Homework 05

Regularized Linear Regression and Bias v.s. Variance Section 1

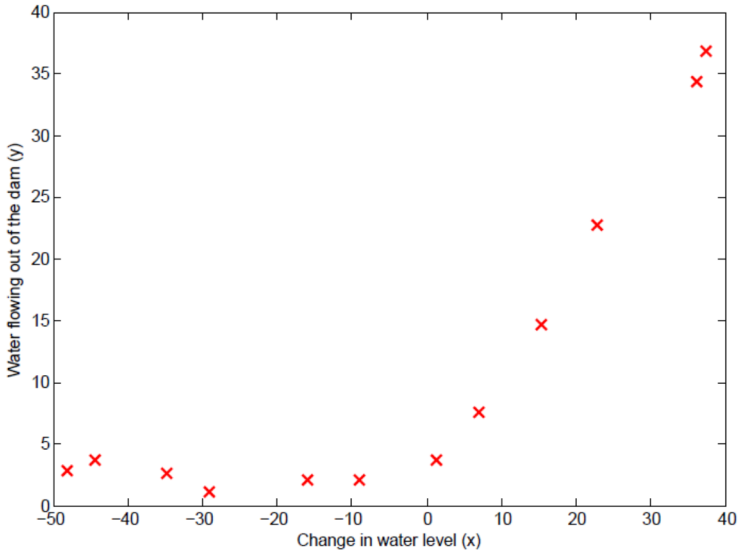
In this exercise, you will implement regularized linear regression and use it to study models with different bias-variance properties. In the first half of the exercise, you will implement regularized linear regression to predict the amount of water owing out of a dam using the change of water level in a reservoir. In the next half, you will go through some diagnostics of debugging learning algorithms and examine the effects of bias v.s. variance. The provided script, **hw5.m**, will help you step through this exercise.

Part 1: Visualizing the data (0 points)

We will begin by visualizing the dataset containing historical records on the change in the water level, x, and the amount of water owing out of the dam, y. This dataset is divided into three parts:

- A **training** set that your model will learn on: **X, y**
- A **cross validation** set for determining the regularization parameter: **Xval, yval**
- A **test** set for evaluating performance. These are "unseen" examples which your model did not see during training: **Xtest, ytest**

The next step of hw5.m will plot the training data (see Figure below). In the following parts, you will implement linear regression and use that to fit a straight line to the data and plot learning curves. Following that, you will implement polynomial regression to find a better fit to the data.



Part 2: Regularized Linear Regression Cost Function (25 points)

Recall that regularized linear regression has the following cost function:

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\right) + \frac{\lambda}{2m}\left(\sum_{j=1}^{n}\theta_j^2\right)$$

where λ is a regularization parameter which controls the degree of regularization (thus, help preventing overfitting). The regularization term puts a penalty on the overall cost J. As the magnitudes of the model parameters θ_j increase, the penalty increases as well. Note that you should not regularize the θ_0 term. (In Octave, the θ_0 term is represented as **theta(1)** since indexing in Octave starts from 1).

You should now complete the code in the file **linearRegCostFunction.m**. Your task is to write a function to calculate the regularized linear regression cost function. If possible, try to vectorize your code and avoid writing loops. When you are finished, the next part of hw5.m will run your cost function using theta initialized at [1; 1]. You should expect to see an output of **303.993**.

Part 3: Regularized linear regression gradient (25 points)

Correspondingly, the partial derivative of regularized linear regression's cost for θ_j is defined as

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad \text{for } j \geq 1$$

In **linearRegCostFunction.m**, add code to calculate the gradient, returning it in the variable grad. When you are finished, the next part of hw5.m will run your gradient function using theta initialized at [1; 1]. You should expect to see a gradient of **[-15.30; 598.250]**.

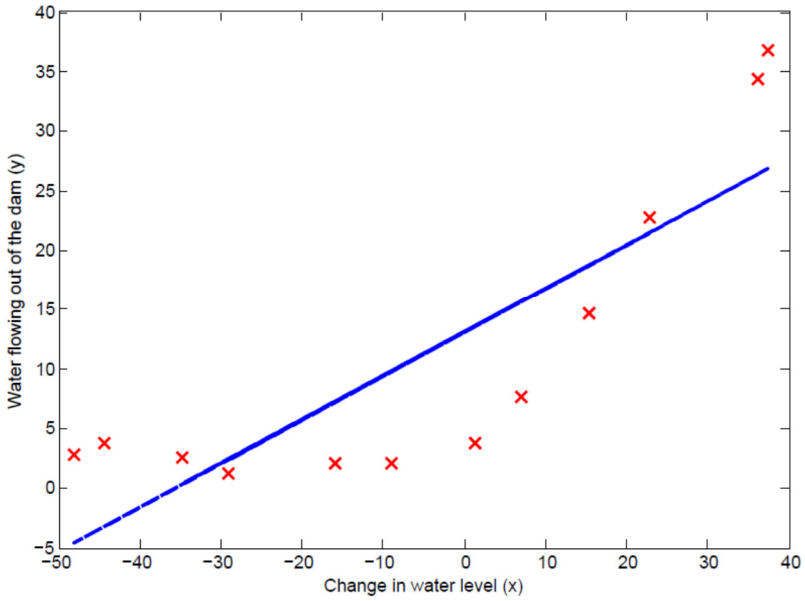
Part 4: Fitting Linear Regression (0 points)

Once your cost function and gradient are working correctly, the next part of hw5.m will run the code in **trainLinearReg.m** to compute the optimal values of θ. This training function uses fmincg to optimize the cost function.

In this part, we set regularization parameter λ to zero. Because our current implementation of linear regression is trying to fit a 2-dimensional θ, regularization will not be incredibly helpful for a θ of such low dimension. In the later parts of the exercise, you will be using polynomial regression with regularization.

Finally, the hw5.m script should also plot the best fit line, resulting in an image similar to the one shown below. The best fit line tells us that the model is not a good fit to the data because the data has a non-linear pattern. While visualizing the best fit as shown is one possible way to debug your learning algorithm, it is not always easy to visualize the data and model. In the next section, you will implement a function to generate learning curves that can help you debug your learning algorithm even if it is not easy to visualize the data.

Homework 05



Submission:

To submit, turn in the following files on Canvas:

- **linearRegCostFunction.m**