Least Squares Optimization with K-Fold Cross-Validation and Analysis of Real-Life Data

Part A) Least Squares Optimization with K-Fold Cross Validation

Abstract

Purpose: The first goal set out is to reproduce plots given by the instructor, it is proposed that our initial solution found by Ordinary Least Squares (OLS) is infeasible so through the implementation of Constrained Least-Squares (CLS) it is possible to find a more optimal solution.

Once two variations of Least Squares has been implemented, a technique called K-Fold Cross Validation, in particular 5-Fold Cross Validation, will be used to confirm the improved accuracy of CLS. By implementing this technique it becomes clear the approach many common packages take when implementing this cross validation an important concept to understand.

Methods: The CLS routine was implemented using MATLAB, the script named a3q1_20051918.m, to take advantage of several built-in functions. A skeleton script was provided by Dr. Ellis allowing for the focus to be on the implementation of the algorithm instead. The CLS routine in the associated script follows from the method described in lecture, class 24.

Similarly, K-Fold Cross Validation was done in MATLAB but in a new script labeled a3q2_20051918. The script still utilizes the least squares methods from the previous script. This is done by a working method which follows the implementation outline done in class 25. The main difficulties arise from coding nuances due to indexing.

Results: The results for the CLS problem can be found in the respective section. These include a plot with the proper reconstruction of the instructor's plot and a table indicating the new weight vector solution and the optimal Lagrange multiplier.

K-Fold Cross Validation allows for insight into the accuracy of our obtained optimal weight vectors. For a more complete understanding of K-Fold Cross Validation, for both Least Squares variations, 10 sets of 5-Fold validation is performed. The code reports back the statistical mean and standard deviation, from these 10 sets.

Conclusions: From the results it is clear that the CLS solution shown in the plot as a blue line, provides a much more sensible solution to the model. The OLS solution both under and overestimates the data whereas the CLS solution does not. This provides a basis to further build upon to properly analyze the data.

Through the statistical results generated by K-Fold Cross Validation, the increase in performance between the two least squares solutions is easily observable.

Introduction

Throughout question one of this assignment the aim is to better understand the benefits provided by Constrained Least Squares as a method of finding optimal values and when CLS should be applied to a situation versus OLS. This idea will be further supported by running ten cases of 5-Fold Cross Validation and viewing the basic statistics which are generated.

Constrained Least Squares provides a particular benefit over the ordinary variation. For problems where the data is overdetermined, that is more equations than unknowns, it is not guaranteed a solution exists and when it does it is not necessarily a feasible solution. As stated in lecture the obtained solution is nonsensical. So, where does K-Fold Cross Validation have an effect on all of this? Well, K-Fold Cross Validation allows the original data to be partitioned into K training and testing folds. This is typically done using pseudo random selection as done in this assignment. K-Fold Cross Validation then provides a method of evaluating the effectiveness of a regression (or classification) model on numerous training and test folds.

The nonsensical solution of OLS appears due to statistical outliers in the data. It is quickly observable by the plot in the results section that the data used in question one and two of this assignment contains outliers. These outliers quadratically influence the OLS solution with distance thus by applying a constraint on the system it is possible to mitigate this effect. K-Fold Cross Validation evaluates both methods and their ability to predict on K test folds from being trained with K training folds. Each iteration of the K-Fold Cross Validation technique produces an RMS of the fit for that particular training and testing fold. Taking the average of these five pairs of RMS values and generating 10 such cases allows for a way to draw insightful conclusions.

Methods

The skeleton script supplied by Dr. Ellis is accompanied with a function that generates a test data set. This is what is used in the plots and for the remainder of question one and two.

The data vectors are augmented with a vector of ones with appropriate length to create the design matrix X.

$$X = [\vec{x}\ \vec{1}]$$

Each data vector has an accompanying y_i which is compiled into the vector $\vec{y}$.

