , el otro método utilizado fue la red backpropagation en donde igual que en la primer parte la superficie se descompone solo en 3 dimensiones y se meten los patrones que varían en x y y a la red esa se entrena de tal modo que la salida es la respuesta de la evaluación de la función dada por z de este modo después de un cierto número de épocas tenemos la aproximación de esta superficie junto con el error que describe cada época y como se entrenó la superficie

Algoritmo

Regla instar

clc,clear all,close all

figure

[X,Y]=meshgrid(-3\*pi:.5:3\*pi);

Z=sin(sqrt(X.^2+Y.^2));

surf(X,Y,Z)

axis([-3\*pi 3\*pi -3\*pi 3\*pi -2 2])

colormap cool

[fi col]=size(X);

N(:,1)=reshape(X,1,fi\*col);

N(:,2)=reshape(Y,1,fi\*col);

N(:,3)=reshape(Z,1,fi\*col);

Neu=100;

W(:,1:2)=-5 + (10)\*rand(100,2);

W(:,3)=rand(100,1);

figure(2)

plot3(W(:,1),W(:,2),W(:,3),'c\*'),grid on

hold on

alpha=0.7;

for i=1:100

rn=randi([1 length(N)],1,length(N));

j=1;

while j<=length(N);

for k=1:100

d(k)=sqrt((N(rn(j),:)'-W(k,:)')'\*(N(rn(j),:)'-W(k,:)'));

end

a = compet(-d');

for k=1:100

W(k,:)=W(k,:)+alpha\*a(k)\*(N(rn(j),:)-W(k,:));

end

j = j+1;

end

plot3(W(:,1),W(:,2),W(:,3),'b+'),grid on

pause(0.001)

alpha=alpha-0.005;

end

**Backpropagation**

clc,clear all,close all

% Comenzamos el foward propagation.

[X,Y]=meshgrid(-3\*pi:.2:3\*pi);

[fi co]=size(X);

ME(:,1)=reshape(X,1,fi\*co);

ME(:,2)=reshape(Y,1,fi\*co);

P=ME';

n1 = 20; %Numero de neuronas en la capa oculta

ep = 1; % Ventana de valores iniciales

Q = length(P);

% Valores iniciales

W1 = ep\*(2\*rand(n1,2)-1);

b1 = ep\*(2\*rand(n1,1)-1);

W2 = ep\*(2\*rand(1,n1)-1);

b2 = ep\*(2\*rand-1);

alfa = 0.04;

T=sin(sqrt(P(1,:).^2+P(2,:).^2));

figure

for Epocas = 1:500

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);

% Retropropagación de la sensibilidades

%T=sin(sqrt(P(1,q).^2+P(2,q).^2));

e = T(q)-a2(:,q);

s2 = -2\*(1-a2(q)^2)\*e;

s1 = diag(1-a1.^2)\*W2'\*s2;

% Actualización de pesos sinapticos y polarizaciones

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;

subplot(1,2,1),plot3(ME(:,1),ME(:,2),a2','.r'),grid on

axis([-3\*pi 3\*pi -3\*pi 3\*pi -2 2])

subplot(1,2,2),plot(emedio),grid on

pause(0.0001)

end

[X,Y]=meshgrid(-3\*pi:.5:3\*pi);

x=-3\*pi:.5:3\*pi;

Z=sin(sqrt(X.^2+Y.^2));

for i=1:length(x)

n1 = (W1\*[X(i);Y(i)])+b1;

a1 = tansig(n1);

a2(i) = (W2\*a1)+b2;

end

figure, subplot(1,3,1), plot(emedio),grid on,title('Error cuadratico Medio')

subplot(1,3,2),surf(X,Y,Z,'FaceColor','blue','FaceAlpha',0.3,'EdgeColor','none'),grid on,title('Señal patron')

axis([-3\*pi 3\*pi -3\*pi 3\*pi -2 2])

colormap spring

subplot(1,3,3),plot3(ME(:,1),ME(:,2),a2','.r'),grid on,title('Señal obtenida')

axis([-3\*pi 3\*pi -3\*pi 3\*pi -2 2])

figure,surf(X,Y,Z,'FaceColor','blue','FaceAlpha',0.9,'EdgeColor','none'),grid on,title('Señal patron'),hold on

plot3(ME(:,1),ME(:,2),a2','.r'),grid on,title('Señal obtenida')

