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

**Xuchen Gong**
**xg54@duke.edu**

## Abstract

Least-square Regression (LSR) and Support Vector Regression (SVR) are applied in this project to predict the traffic flow at different times of a day, and the explored models include LSR (Polynomial) and SVR (Gaussian, Linear and Polynomial). Through controlling variables, some important hyperparameters of each algorithm, such as *PolynomialOrder*, *KernelScale*, and *BoxConstraint*, are optimized and visualized, as well as their effects on the performance of the algorithms.

## 1. Overview

In this project, Least-square Regression (LSR) and Support Vector Regression (SVR) are applied to predict the traffic flow at different times of a day. When utilizing LSR, several polynomial models with different highest order (1-20) are explored, and the best polynomial model is picked to be n=18. When applying SVR (*Gaussian*, *Linear* and *Polynomial*), we first use the referenced functions to better locate the optimal hyperparameters, and then we control variables to further optimize the hyperparameters for each algorithm. In this process, the function and meaning of the hyperparameters are analyzed and visualized. Finally, each model is tested with the evaluation metrics MSE and $r^2$, and the best models are utilized for prediction.

## 2. Mathematical Formulation and implementation

### 2.1 Least Square Regression

The target is to fit the given data of traffic flow by building a polynomial model

$$f(t) = a_n t^n + a_{n-1} t^{n-1} + \dots + a_2 t^2 + a_1 t^1 + a_0$$
$$where\ n = 1, 2, \dots, 9.$$

where $t$ is the different times in a certain day, and $f(t)$ is the prediction of the traffic flow.

To solve this problem, I first implement two functions, *GetData* and *PolynomialTransform*, to do the data pre-processing. Since the input is from a noisy csv file and the class of variable “LocalDate” is datetime when originally read by the matlab command *readtable*, the *GetData* function will select the X and Y of interest and output them as double by applying function *hours* to X.

The function *PolynomialTransform* is to generate the polynomial feature matrix $A$ given a column vector $a$ and an integer that specifies the highest degree of the polynomial, and $a_i$, the ith column of $A$ can be calculated as

$$a_i = x^{i-1}, where\ i = 1,2,\dots,n$$

After the feature matrix $A$ is computed, it is found that the scale of the data on different dimensions of the columns of $A$ is very different. Therefore, standardization is needed and $A$ is then applied to the function *my_standardize* where $a_i'$, the ith column of the output $A'$ is computed as

$$a_i' = \frac{a_i - X}{S},$$

where $X$ is the mean value and $S$ is the variance.

Next, I use the built-in matlab function *lsqr* to solve the system of linear equations $Ax = b$ for $x$. Mathematically, *lsqr* finds a least squares solution for $x$ that

$$minimize\ \|Ax - b\|_2^2 = \sum_{i=1}^{k} (a_i{}^T x - b_i)^2$$

where $A \in R^{k \times n}$ (with k $\geqslant$ n), $a_i{}^T$ are the rows of $A$, and the vector $x \in R^n$ is the optimization variable [2].

The solution of such a problem can be reduced to solving a set of linear equations,

$$(A^T A) = A^T b$$

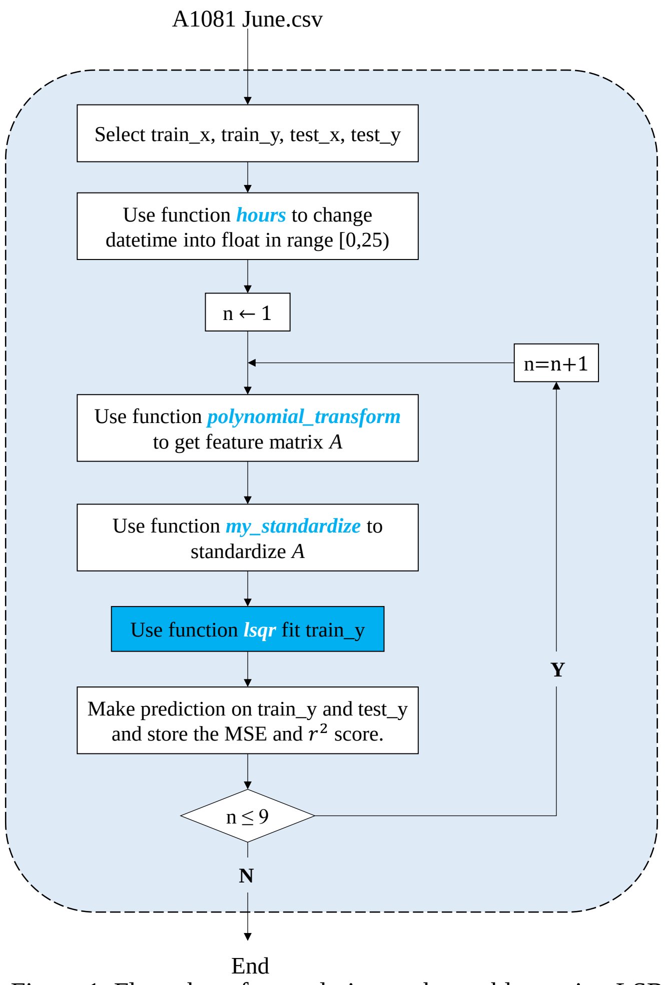
which gives us the solution $x = (A^T A)^{-1} A^T b$.

