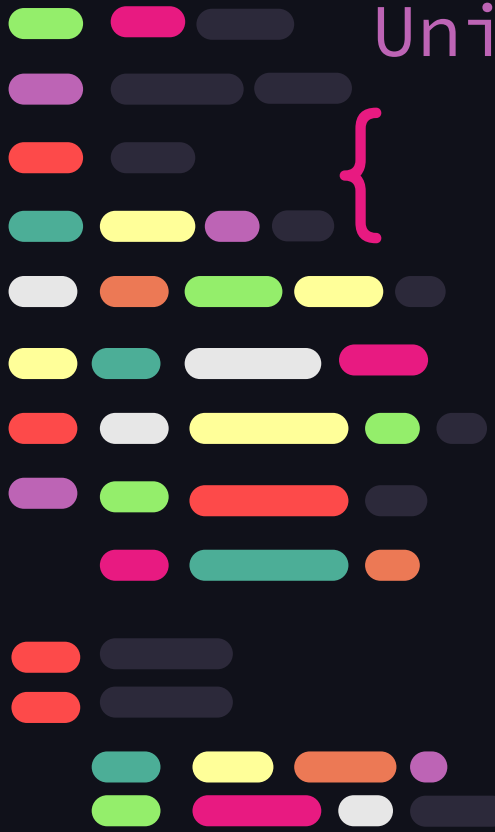


Universidad Autónoma del Carmen

Facultad de Ciencias de la información

Materia:
Reconocimiento de patrones

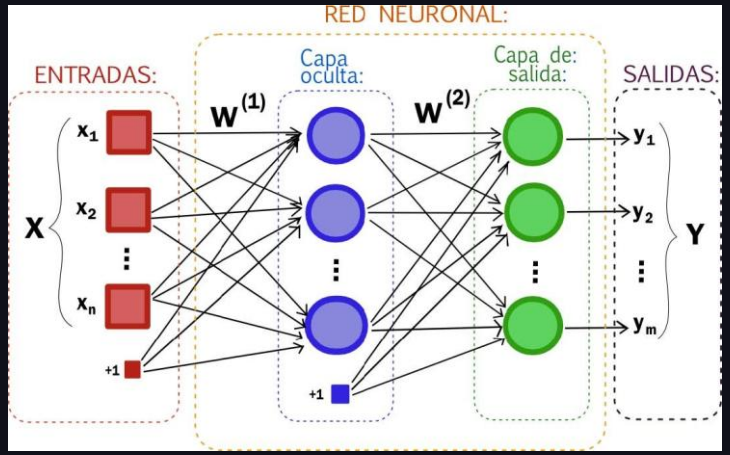
Docente:
Jesús Alejandro Flores Hernandez

Alumna:
Anali del Carmen Perez Martinez

CONJUNTO DE DATOS DE LA FLOR IRIS

Conjuto de datos iris

Diseñar y entrenar una red neuronal para clasificar flores del conjunto de datos Iris.La red neuronal recibe 4 características de la flor y predice a qué especie pertenece (Setosa, Versicolor o Virginica).



Parametros de la red

Definición de la red neuronal

```
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
```

La red tiene:

- 4 entradas (características de la flor).
- 1 capa oculta con 10 neuronas.
- 3 neuronas de salida, una para cada especie de flor.
- Datos de Entrenamiento: 30 observaciones del conjunto Iris con etiquetas correspondientes

Función de activación sigmoide

Función de activación

```
function y = sigmoid(x)

y = 1 ./ (1 + exp(-x));

endfunction
```

Se utiliza la función sigmoide para transformar las entradas ponderadas en valores dentro del rango (0,1).

Derivada de la sigmoide

```
function y = sigmoid_derivada(x)

    y = sigmoid(x) .* (1 - sigmoid(x));

endfunction
```

La derivada se utiliza durante el proceso de retropropagación para ajustar los pesos de la red.

Inicializacion de la red

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
```

Los pesos y los sesgos se inicializan aleatoriamente, lo que permite a la red aprender desde cero.

Carga de datos (Flores iris)

DATOS DE ENTRADA

```
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]
];
```

Cada fila representa una flor con sus 4 características.

```
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]
];
```

Las etiquetas están codificadas en formato one-hot, donde:

- [1, 0, 0] representa Iris setosa.
- [0, 1, 0] representa Iris versicolor.
- [0, 0, 1] representa Iris virginica.

Datos del entrenamiento

```
//Datos del entrenamiento
X=IRIS_DATA;
Y=Y_DATA;
//mostrar entradas
disp("Datos de entrada")
disp(cat(2,X,Y))
```

✔ X contiene las características de cada flor (longitud y ancho de sépalos y pétalos).
✔ Y contiene Etiquetas de salida en formato One-Hot
✔ cat(2, X, Y) concatena las matrices X y Y, mostrando las características y sus etiquetas juntas.