Graficas

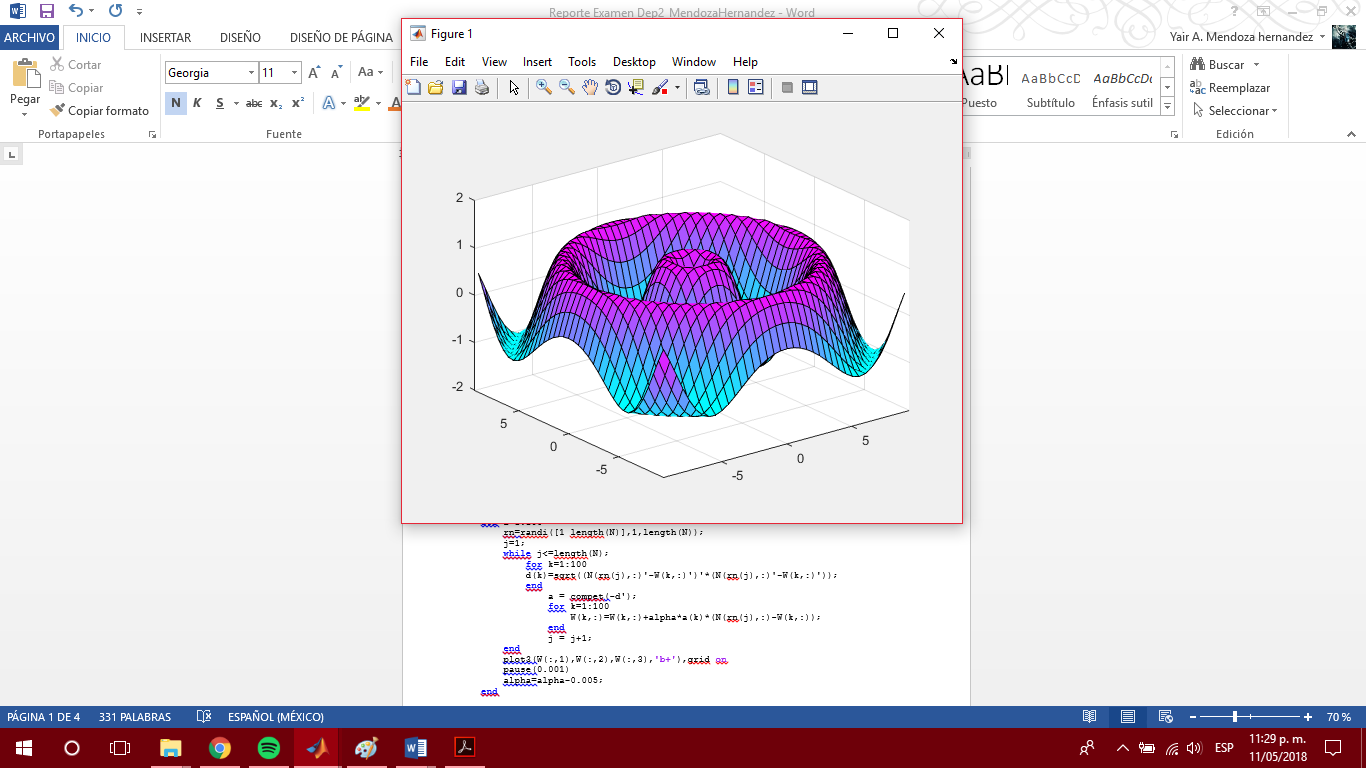


Fig. 1. Superficie a aproximar

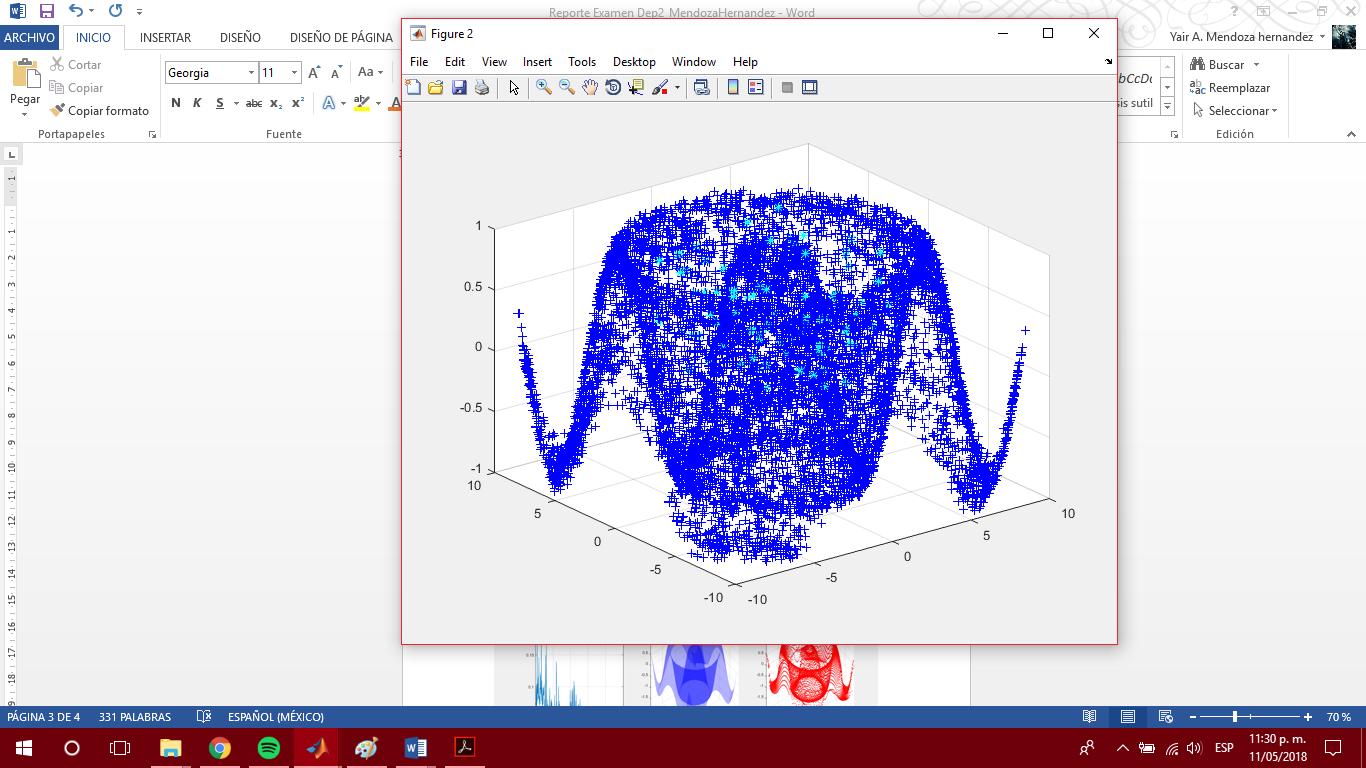