After getting the optimized variable, the prediction $\hat{y}$ can be computed by $Ax$ and the evaluation metrics, such as the mean square error (MSE) and coefficient of determination ($r^2$) can be computed as

$$mse = \sum_{i=1}^{N} \frac{1}{N} (\hat{y}_i - y_i)^2$$

$$r^2 = \frac{[N \sum_{i=1}^{N} (\hat{y}_i y_i) - (\sum_{i=1}^{N} \hat{y}_i)(\sum_{i=1}^{N} y_i)]^2}{[N \sum_{i=1}^{N} \hat{y}_i{}^2 - (\sum_{i=1}^{N} \hat{y}_i)^2][N \sum_{i=1}^{N} y_i{}^2 - (\sum_{i=1}^{N} y_i)^2]}$$

After changing the value of $n$, the evaluation metrics MSE and $r^2$ are utilized to determine the best polynomial model.


A1081 June.csv
Select train_x, train_y, test_x, test_y
Use function hours to change datetime into float in range [0,25)
n ← 1
n=n+1
Use function polynomial_transform to get feature matrix A
Use function my_standardize to standardize A
Use function lsqr fit train_y
Make prediction on train_y and test_y and store the MSE and r² score.
Y
n ≤ 9
N
End
Figure 1. Flow chat of my solution to the problem using LSR.

### 2.2 Support Vector Regression

The target is also to fit the given data of traffic flow, but this time we try to fit the error within a certain threshold by using the algorithm of SVR.

We again first make use of the functions *readtable*, *GetData*, *hour* to read and pre-process the data. We experiment with some important hyperparameters of three kernels, *Linear*, *Polynomial*, and *Gaussian* kernels, which are formulated by

*Linear*

$$k(x,z) = x^T z$$

*Polynomial*

$$k(x,z) = (x^T z)^d$$

*Gaussian*

$$k(x,z) = exp(-\frac{\|x-z\|_2^2}{2\sigma^2})$$

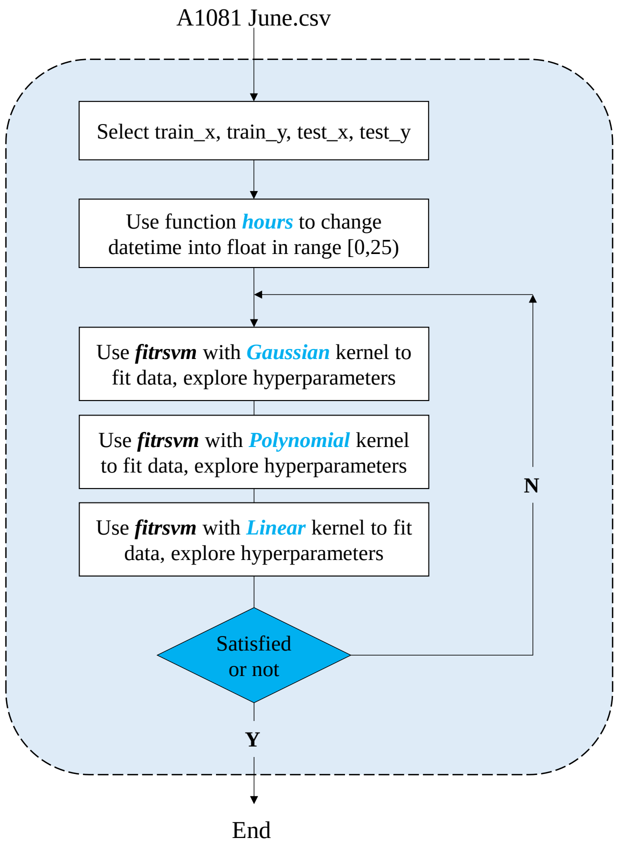
I use the built-in matlab function *fitrsvm* to fit the training data. The goal is to find a function $f(x)$ that deviates from $y_n$ by a value no greater than ε for each training point $x$, and at the same time is as flat as possible [1]. The object function of SVR is

