HYBRID SUPPORT VECTOR REGRESSION FOR SHORT TERM ELECTRICITY LOAD FORECASTING

BY

SU WUTYI HNIN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (ENGINEERING AND TECHNOLOGY)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2017

HYBRID SUPPORT VECTOR REGRESSION FOR SHORT TERM ELECTRICITY LOAD FORECASTING

BY

SU WUTYI HNIN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (ENGINEERING AND TECHNOLOGY)

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2017

HYBRID SUPPORT VECTOR REGRESSION FOR SHORT TERM ELECTRICITY LOAD FORECASTING

A Thesis Presented

By

SU WUTYI HNIN

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE (ENGINEERING AND TECHNOLOGY)

Approved as to style and content by

Advisor and Chairperson of Thesis Committee [illegible]

(Assoc. Prof. Dr. Chawalit Jeenanunta)

Committee Member and

Chairperson of Examination Committee [illegible]

(Asst. Prof. Dr. Somsak Kittipiyakul)

Committee Member Pornthipa Ongkunaruk

(Assoc. Prof. Dr. Pornthipa Ongkunaruk)

NOVEMBER 2017

Abstract

HYBRID SUPPORT VECTOR REGRESSION FOR SHORT TERM ELECTRICITY LOAD FORECASTING

by

Su Wutyi Hnin

Bachelor of Electrical Engineering, Technological University, Mandalay, 2013
Master of Science (Engineering and Technology), Sirindhorn International Institute of Technology, Thammasat University, 2017

The target of this research is to enhance the daily load demand and reduce the error of forecasting in Thailand. Electricity load forecasting presents a core part in power industry. However, it can be seen that electricity load demand demonstrates non-linear nature and distinct seasonal pattern. In the case of complex non-linear problem, Support Vector Regression (SVR) is a well-known technique applying forecasting area due to literature review. Therefore, SVR and three optimization techniques: Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Bayesian Optimization (BO) are proposed to forecast electricity load demand. The data set has 48 periods in one day which has been gathered every 30 minutes for 24 hours from the Electricity Generating Authority of Thailand (EGAT). The electricity load demand, which starts from 2012 (January) to 2017 (July) are used in this research. As it has the seasonal load pattern, window based data cleaning technique is applied to filter the data. The attributes of the model are yesterday load, previous week the same day load, forecasted day temperature, yesterday temperature, day of week and month of year. The performance of forecasting model is utilized by mean absolute percentage error (MAPE). The performance of

Bayesian Optimization is better than other three techniques according to monthly average MAPE.

Keywords: Short-Term Electricity Load Forecasting, Support Vector Regression, Genetic Algorithm, Particle Swarm Optimization, Bayesian Optimization

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Chawalit Jeenanunta for his continuous support, patience, motivation, and immerse knowledge during my study. His valuable guidance and effort helped me a lot with my research and my thesis writing.

Besides, I would like to thank the members of Examination Committees: Asst. Prof. Dr. Somsak Kittipiyakul and Assoc. Prof. Dr. Pornthipa Ongkunaruk, for their helpful comments and suggestions, and more importantly, for their valuable questions which played a vital role in helping me broaden my research from different perspective.

My special thank also goes to Sirindhorn International Institute of Technology (SIIT), Thammasat University for providing me a full scholarship during my master's study. Also, I would like to thank Electricity Generating Authority Thailand (EGAT) for providing me the useful data for my research. Without their precious support, I could not imagine how I could have conducted this research.

Last but certainly not least, I would like to thank my family, friends, especially my seniors: K. Darshana Abeyrathna and M H M R Syamali Dilhani for encouraging and supporting me spiritually through up my good and bad times during my study in Thailand.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 General Statement

Electricity load demand forecasting plays one of the critical role in power industries and a great issue for management of power system mainly in generation and distribution area. The forecasted load manages the operating system which need to switch and schedule demand. Nowadays the consuming of electricity increase day by day due to developing of industries and increasing of population. The utilities need to upgrade the power system to cover the growth of electricity load demand. To determine an accurate technique is important in forecasting sector. If over forecasting, it will cost more money otherwise it will lead lack of electricity. There are three types of electricity load forecasting which are shown in figure.

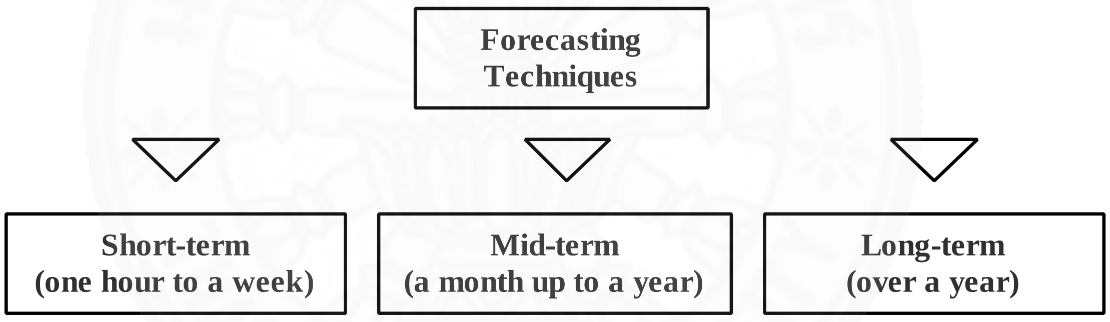

Forecasting Techniques
Short-term (one hour to a week)
Mid-term (a month up to a year)
Long-term (over a year)

Figure 1.1 Types of Forecasting

Short-term electrical demand forecasting (STLF) is to schedule the generation and transmission of electricity (Gross & Galiana, 1987). Medium-term electrical demand forecasting (MTLF) is to arrange the fuel purchase and Long-term electrical demand forecasts (LTLF) is to develop the power supply and delivery system. Temperature is an essential factor in load forecasting for Short-term and Medium-term forecasting. For Long-term forecasting, it relies on the population, number of consumers connected, and gross domestic product.

1.2 Objective of the Study

The main objective of this study is to reduce the forecasting error and improve the model to be a good one. There are many techniques to forecast the electricity load demand. We choose support vector regression among many techniques since it is a good learning algorithm in forecasting areas. Moreover, we apply optimization techniques to optimize the hyper parameters. Therefore, the model gives good performance in hybrid methods.

1.3 Study area scopes

The study area scopes has been defined as follows.

- This study focus on short-term electricity load forecasting in Thailand
- The data are obtained from Electricity Generating Authority in Thailand (EGAT)
- Bangkok and Metropolitan regions' data are used in this study.
- The data has been collected every half an hour so there are 48 periods in 24 hours.
- The experiments are tested from January 2012 to July 2017.
- The historical 30 minutes temperature data are also included in this study. Then, day and month index are also applied.
- The LIBSVM package is used for simulation.

1.4 Organization of the thesis

There are 6 chapters in this research. The chapters are organized as follow:

Chapter (1) "Introduction" divides into 5 sections: the general statement, objective of the study, scopes of the study, and organization of the thesis.

Chapter (2) is literature review which demonstrates the associated documents and related research.

Chapter (3) is methodology that introduces the forecasting techniques, optimization algorithm, data filtering method, and error calculations.

Chapter (4) is experiment of electricity load forecasting. There are 6 sections to present the different kinds of experiments.

Chapter (5) explains the result and discussion which gives the forecasting results on different experiments.

Chapter (6) illustrates the conclusion and recommendation which discusses the overall work for this research and recommends for future research.

Chapter 2
Literature Review

2.1 Forecasting Model

There are various types of methodologies to forecast electrical load demand. Generally, it can be classified into two basic groups. They are Qualitative forecasting method and Quantitative forecasting method. The first one is based on peoples' opinion. The second one is in terms of numerical data. In order to second, there are several techniques such as Regression analysis (Papalexopoulos & Hesterberg, 1990), Stochastic Time Series analysis (STS) (Vemuri, Hill, & Balasubramanian, 1973), Neural Network (Hippert, Pedreira, & Souza, 2001), Fuzzy Logic (Mbarek & Feki, 2016), Expert systems (Ho et al., 1990), Recursive Digital Filters (Maia & Gonçalves, 2009), Machine Learning and much more. The effective ways in electrical load forecasting and modeling is to use the efficient information that is essential for extracting and projecting into the future.

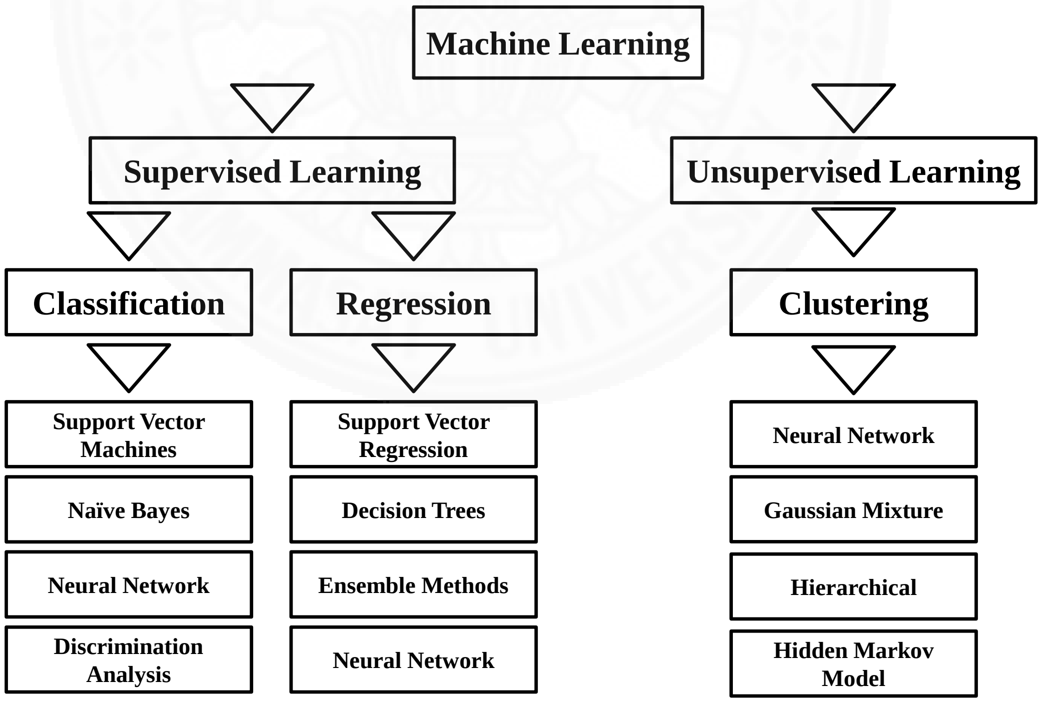


Figure 2.1 Different kinds of Machine Learning (Goyal, 2014)

According to Figure, there are many algorithms used for regression analysis. Neural Network (NN) is one of the most popular techniques in machine learning. NN is based on artificial intelligence (AI). Many researchers approached artificial neural network (ANN) since 1943. Moreover, it has been applied in forecasting areas in the late 1980's and early 1990's (Hippert et al., 2001; Lee, Cha, & Park, 1992). ANN composes of a group of neurons that are interconnected and evaluates the training output by processing and keeping the knowledge. Neural networks with one hidden layer, a nonlinear activation function and a sufficient number of hidden neurons are able to approximate any function with arbitrary precision. However, the error function is not convex and thus the result of the training depends on the initialization.

Support Vector Machine (SVM) was introduced by Vapnik and Chervonenkis. It is based on the statistical learning theory which studied the problem of interference to gain knowledge, make predictions and decisions, and constructing the models by a set of data (Scholkopf & Smola, 2001; Vapnik, 1999). SVM is employed in several areas such as anomaly detection, classification, regression, and text categorization etc. The properties of SVM are duality, kernels, margin, convexity, and sparseness (Scholkopf & Smola, 2001). SVM is a supervised learning algorithm which intend for classification, later regression is introduced for forecasting areas named as Support Vector Regression (SVR). It can robust in high dimension, generalize effectively and no over fitting (Smola & Schölkopf, 2004). So, it is regarded as one of the most efficient family in machine learning algorithm. SVR is characterized by a set of hyper parameters which regulate the function's behavior. This algorithm does not give a good prediction in terms of the uncertainties of the estimation and the determination of parameters for the algorithm is difficult. Consequently, optimizing the hyper parameters plays an important role in the training process.

There are many optimization algorithms to utilize SVR model such as Genetic Algorithm (GA) (Wu, Tzeng, & Lin, 2009), Particle Swarm Optimization (PSO) (Duan, Xie, Guo, & Huang, 2011), Bayesian Optimization (BO) (M. H. Law & Kwok, 2001), and Simulated Annealing (SA) (Geng, Huang, Li, & Hong, 2015) etc.

The population based optimization GA is widely used to optimize hyper parameters in forecasting areas (Chen & Wang, 2007; Pai & Hong, 2005; Saini, Aggarwal, & Kumar, 2010). Wei-Chiang has reported the comparison of three forecasting techniques which are SVR-GA, ANN and regression. The result were compared for four regions in Taiwan. SVR-GA obtained better results than other methods except the northern region (Hong, 2009b; Hong, Dong, Zhang, Chen, & Panigrahi, 2013). Moreover, GA has the ability to escape of trapping local minima (Xiao, Wang, Yang, & Xiao, 2015). Liye Xiao presented the combining of SVR and GA which were forecasted for each day such as Monday, Tuesday etc. According to separate for each day, Saturday's MAPE was the worst one because it did have the same load pattern with others. Wen Yu et.al supposed hybrid GA-SVR model to forecast for china electricity load demand. Seasonal index is used as an input. Traditional time series method ARIMA is compared in their research that GA-SVR outperformed ARIMA model for every month in 2008 (Zhang et al., 2012).

Many researchers have reported that the hybrid model PSO-SVR outperformed the other traditional methods and NN (Duan et al., 2011; Hong, 2009a; Jiang & Wu). According to easy implementation with parameters, PSO is mostly applied in combining with machine learning model (Lin, Ying, Chen, & Lee, 2008; Liu et al., 2011). Lin et.al have reported by using different kinds of kernel function to utilize the PSO-SVR. Optimizing radial basic kernel function by PSO gave the best result among kernel functions (Lin et al., 2008).

Bayesian Optimization (BO) became popular in recent decades. BO is a successful algorithm for machine learning such as neural network, support vector machine (Klein, Falkner, Bartels, Hennig, & Hutter, 2016). It is a useful approach for the selecting a suitable hyper parameters and this method be able to deal with the two mains problems. One is the local minima problem and the other one is the computational complexity problems. Bayesian method is proposed to generalize with SVR to deal with these problems. The results from the study showed that BO-SVR performed well in the large prediction with a reliable sensitivity (Gopi et al., 2013). This proposed method be able to determine the parameter selection more efficiently and provide an estimate of prediction uncertainties (T. Law & Shawe-Taylor, 2017).

2.2 External Factor Effect on Electrical Consumption

The external factors that influence the forecasting of electricity load demand are

- Temperature
- Holidays, Bridging holidays
- Special Events (Example: Songkran in Thailand)
- Seasonality
- Humidity
- Global warming
- Population Growth
- Industrial Growth
- Economic State

The factor for accurate forecast depends on short term, medium term and long term. For short term and medium term, the weather influence and time factors are the main factor. The customer class, GDP, and the population are important factors for long term load forecasting.

Chapter 3
Methodology

3.1 Support Vector Machine

In the recent decade, Support Vector Machine becomes popular in the electricity forecasting. SVM is an innovative of Artificial Intelligence (AI) (Türkay & Demren, 2011)**.** The SVM Model was first innovated for classification and then extended for regression, named Support Vector Regression (SVR). The application of SVM are Time Series Analysis, Classification, Regression, Machine Vision, Anomaly Detection, and Text Categorization. The properties of SVM are duality, kernels, margin, convexity and sparseness. The advantages are that it can generalize well and robust in high dimension (no over fitting).

3.1.1 Support Vector Regression

Support vector regression (SVR) is a well-known machine learning algorithm proposed by Vladimir Vapnik Steven Golowich and Alex Smola in 1997(Smola & Schölkopf, 2004). SVR is based on a statistical learning theory using a high dimensional feature space. The SVR model gives great achievement in many forecasting area (Türkay & Demren, 2011).

SVR is used structural risk minimization (SRM) principle. The great concept of SRM is the application of minimizing an upper bound to the generalization error instead of minimizing the training error. Therefore, SVR will be equivalent to solving a linear constrained quadratic programming problems based on this principle. So, the solution achieves global minimum error (Drucker, Burges, Kaufman, Smola, & Vapnik, 1997).

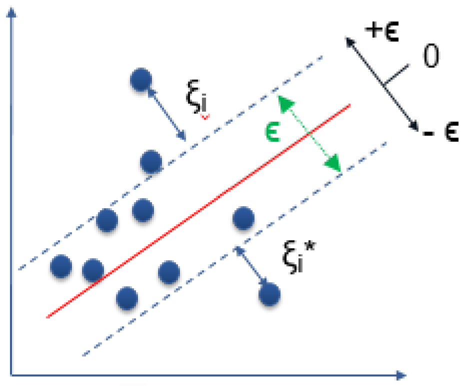


Figure 3.1 SVR Model (Drucker et al., 1997)

There are two important features in the implementation of SVR. The first one is Kernel Functions, and the second one is quadratic programming. SVR parameter can be achieved by the second. To find a large range of the solution space, the kernel functions can be used (Drucker et al., 1997).

Suppose we have a set of training data $\{x_i \ , y_i\}_{i=1}^{N}$, where x_i is the input of training data and y_i is a response value (corresponding output value) and N is the corresponding size of training data. The core idea of SVR is to map the data x_i into a high dimensional feature space via a non-linear mapping and to perform a linear regression. The approximating function is

$$f(x) = w\emptyset(x) + b \tag{1}$$

where $\emptyset(x)$ is the finite-dimensional space approaches to map into a higher dimensional space and w and b is minimized by the following minimizing risk function.

$$R = \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{l} \xi_i + \xi_i^*\right) \tag{2}$$

Subject to the constraints;

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$

$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i^* \geq 0, \xi_i \geq 0$$

ε , represents the Vapnik's ε -sensitive tube. If the observed value (support vector) is within the tube, the error is zero otherwise there is an error. The distance between the observed value and the tube boundary is the error value. The constraint C is the box constraint. It a positive numeric value that maintains the penalty imposed on observations which lie outside the tube (ε). It also helps to prevent overfitting. Then, it determines the trade-off between the flatness of *f(x)* and the amount up to which deviation larger than ε are tolerated.

By Lagrangian Multipliers, we can find the hyperplane parameter.

$$w = \sum_{i=1}^{l} (\propto_i - \propto_i^*) \emptyset(x_i) \tag{3}$$

Then,

$$f(x) = w\emptyset(x) + b$$
$$= w \sum_{i=1}^{l} (\propto_i - \propto_i^*) \emptyset(x_i)\emptyset(x) + b$$
$$f(x) = w \sum_{i=1}^{l} (\propto_i - \propto_i^*) K(x_i, x) + b \tag{4}$$

$K(x_i, x) = \emptyset(x_i)\emptyset(x)$ is kernel function.