Fig. 2. Superficie generada en la regla instar

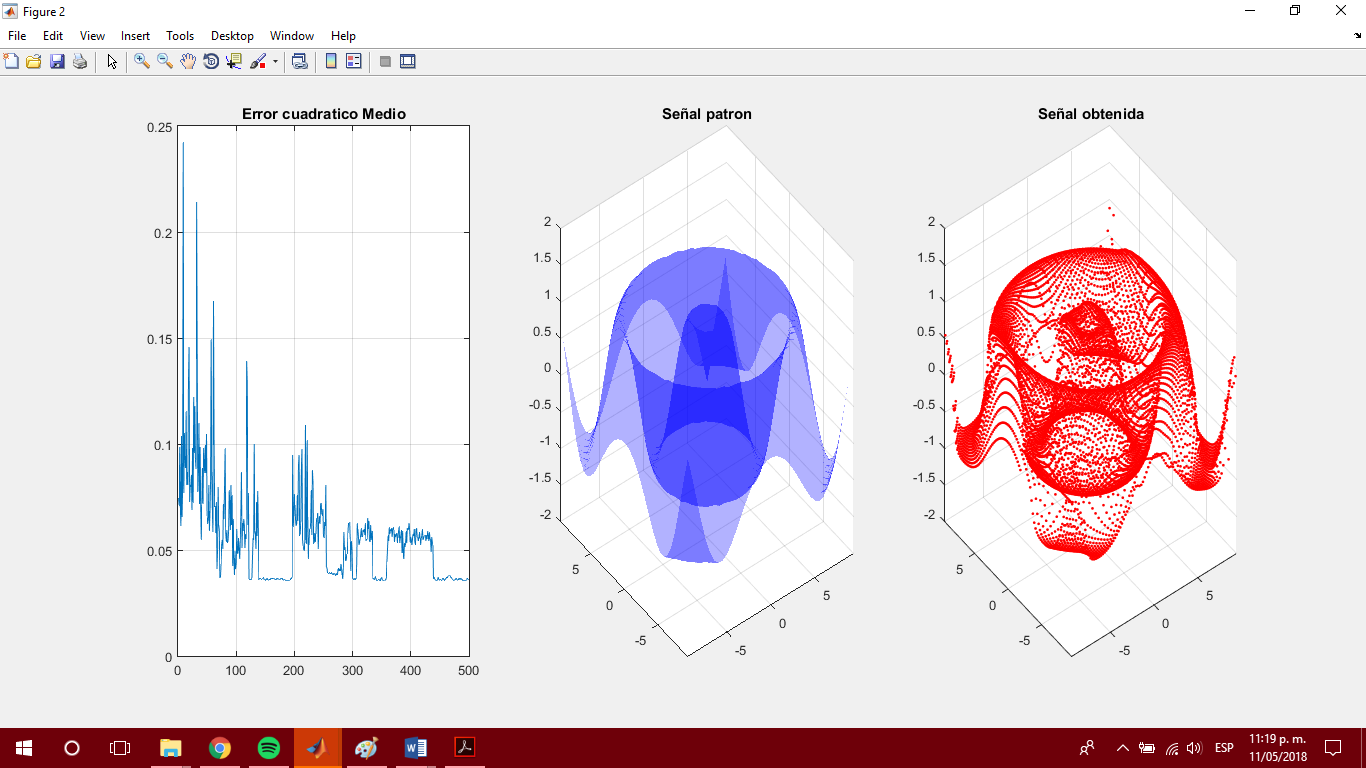


Fig 3. Superficie generada por la red backpropagation junto con el error cuadrático generado después de 600 épocas