$$minimize \quad \frac{1}{2}\|W\|_2^2 + C\sum_{i=1}^{N}(\xi_i - \xi_i{}^*)$$

$$subject \quad y_i - (Wx_i + b) \le \epsilon + \xi_i{}^*, \quad \xi_i{}^* \ge 0$$

$$Wx_i + b - y_i \le \epsilon + \xi_i{}^*, \quad \xi_i{}^* \ge 0$$

where $\epsilon$ is the desired error tolerance, $\xi_n$ and $\xi_n{}^*$ are the "slack" variable for each point that allow regression errors to exist up to the value of $\xi_n$ and $\xi_n{}^*$, and $C$ controls the penalty imposed on observations that lie outside the epsilon margin $\epsilon$.


A1081 June.csv
Select train_x, train_y, test_x, test_y
Use function hours to change datetime into float in range [0,25)
Use fitrsvm with Gaussian kernel to fit data, explore hyperparameters
Use fitrsvm with Polynomial kernel to fit data, explore hyperparameters
Use fitrsvm with Linear kernel to fit data, explore hyperparameters
N
Satisfied or not
Y
End
Figure 2. Flow chat of my solution to the problem using SVR.

## 3. Experimental Results

### 3.1 Least Square Regression

| Model | Training error | | Test error | |
|---|---|---|---|---|
| | MSE | $r^2$ | MSE | $r^2$ |
| n = 1 | 29834 | 0.10841 | 32158 | 0.11421 |
| n = 2 | 13194 | 0.60569 | 15084 | 0.58450 |
| n = 3 | 12732 | 0.61952 | 14313 | 0.60576 |
| n = 4 | 12687 | 0.62084 | 14152 | 0.61019 |
| n = 5 | 11969 | 0.64230 | 13453 | 0.62945 |
| n = 6 | 6315.0 | 0.81128 | 6852.6 | 0.81125 |
| n = 7 | 6269.7 | 0.81263 | 6816.2 | 0.81235 |
| n = 8 | 3080.2 | 0.90794 | 3346.1 | 0.90783 |
| n = 9 | 2953.2 | 0.91174 | 3329.7 | 0.90829 |

| | | | | |
|---|---|---|---|---|
| n = 10 | 2891.8 | 0.91358 | 3295.7 | 0.90922 |
| n = 11 | 2869.1 | 0.91426 | 3284.6 | 0.90953 |
| n = 12 | 2870.4 | 0.91422 | 3284.6 | 0.90953 |
| n = 13 | 2870.4 | 0.91422 | 3289.6 | 0.90939 |
| n = 14 | 2878.8 | 0.91397 | 3309.6 | 0.90884 |
| n = 15 | 2844.0 | 0.91500 | 3188.3 | 0.91218 |
| n = 16 | 2826.1 | 0.91554 | 3135.0 | 0.91365 |
| n = 17 | 2814.0 | 0.91591 | 3094.0 | 0.91478 |
| n = 18 | 2813.1 | 0.91593 | 3075.3 | 0.91529 |
| n = 19 | 2825.6 | 0.91556 | 3082.2 | 0.91510 |
| n = 20 | 2851.3 | 0.91479 | 3113.6 | 0.91424 |

Table 1. MSE and $r^2$ of the predictions made by LSR (Polynomial).

As seen in the above table, the polynomial model n=18 has both the smallest training error and test error, no matter the metric is MSE or $r^2$. Its predicted flow as well as the actual flow is plotted below.

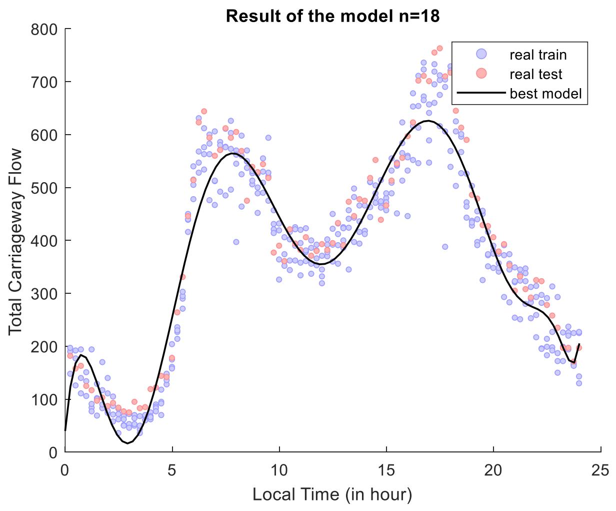


Figure 3. The actual flow and the flow predicted by the polynomial model n=18.