Table 3.1 Three types of kernel functions

Description	**Formula**
Radial Basic Function(RBF)	$K(x, x_i) = exp\ \gamma \|x - x_i\|^2$
Linear Kernel Function (LKF)	$K(x, x_i) = x_i\, x$
Polynomial Kernel Function (PKF)	$K(x, x_i) = (1 + x_i\, x)$

RBF Kernel, *K(x, x') = exp (γ ||x - x'||2)* where || . || is Euclidean distance and γ is negative. We can see that if the distance between 2 points is large then this function gives a very small value. Thus, this function achieves small loss for far away points.

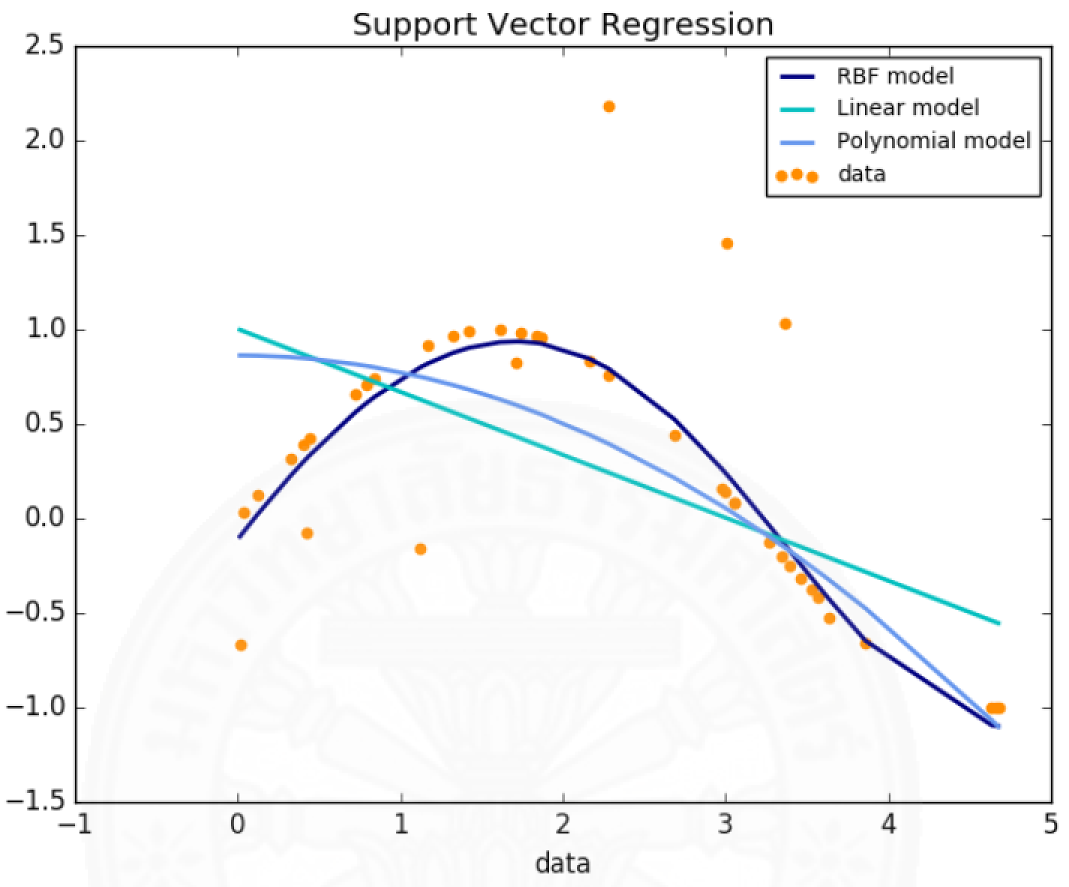


Figure 3.2: Three Types of Kernel Functions

The following figure is the model and process of Support Vector Regression (SVR).

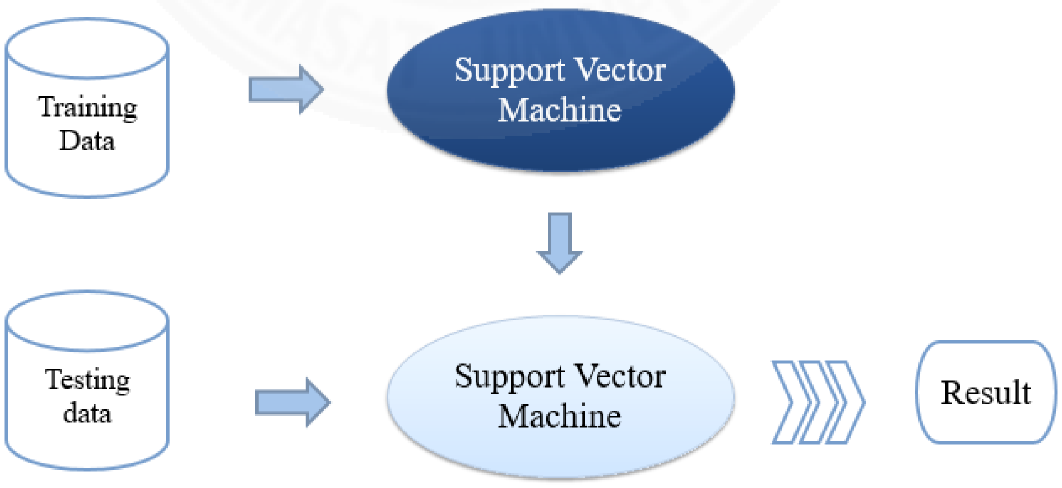


Figure 3.3 Model and Process of SVR

The cost error C, the epsilon tube ε, the mapping function $\emptyset(x)$, the kernel function, and the number of training data are essential factor to get good accuracy in SVR.

3.2 Particle Swarm Optimization

Particle swam optimization (PSO), a population-based algorithm inspired from biological organism such as fish schooling and bird gathering to imitate the food searching behavior, is proposed by Kennedy and Eberhart (Eberhart & Kennedy, 1995). The concept of PSO is to keep finding the best solution by updating its position and velocity. It is calculated the fitness function by each step at each particles. The algorithm finds next velocity and position after evaluation the fitness function.

The outline of the PSO is the following steps. First, we create initial particles and velocities. Then, it calculates the fitness function at each particle position and velocity in every iteration which sets two best solutions. The first one is the particle which flow its own best solution and gives the good fitness value. It is called particle best (*pbest*). The second one is finding the best solution among the swarm which is called global best (*gbest*). The following equation and figure show the way of updating the position and velocity of the particles.

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand_1 \times \left(p - x_i(t)\right) + c_2 \times rand_2 \times \left(g - x_i(t)\right) \quad (5)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (6)$$

where,

v = new velocity
x = new position
i = particle index
t = discrete time index
v_i = velocity of i^{th} particle
c_1 = weight of local information
c_2 = weight of global information
p = personal best position

g = global best position
x_i = position of ith particl

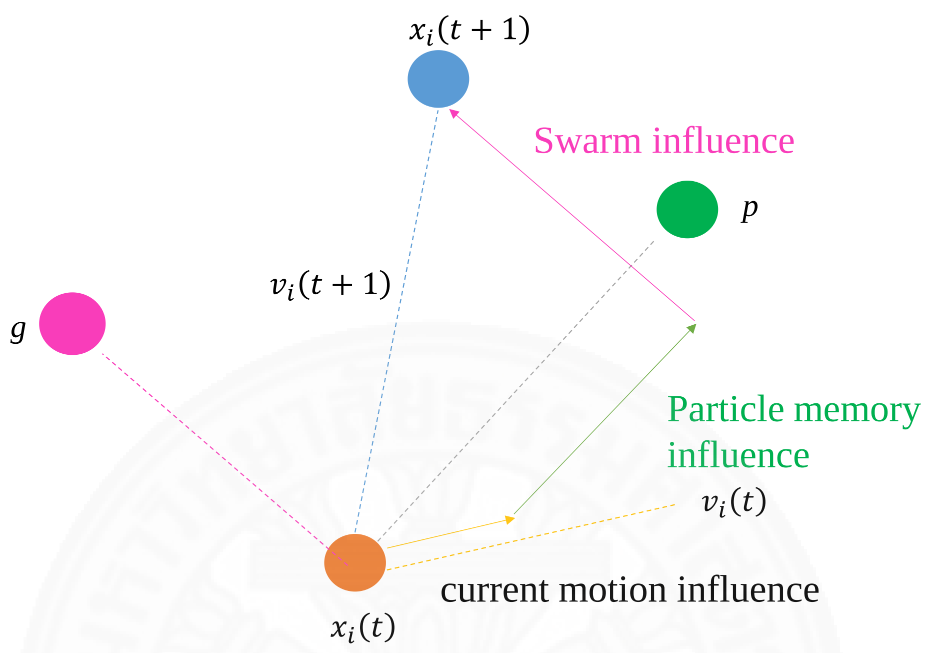


Figure 3.4 Update positions and velocities

3.2 Genetic Algorithm (GA)

Optimization issue which depends on natural selection driving biological interpretation is genetic algorithm (GA). In the 1970's, GA was developed by J. Holland, K.Delong, D.Goldberg. GA reworks a population for individual key repeatedly (Hong et al., 2013). It randomly chooses the individual of current population which will become parent. For next generation, the children was produced by adopting the parent. The solution gets along an optimal solution over consecutive generation.

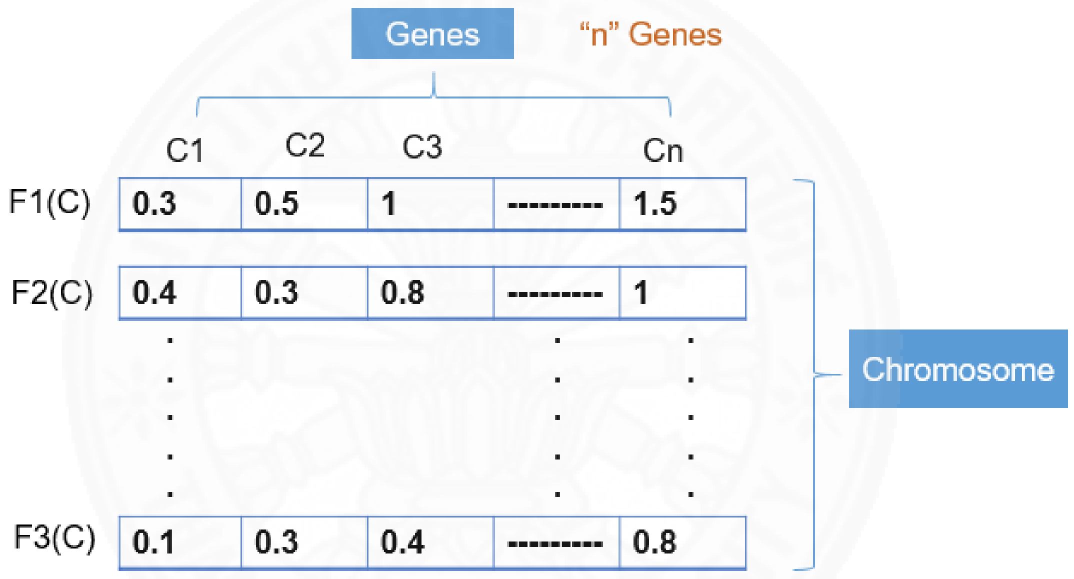


Figure 3.5 GA population

There are three main methods of regulation to produce children from current parent for each step. They are crossover, mutation and selection. In crossover, the offspring chromosome are created by the combination of the parent chromosome vector. In Mutation, it is created by changing the parent chromosome randomly. Selection choose the individual parent chromosome which spread out the population at next offspring generation. The following figures show the three stages of GA.

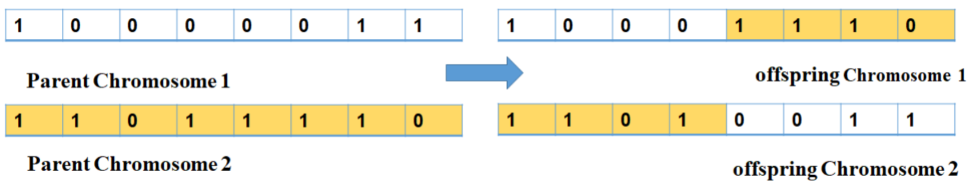


Figure 3.6 Crossover

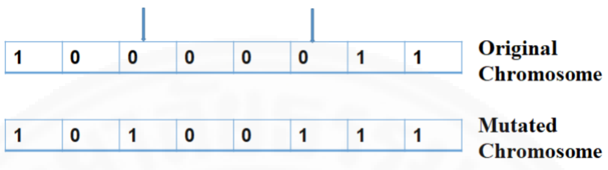


Figure 3.7 Mutation

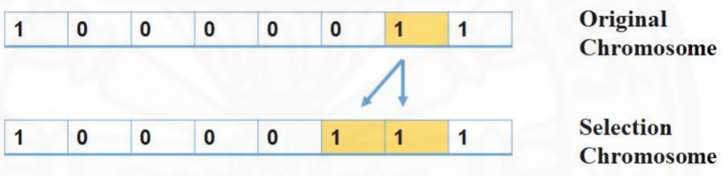


Figure 3.8 Selection

The important key of using GA is the initial population, developing the next generation, selection, crossover, mutation and stopping conditions of the algorithm. Although the features attribution is not brisk, it is a heuristic that is good for combination situation. The best features of GA is that focuses associate information from original chromosome.

3.4 Bayesian Optimization

Another special case for non-linear optimization is Bayesian Optimization (BO). It aims to maximize or minimize the objective function *f(x)* for *x* in bounded area. The results will get different even calculating same point because the function is stochastic and deterministic. The distinct case of BO is that it builds a probabilistic framework for *f(x)* and accomplishes the framework to perform decisions about how to evaluate next value of *X* in the function. The idea is to use all information access from last evaluation of *f(x)* (Snoek, Larochelle, & Adams, 2012). It has the ability of storing a memory of all of the previous observations (Martinez-Cantin, 2014), balancing the sample data points which have low fitness function and seeking the space which the model did not train effectively. Not only finding a superb setting of parameters but also less consuming time is the core idea of BO. In spite of being a global technique, searching for global solution is not like other algorithm. There is no need to begin with various initial values. Moreover, BO is suitable for optimizing parameters of another function.

The two major choice of applying BO is using Gaussian Processes (GP) and Acquisition Function (AF). The first one is selecting a prior value over function which represents assumptions to be optimized. The second one constructs a utility function which refer from the posterior model to evaluate next point of the function. Therefore, the key component of minimization process is GP model of *f(x)*, the updating procedure to modify GP model for new calculation of *f(x)* and an acquisition function which is based on GP model of f which minimize to find the next coming value of x for interpretation.

3.4.1 Gaussian Processes (GP)

GP model of the function, a popular model which is internally maintained by Bayesian Optimization (BO). GP is a prior distribution not only conducive but also effective on function. Mean function and covariance function are specified fully in Gaussian Processes (Gao, Gunn, Harris, & Brown, 2002).

For dataset,$x = \{x_1, \dots, x_n\}$ and the values of the function $f = \{f(x_1), \dots, f(x_n)\}$ is mentioned by GP model

$$f \sim GP(n, K)$$

where $n: X \to R$ is the mean function which tend to $n(x) = E[f(x)]$. $K : X^2 \to R$ is the covariance of function which specify as

$$K(x, x') = [E(f(x) - n(x))(f(x') - n(x')] \qquad (7)$$

According to induce a posterior observation over the function, GP is very useful and popular model which lead to update the beliefs of what function f similar to. .

3.4.2 Acquisition Function (AF)

Applying acquisition function (AF) in BO is a good innovation in recent decade. The determination of the next best point to calculate is the basic fact of using AF. The ability of AF is balancing the sampling of points which have the lowest fitness function and exploring the part which cannot train well. It evaluates the good point formed on the posterior distribution function. There are three types of AF that used for optimizing in machine learning algorithm. They are

- Expected Improvement
- Probability of Improvement
- Lower Confidence Bound

3.4.2.1 Expected Improvement

The scheduled amount of improvement in the fitness function is evaluated by Expected improvement which neglects the point of causing an increment in the object.

$$EI(x, P) = [E_p \max\left(0, \mu_p(x_{best}) - f(x)\right)] \qquad (8)$$

Where x_{best} = the position of the lowest posterior mean

$\mu_p(x_{best})$ = the lowest value of the posterior mean

3.4.2.2 Probability of Improvement

The maximization of the improving probability of the current point is the strategy of the probability of improvement.

$$PI(x, P) = P_Q(f(x) < \mu_p(best) - m) \qquad (9)$$

In Bayeopt, *m* is taken as noise so the probability will be

$$PI = \emptyset\left(\gamma_p(x)\right) \tag{10}$$

$$\gamma_p(x) = \frac{\mu_p(best) - m - \mu_p(x)}{\sigma_p(x)}$$

where, σ_p = the posterior standard deviation of GP

3.4.2.3 Lower Confidence Bound

The recent improvement is that AF follows the curve *G*, two standard deviations of each point under the posterior mean.

$$LCB = 2\sigma_p(x) - \mu_p(x) \tag{11}$$

Based on literature review, Expected Improvement (EI) function is a trendy one. Therefore, EI was applied in this research.

3.5 Error Calculation

Root Mean Square Error (RMSE) is used to calculate the fitness function of SVR-PSO between the actual load $L_t(d)$ and the forecasted load $F_t(d)$. $RMSE(d)$ is calculated by the following equation.

$$RMSE(d) = \sqrt{\frac{1}{t}\sum_{i=1}^{48}(L_t(d) - F_t(d))^2} \tag{12}$$

In order to calculate the forecasting error, Mean Absolute Percentage Error (MAPE) is used. When there are total of 48 observations or 48 forecasted values, $MAPE(d)$ is calculated by the following equation.

$$MAPE(d) = \left[\left(\frac{1}{t}\sum_{t=1}^{48}\left|\frac{L_t(d)-F_t(d)}{L_t(d)}\right|\right)\right] \times 100\% \tag{13}$$

3.6 Data Cleaning

The data from EGAT are applied in this research. The data has been recorded as every half an hour for one day so there are 48 periods in 24 hours. There are five noticeable daily load patterns which are Monday, holidays, bridging holidays, weekdays and weekends. Monday is the start day of the working day which is different load pattern by comparing to other weekdays. Holidays load demand are significantly lower than all other days. Bridging holidays are non-holidays which take between a holiday and a weekend or two holidays. Bridging holidays load demand also have distant load pattern as shown in Figure. Weekends load demand are also significantly low because the offices and industries are closed. Therefore, we noted as five different load demand groups. Figure shows the different load pattern for separate groups in January, 2013.

