Práctica 2: ADALINE
Modelos de la Computación

Emilio Gomez Esteban

1. Enunciado de la prática

En el campus virtual de la asignatura se encuentra el comprimido 'Adalina.zip', similar a la práctica anterior, descárgalo.

1. Crea las funciones 'CheckPattern.m', 'UpdateNet.m' y 'ValoresIOT.m' para que implementen el ADALINA.
2. ¿Qué ocurre si damos un valor negativo de LR? Da una explicación interpretando gráficamente lo que ocurre.
3. Modifica el fichero 'main.m' para calcular el ECM en cada época y almacénalo en un vector, pinta la gráfica con la evolución del ECM cuando intenta aprender la función XOR partiendo de los pesos W=[0 0 0]'.
4. Sube el script 'main.m' modificado que has usado para calcular el ECM.
5. Implementar el algoritmo del ADALINA analítico en un fichero MATLAB. Realizar el cálculo del ECM y del Acc (porcentaje de patrones correctamente clasificados).

En esta práctica hay que subir en total 5 script y un documento de texto con las respuestas, es decir:

- checkPattern.m
- UpdateNet.m
- ValoresIOT.m
- main.m (ECM de la Adalina)
- ADALINAAnalitico.m

2. Resolución de la práctica

Ejercicio 1

Implementación de la función CheckPattern(Data,W):

```
function correct = CheckPattern(Data,W)
    correct = true;

    for i=1:1:size(Data,1)
        [Input, Output, Target] = ValoresIOT(Data,W,i);
        if(Signo(Output)~=Target)   % ~= significa distinto
           que
            correct = false;
            break;
        end
    end
end

%{
    Se compara la salida del perceptron (Output) con el valor
     objetivo (Target). A diferencia del perceptron simple
       , el valor Output es la salida neta del Adaline (es
       decir, la suma ponderada de las entradas menos el
       sesgo), no la salida final despues de aplicar la
       funcion de activacion. Para ello, se aplica la funcion
        Signo antes de comparar con el Target, que convierte
       la salida neta en una decision binaria. Si no
       coinciden (Output es diferente de Target), significa
       que el adaline ha clasificado incorrectamente este
       patron.
    Si correct es true, significa que todos los patrones
       fueron clasificados correctamente por el adaline.
%}
```

Implementación de la función W = UpdateNet(W,LR,Output,Target,Input)

```
function W=UpdateNet(W,LR,Output,Target,Input)
    diffW = LR*(Target - Output)*[Input, -1];
    W = W + diffW';
end

%{
    El objetivo de esta funcion es actualizar los pesos W
       usando la regla de aprendizaje del perceptron simple.
    Se calcula la correccion de los pesos (diffW), usando la
       regla de aprendizaje del perceptron.
    Target - Output es el error o la diferencia entre la
```

```
        salida esperada (Target) y la salida actual (Output).
    [Input, -1] crea un nuevo vector de entrada que incluye
        la entrada original Input y annade un valor -1 al
        final. Esto es importante porque este -1 corresponde
        al peso asociado al sesgo (theta).
    W = W + diffW' actualiza los pesos sumando la correccion
        calculada en el paso anterior.
%}
```

Implementación de la función [Input,Output,Target]=ValoresIOT(Data,W,i):

```
function [Input, Output, Target] = ValoresIOT(Data,W,i)
    Input = Data(i, 1:end-1);
    Target = Data(i,end);
    Output = (Input*W(1:end-1) - W(end));    % W(end) es el
        theta
end

%{
    Esta funcion toma un conjunto de datos (Data), los pesos
        de un perceptron (W), y un indice i para un patron
        especifico del conjunto de datos. Devuelve la entrada
        (Input), la salida calculada por el perceptron(Output)
        , y el objetivo deseado (Target) para ese patron.
    Data(i, 1:end-1) toma todo el contenido de la fila
        excepto la ultima columna (la cual es el target).
    El valor objetivo se encuentra en la ultima columna de la
         fila i.
    Input*W(1:end-1): Esto es el producto escalar entre el
        vector de entradas Input y el vector de pesos
        correspondientes a las entradas W(1:end-1). Es decir,
        multiplica cada entrada por su respectivo peso y suma
        los resultados. En este caso no tenemos que usar la
        funcion de activacion en este momento de la iteracion.
%}
```

Ejercicio 2

Cuando le das un valor negativo a la tasa de aprendizaje (LR, Learning Rate) en el algoritmo Adaline, el modelo actualiza sus pesos en la dirección incorrecta. Esto provoca un comportamiento opuesto al esperado, lo que puede hacer que el modelo se aleje de la solución en lugar de acercarse a ella. Por la forma que tiene la actualización de pesos, en lugar de reducir el error, el modelo lo incrementa, porque los cambios en los pesos están alejando la predicción de la salida deseada.

Gráficamente, vemos como la recta que trata de separar linealmente el problema se ale-

ja de los puntos determinados por los datos, de manera que nunca llega a resolver el problema.