## 3.2 Support Vector Regression

| Model | SVR (Linear kernel) | | | |
|---|---|---|---|---|
| Setting | Default | Case1 | Case2 | Optimized |
| Train MSE | 30325.68 | 30021.82 | 30019.77 | 30019.77 |
| Train r$^2$ | 0.094 | 0.1028 | 0.1029 | 0.1029 |
| Test MSE | 32012.13 | 31699.00 | 31696.34 | 31696.34 |
| Test r$^2$ | 0.118 | 0.1268 | 0.1269 | 0.1269 |

| | |
|---|---|
| Opt. hp | BoxConstraint=58, KernelScale=13.2, Epsilon=0.236 |
| Case 1 | BoxConstraint=20 , KernelScale=13.2, Epsilon=0.236 |
| Case 2 | BoxConstraint=19, KernelScale=12, Epsilon=0.01 |
| Optimized | BoxConstraint=19, KernelScale=12, Epsilon=0.01 |

Table 2. MSE and $r^2$ of the predictions made by SVR(Linear).

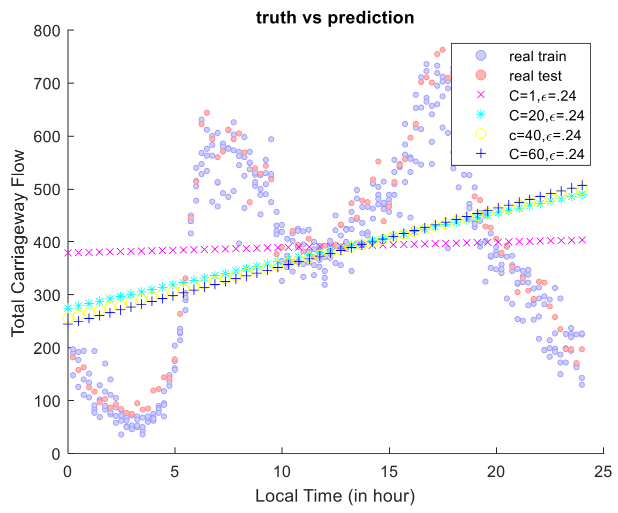


Figure 4. Explore effects of BoxConstraint on SVR(Linear).

| | |
|---|---|
| Difference in $r^2$ train and $r^2$ test for model "C=1" | 0.0146 |
| Difference in $r^2$ train and $r^2$ test for model "C=20" | 0.0241 |

Table 3. Difference in $r^2$of train and test of different models.

| Model | SVR (Polynomial kernel) | | | |
|---|---|---|---|---|
| Setting | Default | Case1 | Case2 | Optimized |
| Train MSE | 13963.17 | 3399.41 | 2701.63 | 2701.63 |
| Train r$^2$ | 0.582713 | 0.8984 | 0.9193 | 0.9193 |
| Test MSE | 16819.28 | 4029.85 | 2974.85 | 2974.85 |
| Test r$^2$ | 0.5367 | 0.8890 | 0.9181 | 0.9181 |

| | |
|---|---|
| Opt. hp | BoxConstraint=521.78, KernelScale=2.47, Epsilon=0.22 |
| Case 1 | PolynomialOrder=17, Opt. hp |
| Case 2 | PolynomialOrder=25, Opt. hp |
| Optimized | PolynomialOrder=25, Opt. hp |

Table 4. MSE and $r^2$ of the predictions made by SVR (Polynomial).

