Práctica 1: El Perceptrón Simple
Modelos de la Computación

Emilio Gomez Esteban

1. Enunciado de la prática

En el campus virtual de la asignatura se encuentra el fichero comprimido 'Perceptron.zip', donde puedes encontrar diferentes ficheros para poder ejecutar el algoritmo del perceptrón. El fichero Main es el fichero principal de dicho algoritmo, el cual no se puede ejecutar correctamente debido a que faltan diferentes funciones que deben ser implementados.

1. Implementar las funciones 'CheckPattern.m', 'UpdateNet.m' y 'ValoresIOT.m' para que el algoritmo se ejecute correctamente. Una vez implementados las funciones para comprobar el correcto funcionamiento de su código, cambie los datos de entrada para comprobar que el algoritmo funciona con diferentes datos de entrada (DatosAND, DatosOR, DatosXOR, DatosLS10, DatosLS50).

   - La funcion CheckPattern(Data,W) tiene como parámetros de entrada el conjunto de datos 'Data' y los pesos sinápticos de la red 'W' y devuelve un booleano que es True cuando todos los datos han sido clasificados de forma correcta y False si algún dato del conjunto no está bien clasificado.
   - La función W=UpdateNet(W,LR,Output,Target,Input) tiene como entrada los pesos sinápticos de la red 'W', el valor de Learning rate 'LR' (η), la salida de mi red 'Output', para un cierto valor de entrada 'Input' y el valor 'Target' para ese valor de entrada. La función actualizará los pesos sinápticos y los devuelve actualizados.
   - La función [Input,Output,Target]=ValoresIOT(Data,W,i) tiene como entrada el conjunto de datos 'Data', los pesos sinápticos 'W' y un valor del índice 'i' que indica que dato del conjunto de datos se selecciona. La función selecciona un dato dentro del conjunto de datos y devuelve un único valor del conjunto 'Input', la salida objetivo de dicho valor 'Target' y la salida que se obtiene de la red 'Output'.

2. Sube la implementación de cada una de las funciones en tres ficheros independientes cuyo nombre sea el de la función que implementan y terminado en '.m'. Realiza comentarios en el código sobre los cálculos que realizas.

3. Comenta con tu palabras lo que ocurre cuando ponemos una tasa de aprendizaje de 0.005 para aprender la función OR en comparación con el caso en que utilizamos una tasa de 0.5.

4. Usa una tasa de 0.5 y comenta lo sucede cuando intenta ajustar la función XOR, ¿por qué ocurre?

5. Propón un valor a añadir al conjunto de entrenamiento DatosAND para que el perceptrón no encuentre una solución.

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(Output~=Target)   % (~= significa distinto que)
            correct = false;
            break;
        end
    end
end

%{
    Se compara la salida del perceptron (Output) con el valor
        objetivo (Target). Si no coinciden (Output es
       diferente de Target), significa que el perceptron ha
       clasificado incorrectamente este patron.
    Si correct es true, significa que todos los patrones
       fueron clasificados correctamente por el perceptron.
%}
```

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).
```

```
10      [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).
11      W = W + diffW' actualiza los pesos sumando la correccion
           calculada en el paso anterior.
12  %}
```

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

```
1   function [Input, Output, Target] = ValoresIOT(Data,W,i)
2       Input = Data(i, 1:end-1);
3       Target = Data(i,end);
4       Output = Signo(Input*W(1:end-1) - W(end));     % W(end) es
            el theta
5   end
6
7   %{
8       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.
9       Data(i, 1:end-1) toma todo el contenido de la fila
           excepto la ultima columna (la cual es el target).
10      El valor objetivo se encuentra en la ultima columna de la
            fila i.
11      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.
12      Signo() es la funcion de activacion que define si el
           perceptron se activa (da una salida de 1) o no (da una
            salida de -1).
13  %}
```

Ejercicio 2

✓

Ejercicio 3

Cuando se establece una tasa de aprendizaje muy baja, como 0.005, el ajuste de los pesos en cada iteración del entrenamiento será pequeño. En primer lugar, el entreamiento será lento y puede llevar más tiempo que los pesos converjan hacia valores que clasifiquen correctamente todos los patrones. Un aprendizaje lento reduce el riesgo de que los

pesos cambien drásticamente en cada actualización, lo que puede hacer que el proceso de aprendizaje sea más estable. Sin embargo, la estabilidad no es tan crítica para una función simple como OR, pero es importante en problemas más complejos.

En resumen, para un problema sencillo como aprender la función OR, una tasa de 0.5 es mucho más efectiva en términos de rapidez y eficiencia, ya que el perceptrón necesita muy pocos ajustes grandes para aprender correctamente la función. La tasa de 0.005 también funcionará, pero será mucho más lenta en converger, lo cual no es necesario dado lo simple del problema.

Ejercicio 4

Hemos tomado una tasa de aprendizaje de 0.5 para intentar ajustar la función XOR, sin embargo hemos visto que la ejecución del programa llega al número máximo de iteraciones sin encontrar una solución correcta. Esto es debido a que la función XOR no es linealmente separable, por lo que el modelo del perceptrón simple es incapaz de encontrar una solución, el cual está preparado para clasificar correctamente conjuntos de datos que sean linealmente separables. Por tanto, el perceptrón, sin importar cuántas iteraciones o cómo ajuste los pesos, no podrá encontrar una solución correcta.

Ejercicio 5

Para hacer que el perceptrón no pueda encontrar una solución, necesitamos añadir un patrón al conjunto de entrenamiento que no sea linealmente separable junto con los demás patrones. Si, por ejemplo, añadimos la fila (0.5, 0.5, -1) se rompe la simetría lineal de la función AND, y al asignarle un valor de salida -1, lo estamos colocando en la misma clase que los puntos (0,0), (0,1) y (1,0), lo cual hará que el perceptrón no pueda encontrar una frontera lineal que separe correctamente todos los puntos.