Ejercicio 3

Implementación del Main_ECM:

```
clear;
clc;
close all;

%load DatosAND
%load DatosLS5
%load DatosLS10
%load DatosLS50
%load DatosOR
load DatosXOR

LR=0.5;
Limites=[-1.5, 2.5, -1.5, 2.5];
MaxEpoc=30;

%W=PerceptronWeigthsGenerator(Data);
W = [0,0,0]';

Epoc=1;
ECM = zeros(MaxEpoc - 1,1);

while ~CheckPattern(Data,W) && Epoc<MaxEpoc
     ET = 0;
     for i=1:size(Data,1)
        [Input,Output,Target]=ValoresIOT(Data,W,i);
        ET = ET + (Output - Target)^2;
        GrapDatos(Data,Limites);
        GrapPatron(Input,Output,Limites);
        GrapNeuron(W,Limites);hold off;
        drawnow
%         pause;

        if Signo(Output) ~= Target
            W=UpdateNet(W,LR,Output,Target,Input);
        end


        GrapDatos(Data,Limites);
        GrapPatron(Input,Output,Limites)
        GrapNeuron(W,Limites);hold off;
```

```
        drawnow
%         pause;

    end
    ECM(Epoc,1) = ET/size(Data,1);   %size(Data,1) = 2
    Epoc=Epoc+1;
end

figure;
plot(1:Epoc-1, ECM(1:Epoc-1), '-o', 'LineWidth',2);
xlabel('Epocas');
ylabel('ECM');
title('Evolucion del Error Cuadrtico Medio (ECM)');
```

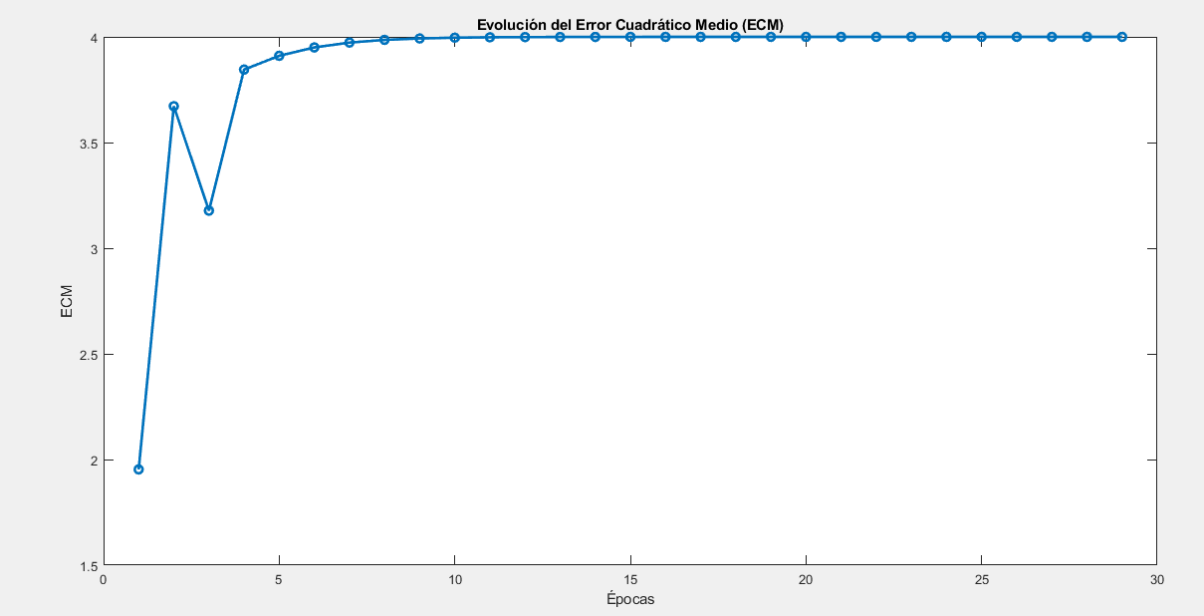

Figura 1: Evolución del ECM

Ejercicio 4

✓

Ejercicio 5

Implementación del ADALINAAnalitico.m

```
clear all;

load DatosAND
%load DatosLS5
```

```
%load DatosLS10
%load DatosLS50
%load DatosOR
%load DatosXOR

X = Data(:,1:end-1);
Y = Data(:,end);
k = size(Data,2) - 1;
N = size(Data,1);

Xext = [X -ones(N,1)];
%size(Xext);
%Xext;

w1 = inv(Xext' * Xext)*Xext' * Y;
w2 = pinv(Xext)*Y;

predicted = Xext * w1;   %predicted = y gorro
label = Signo(predicted);
ccr = sum(label == Y)/N;        %genera un vector (vector de
   booleanos) a partir de comparar ambos vectores componente
   a componente, colocando 1 si el elemento es igual y 0 si
   es distinto
%correct classification rate

ECM = norm(predicted - Y,2);
```