MATH 304 - Numerical Analysis and Optimization
Project – Traffic flow prediction by using LSR and SVR

Haitong Lin
hl345@duke.edu

Abstract

In this project, I use Least Square Regression (LSR) and Support Vector Regression (SVR) to predict the traffic flow information in June 2016. This report contains the overview of this prediction project, related mathematical formulation and implementation. The experimental results and discussion of the results are also included in this report, which contains several comparisons between different models and also analysis of different performances within each model.

1. Overview

This project uses dataset collected by Highways-England[1], which shows the traffic flow information of June 2016. I use the Thursdays traffic flow information for this project. The training set is the first four Thursdays data (June 2, 9, 16 and 23), and the test data set is the data from June 30. I used Matlab for this project.

In the first part of the project, I implement Least Square Regression (LSR) models for prediction. I tried 9 different models for LSR (n=1,2,3……,9), fitted the models with training data and tested the derived models with test data. I compare errors for different models and obtained the best model that has the smallest error on both train and test data.

In the second part of the project, I implement Support Vector Regression (SVR) models for prediction. I implement three different kernels to train the models: gaussian, RBF and polynomial. For each kernel, I test three different settings (one default setting and two other personal selected settings) and also the optimized setting.

Finally, I selected the best models from LSR, SVR (one best model for each kernel) collectively and make visualizations to display the differences.

2. Mathematical formulation and implementation

Least Square Regression (LSR) aims at minimizing the sum of the squares of the residuals between the measured y and the y calculated by the model, which fits a unique line for a given set of data[2]. LSR can be extended to polynomial regression, which is what I used in this project. The logic of LSR is to minimize the prediction error.

Given $A \in R^{mxn}$ and $B \in R^{m}$, a general solution for LSR problems are obtained by minimizing $||AX-B||^2$, which is

$$X = (A^TA)^{-1} A^TB$$

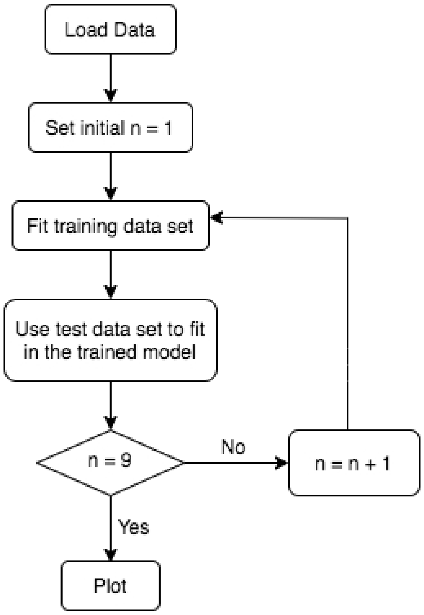

Figure 1 Flowchart for LSR

The flowchart showcases my implementation process for the LSR part in this project. I firstly loaded the data, specified training data and test data. Then I used the training data set to fit the LSR models with different polynomials, from n=1 to n=9, respectively. Then I fit the test data into the trained models, compared the predicted values with the actual data. The results and discussion are stated in later sections of this report.

Support Vector Machine (SVM) is a well-known classification algorithm, which separates two sets of points by a maximum of margin. SVM can also be applied to regression analysis problems, which is referred to as Support Vector Regression (SVR)[3]. For the second part of this project, I applied SVR models for traffic flow prediction.

In SVR, the object is to minimize the error while maximizing the margin, keeping in mind that some errors are tolerated. Figure 2 below illustrates the SVR.

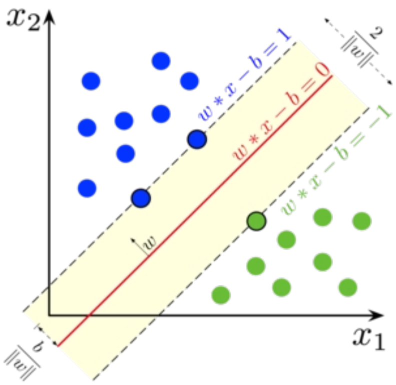

Figure 2 SVR[4]

Similar to SVM, in order for SVR to solve nonlinear regression problems, we need to implement different kernels to SVR such that the original space can be mapped to a new space. For this project, I selected three different kernels: Gaussian, Radical Basis Function (RBF) and Polynomial.

