Programming Exercise 5: Multi-class Classification Using Neural Networks (Forward Propagation)

Introduction

In this exercise, you will implement forward propagation in neural networks to recognize hand-written digits. To get started with the exercise, you will need to download the starter code and unzip its contents to the directory where you wish to complete the exercise. If needed, use the `cd` command in Octave/MATLAB to change to this directory before starting this exercise.

Files included in this exercise

`ex5.m` – Octave/MATLAB script that steps you through the exercise
`ex5data1.mat` – Dataset of hand-written digits
`ex5weights.mat` – Weights for the neural network (already trained)
`displayData.m` – Function to help visualize the dataset
`sigmoid.m` – Sigmoid function
[*] `predict.m` – Predict using a neural network multi-class classifier

For this exercise, you will use neural networks to recognize handwritten digits (from 0 to 9). Automated handwritten digit recognition is widely used today – from recognizing zip codes (postal codes) on mail envelopes to recognizing amounts written on bank checks. This exercise will show you how the methods you've learned can be used for this classification task.
Throughout the exercise, you will be using the script `ex5.m`. This script sets up the dataset for the problem and makes call to the function that you will write. You do not need to modify this script. You are only required to modify a function in some other file, by following the instructions in this assignment.

---

[*] indicates files you will need to complete

1 Dataset

You are given a data set in `ex5data1.mat` that contains 5000 training examples of handwritten digits[1]. The `.mat` format means that that the data has been saved in a native Octave/MATLAB matrix format, instead of a text (ASCII) format like a csv-file. These matrices can be read directly into your program by using the `load` command. After loading, matrices of the correct dimensions and values will appear in your program's memory. The matrices will already be named, so you do not need to assign names to them.

```
% Load saved matrices from file
load('ex5data1.mat');
% The matrices X and y will now be in your Octave/MATLAB environment
```

There are 5000 examples in `ex5data1.mat`, where each example is a 20 pixel by 20 pixel grayscale image of a digit. Each pixel is represented by a floating point number indicating the grayscale intensity at that location. The 20 by 20 grid of pixels is “unrolled” into a 400-dimensional vector. Each of these examples becomes a single row in our data matrix X. This gives us a 5000 by 400 matrix X where every row is an example for a handwritten digit image.

$$X = \begin{bmatrix} - - x^{(1)} - - \\ - - x^{(2)} - - \\ \vdots \\ - - x^{(m)} - - \end{bmatrix}$$

The second part of the training set is a 5000-dimensional vector y that contains labels for the dataset. To make things more compatible with Octave/MATLAB indexing, where there is no zero index, the digit zero is mapped to the value ten. Therefore, a “0” digit is labeled as “10”, while the digits “1” to “9” are labeled as “1” to “9” in their natural order.

[1] This is a subset of the MNIST handwritten digit dataset (http://yann.lecun.com/exdb/mnist/)

2 Visualizing the data

You will begin by visualizing a subset of the dataset. In Part 1 of `ex5.m`, the code randomly selects 100 rows from X, calles them `X_test`, and passes those rows to the `displayData` function. This function maps each row to a 20 pixel by 20 pixel grayscale image and displays the images together. The `displayData` function is provided, and you are encouraged to examine the code to see how it works. After you run this step, you should see an image like Figure 1.

