# 10601 Machine Learning Assignment 4: Linear Regression

Due: Oct. 2nd, 16:30 EST (**Before the lecture**), via AutoLab
Late submission due: Oct. 4th, 23:59 EST with 50% discount of credits

TAs-in-charge: Guanyu Wang, Pengtao Xie

## Policy on Collaboration among Students

These policies are the same as were used in Dr. Rosenfeld's previous version of 2013. The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone, and the student should be ready to reproduce their solution upon request. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment. Specifically, each assignment solution must start by answering the following questions in the report:

- Did you receive any help whatsoever from anyone in solving this assignment? Yes / No. If you answered 'yes', give full details: _____ (e.g. "Jane explained to me what is asked in Question 3.4")
- Did you give any help whatsoever to anyone in solving this assignment? Yes / No. If you answered 'yes', give full details: _____ (e.g. "I pointed Joe to section 2.3 to help him with Question 2".

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism. As a related point, some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be

(or may have been) available online, or from other people. It is explicitly forbidden to use any such sources, or to consult people who have solved these problems before. You must solve the homework assignments completely on your own. I will mostly rely on your wisdom and honor to follow this rule, but if a violation is detected it will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

# 1 Linear Regression

In this assignment, you will need to implement the Linear Regression (LR) method to recover an unknow function $f(\cdot)$. In other words, assume there is a set of data $\{(y_i, \mathbf{x_i})\}_{i=1}^m$, ($\mathbf{x_i}$ is the $n$ dimension feature vector), and we already know that $y_i$ and $\mathbf{x_i}$ satisifies certain kind of relation, i.e. $\forall i \ y_i = f(\mathbf{x_i}) + \epsilon = \theta^T \mathbf{x_i} + \epsilon$ where $\epsilon$ is an error term of unmodeled effects or random noise. You need to find this relaion $f(\cdot)$ on the data we provide in this assignment.

Note that Linear Regression does **NOT** mean it can only deal with linear relationships (even it has the word "linear" in it). You can always design (non-linear) features under it, i.e. instead of using $\mathbf{x_i}$ directly, you can create the nonlinear basis function $\Phi(\mathbf{x})$ then the relation can be $y_i = \theta^T \Phi(\mathbf{x_i}) + \epsilon$ now. Please refer the lecture slides `http://curtis.ml.cmu.edu/w/courses/images/f/f0/Lecture6-LiR.pdf`, page 33 for more details.

# 2 Methods for solving Linear Regression

In order to solve the LR problem, three different kinds of methods have been explained in the class. They are LMS update rule, steepest descent and normal equations.

**Notice** that all the following explanations of different methods do not use the basis functions, but you need to use them in your implementation.

## 2.1 Steepest descent

First define the cost function (Least-Mean-Square):

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (\mathbf{x_i}^T \theta - y_i)^2 \tag{1}$$

where the $(\cdot)^T$ is the transpose operation for matrix, i.e. $(A^T)_{ij} = A_{ji}$.

Using the common gradient descent algorithm, we can get the batch gradient descent update rule (here "batch" means taking all the training data into consideration in each step)

$$\theta^{t+1} = \theta^t + \alpha \sum_{i=1}^{m} (y_i - \mathbf{x_i}^T \theta) \mathbf{x_i} \tag{2}$$

where the $\alpha$ is the learning rate and $\theta^{t+1}$ is the learned $\theta$ after $t+1$ rounds

## 2.2 LMS update rule

Still use the same cost function as Eq. 1, but if we handle all the training data in the one-by-one fashion (online learning), the update rule can be

$$\theta_j^{t+1} = \theta_j^t + \alpha (y_i - \mathbf{x_i}^T \theta) \mathbf{x_{i,j}} \tag{3}$$

where the $\theta_j^{t+1}$ is the $j$th entry of the vector $\theta^{t+1}$, and $\mathbf{x_{i,j}}$ is the $j$th entry of the training data vector $\mathbf{x_i}$.

So the updating process can be done as follows

**while** convergence condition is not satisfied **do**
  shuffle the training examples;
  **for** each training example $(y_i, \mathbf{x_i})$ **do**
    **for** each entry $j$ **do**
      $\theta_j = \theta_j + \alpha (y_i - \mathbf{x_i}^T \theta) \mathbf{x_{i,j}}$;
    **end for**
  **end for**
**end while**

## 2.3 Normal equation

We can also minimize the cost function $J(\theta)$ explicitly and without resorting to an iterative algorithm.

Given a training set $X$ to be the $m$-by-$n$ matrix (actually $m$-by-$n+1$, if we include the bias or intercept[1] term, ) that contains the training examples feature vector in its rows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_m}^T \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}. \tag{4}$$

[1]Most of the time, peope add a bias term $\theta_0$ in the function, i.e. using $\mathbf{x_i}^T = [1 \ \mathbf{x_i}^T]$ as the input data point for $\mathbf{x_i}$, so the training matrix has one more column than the real number of dimensions $\mathbf{x_i}$ has.

In this setting, we can compute in closed-form the value of $\theta$ which minimize the cost function:

$$\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{5}$$

For more details of the proof, please refer the lecture slides `http://curtis.ml.cmu.edu/w/courses/images/f/f0/Lecture6-LiR.pdf`, page 9, and you can find the comparison among three methods in page 13.