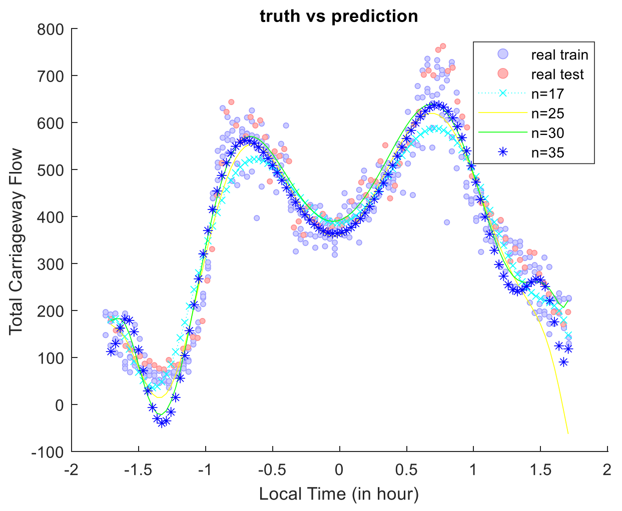


Figure 5. Explore effects of PolynomialOrder on SVR (Polynomial).

| Model | SVR (Gaussian kernel) | | | |
|---|---|---|---|---|
| Setting | Default | Case1 | Case2 | Optimized |
| Train MSE | 1587.85 | 1594.13 | 1752.91 | 1752.91 |
| Train r$^2$ | 0.9525 | 0.9506 | 0.9476 | 0.9476 |
| Test MSE | 1503.10 | 1591.13 | 1470.62 | 1470.62 |
| Test r$^2$ | 0.9586 | 0.9562 | 0.9595 | 0.9595 |

| | |
|---|---|
| Opt. hp | BoxConstraint=137, KernelScale=1, Epsilon=0.45 |
| Case 1 | BoxConstraint=150, KernelScale=1, Epsilon=0.45 |
| Case 2 | BoxConstraint=185, KernelScale=1.3, Epsilon=0.45 |
| Optimized | BoxConstraint=185, KernelScale=1.3, Epsilon=0.45 |

Table 5. MSE and $r^2$ of the predictions made by SVR *Gaussian).

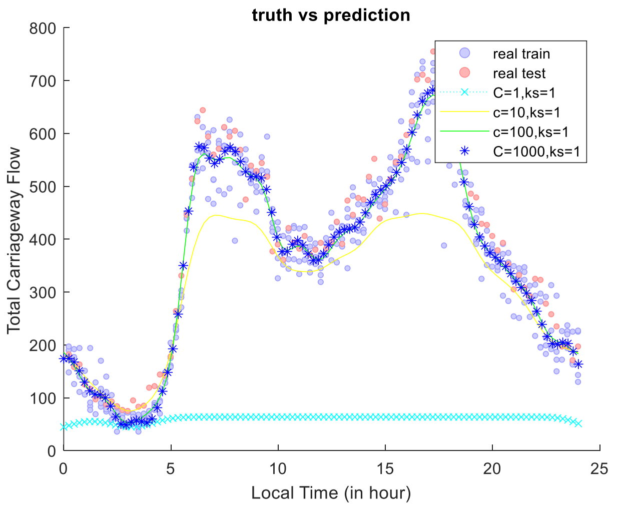


Figure 6. Explore effects of BoxConstraints on SVR (Gaussian).

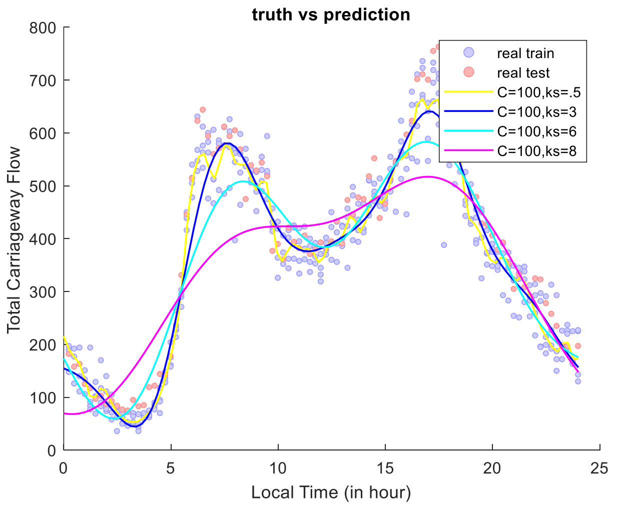


Figure 7. Explore effects of KernelScale on SVR (Gaussian).