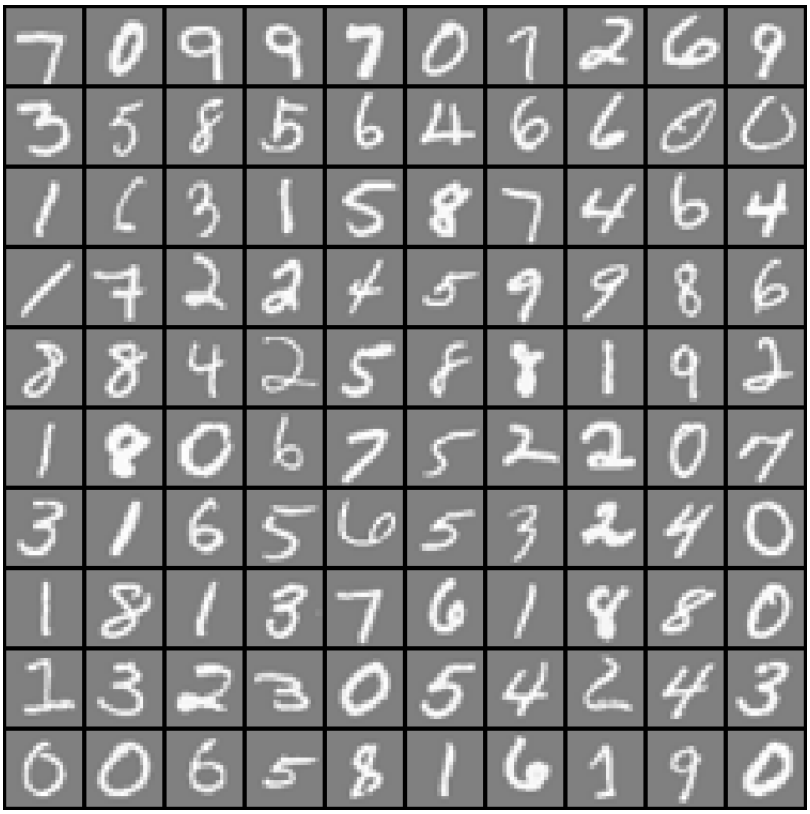

Figure 1: Examples from the dataset

3 Neural Networks

In this part of the exercise, you will implement a neural network to recognize handwritten digits provided in the `X_test`. The neural network will be able to represent complex models that form non-linear hypotheses. For this programming exercise, you will be using parameters from a neural network that have already been trained and provided. Your goal is to implement the forward propagation algorithm to use the given weights for prediction. In the next programming exercise, you will write the backpropagation algorithm for your own to make a neural network learn its parameters.

3.1 Model representation

Our neural network is shown in Figure 2. It has 3 layers – an input layer, a hidden layer and an output layer. Recall that our inputs are pixel values of digit images. Since the images are of size 20×20, this gives us 400 input layer units (excluding the extra bias unit which always outputs +1).
You have been provided with a set of (already trained) network parameters ($\Theta^{(1)}, \Theta^{(2)}$). These are stored in `ex5weights.mat` and will be loaded by `ex5.m` into `Theta1` and `Theta2`. The parameters have dimensions that are sized for a neural network with 25 units in the second layer and 10 output units (corresponding to the 10 digit classes).

```
% Load the weights into variables Theta1 and Theta2
load('ex5weights.mat');
% The matrices Theta1 and Theta2 will now be in your Octave/MATLAB environment
% Theta1 has size 25 x 401
% Theta2 has size 10 x 26
```

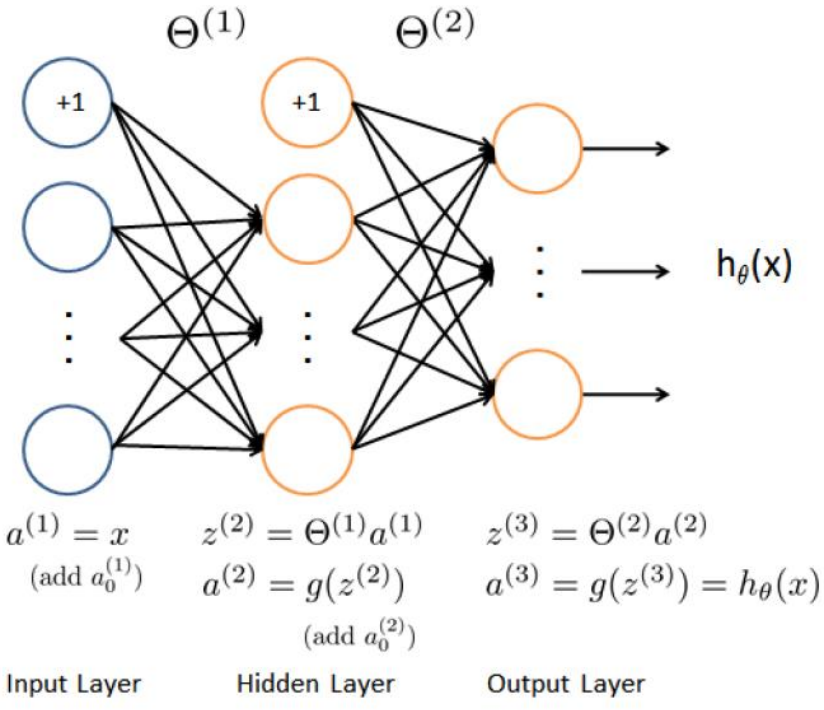

Figure 2: Neural network model

3.2 Forward propagation and prediction

Now you will implement forward propagation for the neural network. You will need to complete the code in `predict.m` to return the neural network's prediction.
You should implement the forward propagation computation that computes $h(x^{(i)})$ for every example i and returns the associated predictions. The prediction from the neural network will be the label that has the largest output $(h(x))_k$.

Implementation Note: The matrix X in `predict.m` contains the examples in rows. When you complete the code in `predict.m`, you will need to add the column of 1's to the matrix. The matrices `Theta1` and `Theta2` contain the parameters for each unit in rows. Specifically, the first row of `Theta1` corresponds to the first hidden unit in the second layer. In Octave/MATLAB, when you compute $z^{(2)} = \Theta^{(1)} * a^{(1)}$, be sure that you index (and if necessary, transpose) X correctly so that you get $a^{(l)}$ as a column vector.

Once you are done, `ex5.m` will call your predict function using `X_test` and the loaded set of parameters from `Theta1` and `Theta2`. You should see that the accuracy is about 99% on the testing dataset. After that, an interactive sequence will launch displaying images from the testing dataset one at a time, while the console prints out the predicted label for the displayed image. To stop the image sequence, press `Ctrl-C`.