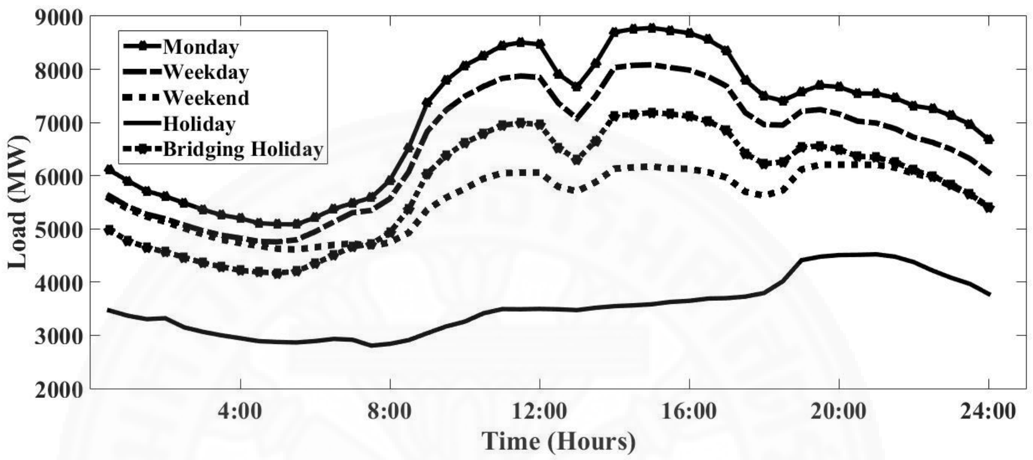


Figure 3.9 Average different load pattern for five groups in January, 2013

It is investigated how many outliers exist after making different groups. For the first step, weighted moving average is used to replace holidays. Then, bridging holidays are put in place by weighted moving average as a second step. There also have some irregular load patterns. So, the outliers are detected by the Time-Window based filtering band. We establish a window band $B_t(d)$ with mean and standard deviation by constructing the same day dataset. The following equation shows how we detect the outliers.

$$B_t(d) = \frac{\left[\sum_{i=1}^{k} L_t(d-7\times i)\right]}{k} \pm N \times SD(V_t(d)) \tag{14}$$

$$V_t(d) = [L_t(d), L_t(d-7), L_t(d-14) \ldots L_t(d-7\times m)] \tag{15}$$

Where,

$L_t(d)$	= Load at period t for day d,
$L_t(d\text{-}1)$	= Yesterday load at period t for day d
m	= Number of week

The size of N changes the width of the band. N is set up 1.6 based on many experiments. The last step is to replace the outliers which locates outside the filtering band by simple moving average. The load pattern after detecting and replacing outliers are as shown in figure.

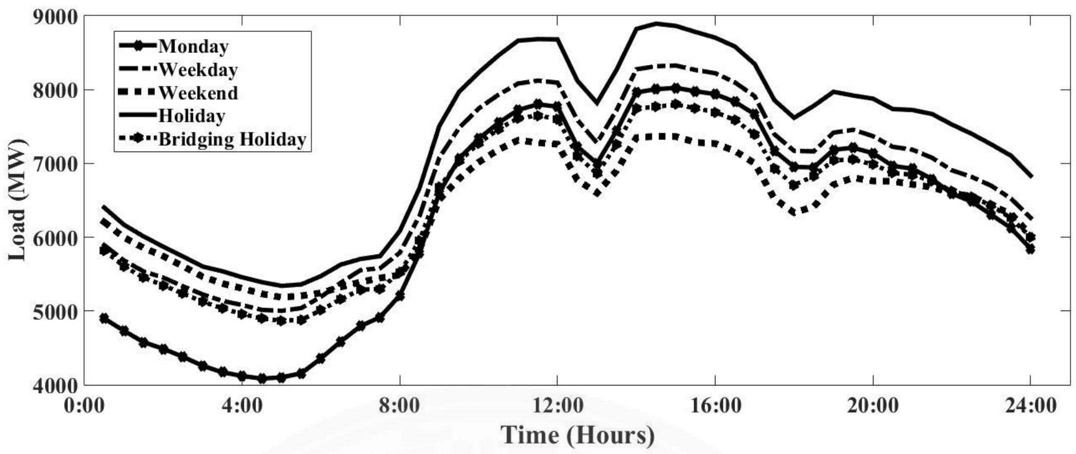


Figure 3.10 Average load pattern after replacing outliers for five different groups in January, 2013

Chapter 4
Experiment of Forecasting Electricity Load

4.1 Experiment 1: Study of support vector regression on different training

In order of being a very specific algorithm, the arrangement of training dataset plays an important issues in SVR as mentioned in section 3. Therefore, the first study of SVR is applied with different training dataset.

Data of Bangkok and Metropolitan region is selected in this study, since it has a large demand variation over time. November is used as the testing month since it does not have special holidays. The training data is two inputs: yesterday's load and same day previous week's load. May, June, July, August, September, and October data are used to train the model. November Data is used for testing performance.

The forecasting model is

$$F_t(d) = w_1 L_t(d-1) + w_2 L_t(d-7) + e_t(d) \tag{16}$$

Where,

$F_t(d)$	= Forecasted load of day *d* at period *t*,
$L_t(d-1)$	= Yesterday's load of day *d* at period *t*,
$L_t(d-7)$	= Same day previous week's load of day *d* at period *t*,

Table 4.1 Example of Training Data Set for Testing Data Set of November 1, 2013

No.	**Input**		**Output**
	$\boldsymbol{L_t(d-1)}$	$\boldsymbol{L_t(d-7)}$	$\boldsymbol{F_t(d)}$
Training 1	Oct 24(Thu)	Oct 18 (Fri)	Oct 25 (Fri)
Training 2	Oct 17(Thu)	Oct 11 (Fri)	Oct 18 (Fri)
Training 3	Oct 10(Thu)	Oct 04 (Fri)	Oct 11 (Fri)
Training 4	Oct 03(Thu)	Sep 27 (Fri)	Oct 04 (Fri)
Testing 1	Oct 31(Thu)	Oct 25 (Fri)	Nov 01 (Fri)

The training data for 1 month include four pairs. In this experiment, we trained up to 6 months. Therefore, there are 8 pairs for 2 months training and 24 pairs for 6 months. In result section, average RMSE and MAPE are summed up.

4.2 Experiment 2: Testing of SVR on different kernel functions

Kernel function is one of the key part in SVR which is to get the large amount of solution space. There are three types of kernel function as described in section 3. This experiment show which kernel function gives the better performance among other. The training data is from 2012 to 2013 and the testing data is 2013. The forecasting model for testing the different kernel function is

$$F_t\,(d) = w_1\,L_t(d-1) + w_2\,L_t(d-7) + w_3T_t(d) + w_4T_t(d-1) + w_5\,DoW + w_6\,MOY \quad (17)$$

where,

$F_t\,(d)$	= Forecasted Load of day *d* at period *t*,
$T_t\,(d)$	= Temperature of day *d* at period *t*,
$T_t\,(d-1)$	= Yesterday temperature of day *d* at period *t*,
$L_t\,(d)$	= Actual load of day *d* at period *t*,
$L_t(d-1)$	= Yesterday load of day *d* at period *t*,
$L_t(d-7)$	= Same day previous week’s load of day *d* at period *t*,

Table 4.2 Example data set for testing November 1, 2013

Training						
$\boldsymbol{L_t(d-1)}$	$\boldsymbol{L_t(d-7)}$	$\boldsymbol{T_t(d-1)}$	$\boldsymbol{T_t(d)}$	$\boldsymbol{DoW}$	$\boldsymbol{MoY}$	**Target** $\boldsymbol{F_t\,(d)}$
06-11-12	31-10-12	06-11-12	07-11-12	4	11	07-11-12
(Wed)	(Thurs)	(Wed)	(Thurs)			(Thurs)
. . .	. . .	. . .	. . .	. . .	. . .	. . .
28-11-13	22-11-13	28-11-13	29-11-13	5	11	29-11-13
(Thurs)	(Fri)	(Thurs)	(Fri)			(Fri)
Testing						
$\boldsymbol{L_t(d-1)}$	$\boldsymbol{L_t(d-7)}$	$\boldsymbol{T_t(d-1)}$	$\boldsymbol{T_t(d)}$	$\boldsymbol{DoW}$	$\boldsymbol{MoY}$	**Compare** $\boldsymbol{F_t\,(d)}$
31-10-13	25-10-13	31-10-13	01-11-13	5	11	01-11-13
(Thurs)	(Fri)	(Thurs)	(Fri)			(Fri)

4.3 Experiment 3: Testing of SVR on different input variables

After testing for 1 month (November), then tests for 1 year by using same day training and different input strategies. When we test for same day training, we do not consider for day of week because the model are tested by the same day training. Therefore, we consider one extra input which is Month of Year (*MOY*).The two forecasting models are shown in the following figures.

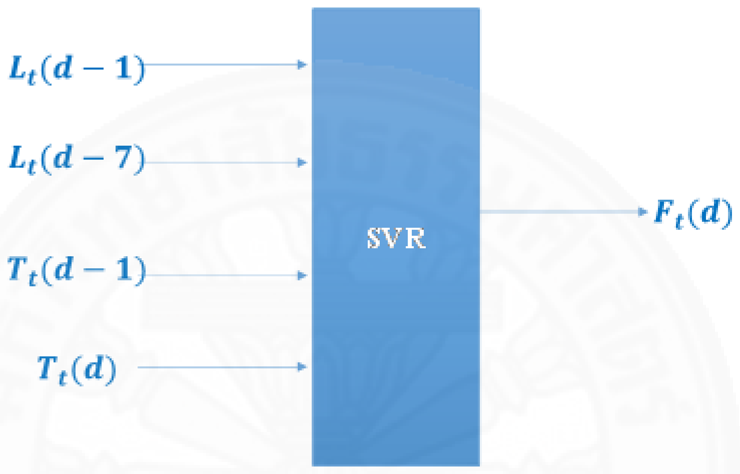


Figure 4.1 Case 1, experiment on 4 inputs

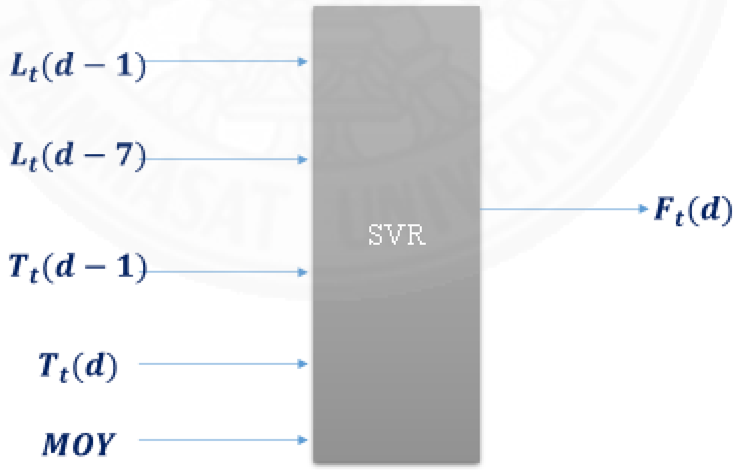


Figure 4.2 Case 2, experiment on 5 inputs

Table 4.3 Example Data Set for Case 1 to test Data Set of January 5(SAT), 2013

Training				
$L_t(d-1)$	$L_t(d-7)$	$T_t(d-1)$	$T_t(d)$	**Target** $F_t\,(d)$
07-01-12	13-01-12	13-01-12	14-01-12	14-01-12
(Sat)	(Fri)	(Fri)	(Sat)	(Sat)
. . .	. . .	. . .	. . .	. . .
22-12-12	28-12-12	28-12-12	29-12-12	29-12-12
(Sat)	(Fri)	(Fri)	(Sat)	(Sat)
Testing				
$L_t(d-1)$	$L_t(d-7)$	$T_t(d-1)$	$T_t(d)$	**Compare** $F_t\,(d)$
29-12-12	04-01-13	04-01-13	05-01-13	05-01-13
(Sat)	(Fri)	(Fri)	(Sat)	(Sat)

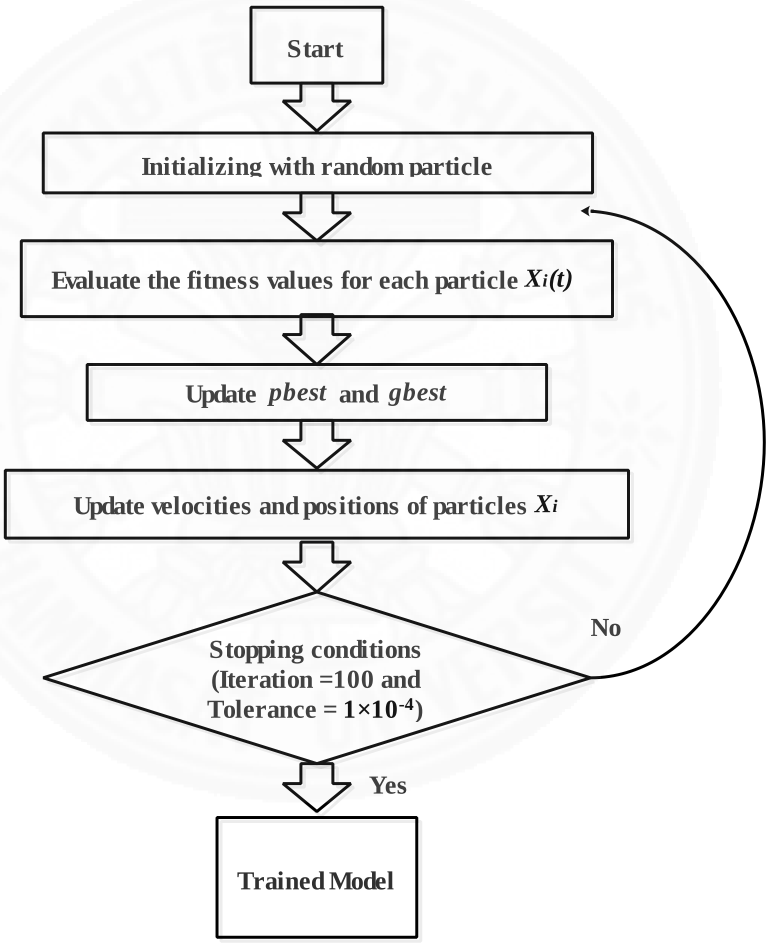
Table 4.4 Example Data Set for Case 2 to test Data Set of January 5(SAT), 2013

Training					
$L_t(d-1)$	$L_t(d-7)$	$T_t(d-1)$	$T_t(d)$	MoY	**Target** $F_t\,(d)$
07-01-12	13-01-12	13-01-12	14-01-12	1	14-01-12
(Sat)	(Fri)	(Fri)	(Sat)		(Sat)
. .	. .	. . .	. . .	. . .	. . .
22-12-12	28-12-12	28-12-12	29-12-12	1	29-12-12
(Sat)	(Fri)	(Fri)	(Sat)		(Sat)
Testing					
$L_t(d-1)$	$L_t(d-7)$	$T_t(d-1)$	$T_t(d)$	MoY	**Compare** $F_t\,(d)$
29-12-12	04-01-13	04-01-13	05-01-13	1	05-01-13
(Sat)	(Fri)	(Fri)	(Sat)		(Sat)

4.4 Experiment 4: Testing of SVR with PSO

4.4.1 Experiment 4: Testing of SVR with traditional PSO

Many optimization algorithm are applied to set the hyper parameters of SVR. The first hybrid experiment for this research is combining with PSO. The encoding part is important before training the hybrid model. The three parameters which need to tune the optimal solution are *C*, ε, and γ. The particle is encoded as $X_i = (C_i, \varepsilon_i, \gamma_i)$. The following figure is shown the outline algorithm of SVR-PSO. The model will update *p* and *g* until it reaches the stopping conditions.



Start
Initializing with random particle
Evaluate the fitness values for each particle $X_i(t)$
Update $pbest$ and $gbest$
Update velocities and positions of particles X_i
Stopping conditions (Iteration =100 and Tolerance = 1×10^{-4})
No
Yes
Trained Model

Figure 4.3 Flow chart of SVR with traditional PSO

The particles setting and bounding intervals are shown in the following tables.

Table 4.5 Interval of SVR hyper parameters

Variables (Hyperparameters)	**Interval (Upper Bound - Lower Bound)**
C	1×10^{-1} - 1000
ε	1×10^{-2} - 10
γ	1×10^{-4} - 200

Table 4.6 PSO's parameters

Parameter of PSO	**Value**
Number of particles	100
c_1 and c_2	2
Maximum number of iteration	100
Tolerance	1×10^{-4}

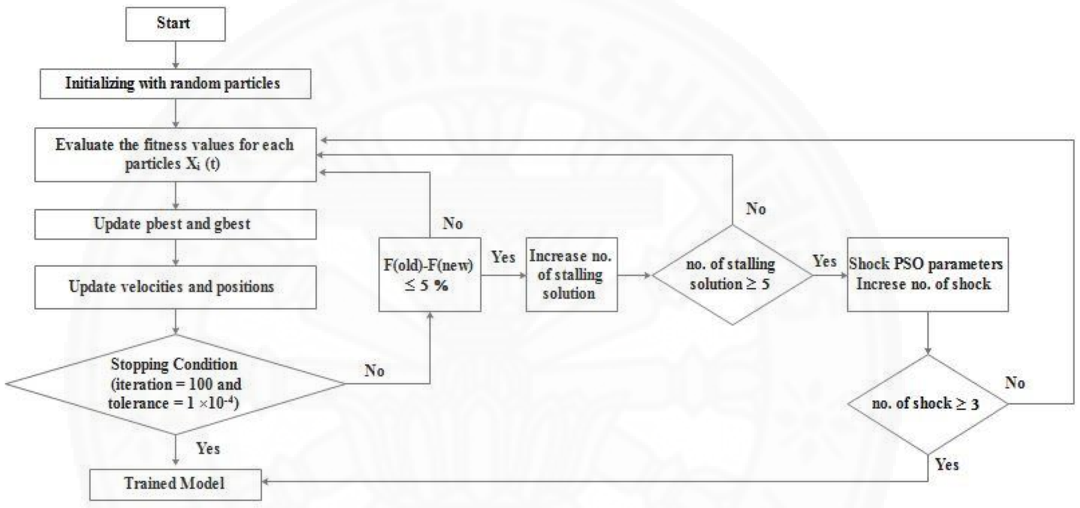
Table 4.7 Example data arrangement of training and testing to forecast 1st August, 2016

Training					
$L_t(d-1)$	$L_t(d-7)$	$T_t(d-1)$	$T_t(d)$	MoY	**Target** $F_t\,(d)$
05-01-14	30-12-13	05-01-14	06-01-14	1	06-01-14
(Sun)	(Mon)	(Sun)	(Mon)		(Mon)
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
24-07-16	18-07-16	24-07-16	25-07-16	7	25-07-16
(Sun)	(Mon)	(Sun)	(Mon)		(Mon)
Testing					
$L_t(d-1)$	$L_t(d-7)$	$T_t(d-1)$	$T_t(d)$	MoY	**Compare** $F_t\,(d)$
31-07-16	25-07-16	31-07-16	01-08-16	8	01-08-16
(Sun)	(Mon)	(Sun)	(Mon)		(Mon)

4.4.2 Experiment 4: Testing of SVR with shocking PSO

The behavior of PSO is easy to trap in local minima. We try to escape for the trapping process. Therefore, testing with some additional technique to improve the local minima is a good way.

