Multiple Linear Regression: Predicting House Prices

Introduction

I have chosen the above problem to propose a solution for. The problem page is found on

https://www.hackerrank.com/challenges/predicting-house-prices/copy-from/14723054.

As the name indicates, this challenge deals with a simple "Multiple Linear Regression" problem. The task is to create a program where regression parameters are learned from the given training data and the predicted values are generated for the given test data. The challenge consists of 3 tasks, and basically the dimensions and the number data samples are different across these tasks.

The proposed code obtained 10 out of 10 marks for this challenge.

Solution outline

Multiple Linear regression is a simple operation which can be summarized with the vector –matrix equations:

$$w = (X'X)^{-1}X't$$

$$X\,w = \hat{t}$$

Where X is the training sample matrix whose rows corresponding to each training vector, w is the regression parameter vector, t is the observed target values for the training samples and $\hat{t}$ is the predicted values for any X and w.

The task is therefore to compute the w vector based on X and t (first equation) and use the w vector to calculate the target values (second equation).

The pitfall is however, that calculation of inverse matrix is computationally heavy. Therefore, it is much cheaper to calculate w vector considering it as a system of linear equations and solving it using a technique like Gauss elimination.

Implementation

The solution is implemented with a C++ class called *mlr* which does the following tasks:

1. Read the training and test data from the standard input (*read_data()* function)
2. Compute the w vector using the Gauss elimination technique (first equation implemented in *fit_data()* function)

3. Compute the predicted values for the test data (second equation implemented in *fit_data()* function)
4. Output the predicted values on the standard output (*fit_data()* function).

There is one supporting class *CMatrix* which implements the matrix operations[1]. Gauss elimination is embedded in this class as well.

Conclusion

The resulting code achieved the intended result, namely to pass all the tests of the challenge within the allocated time. However, there can be several issues which can surface when the problem is scaled up (eg: high feature dimension). One obvious issue is the interface between the *CMatrix* class and the *mlr* class. Since the matrix objects are passed between these classes as return values unnecessary copying of the matrix objects take place. This could have been improved having, for example, output arguments in the function calls.

[1] CMatrix class is based on the code published on the web.