Laboratory 6

Implementation of Multilayer NNs

Due Date: Beginning of Week 8 Lab

Concepts:

- Multilayer Neural Network Implementation
- Matlab Neural Network Toolbox/App

Total Points: 100 points

Objectives:

Multilayer NNs are useful to learn complex non-linear hypothesis. By connecting multiple neurons in multiple layers, NNs can compute the **complicated non-linear functionalities**. In this lab, students will implement the backpropagation neural network to classify the object type in the given images by utilizing the Matlab neural network pattern recognition toolbox.

Files Needed:

- Lab6.mlx
- CIFAR2_data.mat
- displayData.m

Assignment:

Backpropagation NNs implementation using the Neural Network Pattern Recognition App

In Lab6, you will implement backpropagation neural networks to identity the object types in the given images. The provided data set, "CIFAR2_data.mat" is a subset of the CIFAR-10 dataset and consists of 10,000 32x32 color images in 2 classes {"vehicle, "horse"}. In the target data, the class "vehicle" is labeled as "1" and "horse" is labeled as "2".

- The file "CIFAR2_data.mat" contains 10,000 32x32 color images. The 32 by 32 color image is flattened into a 1 by 3071 row vector. Each of these color images becomes a single row in the given data.

- **Task1**: split the data into training and testing sets with an 80-20 split. You can use the function ""mySplitData.m" developed in Lab3.

- **Task2**: **Data normalization**

- In machine learning, data normalization is a procedure that **makes input features are in the similar ranges.** Another reason why the data normalization is applied is that gradient descent converges much faster with feature scaling than without feature scaling.
- Do the data normalization in the range of [0, 1] using the Min-Max method. The equation of the Min_Max method is given below:

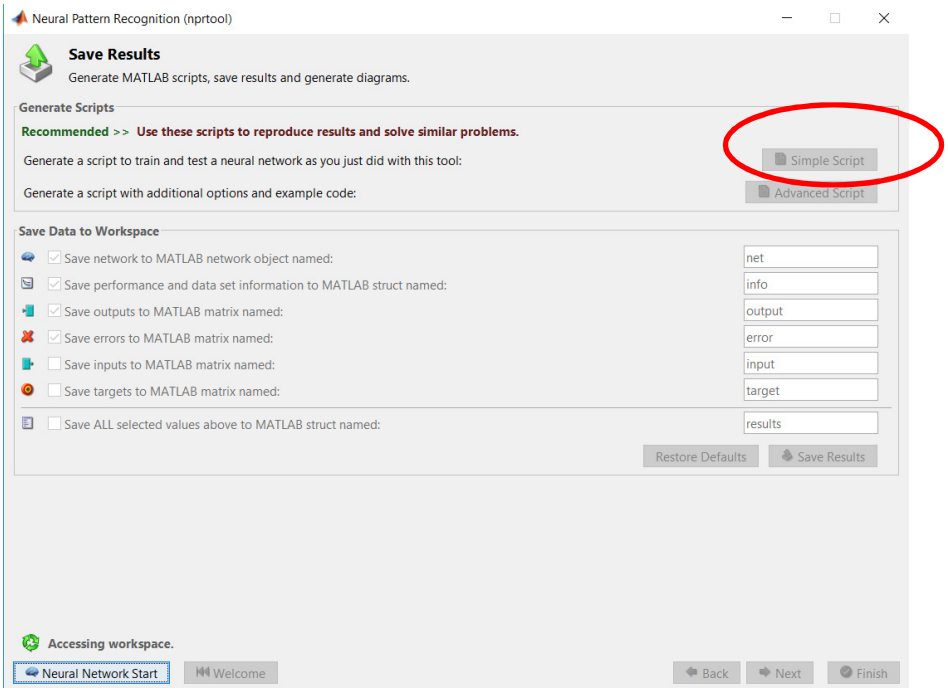
$$norm_x = \frac{x - \min(x)}{\max(x) - min(x)}$$

, where x is a column vector, min(x) is the minimum of the column vector x , and max(x) is the maximum of the column vector x.

Task3: Generate the script from NPRTOOL and edit the script to customize the NN training

Rather than writing code from scratch, you will utilize the **Neural Network Pattern Recognition tool** (NPRTOOL) to generate scripts from NPRTOOL GUIs, and then modify them to customize the network training.

Start NPRTOOL GUI with the command, " **nnstart**". Select Pattern recognition app and train the neural network as you learned in the Lab5-part2. The purpose of using NPRTOOL is to generate the script to customize the neural network training. At the end of the training, you can create MATLAB code by clicking **Simple Script.**



For example, look at the simple script that was created by NPRTOOL.

```
% Solve a Pattern Recognition Problem with a Neural Network
% Script generated by NPRTOOL
%
% This script assumes these variables are defined:
%
%   norm_X - input data.
%   new_Y - target data.

x = norm_X';                  % X is the normalized data
t = new_Y';                   % Y label data is in the converted form [0 1] or [1 0]


% 2. Create a Pattern Recognition Network
hiddenLayerSize = 1000;
net = patternnet(hiddenLayerSize);


% 3. Set up Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 90/100;
net.divideParam.valRatio =  5/100;
net.divideParam.testRatio = 5/100;


% 4. Train the Network
[net,tr] = train(net,inputs,targets);

% 5. Test the Network

y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);


% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
% figure, plotperform(tr)
% figure, plottrainstate(tr)
% figure, plotconfusion(targets,outputs)
% figure, ploterrhist(errors)
```

You can save the script, and then run it from the command line to reproduce the results of the NPRTOOL GUI session. You can also edit the script to customize the training process. In this case, follow each step in the script.