The additional technique is that we stop the process of PSO when the absolute value of subtracting its fitness function values is less than 0.05, then we give a new weight value and new acceleration factor values. Therefore, the process runs until the stopping conditions by changing those values.



Start
Initializing with random particles
Evaluate the fitness values for each particles X_i (t)
Update pbest and gbest
Update velocities and positions
Stopping Condition (iteration = 100 and tolerance = 1×10^{-4})
No
Yes
Trained Model
F(old)-F(new) ≤ 5 %
No
Yes
Increase no. of stalling solution
no. of stalling solution ≥ 5
No
Yes
Shock PSO parameters Increse no. of shock
no. of shock ≥ 3
No
Yes

Figure 4.4 Flow chart of SVR with shocking PSO

4.5 Experiment 5: Testing of SVR with GA

The second experiment of optimization algorithm for this study is combining the GA with SVR. The encoding part is similar to PSO as stated in experiment 4. The data arrangement of the SVR-GA is the same arrangement in section 4.4.

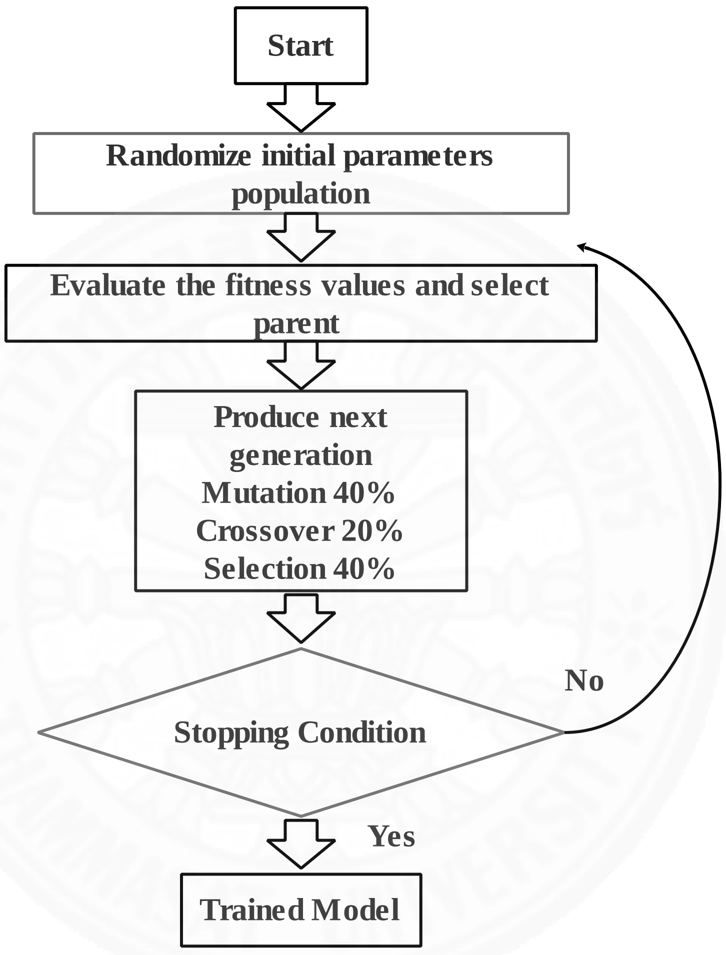


Figure 4.5 Flow Chart of SVR-GA

4.6 Experiment 5: Testing of SVR with BO

The last experiment of this research is combing BO with SVR. BO becomes well-known optimizing techniques in machine learning. The data arrangement is same as the previous two experiments. The following figure shows the outline algorithm of SVR-BO.

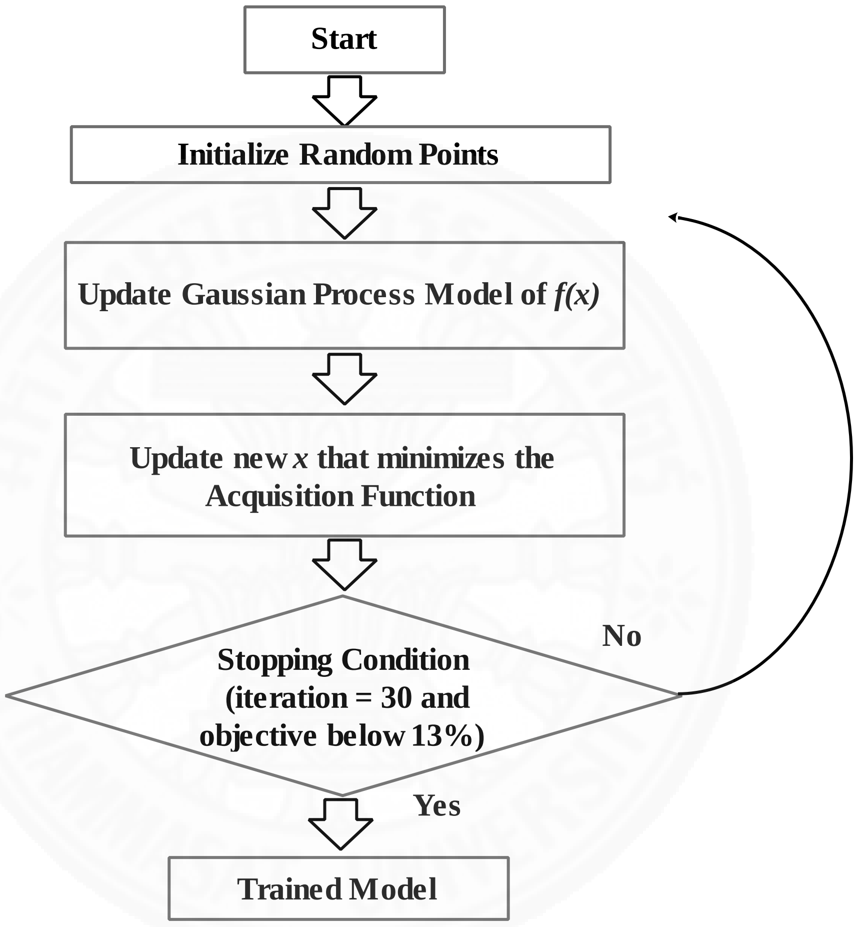

Start
Initialize Random Points
Update Gaussian Process Model of *f(x)*
Update new *x* that minimizes the Acquisition Function
Stopping Condition (iteration = 30 and objective below 13%)
No
Yes
Trained Model

Figure 4.6 SVR-BO's flow chart

CHAPTER 5
RESULT AND DISCUSSION

5.1 Experiment 1: Study of support vector regression on different training

The following Table 5.1 outlines the MAPE of the different training dataset. The experiment runs 6 times because it has 6 training dataset.

Table 5.1 MAPE for different training dataset

	1 month training	2 month training	3 month training	4 month training	5 month training	6 month training
	M=4	M=8	M=12	M=16	M=20	M=24
1/11/2013	1.2075	2.1301	3.0624	1.8675	1.9332	2.1369
2/11/2013	5.7108	4.9767	4.9768	4.4155	4.7282	5.0573
3/11/2013	4.8903	5.1648	4.6276	4.7759	4.6278	4.7376
4/11/2013	4.7768	2.9655	3.3233	3.3837	2.8307	3.3034
5/11/2013	1.4386	1.7687	1.8696	1.9224	1.8699	3.3517
6/11/2013	5.9606	2.8389	2.5566	2.4854	2.3285	2.2492
7/11/2013	3.2643	2.0368	2.1603	2.5685	2.4309	2.3250
8/11/2013	7.4330	5.3945	5.1247	5.4844	5.6575	5.6996
9/11/2013	4.3250	3.2787	3.3853	4.2900	3.9068	3.2648
10/11/2013	4.3879	4.0119	4.7785	4.2294	4.0449	4.4263
11/11/2013	3.7000	3.1578	2.4299	3.8626	3.5281	4.0078
12/11/2013	2.3965	2.2764	1.9254	2.1492	2.3641	2.5233
13/11/2013	3.8181	2.2130	1.8118	2.3069	2.2530	2.3547
14/11/2013	4.7651	2.9620	2.4016	1.8251	2.4355	1.7923
15/11/2013	3.1809	3.5001	4.3801	4.2346	3.7650	2.7262
16/11/2013	6.1298	6.5579	7.0848	7.0275	6.2555	6.7513
17/11/2013	2.8920	4.5278	5.6428	5.5559	5.4463	5.2312
18/11/2013	3.5222	3.5252	4.7294	4.1136	3.2778	3.7578
19/11/2013	2.3386	2.6291	3.5464	3.9817	3.4038	3.0106
20/11/2013	1.6643	2.0835	3.1889	3.7737	3.1675	2.7204
21/11/2013	10.0845	6.3047	4.4520	3.0700	2.5598	2.6514
22/11/2013	1.2488	0.9691	1.1142	0.7728	0.9155	0.7958
23/11/2013	4.0429	3.4757	3.7618	3.4387	3.3625	3.5712
24/11/2013	5.9961	5.0035	4.8208	4.6417	4.7753	4.7440
25/11/2013	2.2734	2.9123	2.3821	3.0153	3.9419	3.4607
26/11/2013	1.9336	1.9770	1.7800	1.5313	1.6652	1.8762

27/11/2013	2.1635	2.0499	1.8618	1.6913	2.2269	1.2410
28/11/2013	3.0519	5.0717	6.4414	4.2162	3.5210	3.5110
29/11/2013	3.7204	4.2302	2.3358	4.2988	4.2012	4.3238
30/11/2013	6.8719	6.4264	6.3926	6.6246	6.5713	6.5290
Average	**3.9730**	**3.5473**	**3.6116**	**3.5851**	**3.4665**	**3.4511**

According to Table 5.1, it shows the MAPE values for November. The average values of MAPE are shown in additional row for each training month.

There is a day (November 21) which has a high MAPE value in data sets. Our forecasting model used the same day previous week load data. When we forecast November 21, it has included October 23 which is a holiday, as an input. So, the holidays which included in training data sets, it causes the high MAPE value. When we increase the size of training data sets, the forecasting performance is good.

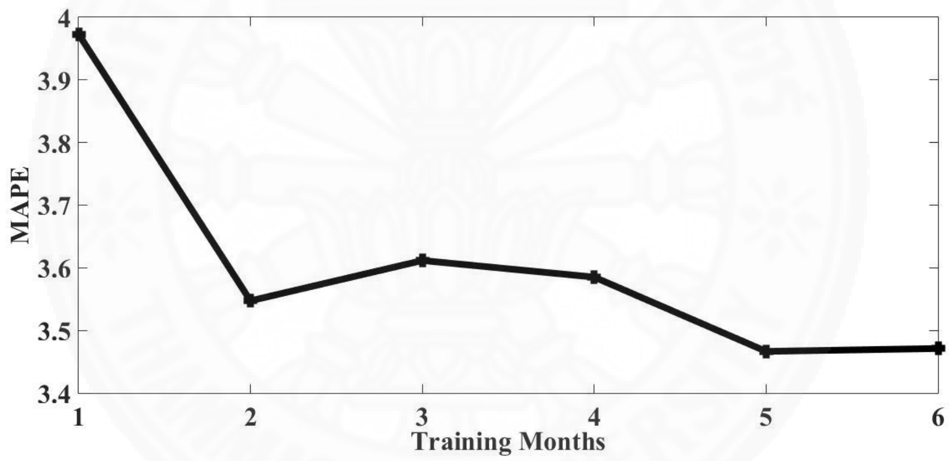


Figure 5.1 MAPE for different training dataset

Average values for each month have been calculated to summarize the results for all six cases. Each month average values provided an idea that how MAPE values change with the different amount of training data sets.

As being the first experiment, the main idea is to change the length of the training dataset to get the good accuracy. According to experiments, the MAPE values decrease when increasing the size of training data set. One more thing of getting inefficient forecasting accuracy is the holiday load. If the holiday load demand include in the training dataset, it can affect the forecasting outcomes. Therefore, the next experiment is to apply outlier detection for improving the accuracy of the forecasting model.

5.2 Experiment 2: Testing on different kernel functions

For experiment 2, the model is trained for six months to test November. The reason for testing November is that there is no holiday in it. So, there is no confusion about getting worse performance on different kernel functions. The following figure seperate into 3 groups which is weekday, weekend and total average of November. As we can see clearly that the performance of radial basis function (RBF) obtain the best performance among the other functions.

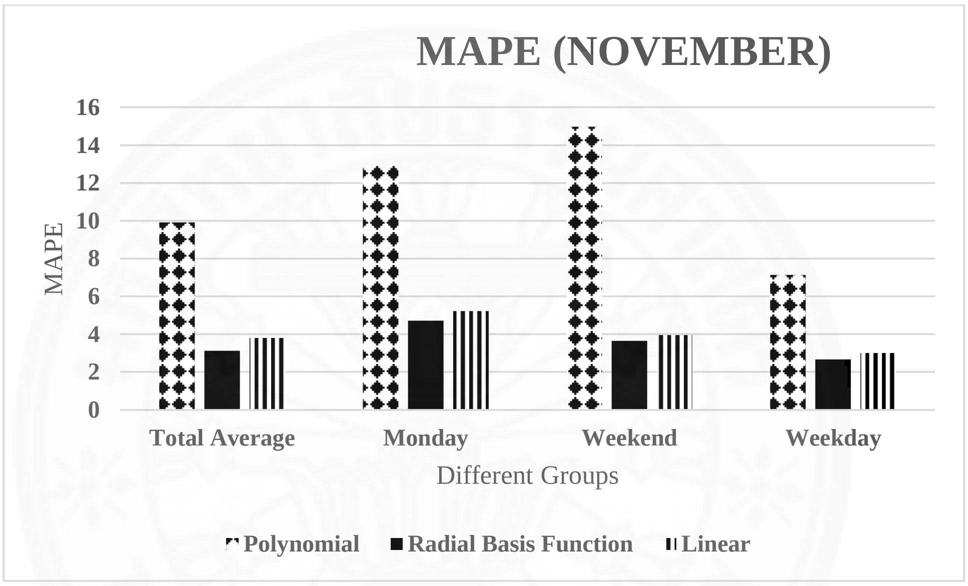


Figure 5.2 Testing on different kernel functions

Kernel function is applied for inner product of SVR. The performance of the model depends on the types of kernel functions. We have to change according to the type of problem. Since our data are non-linear, it is sure that the model does not fit well with linear kernel function. Therefore, the best kernel function for SVR is Radial Basic Function (RBF). RBF is rhythm with the support vectors as mentioned in section 3 in Figure 3.2.

5.3 Experiment 3: Testing on different input variables

According to the load patterns of each day, there are 6 categories to show the forecasting performance.

Table 5.2 MAPE of testing data in each month in 2013 for different inputs

	(Four inputs)	(Five inputs)	(Four inputs)	(Five inputs)	(Four inputs)	(Five inputs)
	January		**February**		**March**	
Total Average	4.1435	2.6426	4.6985	3.4145	3.8108	2.7244
Monday	2.7452	2.3319	3.3321	1.6660	1.9674	1.7402
Holidays	4.7258	2.3206	2.1446	2.1658		
Bridging Holiday	3.0198	1.7692	2.6000	1.8422	2.0880	1.8584
Weekend	4.7906	3.5383	5.6971	5.0456	4.1622	3.1029
Weekdays	4.8681	2.5350	5.3077	3.3312	5.0638	3.2605
	April		**May**		**June**	
Total Average	3.827	2.8432	4.6698	3.2754	4.0111	3.1131
Monday	4.8557	2.7231	2.3612	2.5257	2.9638	2.6774
Holidays	2.9529	2.6072	2.2822	2.7238		
Bridging Holiday	3.8411	1.2858	4.5489	1.9836		
Weekend	3.2575	1.6696	3.7817	2.5879	3.6742	2.903
Weekdays	4.759	4.0949	6.8549	4.3918	5.094	3.6369
	July		**August**		**September**	
Total Average	4.1559	3.2959	3.7362	2.6676	4.2725	2.3896
Monday	2.2466	2.9324	1.5437	3.2155	3.0986	1.7526
Holidays	2.0437	1.7977	1.2858	2.2068		
Bridging Holiday						
Weekend	3.9327	3.1965	3.394	2.3035	3.3398	2.5444
Weekdays	5.2937	3.3719	5.1112	3.142	5.8216	2.4749
	October		**November**		**December**	
Total Average	4.6554	2.8766	4.4412	2.7928	7.2143	5.4363
Monday	4.7572	1.2778	2.3322	1.7976	9.9525	4.2947
Holidays	3.3343	2.8345			8.3299	5.7313

Bridging Holiday					8.9519	6.8758
Weekend	4.5264	3.5842	4.3692	3.6869	6.1712	6.0438
Weekdays	5.2039	3.3502	5.8115	2.4253	8.2376	6.4825

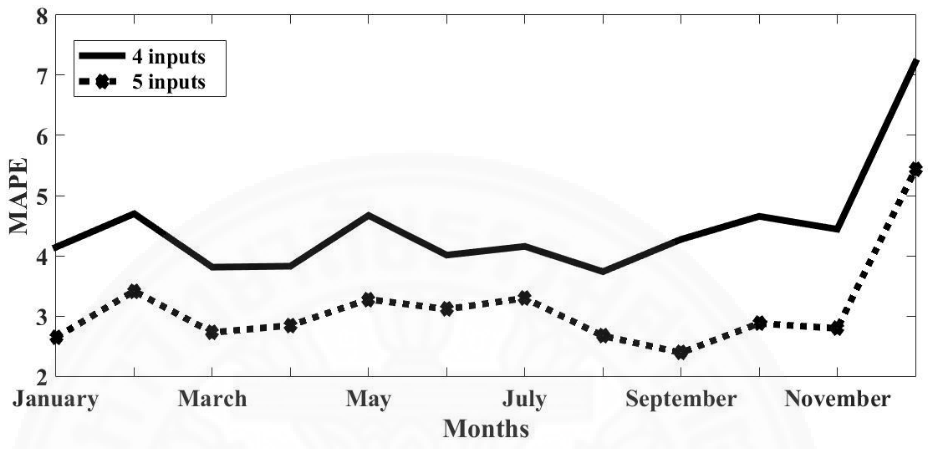


Figure 5.3 Comparison of total average between 4 inputs and 5 inputs variables

The input variables are one of the important part for getting good performance. So, we try to test in this experiment. Consequently, we add one more dammy value (month of year). Since we use same day training , it does not make sense for adding day of week as a dummy value. The outcomes show that adding dummy value gives good performance. By the above result, 5-inputs structure is better than 4-inputs structure. According to literature review, dummy values are one of the important part to get accurate results.