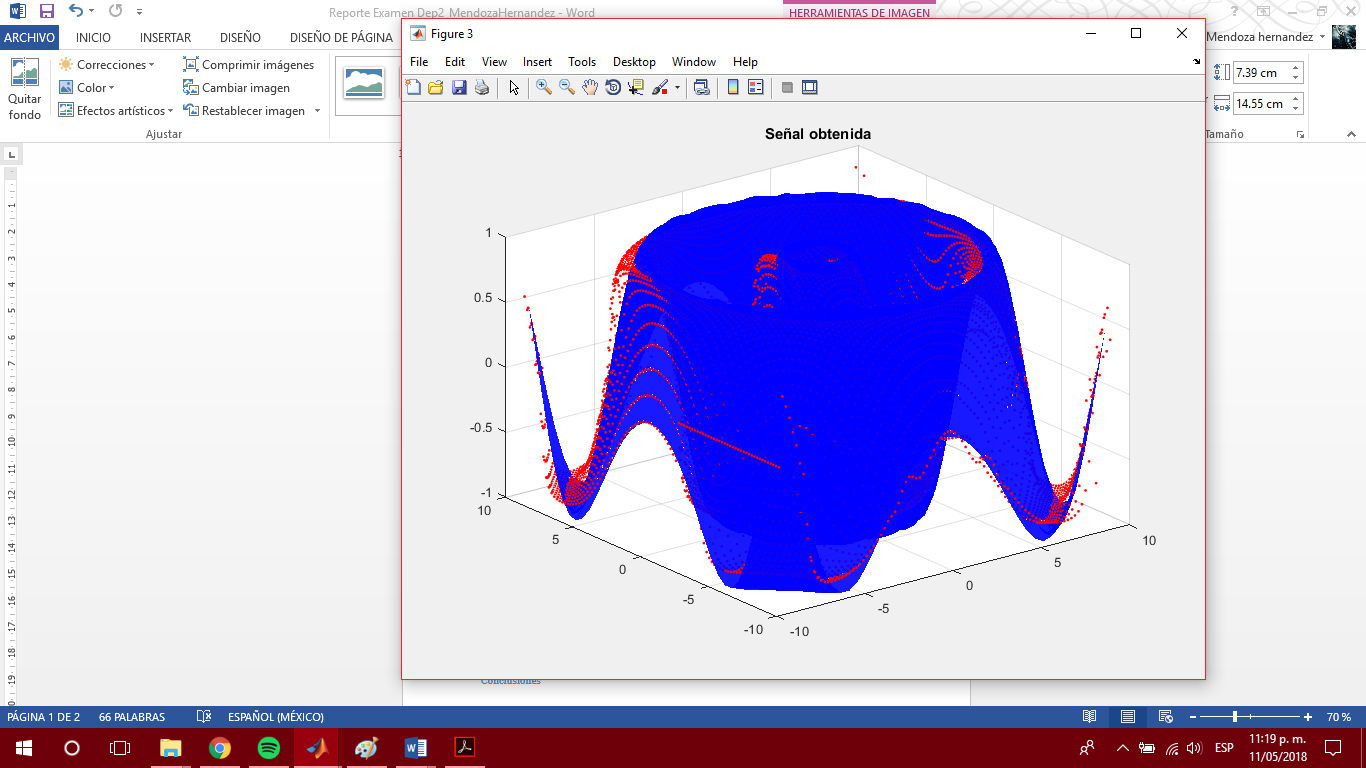


Fig 4. Comparación de la superficie generada por backpropagation con la superficie patrón

Conclusiones

Se puede concluir luego de realizar esta aproximación por dos métodos diferentes que sin lugar a dudas backpragation es una solución más efectiva y rápida que la regla instar esto debido a que con la regla instar se utiliza un mayor número de neuronas y por otro lado backproagation nos proporciona más flexibilidad debido a que los puntos de inflexión de la figura hacen que pueda funcionar de mejor manera que la regla instar además la regla instar depende totalmente de que tan bien pongamos los puntos iniciales

**REFERENCIAS**

* http://www.varpa.org/~mgpenedo/cursos/scx/archivospdf/Tema3-0.pdf
* Medina, E; Cubides, H; Salazar, J y Sigüencia, J. 2010. Redes Adaline – Filtros Adaptativos.
* Galván, I y Valls, J. 2010. Redes de Neuronas Artificiales.
* Valls, J. 2007. Redes de Neuronas Perceptrón y Adaline.
* Digital Signal Processing and Applications