10-601A/SV: Introduction to Machine Learning (F15)

PROGRAMMING ASSIGNMENT 0

Octave and Autolab

Di Jin <dijin@andrew.cmu.edu>, Dawei Wang <daweiwan@andrew.cmu.edu>

DUE: 11:59 PM EDT, September 9th 2015

- Homework policy may be found at: `https://piazza.com/class/idvpb9zviwz6w5?cid=7`. You may also find additional clarification on this homework somewhere on Piazza, if any.
- All programming assignments will be submitted electronically through Autolab. You should be able to sign in with your Andrew ID at `https://autolab.cs.cmu.edu/`; if not, contact the teaching assistants listed above for access privileges.
- All programming assignments will be graded automatically on Autolab using **Octave 3.6.4**. You may develop your code in your favorite IDE, but please make sure that it runs as expected on Octave before submitting. *To assure that your submission can be processed properly by the grading scripts, please adhere to the function signatures, argument formats, file names, and submission instructions we have specified.*
- You may discuss with other students to tackle any of the programming assignments; however, you must write the code completely on your own. Include a text file in the submission that elaborates on how you have collaborated with other students, and what kind of external resources you have made use of substantially. Any violation of academic integrity will result in severe consequences to the maximum extent permitted by university policies.

* * *

This assignment aims to help you get familiar with the programming environment that we will be using throughout this semester. To begin with, install Octave:

`https://www.gnu.org/software/octave/download.html`

Since Autolab uses Octave version 3.6.4, there is a remote possibility that the code you have developed uses some new functions and features that do not yet exist in 3.6.4. In this case, please check the feedback message generated by the automatic grader.

Octave is a high-level interpreted language primarily intended for numerical computations. Its syntax is very similar to Matlab, with quite a few additional features borrowed from other programming languages such as `C++`. Please note that a considerable number of Matlab functions are not available in Octave, and vice versa. [1] Before proceeding to the next section, you may want to go through some online tutorials to familiarize yourself with the basics.

[1] However, Matlab requires additional purchase for applications such as Autolab. Octave is free.

Distributed with the handout is a directory named `prog0`. It typically consists of several files that you may use directly, or have to complete as a part of the assignment. In this case, there are two Octave function files, namely `ulr.m` and `ulr_plot.m`, which you will have to fill-in in the subsequent tasks. The following section depicts the format of a programming problem.

1 Linear Regression

Linear regression is an approach for modelling the relationship between multiple variables. Comprehensive introductions to this topic may be found elsewhere and are hence omitted here for brevity. In this assignment, you will be implementing *univariate linear regression* to determine the relationship between two variables, just to develop some familiarity with Octave.

This model assumes that two variables x and y exhibit the following relationship:

$$y = w_1 x + w_2 + \varepsilon \tag{1}$$

where ε is the *error variable*, and w_1 and w_2 are the *vector parameters*. Given a set of n observations, x_i and y_i, for $i = 1, 2, \ldots, n$, the goal is to determine the most likely values of w_1 and w_2, denoted as $\hat{w}_1$ and $\hat{w}_2$, such that the errors between the predicted values $\hat{w}_1 x_i + \hat{w}_2$ and the observed values y_i are minimal. There are many approaches that we can employ to solve this problem. A popular choice is the *least squares*, which essentially derives w_1 and w_2 by differentiating the sum of squared vertical errors and sets it to zero, resulting in

$$\hat{w}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},\ \hat{w}_2 = \bar{y} - \hat{w}_1 \bar{x} \quad \text{where} \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{2}$$

TASK 1 (20 points)

Write a function with the signature

```
function [w1, w2] = ulr(x, y)
```

that finds the most likely values of w_1 and w_2 using the least squares approach.

INPUT:

- `x`: a column vector that contains n observations of x;
- `y`: a column vector that contains the n corresponding observations of y.

OUTPUT:

- `w1`, `w2`: scalars that correspond to w_1 and w_2 in Equation (1), respectively.

FILE: `ulr.m`

In fact, solving an univariate linear regression problem with the least squares approach has a nice geometric interpretation: it fits a straight line through the set of points (x_i, y_i), $i = 1, 2, \ldots, n$, such that the sum of squared vertical distances from the points of the data set to the fitted line reaches its minimum. We can plot these elements on a figure for a visual inspection.

TASK 2 (20 points)

Write a function with the signature

```
function [h] = ulr_plot(x, y, w1, w2)
```

that plots the data points and the fitted line using `scatter` and `line`. Do not re-compute w_1 and w_2 from x and y; use the passed values instead. Be aware that using other functions may cause your figure to be unrecognizable by the autograder.

INPUT:

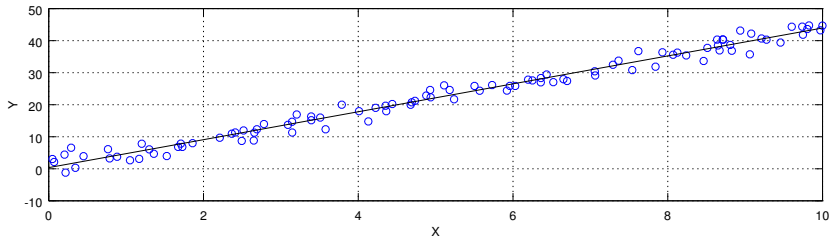
- `x`: a column vector that contains x_i, for $i = 1, 2, \ldots, n$;
- `y`: a column vector that contains y_i, for $i = 1, 2, \ldots, n$;
- `w1`, `w2`: the two scalars obtained by invoking `ulr` with `x` and `y`.

OUTPUT:

- `h`: a handle to a *figure* that contains the data points and the fitted line.

FILE: `ulr_plot.m`

Here is an example:



It is usually a good idea to generate some toy dataset and check to see that the functions that you have developed produce the correct results. As a matter of fact, your code will be trialled against multiple sets of toy datasets generated in accordance with Equation (1).

Now you are ready to submit your code. Produce a single `tar` file by compressing the `prog0` directory that contains both `ulr.m` and `ulr_plot.m`; then, proceed to `https://autolab.cs.cmu.edu` and submit under the appropriate assessment. It will take several seconds for the autograder to run your code. If your code exits without any problems, grades will be assigned for each individual task; if you are getting zero points for some of the tasks, a runtime error might have occurred. Check the job status for details.

You might submit as many times as you would like. Your final grade for this programming assignment will be *the grade for your last submission before the due date.*