5.4 Experiment 5: Testing of SVR with PSO

5.4.1 Testing of SVR with traditional PSO

There are 6 separate groups to present the outcomes of the forecasting as it has different load patterns. The performance of MAPE is shown in the following table.

Table 5.3 MAPE performance by SVR with traditional PSO from August 2016 to July 2017

Month	Total Average	Weekday	Weekend	Monday	Holiday	Bridging Holiday
August	5.3120	3.8704	5.2965	6.6696	5.4113	
September	4.9365	3.4595	4.9609	6.5427		
October	5.1122	4.5085	3.5280	4.3856	8.0266	
November	4.7706	5.6969	5.3995	3.2155		
December	5.9922	5.9996	5.8724	6.1138	2.9813	8.9939
January	5.0623	3.5503	4.9995	5.7305	5.4528	5.5783
February	3.7453	2.9938	4.4678	2.5847	4.0471	4.6329
March	4.5956	4.4959	2.9012	5.4959		5.4892
April	5.2308	3.7915	5.1680	8.0250	4.2911	4.8787
May	4.9366	3.4089	4.0953	7.5644	5.8679	3.7465
June	4.6274	4.5248	4.0314	5.3260		
July	4.8265	4.4874	4.4642	5.1160		

The MAPE of Monday group have higher MAPE results than others. The reason of getting high MAPE is that the load demand of Sunday are included in the input dataset. Sunday's Load demand is significantly lower than weekday's load demand. Moreover, weekend also gives higher MAPE because the load demand of Friday are used as an input to forecast Saturday. Normally, Friday load demand variation is higher than weekend. As seen in Figure 3.9, there are different load variations. Weekends are similar to Monday. The following figure show how previous day load demand affect the forecasted load demand.

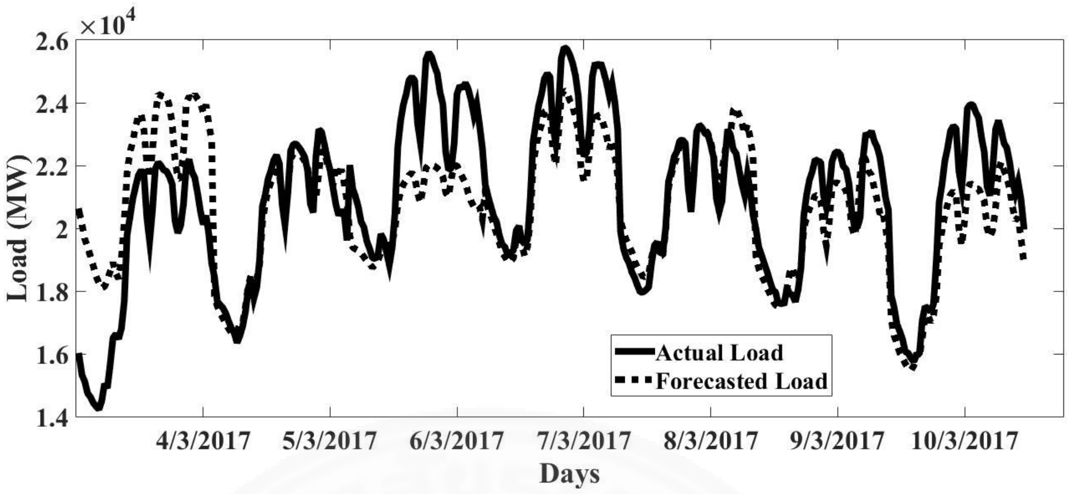


Figure 5.4 Different load patterns for one week

There is a high error in December. Since there are many holidays and many industries shut down their work for holidays in December, the demand of electricity is significantly decreased. One important factor is the temperature in December which is lower than the other month. Therefore, it causes the high MAPE. The following figure shows the MAPE of total average for each month from August 2016 to July 2017.

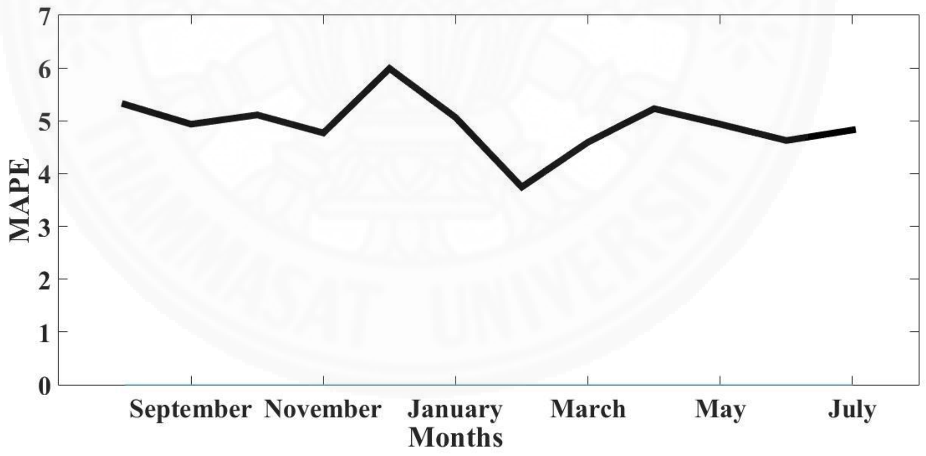


Figure 5.5 Total average MAPE from August 2016 to July 2017

5.4.2 Testing of SVR with shocking PSO

Similar to previous experiment, the performance of MAPE by SVR-PSO shocking is divided into 6 separate groups. The following figure shows the total average for 12 months which starts from August 2016 to July 2017.

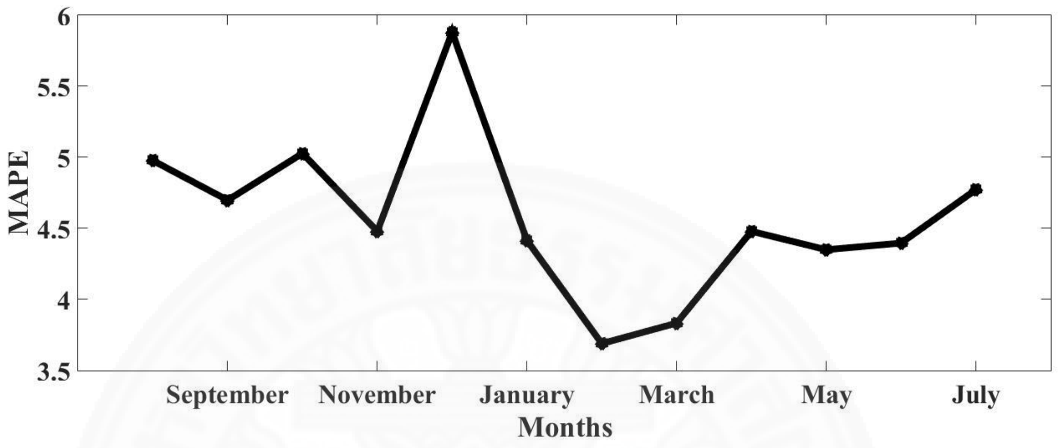


Figure 5.6 Total average's MAPE values for August 2016 to July 2017

As we can see in the above figure, December has the highest MAPE among the other months. The reason of getting highest MAPE is already discussed in above experiment 4. We discuss for each group in this experiment. The five groups are bridging holidays, holidays, weekends, weekdays and Monday. Bridging holidays and holidays are shown in one figure. There is no bridging holidays in August, September, October, November, June and July. The highest MAPE in bridging holidays is December. Similarly, the highest holidays' MAPE is December.

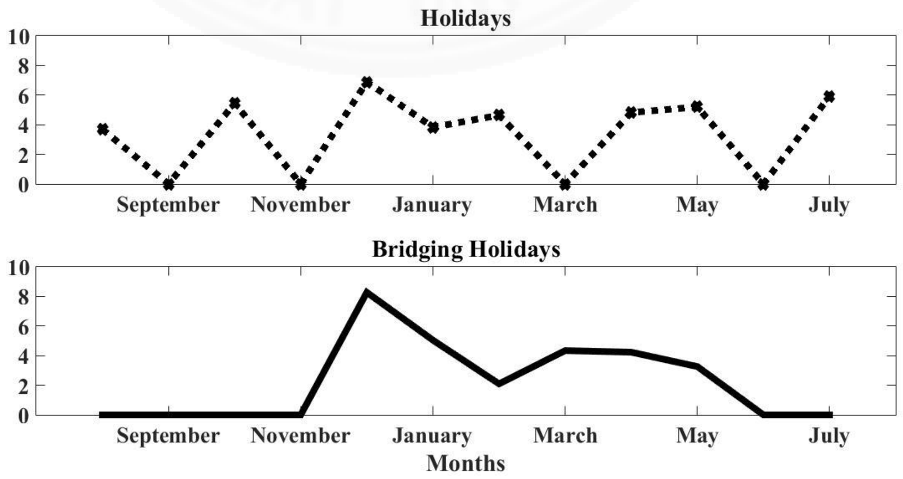


Figure 5.7 MAPE for Holidays and bridging holidays in each month

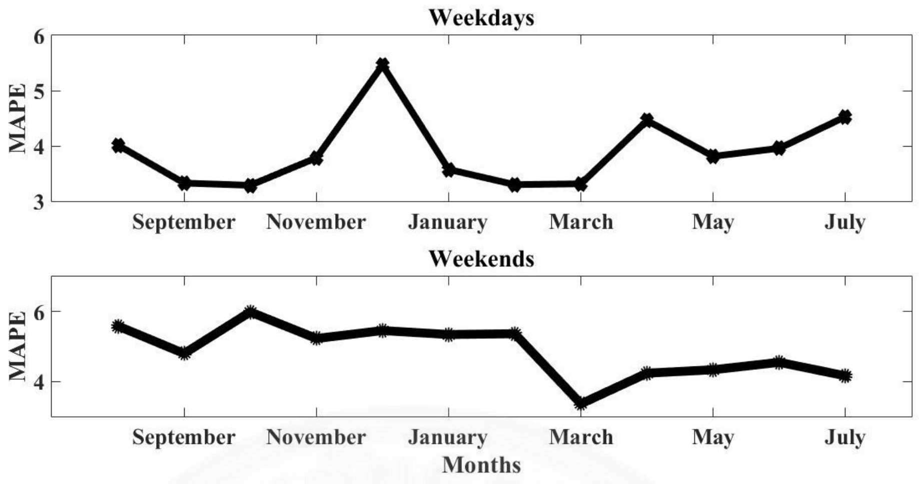


Figure 5.8 MAPE of weekdays and weekends for each month

According to above figure, the lowest MAPE for weekend is February and March. Moreover, the highest MAPE is still taken by December. For the other months, the pattern goes in stable. For Monday group, the MAPE is around 4 and 5 in almost every month except December which MAPE's value is 8.108.

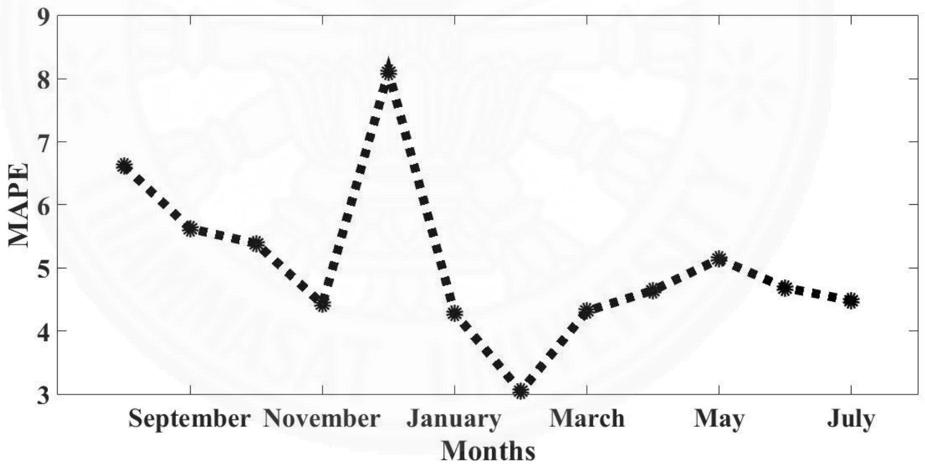


Figure 5.9 Monday's MAPE for each month

The load on Sunday are normally smooth and lower than the load on Monday. Since we use Sunday as an input to forecast Monday, the results of MAPE' on Monday is inefficient performance.

5.5 Experiment 5: Testing of SVR with GA

There are 6 groups to present the accuracy of GA-SVR according to the behavior of load patterns. The following table shows each group of MAPE from August 2016 to July 2017.

Table 5.4 MAPE performance by SVR-GA from August 2016 to July 2017

Month	Total Average	Weekday	Weekend	Monday	Holiday	Bridging Holiday
August	5.8173	7.1963	6.8751	6.6288	8.6112	
September	5.7009	4.7063	5.6515	6.7063		
October	5.3786	3.9410	5.7563	5.9270	5.8903	
November	4.7895	5.1202	5.0050	4.2433		
December	7.8917	7.5553	6.1177	11.6376	3.5478	10.6001
January	5.3257	4.6563	5.2795	4.5439	4.5059	7.6428
February	5.1191	4.7091	4.5318	5.2533	6.4027	4.6985
March	4.4513	3.8746	3.3008	4.8746		5.7552
April	5.2502	4.9845	4.9961	5.9507	5.6552	4.6643
May	5.6257	3.9013	4.1756	9.5387	6.7523	3.7605
June	4.7509	3.6744	4.9891	5.5890		
July	4.4238	3.9430	3.6926	5.0465	5.0130	

The performance of MAPE is good in November, March, June and July. Bridging holidays and Monday give the inefficient performance in December. The following figure shows the total average MAPE values for each month.

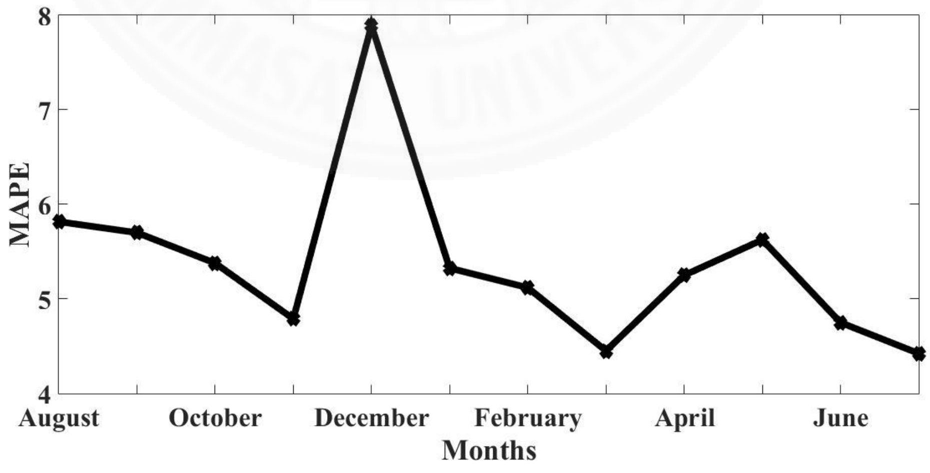


Figure 5.10 MAPE for total average of each month

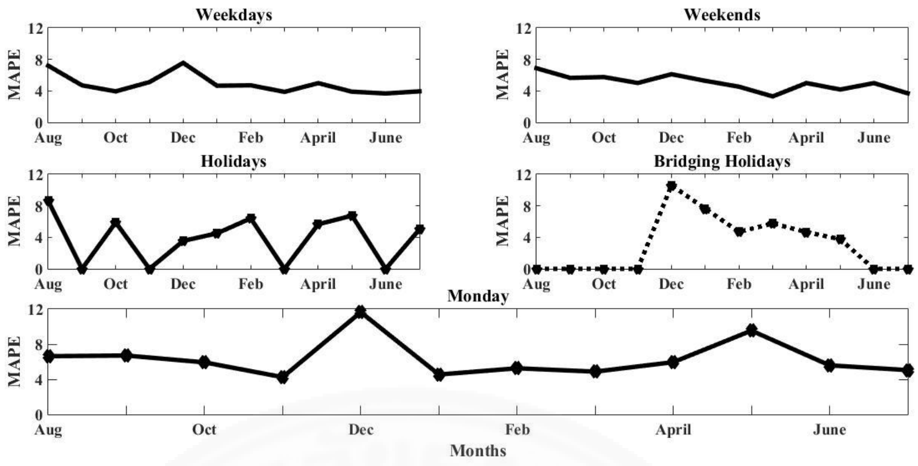


Figure 5.11 MAPE for different group From August 2016 to July 2017

The MAPE value of holidays in December is significantly low compared to other techniques. This is the reason of optimizing the hyper parameters to get good performance. According to SVR-GA model, it can perform well in holidays except August.

5.6 Experiment 5: Testing of SVR with BO

The last experiment for this study is testing of forecasting performance by SVR-BO. This experiment also separates into 6 groups. The following table shows the outcomes of forecasting from August 2016 to July 2017.

Table 5.5 MAPE performance by SVR-BO from August 2016 to July 2017

Month	Total Average	Weekday	Weekend	Monday	Holiday	Bridging Holiday
August	4.4876	3.5790	5.3568	5.0840	3.4077	
September	4.3493	3.3845	3.7368	5.3843		
October	4.7384	3.0186	5.8153	4.4208	5.6989	
November	4.2532	3.9799	5.1027	3.6771		
December	4.9787	7.6322	6.3228	3.3340	4.2707	3.3340
January	4.1184	4.1396	4.4564	3.6448	5.9356	2.4156
February	3.6578	3.3391	3.9201	3.8676	4.2257	2.9368
March	2.7845	1.7136	3.0096	2.7136		3.7011
April	4.2065	3.2541	4.7491	4.7692	4.3522	3.9077
May	2.9442	4.0572	2.4589	4.0918	1.9740	2.1391
June	4.2709	3.0231	5.1051	4.6844		
July	4.4238	3.9430	3.6926	5.0465	5.0130	

The following 4 figures show the MAPE performance of each month from August 2016 to July 2017. Although the performance of BO is better than the other techniques, the MAPE in December is still high. The following subplot shows the MAPE of weekdays, weekends, holidays and bridging holidays.