Furthermore the theta value for the constraint is given as the threshold value…

$$\theta = 8$$

The code first computes the Ordinary Least Squares solution using provided code in the skeleton. The equation for OLS is given below to ensure reproducibility.

$$\vec{w}^*{}_{OLS} = [X^T X]^{-1} X^T \vec{y} \qquad (1)$$

The line of best fit proposed by $\vec{w}^*{}_{OLS}$ is then plotted alongside the dataset.

Next, a solution is found with Constrained Least Squares, plotted on the same figure as created by the above code. Much of the beginning of the cls() function in the MATLAB script is provided set-up code. The important pieces to note include…

$$\vec{w}^*{}_{CLS}(\lambda^*) = [X^T X + \lambda^* I]^{-1} X^T \vec{y} \qquad (2)$$

$$g(\lambda^*) = ||\vec{w}^*(\lambda^*)||^2 - \theta \qquad (3)$$

Where $g(\lambda^*)$ is a temporary function for finding an optimal λ^* that minimizes the condition given in Eq 3 for use in Eq 2. The script also checks the condition number of the matrix X. This is to ensure the value given by the inverse of X or $[X^T X]^{-1}$ aligns with the theoretical result. The condition number in this case is a measure of how inaccurate the variable that is solved for will be.

Furthermore it is assessed that if the threshold is satisfiable to the OLS solution then CLS is foregone opting for $\vec{w}^*{}_{OLS}$ instead.

Finally with preparatory code out of the way, the script will finally perform Constrained Least Squares. Using the fzero function of MATLAB, λ^* is approximated through the condition given by Eq 3.

$$\lambda^* = fzero(g(\lambda^*), 1) \qquad (4)$$

1 is an initial estimate for the value of λ^*. When the function finds an appropriate λ^* value to return it is passed to Eq 2, which in turn returns the optimal $\vec{w}^*{}_{CLS}$ values.

For question two, the same implementation of OLS and CLS from question one is used. Thus the following portion of the methods section pertains to question two and the implementation of K-Fold Cross Validation for K = 5. This is found in the MATLAB script a3q2_20051918.m.

The main function of this script deals with setting the random number generator to a default setting. This is to ensure reproducibility of results. Furthermore it performs two loops of ten iterations. Inside these two loops is where the OLS and CLS methods are called, gathering ten sets of results for 5-Fold Cross Validation. Finally it displays the relevant output, which is the RMS error values and their means and standard deviations.

In the clskfold() function, which takes parameters design matrix X, response vector $\vec{y}$, θ a constraint value, and K the number of folds to use for K-Fold Cross Validation, there is further set up for this question. For computing the RMS of training and testing, two variables are initialized for measuring the variance of each train and test partition.

$$var_{train} = 0.0;$$

$$var_{test} = 0.0;$$

Next, consider partitioning both the design matrix and the response vector into 5 subsets.

$$(X, \vec{y}) = \{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \mathbb{S}_4, \mathbb{S}_5\}$$

On each iteration of 5-Fold Cross Validation, a different subset is selected as the test data while the model is trained with the remaining four.

Iteration…

1) Train the Least Squares model on $\{\mathbb{S}_2, \mathbb{S}_3, \mathbb{S}_4, \mathbb{S}_5\}$ and then test the model on $\mathbb{S}_1$.
2) Train the Least Squares model on $\{\mathbb{S}_1, \mathbb{S}_3, \mathbb{S}_4, \mathbb{S}_5\}$ and then test the model on $\mathbb{S}_2$.
3) Train the Least Squares model on $\{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_4, \mathbb{S}_5\}$ and then test the model on $\mathbb{S}_3$.
4) Train the Least Squares model on $\{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \mathbb{S}_5\}$ and then test the model on $\mathbb{S}_4$.
5) Train the Least Squares model on $\{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \mathbb{S}_4\}$ and then test the model on $\mathbb{S}_5$.

