MATLAB Companion Script for *Machine Learning* ex4 (Optional)

Introduction

Coursera's *Machine Learning* was designed to provide you with a greater understanding of machine learning algorithms- what they are, how they work, and where to apply them. You are also shown techniques to improve their performance and to address common issues. As is mentioned in the course, there are many tools available that allow you to use machine learning algorithms *without* having to implement them yourself. This Live Script was created by MathWorks to help *Machine Learning* students explore the data analysis and machine learning tools available in MATLAB.

FAQ

Who is this intended for?

- This script is intended for students using MATLAB Online who have completed ex4 and want to learn more about the corresponding machine learning tools in MATLAB.

How do I use this script?

- In the sections that follow, read the information provided about the data analysis and machine learning tools in MATLAB, then run the code in each section and examine the results. You may also be presented with instructions for using a MATLAB machine learning app. This script should be located in the ex4 folder which should be set as your Current Folder in MATLAB Online.

Can I use the tools in this companion script to complete the programming exercises?

- No. Most algorithm steps implemented in the programming exercises are handled automatically by MATLAB machine learning functions. Additionally, the results will be similar, but not identical, to those in the programming exercises due to differences in implementation, parameter settings, and randomization.

Where can I obtain help with this script or report issues?

- As this script is not part of the original course materials, please direct any questions, comments, or issues to the *MATLAB Help* discussion forum.

Create and Train a Neural Network in MATLAB

In this Live Script, we will use the *Neural Net Pattern Recognition App* from the Neural Network Toolbox to build and train a neural network for hand-written digit recognition.

Files needed for this script

- `ex4data1.mat` - Training set of hand-written digits

Table of Contents

Create a Pattern Recognition Network using `nprtool`

Load the data

Run the code in this section to load the image data matrix `X` and the digit label vector `y`. Note that before creating and training a multi-class neural network classifier in MATLAB, the label vector `y` should be transformed into a *binary class matrix* `Y` such that `Y(i,j)` equals `1` if y(`i)==j`, and 0 otherwise.

```
clear;
load('ex4data1.mat');
Y = (1:10) == y;
```

Choose an appropriate neural network app

As mentioned in the companion script for ex3, MATLAB neural network model variables are complicated structures that contain all of the information about the network, including the structure, weights, and other details. Instead of writing code to build and train a neural network and create a model variable from scratch, it is much easier to use one of the available apps from the Neural Network toolbox. The trained network can then be exported as a network variable to the workspace, or a script can be automatically generated which contains the code necessary for recreating and retraining a similar network at a later date. If further customization or duplication of the network is desired, one can then adapt the code in this script.

In the Machine Learning section of the MATLAB Apps tab, you will find several interactive apps for quickly generating and training a neural network for the following applications:

- Clustering (`nctool`)
- Regression (`nftool`)
- Time Series (`ntstool`)
- Pattern Recognition (`nprtool`)

The apps above are launched from the 'Apps' menu or using the (`*tool`) commands listed above. Alternatively, you can launch the general Neural Network Start App using the command `nnstart` if you are not yet sure which app to choose. Run the code below to open the Neural Network Start app.

```
nnstart
```

Create and train a neural network for pattern recognition

Once the app opens, you should see a button to launch each of the above apps listed above. Note that there is additional information and helpful examples to you get started in the 'More Information' tab. The problem of classifying the digit images is a pattern recognition and classification problem. The Pattern Recognition App is designed to help create and train a two layer neural network suitable for this type of problem. Follow the steps in the next few sections to train a digit classifier using the Pattern Recognition App.

Note: If you have difficulty reading the instructions below while the app is open in MATLAB Online, export this script to a pdf file which you can then use to display the instructions in a separate browser tab or window. To export this script, click on the 'Save' button in the 'Live Editor' tab above, then select 'Export to PDF'.

Open the Pattern Recognition App and select the data

1. In the **Neural Network Start** app opened previously, select the **Pattern Recognition App**.
2. In the **Welcome** page, read the information about the pattern recognition network uses and structure, then click **Next**.
3. In the **Select Data** page, select X from the **Inputs** list, select Y from the **Targets** list, and select '**Matrix rows**' for the sample orientation, then click '**Next**'.
4. In the **Validation and Test Data** page, select 5% for '**Validation**' and '**Testing**' and click '**Next**'.

Create and train the network

The network input and output layer sizes are determined from the data selected. The number of layers is automatically determined from the network type (pattern recognition in this case). All that is left is to select the number of neurons in the hidden layer and train the network.

1. In the **Network Architecture** page, enter '25' for the '**Number of Hidden Neurons**' and click '**Next**'.
2. In the **Train Network** page, click the '**Train**' button. The **Neural Network Training Tool** app will automatically open and train network. When the training is finished, several plots are available for inspection which provide information on the training and performance of the network model in the '**Plots**' section of the **Neural Network Training Tool**.
3. Close the **Neural Network Training Tool**, return to the **Pattern Recognition App**, and click '**Next**'.