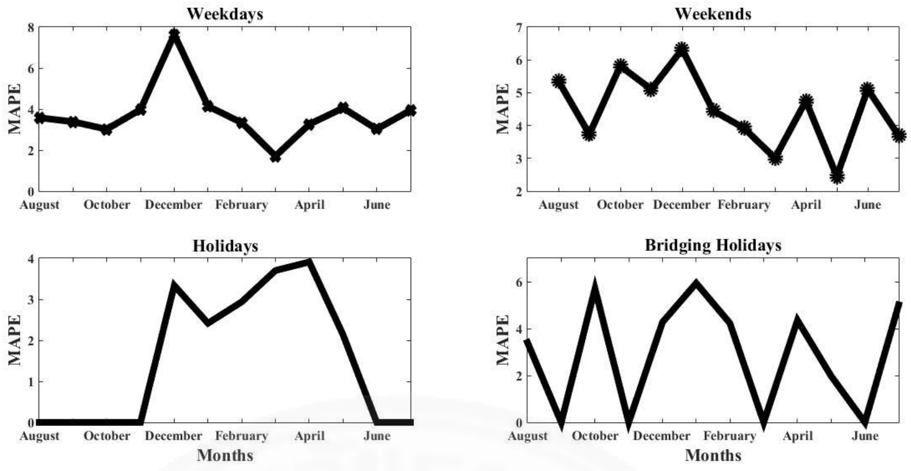


Figure 5.12 MAPE outcomes for 4 groups from August 2016 to July 2017

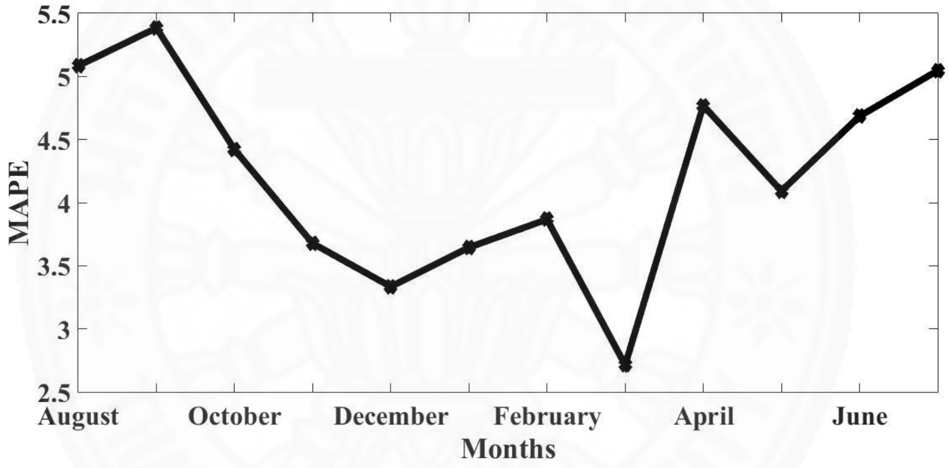


Figure 5.13 Monday’s MAPE for each month

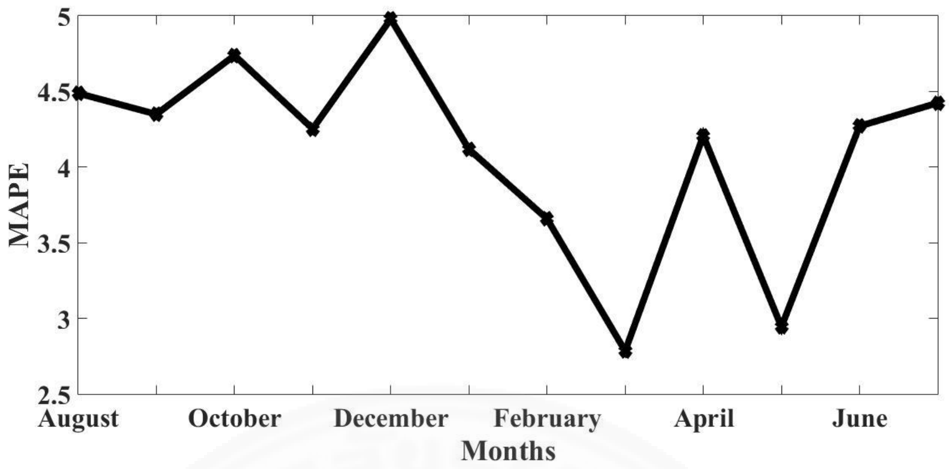


Figure 5.14 Total average's MAPE for August 2016 to July 2017

The following figure shows the best forecasting performance of BO-SVR model which MAPE is 1.2465 in May 23, 2016. It can be seen that the forecasted load demand is almost the same with the actual demand in the first subplot. The error between actual load demand and forecasted load demand of BO-PSO are also shown in second subplot of Figure 5.11. The error is calculated by subtracting from the actual load demand to forecasted load demand. As we can see that the highest errors are at 12:30 pm and 4:30 pm with approximately 900 MW. Normally, the load demand at 12:30 pm and 4:30 pm are lower because it is the time for lunch and the way to back home. So, the error is big because the forecasted load demand is higher than normal one at those times.

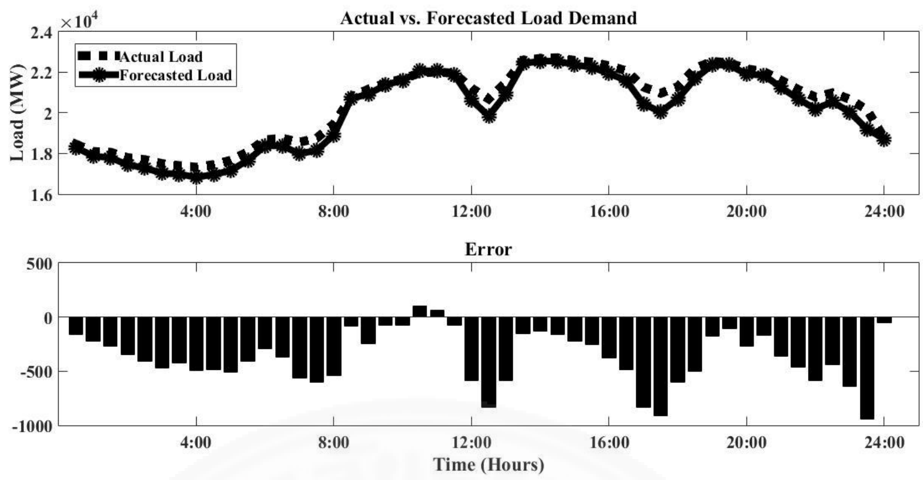


Figure 5.15 Actual vs. forecasted load and error between actual and forecasted load for May 23, 2016

5.7 Testing on different optimization methods

The experiment is conducted by three different optimization techniques. Moreover, PSO is tested by two methods. The first one is traditional PSO. The second one is shocking PSO. The following figure shows the improvement of local minima.

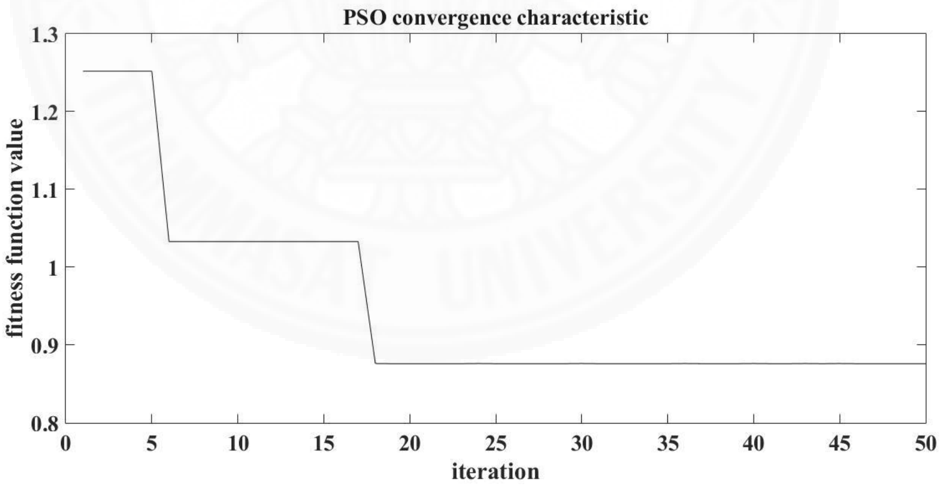


Figure 5.16 PSO convergence characteristic

The reason that the second PSO performs effectively, it supports to decrease the training error (RMSE) when trapping in the local minima. According to experiment 4, 5 and 6, we test SVR combining with different optimization techniques. As we can see clearly that SVR-BO is the best one among the other algorithms. The following table shows the total average value of five algorithms.

Table 5.6 MAPE of total average values for five algorithms

Month	SVR	SVR-GA	SVR-PSO (Traditional)	SVR-PSO (Shock)	SVR-BO
August	8.1392	5.8173	5.3120	4.9776	4.4876
September	6.7606	5.7009	4.9365	4.6987	4.3493
October	7.5300	5.3786	5.1122	5.0282	4.7384
November	6.5723	4.7895	4.7706	4.4799	4.2532
December	8.7238	7.8917	5.9922	5.8801	4.9787
January	6.4701	5.3257	5.0623	4.4154	4.1184
February	6.4467	5.1191	3.7453	3.6898	3.6578
March	5.9400	4.4513	4.5956	3.8337	2.7845
April	6.2387	5.2502	5.2308	4.4799	4.2065
May	7.4799	5.6257	4.9366	4.3503	2.9442
June	7.2489	4.7509	4.6274	4.3969	4.2709
July	7.4900	4.4238	4.8265	4.7730	4.4238
Yearly Average	7.0867	5.3771	4.9290	4.5836	4.1011

.

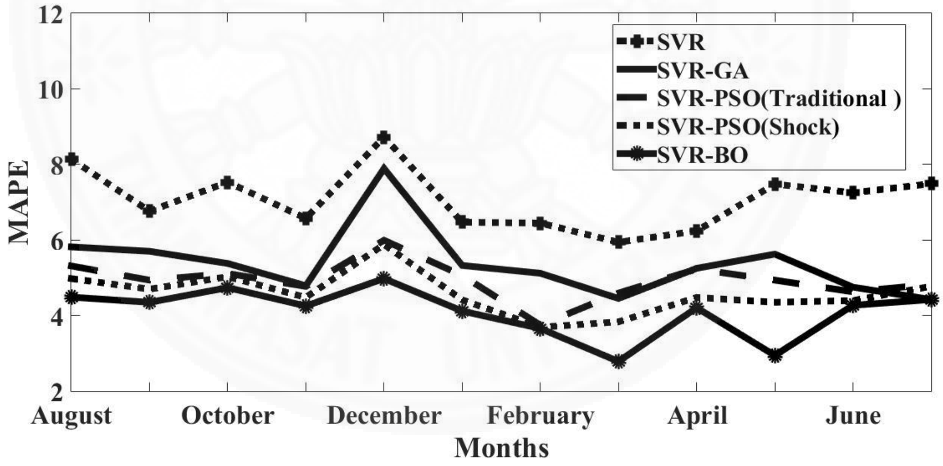


Figure 5.17 Comparison of total average load’s MAPE on Different optimization algorithms

The MAPE of March is good in all algorithms because the load demand in February is stable compared to other months. So, the model see the good pattern. The MAPE of December is high in every optimization algorithms even though we optimize hyper parameters of SVR. The following figure shows that December has different load pattern with other months.

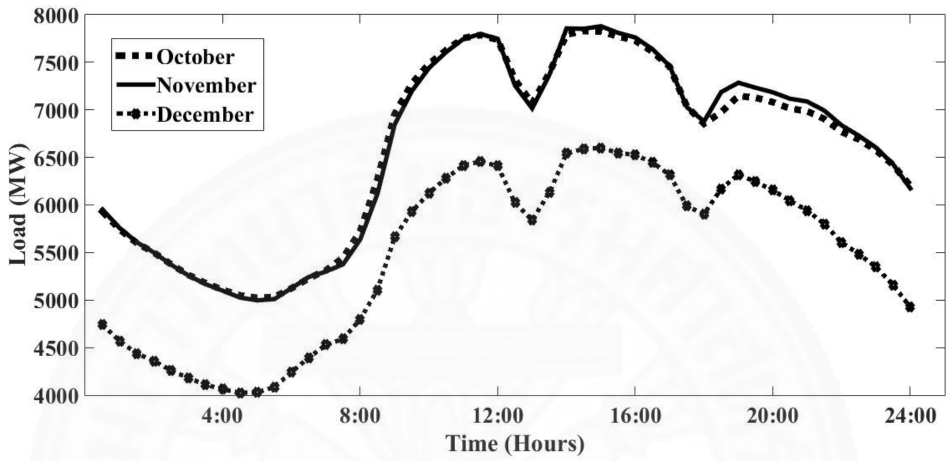


Figure 5.18 Average load pattern for October, November, and December

Moreover, the forecasting accuracy is still high that the MAPE values are over 4 and 5. One reason is that the data filtering method. There are still unsmooth data in the band as shown in figure.

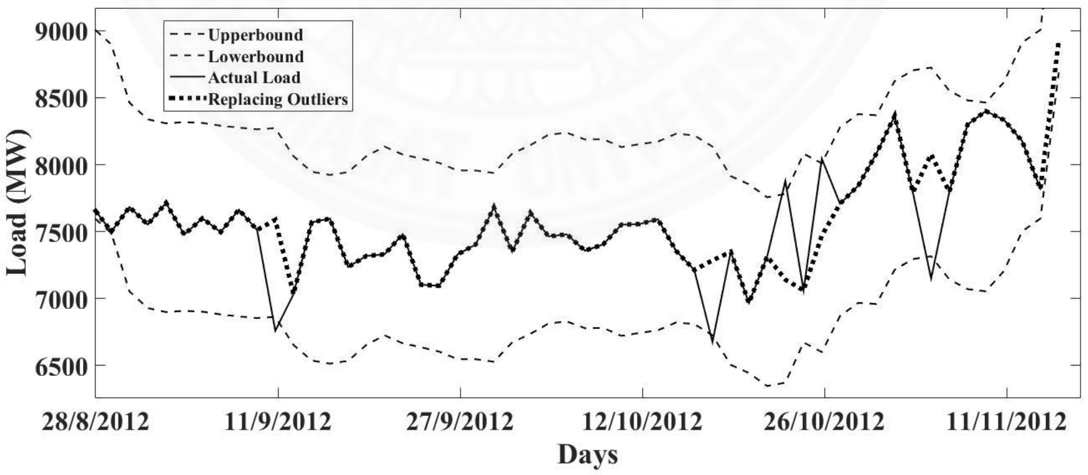


Figure 5.19 Replacing outliers of the load demand with the filtering band

CHAPTER 6
CONCLUSION AND RECOMMANDATION

This research aims to forecast the short term load forecasting by hybrid support vector regression model. The historical data from 2009 to 2017 which are collected from Electricity Generating Authority of Thailand (EGAT) are used for this research.

There are 6 experiments that to test the data to get better outcomes. First, we test the amount of training data. The more we train, the better results we get. Then, we test on different kernel functions which is important in constructing the model. RBF give the best performance since it performs well in non-linear problem. The training inputs play an important role in modelling. Therefore, we compare the results by using different inputs. The result which includes dummy values is better than without including dummy values. Later, we combine the SVR with different optimization model which are PSO, GA and BO. We test two styles for PSO. To improve from trapping local minima, we introduce shocking PSO. The performance of shocking PSO is better than the traditional PSO. Among all three optimization algorithms, SVR-BO give the best outcomes.

Moreover, the MAPE of December is still high, even though we optimize hyper parameters of SVR. The result for December would improve by training the load of December only. Then, the forecasting accuracy of the data from August, 2016 to July, 2017 is inefficient. Accordingly, future research can change the training inputs and modify the filtering techniques.

References

Chen, K.-Y., & Wang, C.-H. (2007). Support vector regression with genetic algorithms in forecasting tourism demand. *Tourism Management, 28*(1), 215-226.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). *Support vector regression machines.* Paper presented at the Advances in neural information processing systems.

Duan, P., Xie, K., Guo, T., & Huang, X. (2011). Short-term load forecasting for electric power systems using the PSO-SVR and FCM clustering techniques. *Energies, 4*(1), 173-184.

Eberhart, R., & Kennedy, J. (1995). *A new optimizer using particle swarm theory.* Paper presented at the Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on.

Gao, J. B., Gunn, S. R., Harris, C. J., & Brown, M. (2002). A probabilistic framework for SVM regression and error bar estimation. *Machine Learning, 46*(1-3), 71-89.

Geng, J., Huang, M.-L., Li, M.-W., & Hong, W.-C. (2015). Hybridization of seasonal chaotic cloud simulated annealing algorithm in a SVR-based load forecasting model. *Neurocomputing, 151*, 1362-1373.

Gopi, G., Dauwels, J., Asif, M. T., Ashwin, S., Mitrovic, N., Rasheed, U., & Jaillet, P. (2013). *Bayesian support vector regression for traffic speed prediction with error bars.* Paper presented at the Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on.

Goyal, P. (2014). MATLAB for Machine Learning.

Gross, G., & Galiana, F. D. (1987). Short-term load forecasting. *Proceedings of the IEEE, 75*(12), 1558-1573.

Hippert, H. S., Pedreira, C. E., & Souza, R. C. (2001). Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on Power Systems, 16*(1), 44-55.

Ho, K.-L., Hsu, Y.-Y., Chen, C.-F., Lee, T.-E., Liang, C.-C., Lai, T.-S., & Chen, K.-K. (1990). Short term load forecasting of Taiwan power system using a knowledge-based expert system. *IEEE Transactions on Power Systems, 5*(4), 1214-1221.

Hong, W.-C. (2009a). Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model. *Energy Conversion and Management, 50*(1), 105-117.

Hong, W.-C. (2009b). Hybrid evolutionary algorithms in a SVR-based electric load forecasting model. *International Journal of Electrical Power & Energy Systems, 31*(7), 409-417.

Hong, W.-C., Dong, Y., Zhang, W. Y., Chen, L.-Y., & Panigrahi, B. (2013). Cyclic electric load forecasting by seasonal SVR with chaotic genetic algorithm. *International Journal of Electrical Power & Energy Systems, 44*(1), 604-614.

Jiang, F., & Wu, W. Hybrid Particle Swarm Optimization and Support Vector Regression Performance in Exchange Rate Prediction.

Klein, A., Falkner, S., Bartels, S., Hennig, P., & Hutter, F. (2016). Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079.*

Law, M. H., & Kwok, J. T.-Y. (2001). *Bayesian Support Vector Regression.* Paper presented at the AISTATS.

Law, T., & Shawe-Taylor, J. (2017). Practical Bayesian support vector regression for financial time series prediction and market condition change detection. *Quantitative Finance*, 1-14.

Lee, K., Cha, Y., & Park, J. (1992). Short-term load forecasting using an artificial neural network. *IEEE Transactions on Power Systems, 7*(1), 124-132.

Lin, S.-W., Ying, K.-C., Chen, S.-C., & Lee, Z.-J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications, 35*(4), 1817-1824.

Liu, Y., Wang, G., Chen, H., Dong, H., Zhu, X., & Wang, S. (2011). An improved particle swarm optimization for feature selection. *Journal of Bionic Engineering, 8*(2), 191-200.

Maia, C., & Gonçalves, M. (2009). A methodology for short-term electric load forecasting based on specialized recursive digital filters. *Computers & Industrial Engineering, 57*(3), 724-731.

