Thank you to Machine Learning Mentor, Tom Mosher, for compiling this list

Subject: Confused about "h(x) = theta' * x" vs. "h(x) = X * theta?"

Text:

The lectures and exercise PDF files are based on Prof. Ng's feeling that novice programmers will adapt to for-loop techniques more readily than vectorized methods. So the videos (and PDF files) are organized toward processing one training example at a time. The course uses column vectors (in most cases), so h (a scalar for one training example) is theta' * x.

Lower-case x typically indicates a single training example.

The more efficient vectorized techniques always use X as a matrix of all training examples, with each example as a row, and the features as columns. That makes X have dimensions of (m x n). where m is the number of training examples. This leaves us with h (a vector of all the hypothesis values for the entire training set) as X * theta, with dimensions of (m x 1).

X (as a matrix of all training examples) is denoted as upper-case X.

Throughout this course, dimensional analysis is your friend.

Subject: Tips from the Mentors: submit problems and fixing program errors

Text:

This post contains some frequently-used tips about the course, and to help get your programs working correctly.

The Most Important Tip:

Search the forum before posting a new question. If you've got a question, the chances are that someone else has already posted it, and received an answer. Save time for yourself and the Forum users by searching for topics before posting a new one.

Running your scripts:

At the Octave/Matlab command line, you do not need to include the ".m" portion of the script file name. If you include the ".m", you'll get an error message about an invalid indexing operation. So, run the Exercise 1 script by typing just "ex1" at the command line.

You also do not need to include parenthesis () when using the submit script. Just type "submit".

You cannot execute your functions by simply typing the name. All of the functions you will work on require a set of parameter values, enter between a set of parenthesis. Your three methods of testing your code are:

1 - use an exercise script, such as "ex1"

2 - use a Unit Test (see below) where you type-in the entire command line including the parameters.

3 - use the submit script.

Making the grader happy:

The submit grader uses a different test case than what is in the PDF file. These test cases use a different size of data set and are more sensitive to small errors than the ex test cases. Your code must work correctly with any size of data set.

Your functions must handle the general case. This means:

- You should avoid using hard-coded array indexes.

- You should avoid having fixed-length arrays and matrices.

It is very common for students to think that getting the same answer as listed in the PDF file means they should get full credit from the grader. This is a false hope. The PDF file is just one test case. The grader uses a different test case.

Also, the grader does not like your code to send any additional outputs to the workspace. So, every line of code should end with a semicolon.

Getting Help:

When you want help from the Forum community, please use this two-step procedure:

1 - Search the Forum for keywords that relate to your problem. Searching by the function name is a good start.

2 - If you don't find a suitable thread, then do this:

2a - Find the unit tests for that exercise (see below), and run the appropriate test. Attempt to debug your code.

2b - Take a screen capture of your whole console workspace (including the command line), and post it to the forum, along with any other useful information (computer type, Octave/Matlab version, other tests you've tried, etc).

Debugging:

If your code runs but gives the wrong answers, you can insert a "keyboard" command in your script, just before the function ends. This will cause the program to exit to the debugger, so you can inspect all your variables from the command line. This often is very helpful in analysing math errors, or trying out what commands to use to implement your function.

There are additional test cases and tutorials listed in pinned threads under "All Course Discussions". The test cases are especially helpful in debugging in situations where you get the expected output in ex but get no points or an error when submitting.

Unit Tests:

Each programming assignment has a "Discussions" area in the Forum. In this section you can often find "unit tests". These are additional test cases, which give you a command to type, and provides the expected results. It is always a good idea to test your functions using the unit tests before submitting to the grader.

If you run a unit test and do not get the correct results, you can most easily get help on the forums by posting a screen capture of your workspace - including the command line you entered, and the results.

Having trouble submitting your work to the grader?:

- This section will need to be supplemented with info appropriate to the new submission system. If you run the submit script and get a message that your identity can't be verified, be sure that you have logged-in using your Coursera account email and your Programming Assignment submission password.

- If you get the message "submit undefined", first check that you are in the working directory where you extracted the files from the ZIP archive. Use "cd" to get there if necessary.

- If the "submit undefined" error persists, or any other "function undefined" messages appear, try using the "addpath(pwd)" command to add your present working directory (pwd) to the Octave execution path.

-If the submit script crashes with an error message, please see the thread "Mentor tips for submitting your work" under "All Course Discussions".

-The submit script does not ask for what part of the exercise you want to submit. It automatically grades any function you have modified.

Found some errata in the course materials?

This course material has been used for many previous sessions. Most likely all of the errata has been discovered, and it's all documented in the 'Errata' section under 'Supplementary Materials'. Please check there before posting errata to the Forum.

Error messages with fmincg()

The "short-circuit" warnings are due to use a change in the syntax for conditional expressions (| and & vs || and &&) in the newer versions of Matlab. You can edit the fmincg.m file and the warnings may be resolved.

Warning messages about "automatic broadcasting"?

See this link for info.

Warnings about "divide by zero"

These are normal in some of the exercises, and do not represent a problem in your function. You can ignore them - Octave senses the issue and substitutes a +Inf or -Inf value so your program continues to execute.