CS 341: Machine Learning
Homework 2

Dan Sheldon

September 21, 2012

Instructions

Complete all problems and submit by the beginning of class on Wed. Sep 26. You may work together with other students, but *I highly encourage you to attempt the problems first on your own, especially the simpler ones.* Please make sure to:

- Write your name on your submission
- Write the name of all students with whom you collaborated
- Cite any sources you used other than the textbook, course notes, or Coursera

Files

The zip archive `hw2-files.zip` contains the files needed for this assignments. When unzipped, it will expand into a directory `hw2-files` with the following files:

- `learn_matlab.m`
- `problem_1.m`
- `problem_2.m`
- `cost_function.m`
- `normal_equations.m`
- `problem_3.m`
- `gradient_descent.m`
- `problem_4.m`
- `book-data.csv`

You will need to add code to some of these files as instructed below.

What to Submit

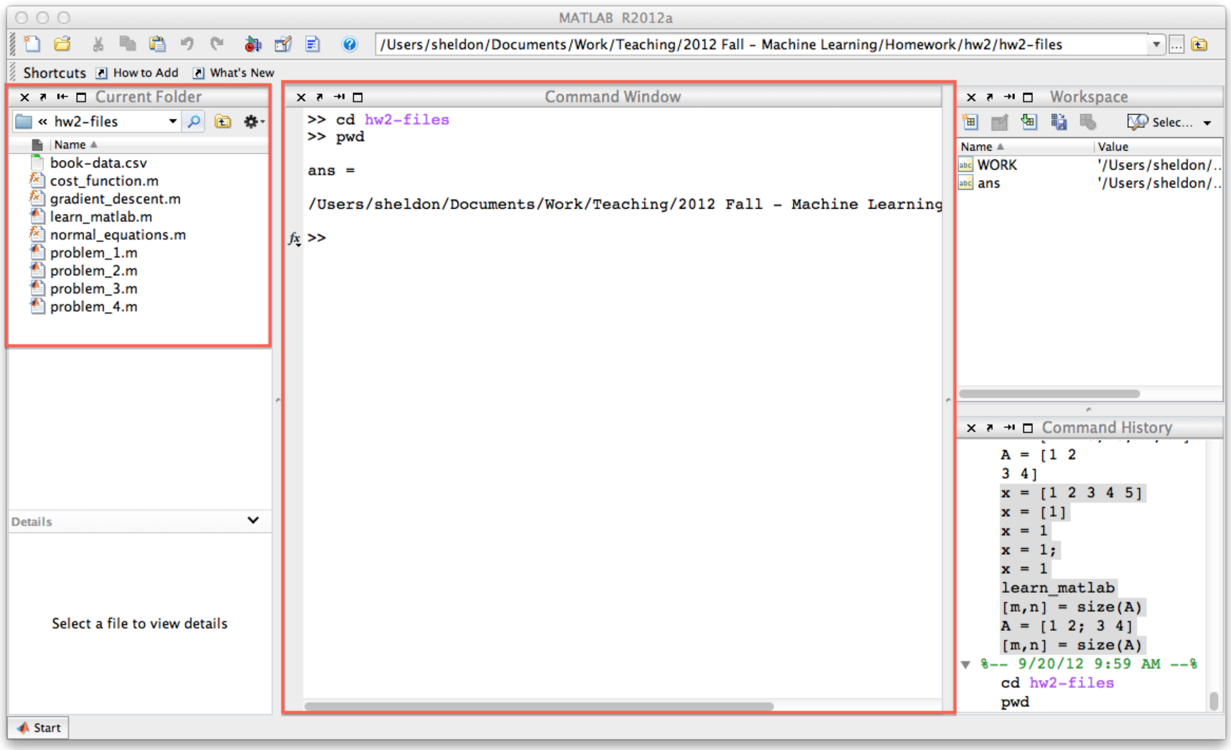
When you are done editing the code, rename the directory `hw2-files` to `hw2-files-<yourlastname>` and then zip it into a file `hw2-files-<yourlastname>.zip`. E.g., you could do this in the terminal via the two commands:

```
mv hw2-files hw2-files-sheldon
zip -r hw2-files-sheldon.zip hw2-files-sheldon
```

Submit the zip archive via ella (by the beginning of class) and hand in written work in class.

MATLAB Instructions

You will need to know a few basic things about the MATLAB environment to get started on this homework. The two parts of the workspace you will interact with most are the 'Command Window' and the 'Current Folder' browser, outlined in red in the figure below.