# 3 Prediction

After achieving the $\theta$, you can use it to predict the output value $y'$ given a new input $\mathbf{x}'$ by simply computing

$$y' = \theta^T \mathbf{x}'_{with\ bias} \tag{6}$$

Remember, since you add the intercept term, $\mathbf{x}'^T_{with\ bias} = [1\ \mathbf{x}'^T]$

# 4 Implementation Notes

You should implement all three methods to solve the LR task in this assignment. Your implementation should have three function:

```
LR_SteepestDescent(Xtrain, Ytrain, Xtest, Ytest);
LR_LMS(Xtrain, Ytrain, Xtest, Ytest);
LR_NormalEquation(Xtrain, Ytrain, Xtest, Ytest);
```

The $Xtrain$ and $Xtest$ both have the same format as shown in section. 2.3, i.e. these matrices contains the data in their rows. And $Ytrain$, $Ytest$ are their corresponding target values. All your three functions should save the predicted values, i.e. Ytest, as .mat files with name **Ypred_SD.mat**, **Ypred_LMS.mat**, **Ypred_NE.mat** (same method as previous homeworks). And print out the testing error. You can compute the error with

```
error = norm(Ypred_XXX - Ytest);
fprintf('Error = %.3f\n', error);
```

**For intercept term:** after adding the intercept term, the input $X$ matrix should have the following format (i.e. a $m$-by-$n+1$ matrix)

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbf{x_1}^T \\ 1 & \mathbf{x_2}^T \\ \vdots & \\ 1 & \mathbf{x_m}^T \end{bmatrix} \tag{7}$$

**For basis functions:** the data set we provide to you in this assignment is just the one dimension vector, i.e. all $\mathbf{x_i}$s are just the single values. And the unknown relation $f(x)$ is not necessarily a linear one, so you might want to (actually you **MUST**) use the basis functions. We suggest the 3-order polynomial basis functions, i.e. $\Phi(x) = [1, x, x^2, x^3]$. You can choose your own basis functions, but you have to make sure they provide no worse performance than the suggested one and the computational complexity is decent. Your code should not cause timeout error when running on the AutoLab!

**For learning rate:** please refer lecture note page 12 for the learning rate selection. I would suggest that you can start with some small value like $10^{-5}$, $10^{-6}$, make sure the algorithm can converge and then gradually increase it to satisfy the speed requirement.

**For convergence condition:** you should **NOT** use the testing error as the convergence condition. We will **NOT** provide the true $Ytest$ matrix to your function when we do the held-out test. Instead, you should use the changes of tranining error (i.e. apply the learned function to the $Xtrain$ and compare the result to $Ytrain$ ) after each round as the convergence condition. You should choose an appropriate threshold to make sure the performances of LMS and Steepest Descent are close to what you get with Normal Equation method.

# 5 Data Set

For this assignment, you should download the handout data from `http://curtis.ml.cmu.edu/w/courses/images/d/da/Assignment4-handout.mat`. It contains four matrices:"Xtrain", "Ytrain", "Xtest", "Ytest".

In the X (training set and testing set) matrices, each row is a feature vector (actually just a single value in this assignment). The Y matrices provide the target values generated by $f(\mathbf{x_i}) + \epsilon$ for each $x_i$.

If you plot the traning data $Xtrain$ and $Ytrain$ in 2D with

```
plot(Xtrain, Ytrain, '*');
```

You can get a figure just like Fig. 1

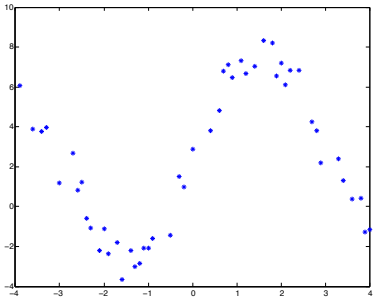

Figure 1: Training data in 2D

# 6 Deliverables

Submit your codes of all three functions via AutoLab. You should implement the linear regression algorithm by yourself instead of using any built-in functions. Also, you are NOT allowed to use any off-the-shelf optimizer.

You should upload your codes (including all 3 function files) along with a report, which should solve the following question:

Compare all three different methods for solving LR. For this purpose, pick updating rounds (number of passes over all the data), 1, 2, 5, 10, 20, 50, 100 to compute $\theta$. Plot the testing error as a function of the updating rounds for all three functions (x-axis should be "number of rounds" and y-axis should be "testing error"). You can use the same way to compute the So there will be three curves in your figure.

**Notice** that the **Normal Equation** method can achieve the solution in a single step, so the testing error curve for function LR_NormalEquation should be just a horizontal line. You can use the **inv**$(\cdot)$ or matrix division operations to compute the equation directly.

You should tar gzip the following items into **hw4.tgz** and submit to the homework 4 assignment under Autolab:

- LR_SteepestDescent.m

- LR_LMS.m
- LR_NormalEquation.m
- and all other auxiliary functions you have written
- report.pdf

Tar gzip the files directly using "tar -cvf hw4.tgz *.m report.pdf". Do **NOT** put the above files in a folder and then tar gzip the folder. You do not need to upload the saved predicted labels (i.e. the .mat files). Please make sure your code is working fine under Octave before you submit.

# 7 Submission

You must submit your homework through Autolab via the "Homework4-submission" link. In this homework, we provide an additional tool called "Homework4-validation":

- Homework4-validation: You will be notified by Autolab if you can successfully finish your job on the Autolab virtual machines. Note that this is not the place you should debug or develop your **algorithm**. All development should be done on linux.andrew.cmu.edu machines. This is basically a Autolab debug mode. There will be **NO** feedback on your **performance or score** in this debugging mode. You have unlimited amount of submissions here (All the same as homework3).
- Homework4-submission: This is where you should submit your validated final submission. You have a total of 5 possible submissions. Your performance will be evaluated, and feedback will be provided immediately.