1. The script assumes that the input vectors and target vectors are already loaded into the workspace.

2. Create the network.
   - The network has two output neurons, because there are two target values (categories) associated with each input vector.
   - Each output neuron represents a category: "vehicle" = [1 0] , "horse"=[0 1]
   - When an input vector of the appropriate category is applied to the network, the corresponding neuron should produce a 1, and the other neurons should output a 0.

   To create the network, you need to specify the number of nodes in each hidden layer. For example, the following command will create the NN with 5 hidden layers. Each hidden layer contains the 1000, 500, 100, 50, and 20 nodes, respectively. The function " **patternnet**()" will create the NN architecture based on the parameter values you passed.

```
% Create a Pattern Recognition Network
h1_Size = 1000;
h2_Size = 500;
h3_Size = 100;
h4_Size = 50;
h5_Size = 20;
hidden_layers = [h1_Size h2_Size  h3_Size  h4_Size  h5_Size ];

net = patternnet (hidden_layers)
```

3. Set up the division of data.
   - `net.divideParam.trainRatio = 90/100;`
   - `net.divideParam.valRatio   = 10/100;`
   - `net.divideParam.testRatio  = 0 /100;`

   With these settings, the input vectors and target vectors will be randomly divided, with 90% used for training, 10% for validation and 0% for testing because you have already the separate test data set.

4. Train the network. The pattern recognition network uses the default Scaled Conjugate Gradient (`trainscg`) algorithm for training. To train the network by passing network architecture, input and target as below, enter this command:

```
[net,tr] = train(net, inputs, targets);
```

   During training, as in function fitting, the training window opens. This window displays training progress. To interrupt training at any point, click **Stop Training**.

   This training stopped when the validation error increased for six iterations.

5. Calculate the training performance of the network. After the network has been trained, you can use it to compute the network outputs. The following code calculates the network outputs, errors and overall performance.

```
y = net(x);

tind = vec2ind(t);
yind = vec2ind(y);
Training_percentErrors = sum(tind ~= yind)/numel(tind)*100

Tip: To get the help of the function, move the cursor over the function name,
highlight the name and click the right mouse button.
```

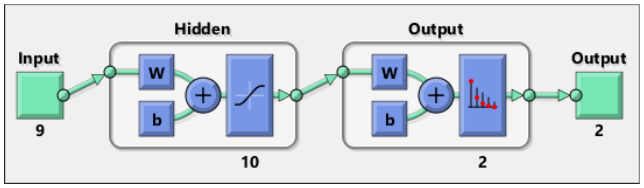
6. It is also possible to calculate the network performance only on the test set, by using the test data,

```
x = norm_xTest';        % normalized data
t = new_yTest';         % new converted label format: label 1= [1 0],2=[0 1 ]

y = net(x);
tind = vec2ind(t);
yind = vec2ind(y);
Test_percentErrors = sum(tind ~= yind)/numel(tind)*100
```

7. View the network diagram.

```
view(net)
```



8. Plot the training, validation, and test performance.

```
figure, plotperform(tr)
```

9. Use the `plotconfusion` function to plot the confusion matrix. It shows the various types of errors that occurred for the final trained network.

```
figure, plotconfusion(targets,outputs)
```

Each time a neural network is trained, can result in a different solution due to different initial weight and bias values and different divisions of data into training, validation, and test sets. As a result, different neural networks trained on the same problem can give different outputs for the same input. To ensure that a neural network of good accuracy has been found, retrain several times.

Task4: Build a neural network with **5 hidden layers by modifying code generated by NPRTOOL**. You need to design the number of hidden nodes in each hidden layer.
Question1: What are the best performances of the trained network on the training data and the test data?

Task5: Build a neural network with 3 hidden layers by modifying code generated by NPRTOOL. You need to design the number of hidden nodes in each hidden layer.
Question2: What are the best performances of the trained neural network on the training data and the test data?

You can work in groups of up to two people

What to Submit to Blackboard:

- **one** ZIP file that includes :
  1) **A report (10pts): contains answers for Question1 –Question2.**
  2) (90 pts): The live script file, "Lab6.mlx", in that all outputs are generated either inline or on the right side.

- **Pay attention to the zip file name convention:**
  Lab6_Student1 Lastname_Student2 Lastname.zip
  Ex) Lab6_Green_Smith.zip