Where the training subset is composed of a design matrix and response vector denoted X_{train} and $\vec{y}_{train}$. Likewise the testing subset has a design matrix and response vector component denoted X_{test} and $\vec{y}_{test}$.

At the end of each of these iterations we add the corresponding variances to var_{trainS} and var_{test}.

$$var_{train} = var_{train} + \frac{1}{N}\sqrt{X_{train} \cdot \vec{w}_{cls} - \vec{y}_{train}} \quad (5)$$

$$var_{test} = var_{test} + \frac{1}{N}\sqrt{X_{test} \cdot \vec{w}_{cls} - \vec{y}_{test}} \quad (6)$$

In Eq 5 & 6. N refers to the problem size, in this case $N = 10$. $\vec{w}_{cls}$ is the optimal weight vector found by making a call to the least squares function coded for question one.

Once all iterations are complete the average of var_{train} and var_{test} are taken then square rooted giving the returnable RMS values.

$$rms_{train} = \sqrt{\frac{var_{train}}{K}} \quad (9)$$

$$rms_{test} = \sqrt{\frac{var_{test}}{K}} \quad (8)$$

Results

Table 1: Displays the solution obtained through Ordinary and Constrained Least Squares.

OLS $\overrightarrow{w}^*$	CLS $\overrightarrow{w}^*$
3.1546	2.7887
0.9780	0.4724

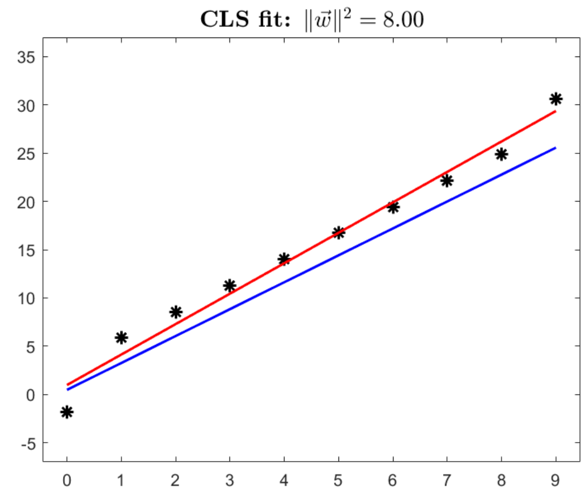

Figure 1: Plots the weight vector solutions obtained by Ordinary and Constrained Least Squares.

Table 2: Displays the mean and standard deviation of the training and testing RMS of ten 5-Fold Cross Validation calls using two variations of a Least Squares model.

	OLS		**CLS**	
	Train	**Test**	**Train**	**Test**
Mean	1.2678	1.9105	2.7409	2.7675
Std. Dev.	0.0210	0.1523	0.0014	0.0139

Conclusions

From *Figure 1* we can see that the blue line which displays the weight vector solution computed by Constrained Least Squares better fits the dataset X generated for this part of the assignment. The red line, the solution from Ordinary Least Squares both under and overestimates the data while the CLS solution solely underestimates aside from one outlier and is less affected by both outliers given at the upper and lower extremities of the plot, *Figure 1*.

The blue line serves as a better line of best fit for the data, particularly for analytics as we now have the notion that it underestimates the majority of the data, which will give more consistent results across test cases. This is in contrast to the previous case displayed by the red line where unseen data will have a higher response variability. This lower variability helps eliminate confusion when attempting to reach conclusions from the results of a model.

Referencing *Table 2*, we can see support for our conclusions about testing variability. For OLS the mean error and standard deviation of the test data is noticeably higher than that of the training data. This indicates that the model tests poorly on unseen data regardless of the training folds used. This implies a higher variability among responses for data the model has not seen prior.

Whereas in CLS there is a greater parity between the training and testing data statistics. Both the mean and standard deviation for the two are within a small range implying that the data we train on creates a better model for prediction on unseen data due to less variability.