Martinez-Cantin, R. (2014). Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research, 15*(1), 3735-3739.

Mbarek, M. B., & Feki, R. (2016). Using fuzzy logic to renewable energy forecasting: a case study of France. *International Journal of Energy Technology and Policy, 12*(4), 357-376.

Pai, P.-F., & Hong, W.-C. (2005). Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. *Electric Power Systems Research, 74*(3), 417-425.

Papalexopoulos, A. D., & Hesterberg, T. C. (1990). A regression-based approach to short-term system load forecasting. *IEEE Transactions on Power Systems, 5*(4), 1535-1547.

Saini, L. M., Aggarwal, S. K., & Kumar, A. (2010). Parameter optimisation using genetic algorithm for support vector machine-based price-forecasting model in National electricity market. *IET generation, transmission & distribution, 4*(1), 36-49.

Scholkopf, B., & Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*: MIT press.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing, 14*(3), 199-222.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). *Practical bayesian optimization of machine learning algorithms.* Paper presented at the Advances in neural information processing systems.

Türkay, B. E., & Demren, D. (2011). *Electrical load forecasting using support vector machines.* Paper presented at the Electrical and Electronics Engineering (ELECO), 2011 7th International Conference on.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks, 10*(5), 988-999.

Vemuri, S., Hill, D., & Balasubramanian, R. (1973). *Load forecasting using stochastic models.* Paper presented at the Proceeding of the 8th power industrial computing application conference.

Wu, C.-H., Tzeng, G.-H., & Lin, R.-H. (2009). A Novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert systems with applications*, *36*(3), 4725-4735.

Xiao, L., Wang, J., Yang, X., & Xiao, L. (2015). A hybrid model based on data preprocessing for electrical power forecasting. *International Journal of Electrical Power & Energy Systems, 64*, 311-327.

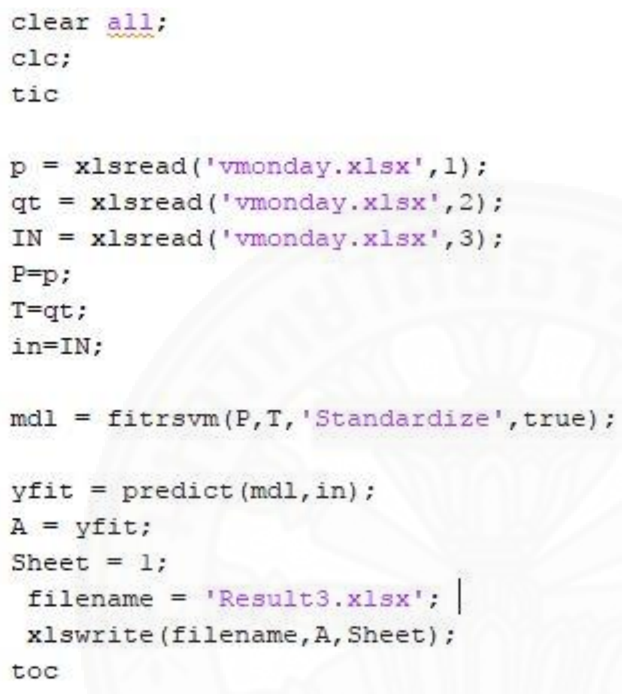
Zhang, W. Y., Hong, W.-C., Dong, Y., Tsai, G., Sung, J.-T., & Fan, G.-f. (2012). Application of SVR with chaotic GASA algorithm in cyclic electric load forecasting. *Energy, 45*(1), 850-858.

Appendices

Appendix A

MATLAB code: Training SVR

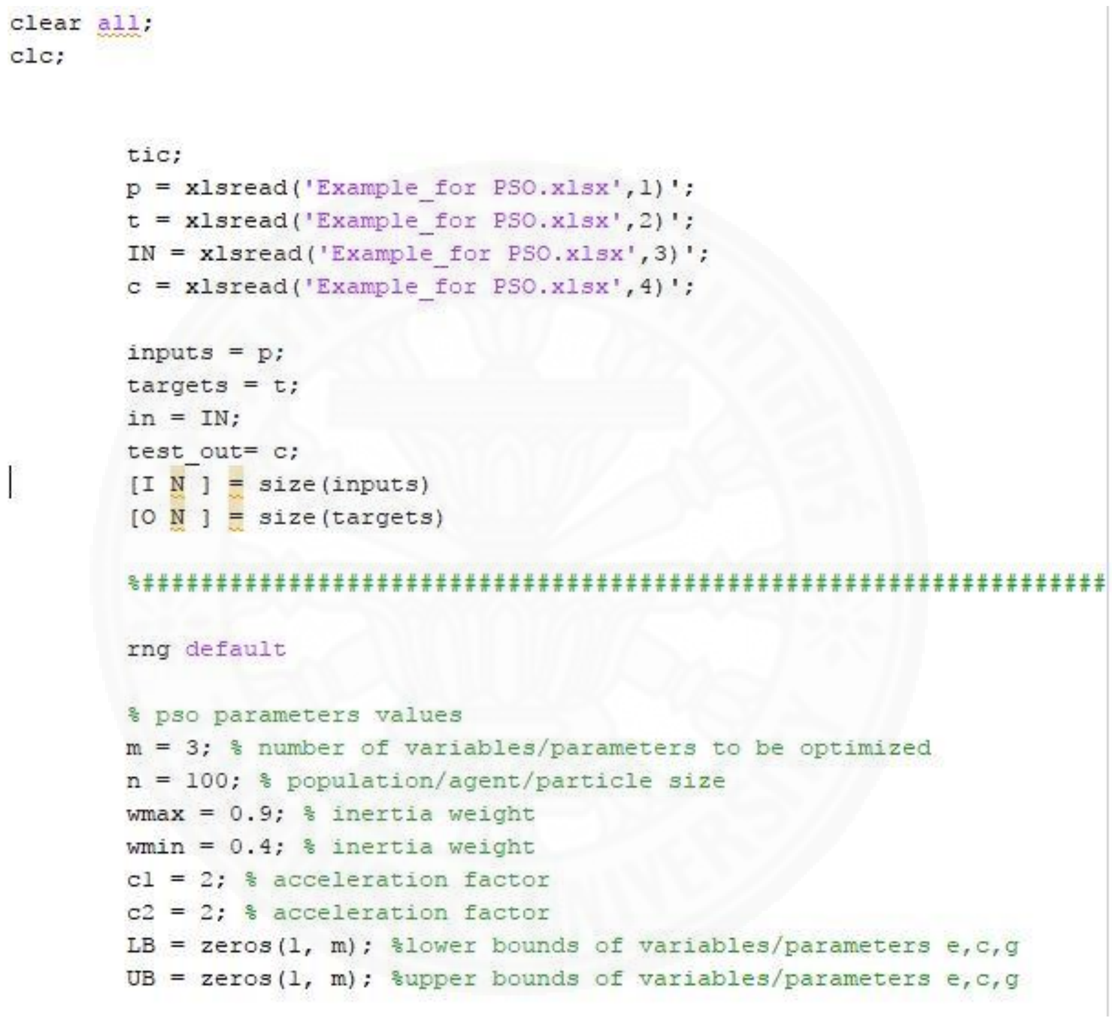
Constructing the SVR, Putting Data, Noting Outputs



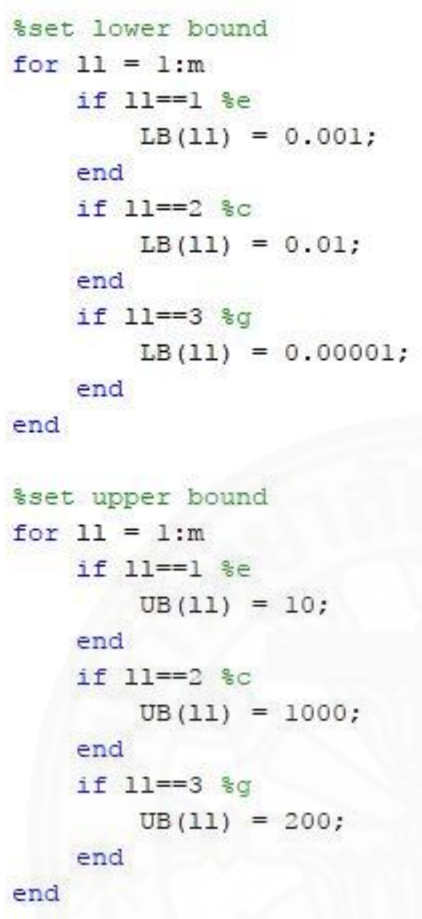
```
clear all;
clc;
tic

p = xlsread('vmonday.xlsx',1);
qt = xlsread('vmonday.xlsx',2);
IN = xlsread('vmonday.xlsx',3);
P=p;
T=qt;
in=IN;

mdl = fitrsvm(P,T,'Standardize',true);

yfit = predict(mdl,in);
A = yfit;
Sheet = 1;
 filename = 'Result3.xlsx';
 xlswrite(filename,A,Sheet);
toc
```

Appendix B
MATLAB code: Training SVR with traditional PSO

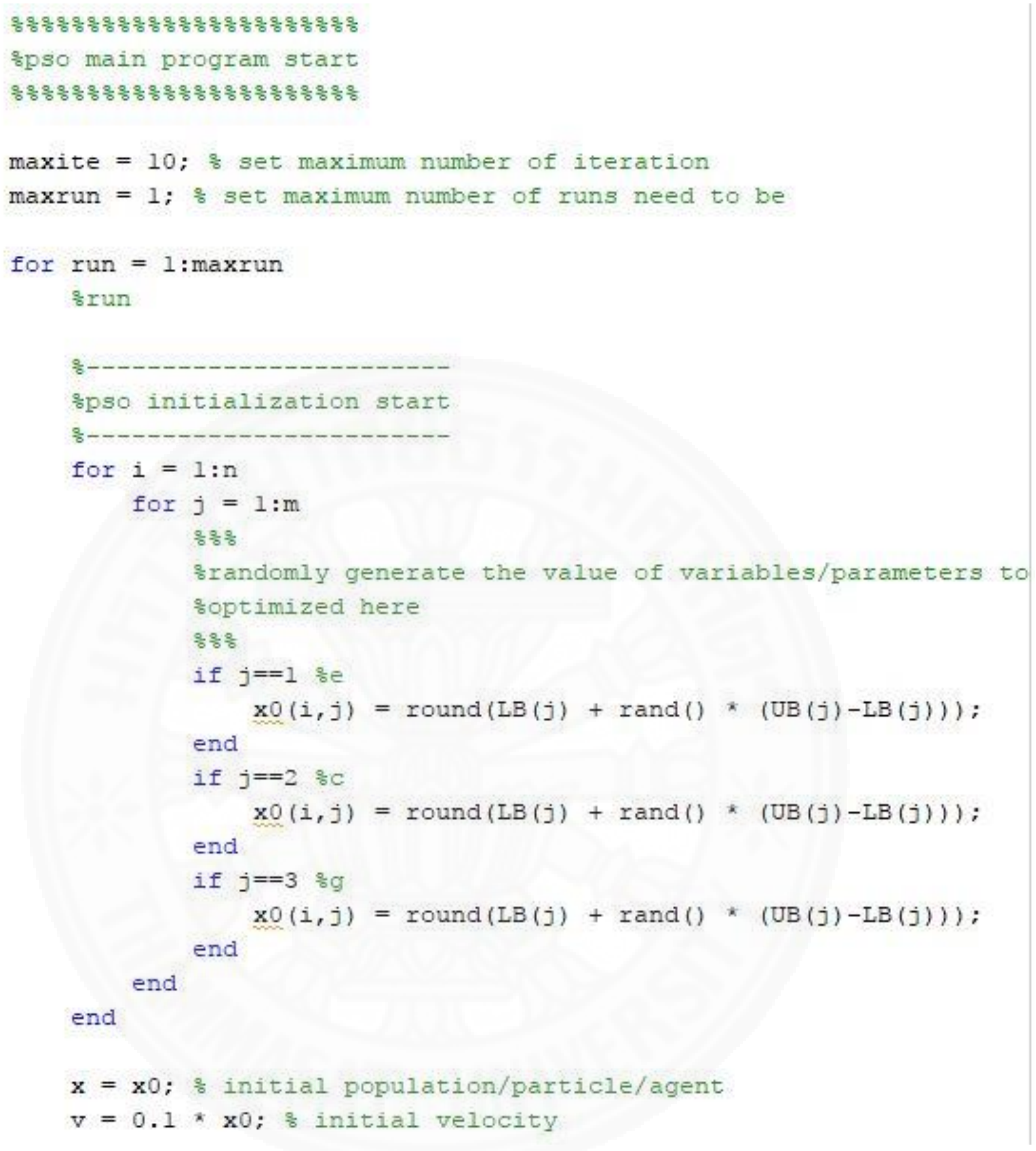
Constructing the SVR-PSO, Putting Data, Noting Outputs



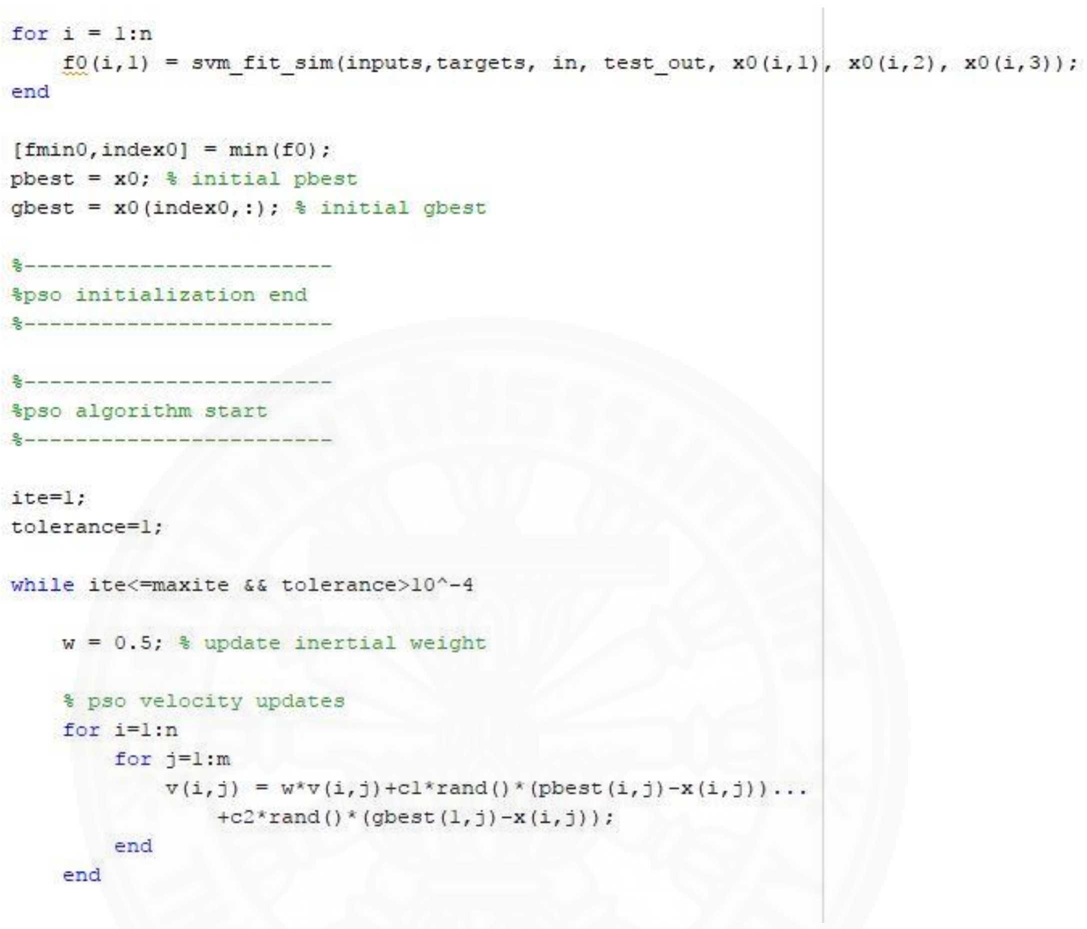
```
clear all;
clc;


        tic;
        p = xlsread('Example_for PSO.xlsx',1)';
        t = xlsread('Example_for PSO.xlsx',2)';
        IN = xlsread('Example_for PSO.xlsx',3)';
        c = xlsread('Example_for PSO.xlsx',4)';

        inputs = p;
        targets = t;
        in = IN;
        test_out= c;
        [I N ] = size(inputs)
        [O N ] = size(targets)

        %##################################################################

        rng default

        % pso parameters values
        m = 3; % number of variables/parameters to be optimized
        n = 100; % population/agent/particle size
        wmax = 0.9; % inertia weight
        wmin = 0.4; % inertia weight
        c1 = 2; % acceleration factor
        c2 = 2; % acceleration factor
        LB = zeros(1, m); %lower bounds of variables/parameters e,c,g
        UB = zeros(1, m); %upper bounds of variables/parameters e,c,g
```



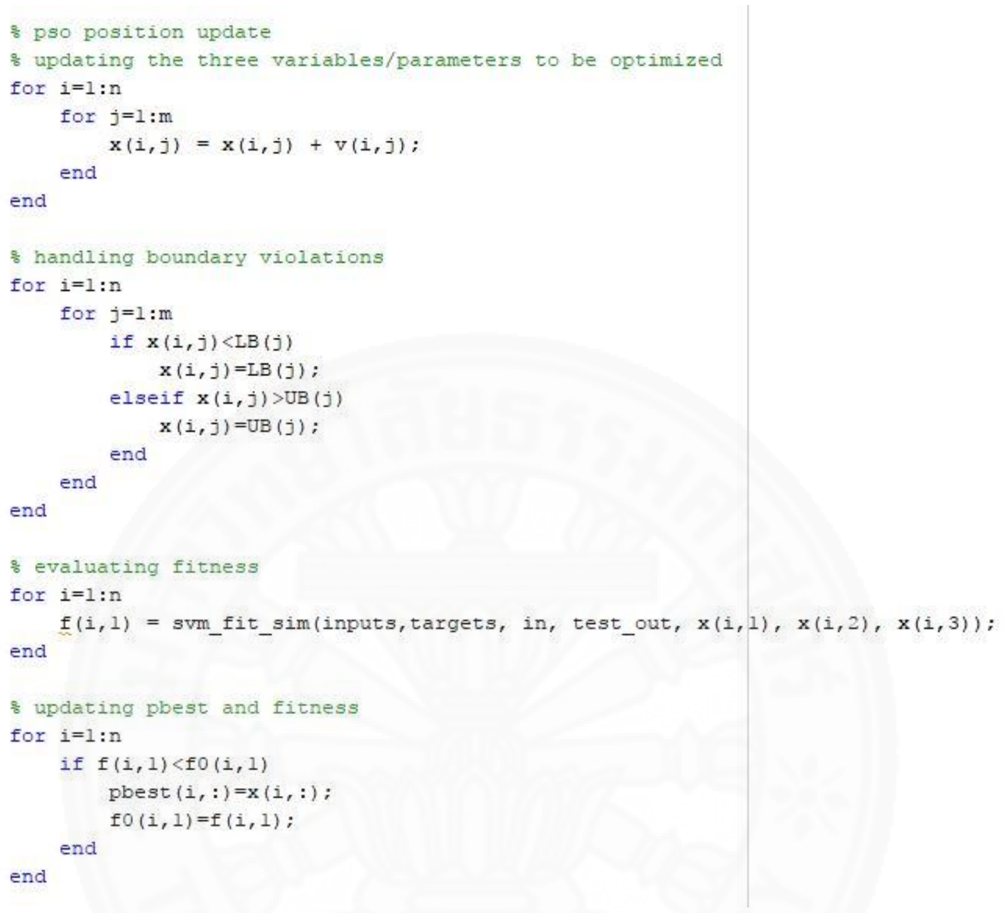
```
%set lower bound
for ll = 1:m
    if ll==1 %e
        LB(ll) = 0.001;
    end
    if ll==2 %c
        LB(ll) = 0.01;
    end
    if ll==3 %g
        LB(ll) = 0.00001;
    end
end

%set upper bound
for ll = 1:m
    if ll==1 %e
        UB(ll) = 10;
    end
    if ll==2 %c
        UB(ll) = 1000;
    end
    if ll==3 %g
        UB(ll) = 200;
    end
end
```