```
// Entrenamiento
disp("Entrenamiento:")
// Hiperparámetros
tasa_aprendizaje = 0.1;
max_iter = 1000;
```

✔ Tasa de aprendizaje (0.1): Controla cuánto se ajustan los pesos en cada iteración.
Un valor muy alto hace que el modelo aprenda rápido, pero puede ser inestable.
Un valor muy bajo hace que el modelo aprenda lento.
✔ Máximo de iteraciones (1000):Define cuántas veces se repite el proceso de entrenamiento.
Un número alto permite más aprendizaje, pero puede hacer que tarde más en converger.

Entrenamiento

```
//Datos del entrenamiento

//% 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)
```

El entrenamiento se realiza mediante el algoritmo de propagación hacia adelante (forward propagation) y retropropagación (backpropagation).

Se calculan las salidas de cada capa aplicando la función sigmoide.

Entrenamiento

Calculo del error y la propagación

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

//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);
```

Se calcula la diferencia entre la salida real y la salida predicha. Luego, se ajustan los pesos y sesgos usando descenso del gradiente

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

Se actualizan los pesos para mejorar la precisión de la red en cada iteración

Prueba de Red

```
//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);
```

Se usa la red neuronal entrenada para predecir la especie de cada flor.Se redondean las predicciones (fix(Y_pred+0.5)) para que sean 0 o 1.

"Datos de entrada"

5.1	3.5	1.4	0.2	1.	0.	0.
4.9	3.	1.4	0.2	1.	0.	0.
4.7	3.2	1.3	0.2	1.	0.	0.
4.6	3.1	1.5	0.2	1.	0.	0.
5.	3.6	1.4	0.2	1.	0.	0.
5.4	3.9	1.7	0.4	1.	0.	0.
4.6	3.4	1.4	0.3	1.	0.	0.
5.	3.4	1.5	0.2	1.	0.	0.
4.4	2.9	1.4	0.2	1.	0.	0.
4.9	3.1	1.5	0.1	1.	0.	0.
6.4	3.2	4.5	1.5	0.	1.	0.
6.9	3.1	4.9	1.5	0.	1.	0.
5.5	2.3	4.	1.3	0.	1.	0.
6.5	2.8	4.6	1.5	0.	1.	0.
5.7	2.8	4.5	1.3	0.	1.	0.
6.3	3.3	4.7	1.6	0.	1.	0.
4.9	2.4	3.3	1.	0.	1.	0.
6.6	2.9	4.6	1.3	0.	1.	0.
5.2	2.7	3.9	1.4	0.	1.	0.
5.	2.	3.5	1.	0.	1.	0.
6.	3.	4.8	1.8	0.	0.	1.
6.9	3.1	5.4	2.1	0.	0.	1.
6.7	3.1	5.6	2.4	0.	0.	1.
6.9	3.1	5.1	2.3	0.	0.	1.
5.8	2.7	5.1	1.9	0.	0.	1.
6.8	3.2	5.9	2.3	0.	0.	1.
6.7	3.3	5.7	2.5	0.	0.	1.
6.7	3.	5.2	2.3	0.	0.	1.
6.3	2.5	5.	1.9	0.	0.	1.
6.5	3.	5.2	2.	0.	0.	1.

"Entrenamiento:"
"Predicciones:"

5.1	3.5	1.4	0.2	1.	0.	0.
4.9	3.	1.4	0.2	1.	0.	0.
4.7	3.2	1.3	0.2	1.	0.	0.
4.6	3.1	1.5	0.2	1.	0.	0.
5.	3.6	1.4	0.2	1.	0.	0.
5.4	3.9	1.7	0.4	1.	0.	0.
4.6	3.4	1.4	0.3	1.	0.	0.
5.	3.4	1.5	0.2	1.	0.	0.
4.4	2.9	1.4	0.2	1.	0.	0.
4.9	3.1	1.5	0.1	1.	0.	0.
6.4	3.2	4.5	1.5	0.	1.	0.
6.9	3.1	4.9	1.5	0.	1.	0.
5.5	2.3	4.	1.3	0.	1.	0.
6.5	2.8	4.6	1.5	0.	1.	0.
5.7	2.8	4.5	1.3	0.	1.	0.
6.3	3.3	4.7	1.6	0.	1.	0.
4.9	2.4	3.3	1.	0.	1.	0.
6.6	2.9	4.6	1.3	0.	1.	0.
5.2	2.7	3.9	1.4	0.	1.	0.
5.	2.	3.5	1.	0.	1.	0.
6.	3.	4.8	1.8	0.	0.	1.
6.9	3.1	5.4	2.1	0.	0.	1.
6.7	3.1	5.6	2.4	0.	0.	1.
6.9	3.1	5.1	2.3	0.	0.	1.
5.8	2.7	5.1	1.9	0.	0.	1.
6.8	3.2	5.9	2.3	0.	0.	1.
6.7	3.3	5.7	2.5	0.	0.	1.
6.7	3.	5.2	2.3	0.	0.	1.
6.3	2.5	5.	1.9	0.	0.	1.
6.5	3.	5.2	2.	0.	0.	1.