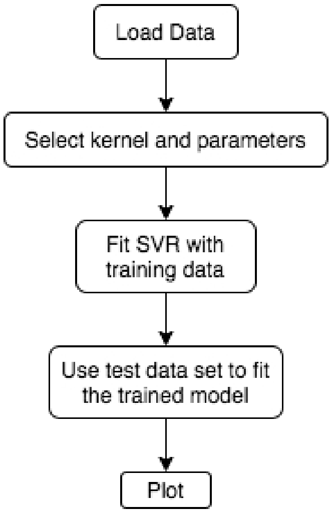
Load Data
Select kernel and parameters
Fit SVR with training data
Use test data set to fit the trained model
Plot

Figure 3 Flowchart for SVM

As is shown in the flowchart above, the implementation for the second part of the project is very straightforward. I select kernel and parameters, fit training data and use test data to make the predictions. This flowchart does not contain a loop because I selected the kernel and parameters randomly by myself. Results, error analysis and discussion can be found in following sections.

3. Experimental Results

As is shown in the figure 4 above, obvious pattern can be found on the scatter plot which has all the training and testing data set. The test data (orange dots) also follows the pattern nicely. We can observe two peaks of traffic flows, one at between 5-10, and one at around 15-20.

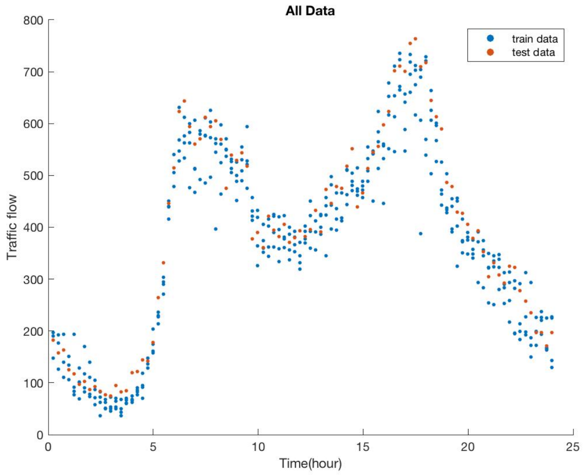

Figure 4 All data

Part 1: LSR

Model		n=1	n=2	n=3	n=4	n=5
Training error	MSE	29834	13195	12732	12687	11969
	r^2	0.1084	0.6057	0.6195	0.6208	0.6423
Test error	MSE	32158	15085	14313	14152	13453
	r^2	0.1305	0.6028	0.6237	0.6283	0.647

Model		n=6	n=7	n=8	n=9
Training error	MSE	6315	6269.7	3080.2	2953.2
	r^2	0.8113	0.8126	0.9079	0.9117
Test error	MSE	6852.6	6816.2	3346.1	33330
	r^2	0.8285	0.8296	0.9258	0.9261

Table 1 LSR models

Table 1 above showcases the errors for different LSR models. We can observe that the MSE is decreasing as the number of n grows. Among all the models we obtained, n=9 is the best model, and its illustration is as above.

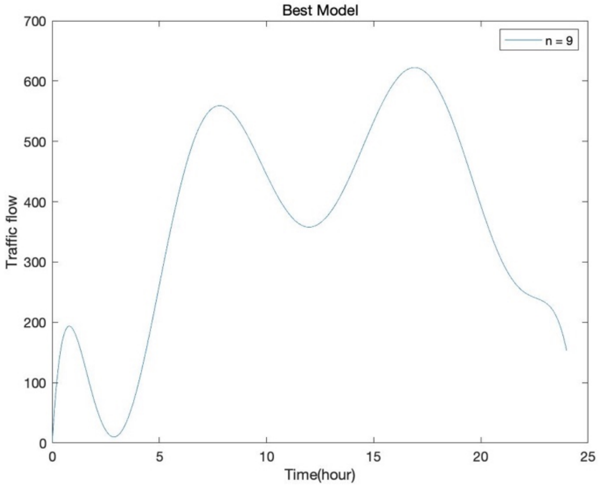

Figure 5 Best model for LSR

Part 2: SVR

a. SVM + Kernel 1 (Gaussian) Regression

Model	SVM+Kernel 1(Gaussian)			
Setting	case 1	case 2 (50)	case 3 (100)	optimized
MSE	1426.8	1644.5	1466.4	2012.8
r^2	0.9771	0.9758	0.9772	0.9670

Table 2 SVM + Gaussian models

Table 2 showcases the results for SVM with gaussian kernel regression. I choose three different settings: default (case 1), box constraint = 50 (case 2) and box constraint = 100 (case 3), and I also included the optimized case (observed box constraint = 595.01 and estimated box constraint = 723.66). The model with the smallest error in this model is case 1 (with default settings).