MATLAB R2012a
/Users/sheldon/Documents/Work/Teaching/2012 Fall – Machine Learning/Homework/hw2/hw2-files
Shortcuts
How to Add
What's New
Current Folder
Command Window
Workspace
Command History
Details
Select a file to view details
Start

Current Folder: hw2-files

```
book-data.csv
cost_function.m
gradient_descent.m
learn_matlab.m
normal_equations.m
problem_1.m
problem_2.m
problem_3.m
problem_4.m
```

Command Window:

```
>> cd hw2-files
>> pwd

ans =

/Users/sheldon/Documents/Work/Teaching/2012 Fall - Machine Learning

>>
```

Workspace:

Name	Value
WORK	'/Users/sheldon/..
ans	'/Users/sheldon/..

Command History:

```
A = [1 2
3 4]
x = [1 2 3 4 5]
x = [1]
x = 1
x = 1;
x = 1
learn_matlab
[m,n] = size(A)
A = [1 2; 3 4]
[m,n] = size(A)
%-- 9/20/12 9:59 AM --%
cd hw2-files
pwd
```

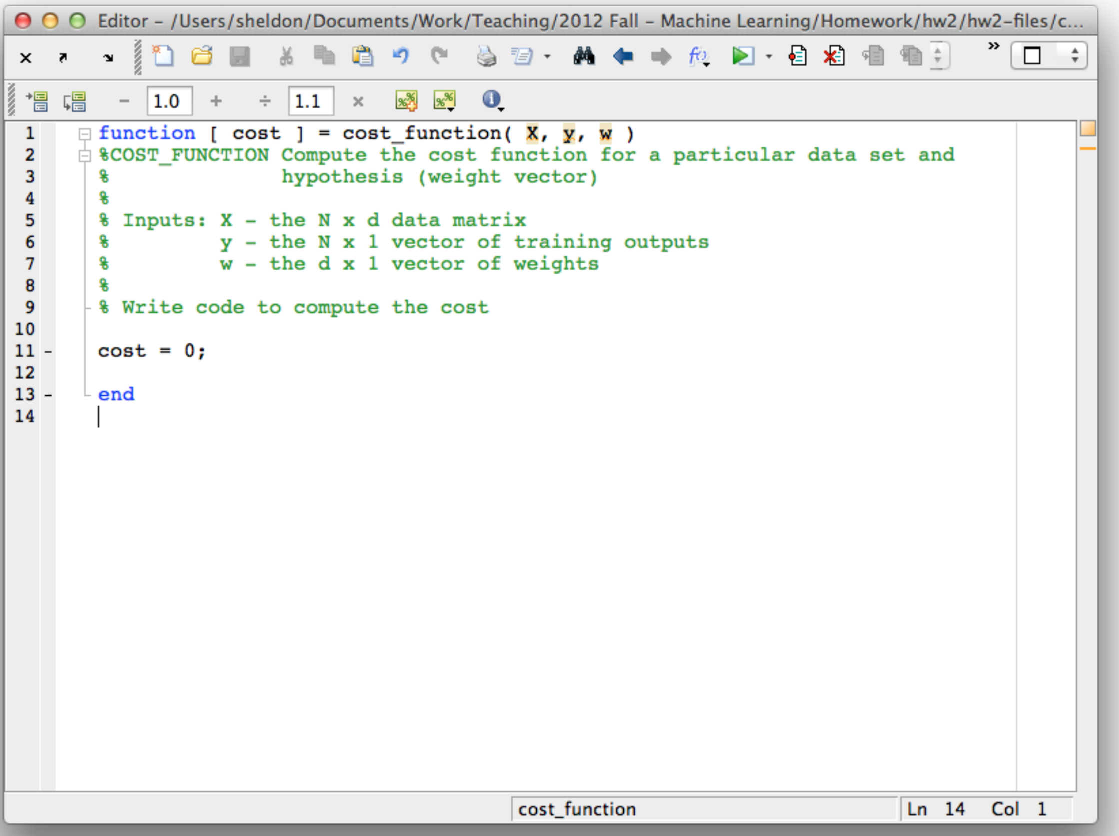
Navigate to the `hw2-files` directory. After you unzip the files for this assignment, you will need to navigate to the `hw2-files` directory from within MATLAB so you can edit the files and execute the scripts. The easiest way to do this is using the Current Folder browser. This is similar to a file browser in a window-based operating system. Click around and figure out how to navigate to the `hw2-files` directory. When you are done, you should see the list of homework files, as in the screenshot above.

Scripts. A MATLAB script is stored in a file with the extension `.m`, such as `learn_matlab.m`. The name of the script is the part of the file name preceding `.m`. There are several ways to execute a script. One way is to type the name of the script in the Command Window:

```
>> learn_matlab
```

Another way is to right click on the `.m` file (e.g., `learn_matlab.m`) in the Current Folder browser, and then select ‘Run’ from the context menu.

Editing Files. The other MATLAB feature you will be using extensively is the editor, so that you can modify and save `.m` files containing scripts and functions. The editor is depicted below.


