```
sigmoid = @(x) 1./(1 + exp(-x));
sigmoid_derivative = @(x) sigmoid(x) .* (1 - sigmoid(x));

softmax = @(x) exp(x) ./ sum(exp(x))
```

```
softmax = function_handle with value:
    @(x)exp(x)./sum(exp(x))
```

Definimos los datos de entrada

```
iris_table = readtable('./iris_dataset.csv');
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.
```

```
iris_data = table2array(iris_table);
X = iris_data(:, 1:4)';
X = (X - mean(X, 2))./std(X, 0, 2);
```

Dividir los datos en conjuntos de entrenamiento y prueba

```
rng(42); % establecer la semilla para la reproducibilidad
data_size = size(X, 2);
train_size = round(0.8 * data_size);

% Conjunto de entrenamiento
train_indices = randperm(data_size, train_size);
X_train = X(:, train_indices);

% Asegurémonos de tener y definido (puedes ajustarlo según tus necesidades)
y_train = zeros(train_size, 3); % Cambia 3 a tu número real de clases

% Conjunto de prueba
test_indices = setdiff(1:data_size, train_indices);
X_test = X(:, test_indices);
y_test = zeros(data_size - train_size, 3); % Cambia num_classes a tu número real de clases

% Definimos el número de clases (ajústalo según tus necesidades)
num_classes = 3;
```

A continuación, definimos la red neuronal

```
% Crear una instancia de la red neuronal
addpath('neuralnet/matlab_net');
```

```
Warning: Name is nonexistent or not a directory: C:\Users\fenix\OneDrive\Escritorio\GCID\asignaturas
tercero\OH\Backpropagation\MatlabNeuralNetworkImpl-main\neuralnet\matlab_net
```

```
net = NeuralNetwork();

% Definir capa 1 con 2 entradas, 3 salidas, función de activación sigmoidal y su derivada
layer1 = SequentialLayer(4, 5, sigmoid, sigmoid_derivative);
net = net.set(layer1);

% Definir capa 2 con 3 entradas (salidas de la capa anterior), 1 salida, función de activación lineal y su derivada
layer2 = SequentialLayer(5, num_classes, softmax, false);
net = net.set(layer2);
```

Entrenamos la red neuronal

```
% Entrenamos la red neuronal
num_samples_train = size(X_train, 2);
learning_rate = 0.01;

for iter = 1:10000
    permuted_indices = randperm(num_samples_train);
    for sample = 1:num_samples_train
        random_index = permuted_indices(sample);
        X_sample = X_train(:, random_index);
        y_sample = y_train(random_index, :);

        % Realizar retropropagación con la muestra seleccionada
        net = net.backpropagation(X_sample, y_sample', learning_rate);
```

```
    end
end
```

Calculamos y mostramos la precisión en el conjunto entrenamiento

```
accuracy_train = net.calculateAccuracy(X_train, y_train);
fprintf('Accuracy en el conjunto de datos de entrenamiento: %.2f%%\n', accuracy_train * 100);
```

```
Accuracy en el conjunto de datos de entrenamiento: 100.00%
```

Calculamos y mostramos la precisión total

```
predictions_test = net.feedforward(X_test);
accuracy_test = net.calculateAccuracy(X_test, y_test);
fprintf('Accuracy en el conjunto de datos de prueba: %.2f%%\n', accuracy_test * 100);
```

```
Accuracy en el conjunto de datos de prueba: 100.00%
```