The last important point to focus on which supports the theory that CLS offers a better overall model then OLS is that although the training data for CLS has a higher mean than the training data for OLS it has variability an order lower than that of the OLS results once again making it more appropriate for use on unseen data. The higher mean in CLS can be chalked up to the fact that CLS attempts to mitigate the effect of outliers. So when the outliers are estimated the model believes they should be a much different value than they are resulting in a larger RMS value.

Graphically this can be seen by an optimal weight vector with a "lower slope". For each test on unseen test data, the regression value will lie along this linear line in which the estimated response value varies by a smaller amount.

Part B) K-Fold Analysis of Financial Data

Abstract

Purpose: The objective of this question is to analyze a set of commodities data, pertaining to the historical price of publicly traded commodities. The specific analysis question to be answered is what method of regression provides best model when predicting the price of copper and should the data be standardized prior to regression. Once again, the RMS values will provide helpful results.

Methods: The three methods of regression implemented in this question are the two variations of Least Squares from question one and two as well as LASSO regression. The implementation of LASSO is done through the lasso() function of MATLAB while the more complicated details are drawn from lecture, class 27.

Results: By measuring the RMS values gathered by each method with and without standardization, the prediction methods can be compared to figure out the best model for the given data. To help make these conclusions the responses from our MATLAB script are tabulated in the corresponding section.

Conclusions: From our noted results. Without standardizing the data, it is clear that lasso performs the worst, then CLS and finally OLS performs the best but still not well. When a standardization routine is implemented there are common RMS values for both the training and testing set for all three methods. This implies that standardization plays a role in the predictability of a model.

Introduction

The motivation behind analyzing historical prices of sets of commodities is to develop better models for predicting future prices. This is important because these commodities are so common in everyday use knowing when or when not to buy them is valuable information to many companies. In this assignment the commodity to be predicted is copper, that is it acts as the response vector and the rest of our data is used as a design matrix.

Three different analysis methods will be used on the data. The results are recorded then the data will be standardized, and the process of each analysis method will be repeated. The three methods in use for this part of the assignment are Ordinary Least Squares, Constrained Least Squares and LASSO Regression. Two of these three have been covered extensively in the previous parts of the assignment and will be lightly touched upon. Lasso will still be covered in more detail in the following section for reproducibility purposes.

The results by the three methods mentioned above will be compared through the analysis of the RMS values which they generate. As well the intuition behind these results will be explored to form a deeper understanding of the methods in use.

Methods

The script for this part of the assignment is located in the MATLAB file, a3q3_20051918. The main function holds much of the set-up required for analysis; the skeleton was given by Dr. Ellis. A first important line to note is the setting of constraint values for Least Squares.

$$\lambda_0 = 0$$

$$\lambda = 0.5$$

λ_0 is the constraint value for OLS, whereas λ is the constraint value passed to the kfold() function for CLS. For the purpose of results to be constant across multiple runs of identical scripts the random number generator is set to its default setting.

The option of standardization is done using a conditional statement. In the case of this assignment for comparing the non-standardized and standardized results we set the flag to false and then re-compute while it is set to true. MATLAB makes standardization easy with the zscore() function. What this does is transforms the passed vector or matrix to follow a standard normal distribution with mean zero and variance one. The general algorithm for doing this computation is subtracting the vector's mean and dividing by its standard deviation.

$$\vec{a} = \frac{a_i - \mu}{\sigma}$$

From the main function, three sequential calls are made to the kfold() function with varying parameters. This is how the analysis method is selected.

$$[trainbig\ testbig] = kfold(X, \vec{y}, lsflag, \lambda_0, 5)$$

$$[trainsmL\ testsml] = kfold(X, \vec{y}, lsflag, \lambda, 5)$$

$$[trainlss\ testlss] = kfold(X, \vec{y}, lssflag, \lambda, 5)$$

The parameters are given to be, X is the commodities data minus the column of copper data, $\vec{y}$ is the response vector or column of copper data, lsflag is set to true telling MATLAB to use least squares regression while lssflag is set to false telling MATLAB to perform Lasso regression, $\lambda_0$and λ are set to the values above, and 5 is the number of folds to use in K-Fold Cross Validation.