Editor – /Users/sheldon/Documents/Work/Teaching/2012 Fall – Machine Learning/Homework/hw2/hw2-files/c...
1.0
1.1
cost_function
Ln 14 Col 1

```
function [ cost ] = cost_function( X, y, w )
%COST_FUNCTION Compute the cost function for a particular data set and
%               hypothesis (weight vector)
%
% Inputs: X - the N x d data matrix
%         y - the N x 1 vector of training outputs
%         w - the d x 1 vector of weights
%
% Write code to compute the cost

cost = 0;

end

```

An easy way to open a file in the editor is to use the `open` command in the Command Window.

```
>> open learn_matlab.m
```

You can also:

- Right click on the file in the Current Folder browser, and then select ‘Open’ from the context menu.
- Click on the ‘File’ → ‘Open...’ from the main MATLAB menu bar.

After you have edited a file in the editor, make sure to save it before running the code.

Problems

Note: Problems 1–4 should be completed in order.

1. In this problem you will exercise some basic MATLAB and linear algebra concepts. Review the file `learn_matlab.m` to learn about basic MATLAB operations. Then, open the file `problem_1.m`, and write code to do the following:

   (a) Enter the following matrices and vectors

   $$A = \begin{bmatrix} -2 & -3 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

   (b) Compute $C = A^{-1}$

   (c) Check that $AC = I$ and $CA = I$

   (d) Compute $A\mathbf{x}$

   (e) Compute $A^T A$

   (f) Compute $A\mathbf{x} - B\mathbf{x}$

   (g) Compute $\|\mathbf{x}\|$ (use the dot product)

   (h) Compute $\|A\mathbf{x} - B\mathbf{x}\|$. (Errata: was $\|A\mathbf{x} - \mathbf{y}\|$, but $\mathbf{y}$ was not defined.)

   (i) Print the first column of B (do not use a loop)

   (j) Compute the element-wise product between the first column of A and the second column of A (use the `.*` operator)

   (k) Write a double for loop to print all the entries of A

2. In this problem, you will implement multivariate linear regression using the normal equations. Open the file `problem_2.m` and look at the code. It generates some training data from a fourth degree polynomial and constructs a multivariate linear regression problem to learn the coefficients of the polynomial. It solves the problem by calling the function `normal_equations()` (in the file `normal_equations.m`), and then it computes the cost by calling the function `cost_function()`. It then prints the cost and plots the original polynomial, the data, and the fitted polynomial.

   Run the script `problem_2.m`. You will see that the fitted polynomial and the computed cost are incorrect, because the functions `normal_equations()` and `cost_function()` are currently only stubs. Complete the two functions so the code works correctly. You should see that the fitted polynomial closely matches the original one.

3. Now you will solve the same polynomial regression problem by gradient descent. Open the file `problem_3.m`. It is very similar to problem `problem_2.m`, except it calls the function `gradient_descent()` to solve the problem. This time, in addition to plotting the fit, it also plots the cost function at each iteration of gradient descent. In a correct implementation, the cost should decrease with each iteration. Run the script. You will see that the cost does not decrease and the learned polynomial is incorrect, because the function `gradient_descent()` is currently only a stub.

   (a) Complete the function `gradient_descent()` so the code works correctly.

   (b) Experiment with different values of the step size α and the number of iterations by changing the parameters in the file `problem_3.m`. Report a value of α for which gradient descent does *not* converge.

   (c) How many iterations do you need to run so that the model is as accurate as the one learned by the normal equations? Report values of α and `num_iters` so that the cost of the solution found by gradient descent matches the cost of the solution found by the normal equations to four decimal places.

4. In this problem you will experiment with feature engineering for the book data to improve the fit of the model to the data. Look at and run the script `problem_4.m`. It loads the book data and fits a linear regression model (using your implementation of `normal_equations()`). It then creates one new feature by adding two existing features, and fits the model again. Notice that this new feature does not help (the cost does not decrease), so you will have to be more creative about designing features.

   (a) Write code to add one or more new features by transforming or combining existing features. Show that you can improve the model fit. That is, after adding your feature(s) and solving the linear regression problem to find $\mathbf{w}$, the overall cost $J(\mathbf{w})$ should be lower than it was before adding the feature. (If not, keep trying).

   (b) See how low you can drive the cost by adding more features. Be creative.

   (c) Suppose you add 20 new features and observe that the cost goes down close to zero on your training data. Does this mean you have fit a better model? Briefly explain.

5. Prove that designing a new feature by adding two existing features cannot lead to a better fit. (Hint: look at the simplest case: $\mathbf{x} \in \mathbb{R}^2$, and add a new feature $x_3 = x_1 + x_2$. Can the minimum of $J(w_0, w_1, w_2, w_3)$ be smaller than the minimum of $J(w_0, w_1, w_2)$?)