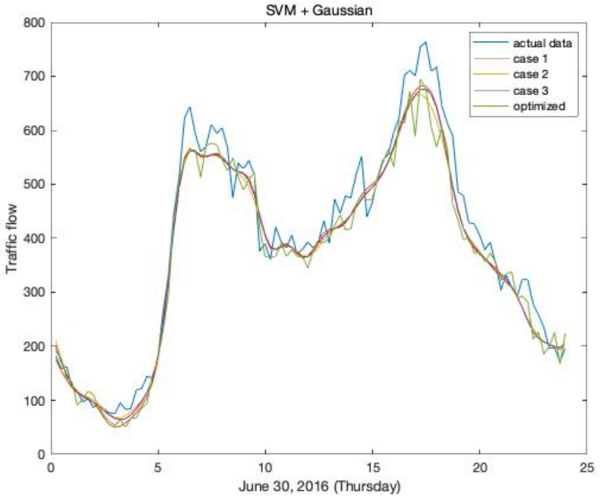

Figure 6 SVM + Gaussian models

Figure 6 contains the visualizations of all the cases for this model, along with the actual data. We can see that the shape of the optimized case is actually more similar to the actual data, despite that case 1 has smaller error.

b. SVM + Kernel 2 (RBF) Regression

Model	SVM+Kernel 2 (RBF)			
Setting	case 1	case 2 (100)	case 3 (300)	optimized
MSE	1426.8	1466.4	1459.1	1565.5
r^2	0.9771	0.9772	0.9766	0.9740

Table 3 SVM + RBF models

Table 3 is the results for SVM with RBF kernel regression. The three different settings for this model are: default (case 1), box constrain = 100 (case 2) and box constraint = 300 (case 3). I also include the optimized case (both observed and estimated box constraint = 433.45). The model with the smallest error in this model is case 1 (with default settings).

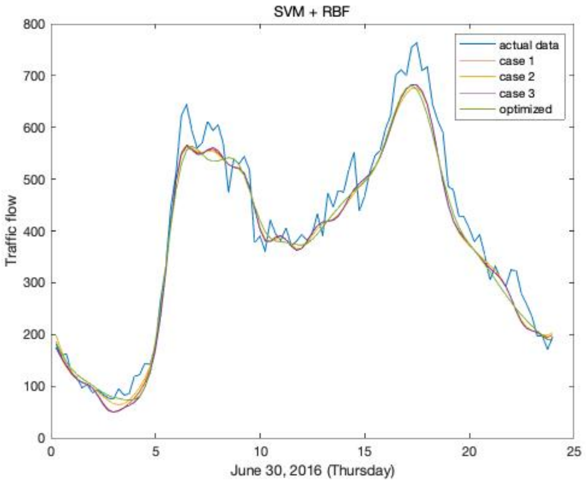

Figure 7 SVM + RBF models

Figure 7 is the visualizations for all cases in the SVM+RBF model. The difference between cases are not significant, which is also reflected in table 3 as the difference between error for each case is relatively small.

c. SVM + Kernel 3 (Polynomial) Regression

Model	SVM+Kernel 3 (Polynomial)			
Setting	case 1	case 2 (5)	case 3 (10)	optimized
MSE	16898	15912	6630	16785
r^2	0.6155	0.6422	0.8565	0.6198

Table 4 SVM + Polynomial models

Table 4 showcases the results for SVR with polynomial kernel regression. The different settings for this model are: default (case 1), polynomial order = 5 (case 2), polynomial order = 10 (case 3) and the optimized case. The error in case 3 is significantly smaller than other cases, which is also shown in the figure 8 below, which shows the visualization for this model.