Evaluate and export the network

After creating and training the network, there are available options for visualizing the network's performance and making adjustments if needed. Once the network is performing well, it can be exported as a network variable to the workspace, or a script can be generated for building and training similar networks in the future.

1. Inspect the options for improving the neural network performance or testing the network on additional data in the **Evaluate Network** page.
2. Since the performance is satisfactory (and we have no additional data) click '**Next**'.
3. In the **Deploy Solution** page, available options are shown for deploying the network. Click '**Next'** to skip deployment.
4. In the **Save Results** page, click the '**Simple Script**' button in the '**Generate Scripts**' section to generate a script for recreating and training the network in the future. Inspect the script and discard it when you are done (this script has already been adapted to train a custom network in the next part of this script).
5. Select the '**Save network to MATLAB Object named:**' checkbox in the '**Save Data to Workspace**' section and de-select the other boxes.
6. Enter the network variable name, '**net**', in the corresponding box, then click the '**Save Results**' button to export the neural network variable to the workspace.
7. Click the '**Finish**' button to exit the **Pattern Recognition App** and close the **Neural Network Start App**.

Predict digits using the trained network variable

Run the code below to estimate the class of a random example using the network variable you trained above, compare with the true class, and display the corresponding image using the trained network:

```
i = randi(length(y));
ysim = net(X(i,:)');
```

```
Unrecognized function or variable 'net'.
```

```
[~,class] = max(ysim);
imshow(reshape(X(i,:),20,20))
fprintf('True class: %d  |  Predicted class: %d | Probability of match: %.1f%%',y(i),cl
```

Build and Train a Customized Neural Network Programatically

In the first part of this live script you used the Pattern Recognition App to build and train a neural network. If you wish to utilize additional options when building or training a network, for example using regularization or a custom activation function, the neural network apps can generate a script which can be adapted with the desired customizations. This is often more efficient and robust method than writing your own code to build and train a network.

Train a custom network with regularization

A local function for building and training a pattern recognition network, `imageNet`, has been provided at the end of this script. When called, it will build, train, and return neural network model variable which can be used for classifying digit images. The code in `imageNet` was generated by the Neural Net Pattern Recognition App using the steps in the previous section and then adapted with custom options and parameters, including regularization.

Run the code below to call `imageNet`, which will create and train a two-layer, feed-forward neural network using the settings for `lambda` and `MaxIter` you provide below. Note that `imageNet` is a local function - it can only be called from inside this Live Script. The training accuracy is then displayed for comparison with your previous

results. Note that without regularization, it is possible to achieve a training accuracy 100% if training is allowed to continue for enough epochs (iterations). However, the model will likely not generalize well to new data as discussed in the lectures. Use the controls below to explore the effect of regularization and training epochs on the final model.

```
% Change lambda and MaxIter to see how it affects the result
lambda = 0.1;
MaxIter = 40;
net = imageNet(X,Y,lambda,MaxIter);
[~,ysim] = max(net(X'));
fprintf('Training accuracy: %g%%',100*sum(y == ysim')/length(y));
```

Local Functions

imageNet

`imageNet` trains a neural network using custom parameter settings and returns the network variable. This function was adapted from a script generated by the Pattern Recognition App.

```
function net = imageNet(X,Y,lambda,MaxIter)
% Build, train, and return a neural network model to classify the 1x400 images
% of digits in X with labels in Y

% MATLAB neural networks prefer examples in columns and labels in rows
X = X';
Y = Y';

% Set network options:
hiddenLayerSize = 25; % Create a neural net with a single hidden layer with 25 output r
trainFcn = 'trainscg';  % Use scaled conjugate gradient backpropagation
net = patternnet(hiddenLayerSize, trainFcn); % Create the net variable
net.layers{1}.transferFcn = 'logsig'; % Change default transfer function to match ex3/4

% Add regularization:
% Note that the MATLAB regularization parameter must lie between 0 and 1,
% where 0 correspnds to lambda = 0 (no regularization) and 1 corresponds to lambda = in
% We map the input lambda value to [0,1] using tanh function
net.performParam.regularization = tanh(lambda);
net.trainParam.epochs = MaxIter;
net.trainParam.showWindow = false; % Don't show training GUI

% Setup Division of Data for Training, Validation, Testing
% This example uses all of the data for training (to match ex3/ex4 network example)
% Validation and test data is discussed later in the course
net.divideParam.trainRatio = 100/100;
net.divideParam.valRatio = 0/100;
net.divideParam.testRatio = 0/100;

% Train the Network
net = train(net,X,Y);
end
```