EXPLICACION DEL CODIGO IRIS

Red Neuronal para la clasificación de la flor IRIS

Universidad Autónoma del Carmen
Facultad de Ciencias de la Información

Autor: Carlos Alejandro Acosta Méndez
20 de Marzo de 2025

Docente:
Jesús Alejandro Flores Hernández

Descricion del Modelo

Objetivo del código: Entrenar una red neuronal para predecir la especie de la flor Iris basándose en 4 características.

- Se implemento una red neuronal con las siguientes caracteristicas:
- 4 neuronas de entrada (caracteristicas de IRIS)
- 10 neuronas en la capa oculta
- 3 neuronas de salida (clases de IRIS)
- **Funcion de activacion:** Sigmoide
- Entrenado con 30 observaciones

```
// Parámetros de la red
n_entradas = 4;  // Número de entradas
n_ocultas = 10;  // Neuronas en la capa oculta
n_salidas = 3;   // Neuronas en la capa de salida
n_num_dat_ent = 30 //numero de datos de entrenamiento
```

Función de activación sigmoide

- La función sigmoide es una función matemática que transforma cualquier valor real en un valor entre 0 y 1. Se utiliza en redes neuronales para procesar y categorizar datos de manera efectiva.

- Fórmula:
- σ(x)=11+e−xσ(x)=1+e−x1.
- Formula derivada:
- σ'(x)=σ(x)·(1−σ(x)).

```
//% Función de activación sigmoide
function y = sigmoid(x)
    y = 1 ./ (1 + exp(-x));
endfunction

//% Derivada de la sigmoide
function y = sigmoid_derivada(x)
    y = sigmoid(x) .* (1 - sigmoid(x));
endfunction
```

ESTRUCTURA DE LA RED NEURONAL

- **La arquitectura de la red:**
- Cuenta con 4 entradas: Corresponden a las 4 características de la flor.
- 10 neuronas en la capa oculta.
- 3 salidas: Corresponden a las 3 especies de Iris.

```
// Inicializar pesos y sesgos aleatoriamente
W1 = rand(n_entradas, n_ocultas);  // Pesos capa oculta
b1 = rand(1, n_ocultas);           // Sesgos capa oculta
W2 = rand(n_ocultas, n_salidas);   // Pesos capa salida
b2 = rand(1, n_salidas);           // Sesgos capa salida
```

CONJUNTO DE DATOS DE IRIS

30 Datos de las observaciones IRIS

4 características: Longitud y ancho del sépalo, longitud y ancho del pétalo.

3 especies: Setosa, Versicolor, Virginica.

```
//30 datos de las observaciones IRIS y sus etiquetas
IRIS_DATA=[
[5.1,3.5,1.4,0.2];[4.9,3,1.4,0.2];[4.7,3.2,1.3,0.2];[4.6,3.1,1.5,0.2];[5,3.6,1.4,0.2];
[5.4,3.9,1.7,0.4];[4.6,3.4,1.4,0.3];[5,3.4,1.5,0.2];[4.4,2.9,1.4,0.2];[4.9,3.1,1.5,0.1];
[6.4,3.2,4.5,1.5];[6.9,3.1,4.9,1.5];[5.5,2.3,4,1.3];[6.5,2.8,4.6,1.5];[5.7,2.8,4.5,1.3];
[6.3,3.3,4.7,1.6];[4.9,2.4,3.3,1];[6.6,2.9,4.6,1.3];[5.2,2.7,3.9,1.4];[5,2,3.5,1];
[6,3,4.8,1.8];[6.9,3.1,5.4,2.1];[6.7,3.1,5.6,2.4];[6.9,3.1,5.1,2.3];[5.8,2.7,5.1,1.9];
[6.8,3.2,5.9,2.3];[6.7,3.3,5.7,2.5];[6.7,3,5.2,2.3];[6.3,2.5,5,1.9];[6.5,3,5.2,2]
];
Y_DATA=[
[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];
[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];
[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1]
];
// Datos de entrenamiento (ejemplo)
//X = rand(n_num_dat_ent, n_entradas);  // 100 muestras, 4 características
//Y = rand(n_num_dat_ent, n_salidas);   //% 100 etiquetas, 3 salidas
```

PROPAGACION HACIA ADELANTE

Primero, expande el vector de sesgos b1 para que coincida con el número de datos de entrada y lo suma al producto de X (las entradas) con W1 (los pesos de la capa oculta), obteniendo Z1.

Luego, aplica la función sigmoide para calcular A1, que representa las activaciones de la capa oculta.

Después, repite el mismo proceso con A1, multiplicándolo por los pesos W2 y sumando el sesgo expandido b2, obteniendo Z2.

Finalmente, se aplica la función sigmoide a Z2 para calcular A2, que es la salida final de la red y representa las probabilidades de cada clase.

```
//% Entrenamiento
for iter = 1:max_iter
    //% Propagación hacia adelante
    b1_expanded = repmat(b1, n_num_dat_ent,1);
    //Expande b1[1,10] para que sea [30,10]
    //para sumar a esto:  X * W1 = [30,4]*[4,10]=[30,10]
    //suma de las entradas ponederadas + los sesgos
    Z1 = X * W1 + b1_expanded
    //A1 salidas de la capa oculta
    A1 = sigmoid(Z1);
    //A2 salidas de la capa de salida
    b2_expanded = repmat(b2, n_num_dat_ent,1);
    Z2 = A1 * W2 + b2_expanded;
    A2 = sigmoid(Z2);
```

CALCULO DEL ERROR

```
//Cálculo del error
error = Y - A2;
```

- Diferencia entre las salidas predichas y las reales.

- Línea: error = **Y – A2**

Esta línea calcula la diferencia entre las salidas esperadas **“Y”** y las salidas predichas **A2** de la red neuronal. Este error indica cuánto se aleja la red de la clasificación correcta y será utilizado en la fase de retropropagación que será explicado en la siguiente diapositiva.

RETROPROPAGACION

Este bloque de código implementa el algoritmo de retropropagación, que ajusta los pesos de la red para minimizar el error. Al final se ajustan los pesos y sesgos de ambas capas utilizando la tasa de aprendizaje (tasa_aprendizaje), lo que permite que la red aprenda gradualmente a clasificar mejor los datos.

```
//Retropropagación
dZ2 = error .* sigmoid_derivada(Z2);
dW2 = A1' * dZ2;
db2 = sum(dZ2, 1);

dZ1 = (dZ2 * W2') .* sigmoid_derivada(Z1);
dW1 = X' * dZ1;
db1 = sum(dZ1, 1);

//% Actualizar pesos y sesgos
W2 = W2 + tasa_aprendizaje * dW2;
b2 = b2 + tasa_aprendizaje * db2;
W1 = W1 + tasa_aprendizaje * dW1;
b1 = b1 + tasa_aprendizaje * db1;
```

PREDICCIONES

Aquí prueba la red neuronal después del entrenamiento para hacer predicciones sobre los datos de entrada.

Primero, expande los sesgos para que coincidan con el número de muestras.

Luego, realiza la propagación hacia adelante usando los pesos ajustados **W1** y **W2**, que representa las probabilidades de pertenecer a cada una de las tres clases del conjunto de datos Iris.

```
//Probar la red
b1_exp=repmat(b1,n_num_dat_ent,1)
b2_exp=repmat(b2,n_num_dat_ent,1)
Y_pred = sigmoid(sigmoid(X * W1 + b1_exp) * W2 + b2_exp);
disp("Predicciones:");
disp(cat(2,X,fix(Y_pred+0.5)));
//disp(Y_pred);
```