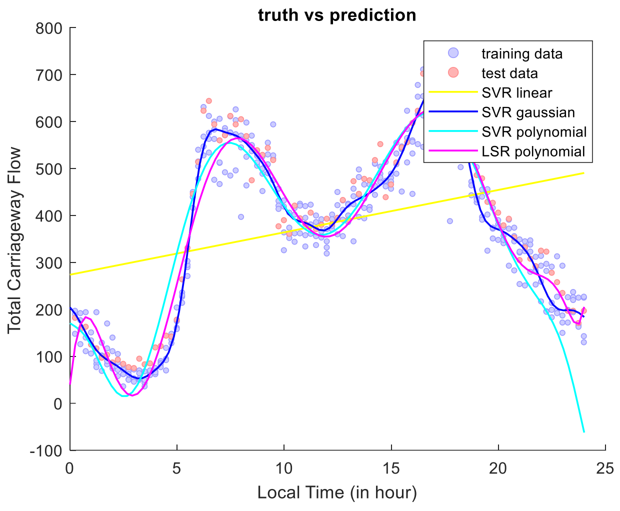


Figure 8. Predictions made by the optimized models.

## 4. Discussion

### 4.1 Factors that may impact prediction accuracy

**For Least Square Regression,**

The only hyperparameter we plan to optimize is the polynomial order $n$.

As seen in Table 1, when polynomial order $n$ is relatively small, the larger the $n$ is, the better the performance the model tends to have on the test data. After $n$ increases to a certain value, for example, 18 in our case, the performance on test data begins to decline because of overfitting.

**For Support Vector Regression,**

The object function of SVR is

$$minimize \quad \frac{1}{2}\|W\|_2^2 + C\sum_{i=1}^{N}(\xi_i - \xi_i{}^*)$$

$$subject \quad y_i - (Wx_i + b) \le \epsilon + \xi_i{}^*, \quad \xi_i{}^* \ge 0$$

$$Wx_i + b - y_i \le \epsilon + \xi_i{}^*, \quad \xi_i{}^* \ge 0$$

*BoxConstraint*

Since $C$ determines the trade-off between the flatness of $f(x)$ and the amount up to which deviations larger than $\epsilon$ are tolerated, a small $C$ helps prevent overfitting (regularization). In one word,

Large C: lower bias, higher variance
Small C: higher bias, lower variance

As seen in Fig. 4, the curves of $f(x)$, or the predictions made by the SVR (Linear) models, are flatter when BoxConstraint is small. Moreover, Table 4 shows that the difference between $r^2$ of train and test, or the extent to which the model is overfit, tends to be smaller when C is smaller.

More evidence can be found in Fig. 6, where the predictions does not accurately fit the data when $C$ is small and the predictions fit the data well when $C$ is large.

*PolynomialOrder*

Similar to the cases using LSR (Polynomial), the test performance will improve as $n$ becomes larger if $n$ stays relatively small. After $n$ becomes larger than 25 in our case, the test performance displays a tendency of decline because of overfitting.

*KernelScale*

The software divides all elements of the predictor matrix $X$ by the value of KernelScale [2].

Since KernelScale determines the extent to which the features vary smoothly,

Large KernelScale: higher bias, lower variance
Small KernelScale: lower bias, higher variance

As seen in Fig. 7, the curve of the predictions made by the SVR (Gaussian) model is flatter when KernelScale is large and fits the data better when KernelScale is small. Moreover, as the KernelScale gets smaller, the model displays more overfit.

Finally, the finalized models for each algorithm are plotted in Fig. 8, and the best performance is given by the algorithm SVR (Gaussian).

### 4.1 Limits of my approach

*Firstly*, comparing the results in Table 1 and Table 4, we find that under the current setting of the hyperparameters, LSR (Polynomial) has relatively better performance than SVR (Polynomial) when their $n$ are the same. Therefore, it is indicated that the other hyperparameters of SVR (Polynomial) also plays an important role and are under-explored in our project currently.

*Secondly*, we search for optimal hyperparameters by first running *fitrsvm* and use the optimal hyperparameters suggested as my starting point. When experimenting with the hyperparameters of SVR (Polynomial) and SVR (Gaussian), we do not consider epsilon as one hyperparameter to optimize and directly use the suggested value in in our trials instead.

*Thirdly*, when learning the effects of each hyperparameter on the predictions, our method is to fix one hyperparameter and treat the other as a variable at a time. This method, however, only serves the scenarios where the number of hyperparameters to optimize are small, which is barely the case.

### 4.2 Possible improvements

When given more time, our plan is to first decide on the bond of each hyperparameter and then do gridsearch. If there are too many hyperparameters to optimize, random search is also feasible as long as the experiments are done for a large number of times.

## References

[1] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
[2] Matlab Documentation. Oct. 23, 2020. Available at https://www.mathworks.com/help/matlab/.