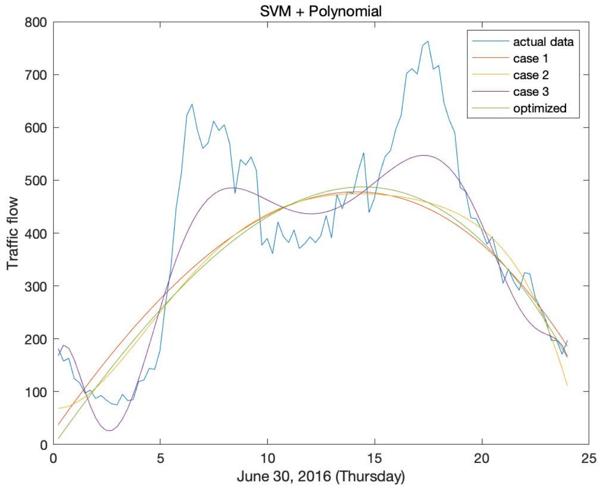

Figure 8 SVM + Polynomial models

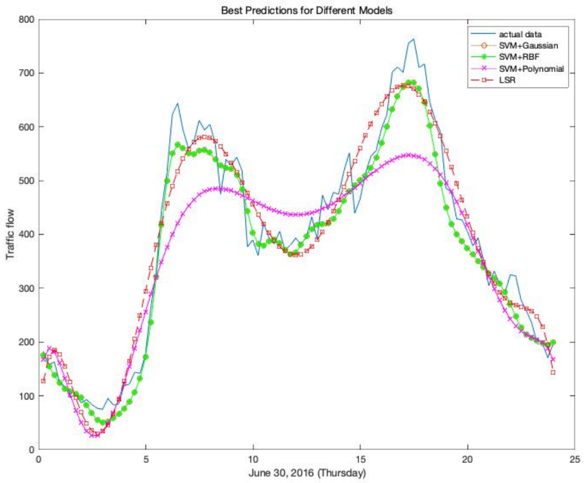

Figure 9 Best predictions for different models

Figure 9 puts together best predictions for each model: LSR, SVR (selected one for each kernel) as well as the actual traffic flow. The best prediction I selected are the ones with the smallest error (MSE) in their category.

For LSR, I choose n = 9 as best prediction (MSE = 2953.2). For SVR, I choose case 1 (default settings) for both gaussian kernel and RBF kernel. The errors for both cases are the same (MSE = 1426.8). The case selected for SVR with polynomial kernel is case 3 (MSE = 6630), which has a polynomial order of 10.

The visualization fits well with the error we obtained: SVR with Gaussian and RBF go with the pattern of our actual data, LSR has a similar curve but looks slightly less accurate than the SVR + Gaussian/RBF model. The SVR + Polynomial model has the worst prediction among all. The curve also fits badly with our actual traffic flow.

To sum up, with the error we obtained, SVR + Gaussian and SVR + RBF models (with default settings) are the best predictions for traffic flow in this project. LSR is slightly less accurate but also has a decent performance. The SVR + Polynomial model has the worst performance.

4. Discussion

a. Factors that may impact prediction accuracy

For this project, the most obvious factor that might affect the accuracy is the size of data. I think the size of training data set can be enlarged to improve the accuracy of the prediction results.

b. Limits or problems of my approach

I think one limit for this project is that we are only selecting a few cases. For example, we only observe 9 cases in LSR, while data shows that the accuracy actually increases as the number of n increases. Similarly, if we select more settings for the models in SVR, it is very likely that we will obtain more accurate results.

In addition, the data I obtained shows that in all three SVR models, none of the best predictions is the optimized case. I am slightly confused, but I also notice that every time I run my code, the error of the optimal outputs different results. Maybe it is possible to dive into the line of code and fix some settings, which could potentially solve the problem.

c. Possible improvements that can be done

As is mentioned in the points above, I think this project can be improved by increasing dataset size, increasing the number of different models and modified optimization code.

d. Anything unique you have done to improve/validate your program's accuracy/efficiency

As is seen in the parts above, after running the code and collecting the required data, I made visualizations for each model. I think visualizing these models is a great way to validate the program's accuracy. For example, in the SVR with polynomial kernel model, the huge differences between error for different cases look strange to me at first, but as I finish the visualization in figure 8, the large difference in accuracy for each model is clearly shown, which validates the accuracy of my program.

References

[1] Highways-England. http://webtris.highwaysengland.co.uk/.

[2] S. C. Chapra, R. P. Canale, Numerical methods for Engineers (seventh edition), McGraw-Hill Education, 2015, Chapter 17

[3] S. Boyd and L. Vandenberghe, Convex Optimization, (Seventh Edition), Cambridge University Press, 2009. Chapter 8.

[4] Wikipedia. https://en.wikipedia.org/wiki/Support_vector_machine.