As with question two, the beginning of the kfold() function solely pertains to set up such as the partitioning of the data. For the most part the kfold() function remains the same as that of the clskfold() function in question two. The important difference comes when the script branches based on the Boolean flag that is passed. When true it computes least squares as expected. When this flag is false though, it executes the following call using the lasso() function of MATLAB.

$$[b, fitinfo] = lasso(X_{train}, \vec{y}_{train}); \qquad (1)$$

$$lam = \min(fitinfo.MSE)\,; \qquad (2)$$

$$\vec{w}^* = b(:, fitinfo.MSE == lam) \qquad (3)$$

What these calls do are first perform lasso regression on the training data. This returns a large matrix b and a MATLAB "struct" object which is set to fitinfo. This contains many key values returned by the regression fit. The following line highlights what is important for this assignment, that is the value which provides the constraint value resulting in the lowest MSE. By using the index of that value in the b matrix the optimal weight vector is obtained. With $\vec{w}^*$ the same next steps are taken as if least squares regression had performed. This is to find the variance for the training and testing data then after the five iterations which implements 5-Fold Cross-Validation is complete the RMS of the sum of the training and testing variances are computed then returned.

In the background what LASSO regression is attempting to do is actually a variation of a least squares approach. It uses a different vector norm, the L_1norm, which differs from the L_2 norm generally used with least squares. The objective to be minimized by LASSO regression is…

$$\mathcal{L}(\vec{w}, \vec{y}) = f(\vec{w}) + \lambda \|\vec{w}\|$$

Where the latter term is the L_1 norm which is defined as…

$$L_1 \, \|\vec{w}\| \coloneqq \sum_{i=1}^{n} |w_i|$$

Results

Table 1: Displays the non-standardized and standardized RMS values for each analysis method performed on the commodities dataset.

	OLS		**CLS, $\lambda = 0.5$**		**LASSO**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Ordinary (RMS value)	263.393	573.236	573.236	671.950	876.922	900.820
Standardized (RMS value)	0.103	0.143	0.103	0.143	0.103	0.143

Conclusions

Just by looking at *Table 1* our attention should be directly drawn to the absurdly high RMS values generated when the data is non-standardized for not just one but all three analysis methods. Without standardizing the data we would want to select Ordinary Least Squares as our primary analysis method for this situation as it evidently has the lowest RMS value, although still high. The reason OLS performs the best in this situation is actually due to "outliers". In question one we saw how CLS mitigates the effects of outliers on the resulting optimal weight vector. Well if we consider the values that are given in the commodities dataset, they are very disproportionate with some commodities ranging in price from \$5.00 - \$15.00 and some commodities have ranges in the hundreds. By not accounting for these differences in scales between commodities both CLS and LASSO, both methods reliant on constraints, will have trouble predicting a commodity that's price lies in a totally different range.

This is where standardization has a substantial effect on the output of a model. By transforming the data to follow a standard normal distribution, that is the column vectors of our matrix now have mean zero and variance of one, likewise with the response vector, all commodities will fall in very similar ranges.

With no commodity column in our dataset now having prices much higher or lower than the others, all our analysis methods should work to the same effect. This can be noted in *Table 1* with the row of results for the standardized data. Both the training fold and testing fold bare the same results for each method. This is because with no obvious outliers OLS and CLS should perform identically. Furthermore LASSO regression is just another form of least squares regression as explored in lecture. Hence our results in the table for the data being standardized makes sense with our intuition of the topic.

So, when standardizing the data in this situation, the choice of analysis method is less important compared to the fact that yes, we should be standardizing our data when it falls on drastically different scales as the commodity prices do in our dataset.