# DS1003 Machine Learning and Computational Statistics
# Spring 2015: Homework 1

Due: Tuesday, October $14^{th}$, beginning of class

## 1 Introduction

In this exercise, you will implement stochastic gradient descent for both linear regression. To get started with the exercise, you will need to download the dataset and starter code and unzip its contents to the directory where you wish to complete the exercise.

## 2 Linear Regression [8 points]

### 2.1 Feature Normalization

In the dataset, when features differ by orders of magnitude, it might cause the problem of slow convergence speed. Performing feature scaling can make gradient descent converge much more rapidly. Modify function `feature_normalize` to normalize all the features to [0, 1].

### 2.2 Batch Gradient Descent

Now you will fit the linear regressions parameters $\theta$ to the dataset using gradient descent. The objective of linear regression is to minimize the cost function

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

Where the $h_\theta(x_i)$ is given by the linear model:

$$h_\theta(x) = \theta^T x = \theta_0 + \sum_{j=1}^{l} \theta_j x_j$$

In batch gradient descent, your parameters $\theta_j$ should come closer to the optimal values that will archive a lower cost $J(\theta)$.

1. *(3 points)* Write down the formula for updating $\theta$.
2. *(4 points)* Modify the function `compute_cost`, this function is used to compute the loss given $\theta$.
3. *(4 points)* Find the optimal $\theta$ using batch gradient descent, you need to modify function `gradient_descent`. In this question you should use the default parameters provided by skeleton code.
4. *(4 points)* In batch gradient descent, the lost function $J(\theta)$ should get smaller in each iteration, to verify your gradient descent implementation is correct, plot the curve of cost J as number of iterations grows. You may need to tune the update step size $\alpha$ to reach a good convergence speed.

## 2.3 Gradient Checker

In gradient descent algorithms, to avoid falling to the wrong optima, we want to make sure in each update step the gradient we computed is correct. A basic idea is numerically approximate the derivatives. For example, the mathematical definition of the derivative is:

$$\frac{d}{d\theta}J(\theta) = \lim_{\epsilon \to 0} \frac{J(\theta+\epsilon) - J(\theta+\epsilon)}{2\epsilon}$$

Then at any specific value of $\theta$, we can numerically approximate the derivative using the following form, in which we set $\epsilon$ to some small constant

$$g(\theta) \approx \frac{J(\theta+\epsilon) - J(\theta+\epsilon)}{2\epsilon}$$

In this function you will modify function `grad_checker` and implement the function in `batch_grad_descent`
See `http://ufldl.stanford.edu/wiki/index.php/Gradient_checking_and_advanced_optimization` for details.

## 2.4 Regularization

Firstly, also visualize the loss on validation dataset as the number of iterations grow. Plot the loss of both training and validation set in the same figure, what is the plot like? Could you explain the reason?
So we modify the loss function of linear regression to the form:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 + \frac{\lambda}{2m}(\sum_{j=1}^{n}\theta_j^2)$$

where $\lambda$ is the regularization parameter which controls the degree of regularization (ie, help preventing overfitting).

1. *(3 points)* We have regularization term in the loss function now, so the gradient of loss function also changed. Write down the new formula for updating $\theta$.
2. *(4 points)* Now you should design a new function to calculate the regularized loss function.
3. *(4 points)* Design a new function to find the optimal $\theta$ using batch gradient descent, in this question fix the regularization parameter to $\lambda = 0.1$.
4. *(4 points)* Visualize the loss as iteration number increases, in visualization the loss should only be the sum of square loss, and not include the regularization term. Your visualization should display the trend in both training and validation set, report the loss at step1, 5, 10, 20. What's the optimal iteration number?
5. *(4 points)* Try different regularization parameter $\lambda$. You should find the corresponding optimal iteration number with each $\lambda$. Plot the loss of both training and validation data with different $\lambda$. What's the plot is like? What's the optimal $\lambda$ based on your result? Could you explain the reason?

## 2.5 Stochastic Gradient Descent

When training data size is extremely large, evaluating the gradient of full loss $J(\theta)$ will be very expensive, since it requires evaluating the gradient on each instance. In this case, stochastic gradient descent become very effective. In SGD, the true gradient is approximated by a gradient at a single example. The algorithm sweep through the whole training set one by one, and perform update for each training example individually.

1. *(3 points)* Now write down the update rule in SGD

2. *(4 points)* Modify your `stochastic_grad_descent` code to make it implement SGD.

3. *(4 points)* Visualize the convergence of SGD. As a comparison, report the time elapsed and the loss in batch/stochastic gradient descent. Which one can reach a smaller total loss? Which one takes less time until convergence? Explain why.

### 2.6 Adaptive Learning Rate

TODO