IE 7374 Machine Learning in Engineering

LAB 1
Due by Oct, 9, 2020

Gradient Descent for Linear Regression

For this lab, you will work as a team and turn in your team work. You will have the same lab group throughout the semester. The lab assignment is to develop gradient-based algorithm to estimate regression coefficients, and then compare with the analytical solution we discussed and you replicated in Python.

In this lab you will be working with three datasets for regression. The data are in the file housing.csv, yachtData.csv and concreteData.csv.

- **Housing**: This is a regression dataset where the task is to predict the value of houses in the suburbs of Boston based on thirteen features that describe different aspects that are relevant to determining the value of a house, such as the number of rooms, levels of pollution in the area, etc.
- **Yacht**: This is a regression dataset where the task is to predict the resistance of a sailing yacht's structure based on six different features that describe structural and buoyancy properties.
- **Concrete**: This is a regression dataset where the task is to predict the compressive strength of concrete on eight different features. There are a total of 1030 instances and all the features are numeric.

1. **Linear Models for Regression**. Recall the model for linear regression: $y = \hat{y} + \epsilon$, where ϵ is the error term with normal distribution $\mathbb{N}(0, \sigma^2)$ and

$$\hat{y} = f(x) = w^T x.$$

We choose parameter w to minimize the following square loss objective function

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(w_i x_i - y_i)^2,$$

where $\{(x_1, x_2), \cdots, (x_n, y_n)\}$ is our training data.

Your task is to estimate regression coefficients w_i using gradient descent methods and then predict the output y based on the observed data x.

2. **Feature Normalization**. When feature values differ greatly, we can get much slower rates of convergence of gradient-based algorithms. Furthermore, when we start using regularization. Features with larger values are treated as "more important", which is not usually what you want. One common approach to feature normalization is perform an affine transformation (i.e. shift and rescale) on each feature so that all feature values in the training set are in $[0, 1]$. Each feature gets its own transformation. We then apply the same transformations to each feature on the test set. It's important that the transformation is "learned" on the training set, and then applied to the test set. It is possible that some transformed test set values will lie outside the $[0, 1]$ interval.

   In initial efforts we normalized the entire dataset prior to learning. However, the correct way of normalizing data would be to normalize the training data and record the normalization parameters i.e., mean and standard deviation for z-score normalization and min and max feature values for feature re-scaling. This minimizes the chances of introducing any bias during the performance evaluation of the learning algorithm. In a real-world scenario you would not have access to test data during the training phase.

   To summarize: only normalize your training data, and then use the normalization parameters to normalize your test data and then estimate the accuracy/error.

   For every regression problem remember adding a column of ones to your dataset to capture the regression intercept.

3. **Major Steps**.

   - Normalize the features in the training data using z-score normalization.
   - Initialize the weights for the gradient descent algorithm to all zeros i.e., $w = [0, 0, \cdots, 0]^T$.
   - Calculate the risk function and its gradient with respect to the parameters.
   - Update the estimates of regression coefficients over iterations while using the following set of parameters

     (a) Housing: learning rate=0.4×10^{-3}, tolerance=0.5×10^{-2}
     (b) Yacht: learning rate=0.1×10^{-2}, tolerance=0.1×10^{-2}
     (c) Concrete: learning rate=0.7×10^{-3}, tolerance=0.1×10^{-3}

     NOTE: the tolerance is defined based on the difference in root mean squared error (RMSE) measured on the training set between successive iterations. Where, RMSE is defined as:

     $$\textbf{RMSE} = \sqrt{\frac{SSE}{n}},$$

     SSE=error sum of squares and n is the number of training cases.

   - Please also experiment with other learning rates and tolerance such as $0.1, 0.01, 0.05$ and report out.

- Be sure to set a maximum number of iterations (recommended maximum iterations = 50000) so that the algorithm does not run forever.

Least Squares Estimation using Normal Equations

Use the housing and yacht dataset to estimate the regression weights by normal equations. Compare the performance (measured through RMSE) with the results obtained using the gradient descent algorithm.

In this section you will calculate the analytical solution in Python that we obtained through Normal equations to learn your weight vector, and contrast the performance (training and test RMSE) for your gradient-descent based implementation.

Submission

When you turn in your assignment, please submit electronically (on Canvas) **BOTH** your results and your Python code that you used to solve the problems. Please make sure you codes are reproducible so TA can grade accordingly.