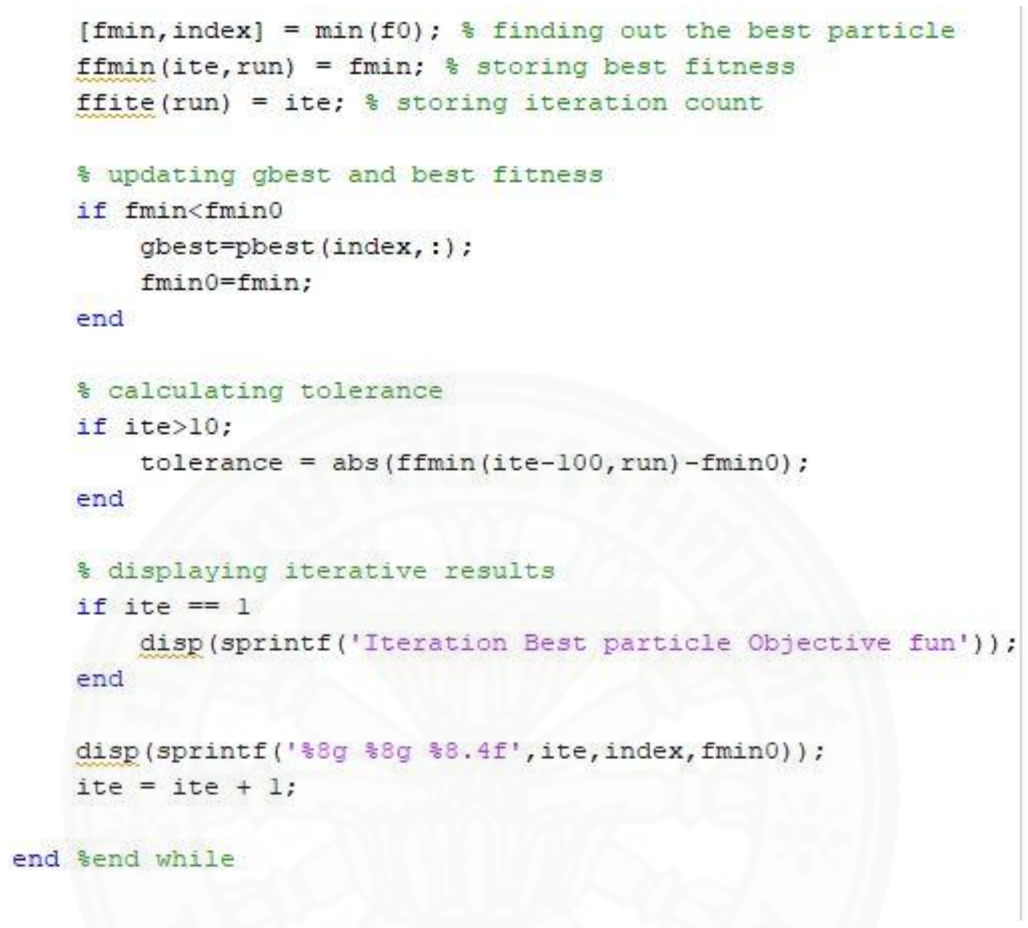
```
%%%%%%%%%%%%%%%%%%%%%%%
%pso main program start
%%%%%%%%%%%%%%%%%%%%%%%

maxite = 10; % set maximum number of iteration
maxrun = 1; % set maximum number of runs need to be

for run = 1:maxrun
    %run

    %------------------------
    %pso initialization start
    %------------------------
    for i = 1:n
        for j = 1:m
            %%%
            %randomly generate the value of variables/parameters to
            %optimized here
            %%%
            if j==1 %e
                x0(i,j) = round(LB(j) + rand() * (UB(j)-LB(j)));
            end
            if j==2 %c
                x0(i,j) = round(LB(j) + rand() * (UB(j)-LB(j)));
            end
            if j==3 %g
                x0(i,j) = round(LB(j) + rand() * (UB(j)-LB(j)));
            end
        end
    end

    x = x0; % initial population/particle/agent
    v = 0.1 * x0; % initial velocity
```



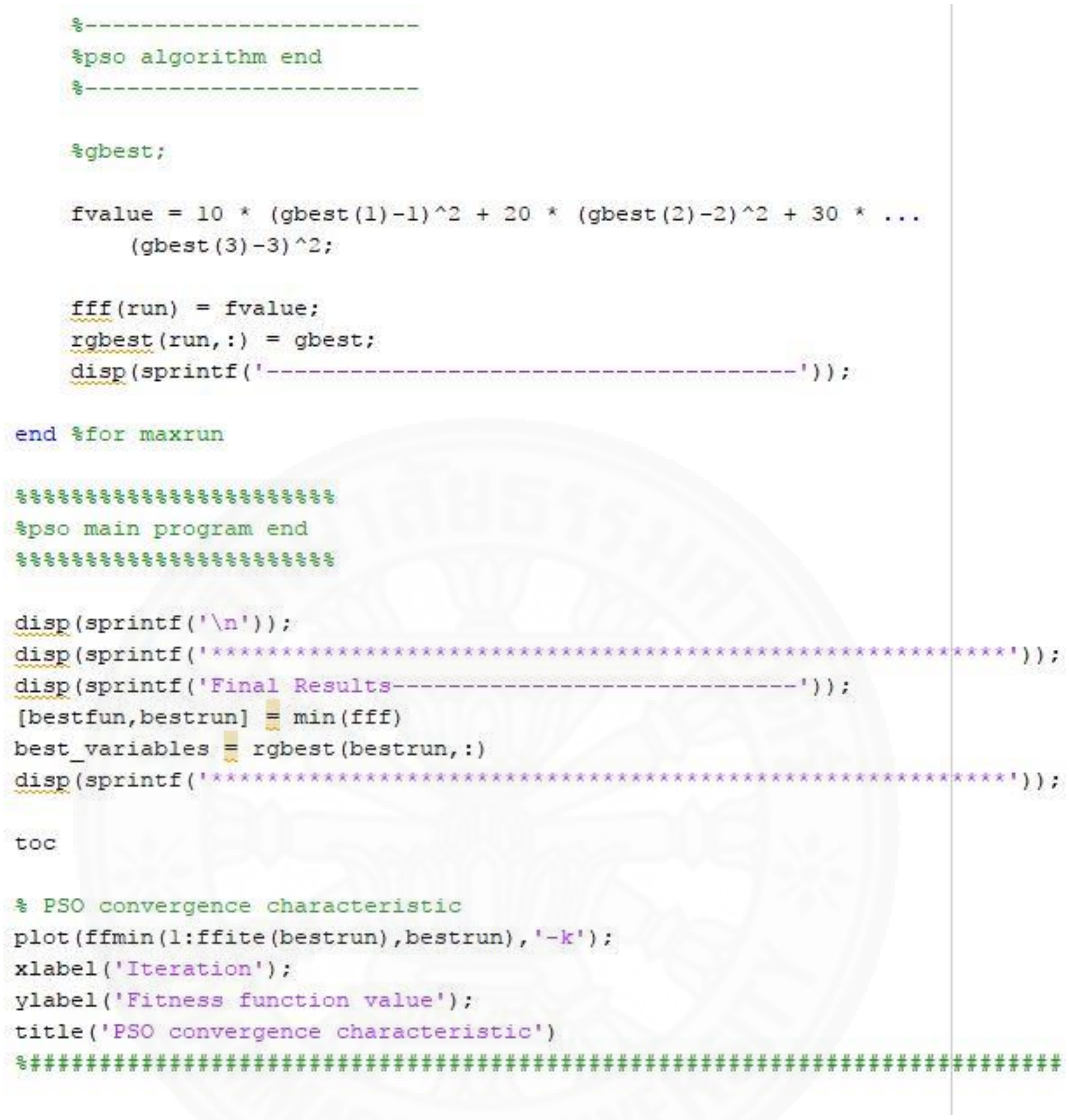
```
for i = 1:n
    f0(i,1) = svm_fit_sim(inputs,targets, in, test_out, x0(i,1), x0(i,2), x0(i,3));
end

[fmin0,index0] = min(f0);
pbest = x0; % initial pbest
gbest = x0(index0,:); % initial gbest

%------------------------
%pso initialization end
%------------------------

%------------------------
%pso algorithm start
%------------------------

ite=1;
tolerance=1;

while ite<=maxite && tolerance>10^-4

    w = 0.5; % update inertial weight

    % pso velocity updates
    for i=1:n
        for j=1:m
            v(i,j) = w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))...
                +c2*rand()*(gbest(1,j)-x(i,j));
        end
    end
```



```
% pso position update
% updating the three variables/parameters to be optimized
for i=1:n
    for j=1:m
        x(i,j) = x(i,j) + v(i,j);
    end
end

% handling boundary violations
for i=1:n
    for j=1:m
        if x(i,j)<LB(j)
            x(i,j)=LB(j);
        elseif x(i,j)>UB(j)
            x(i,j)=UB(j);
        end
    end
end

% evaluating fitness
for i=1:n
    f(i,1) = svm_fit_sim(inputs,targets, in, test_out, x(i,1), x(i,2), x(i,3));
end

% updating pbest and fitness
for i=1:n
    if f(i,1)<f0(i,1)
        pbest(i,:)=x(i,:);
        f0(i,1)=f(i,1);
    end
end
```



```
    [fmin,index] = min(f0); % finding out the best particle
    ffmin(ite,run) = fmin; % storing best fitness
    ffite(run) = ite; % storing iteration count

    % updating gbest and best fitness
    if fmin<fmin0
        gbest=pbest(index,:);
        fmin0=fmin;
    end

    % calculating tolerance
    if ite>10;
        tolerance = abs(ffmin(ite-100,run)-fmin0);
    end

    % displaying iterative results
    if ite == 1
        disp(sprintf('Iteration Best particle Objective fun'));
    end

    disp(sprintf('%8g %8g %8.4f',ite,index,fmin0));
    ite = ite + 1;

end %end while
```

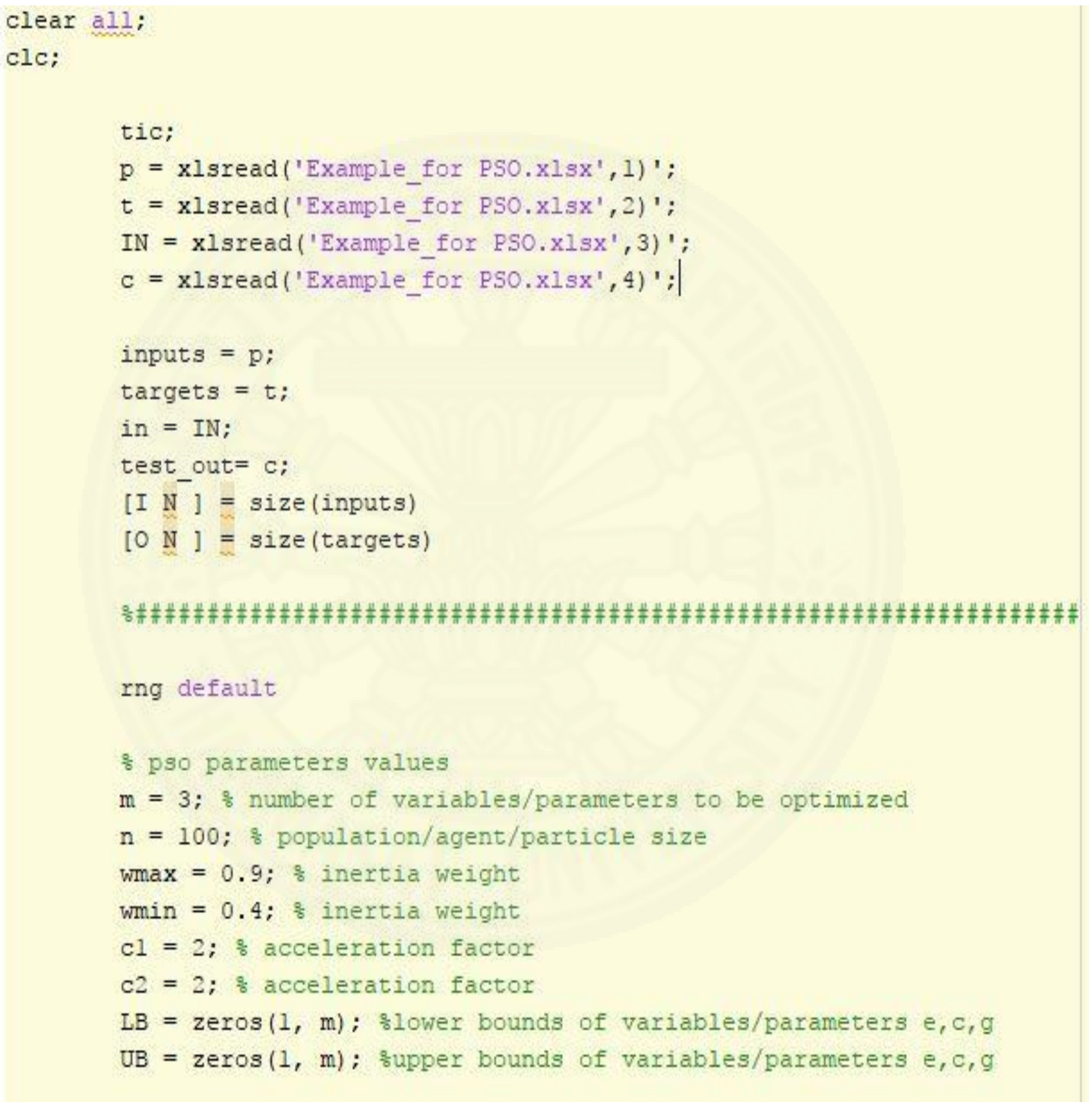


```
    %------------------------
    %pso algorithm end
    %------------------------

    %gbest;

    fvalue = 10 * (gbest(1)-1)^2 + 20 * (gbest(2)-2)^2 + 30 * ...
        (gbest(3)-3)^2;

    fff(run) = fvalue;
    rgbest(run,:) = gbest;
    disp(sprintf('-------------------------------------'));

end %for maxrun

%%%%%%%%%%%%%%%%%%%%%%%%
%pso main program end
%%%%%%%%%%%%%%%%%%%%%%%%

disp(sprintf('\n'));
disp(sprintf('***************************************************************'));
disp(sprintf('Final Results-----------------------------'));
[bestfun,bestrun] = min(fff)
best_variables = rgbest(bestrun,:)
disp(sprintf('***************************************************************'));

toc

% PSO convergence characteristic
plot(ffmin(1:ffite(bestrun),bestrun),'-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
%#####################################################################
```

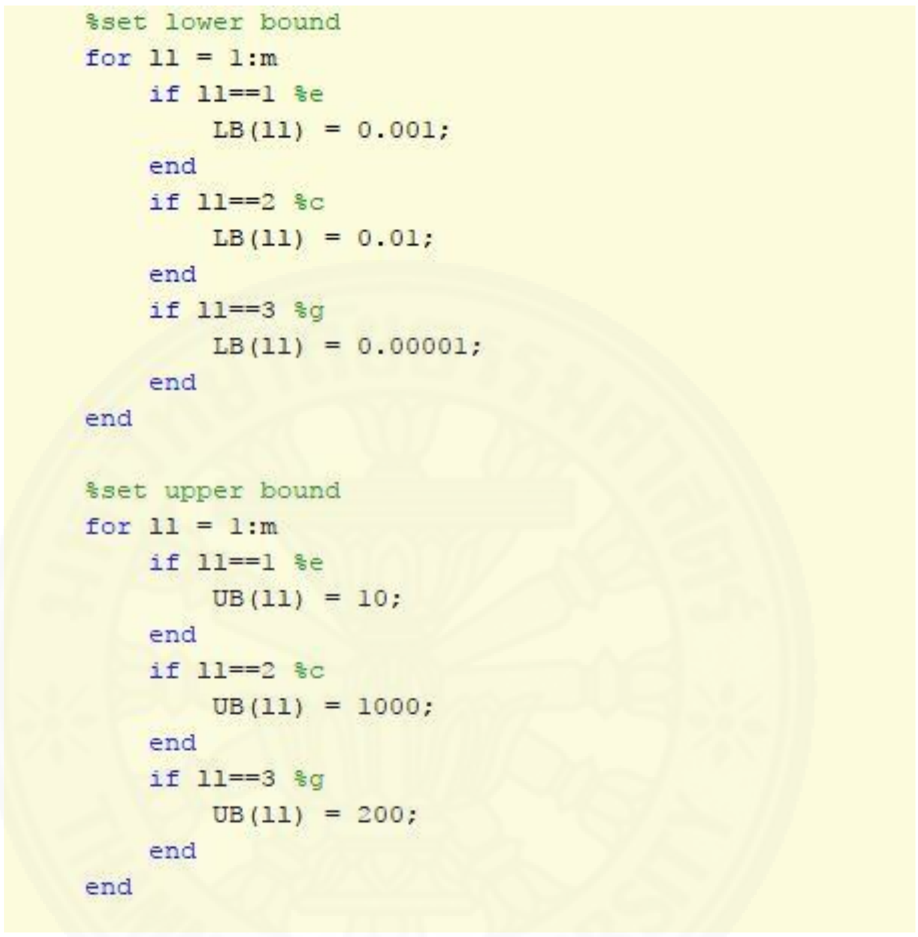
Appendix C

MATLAB code: Training SVR with shocking PSO

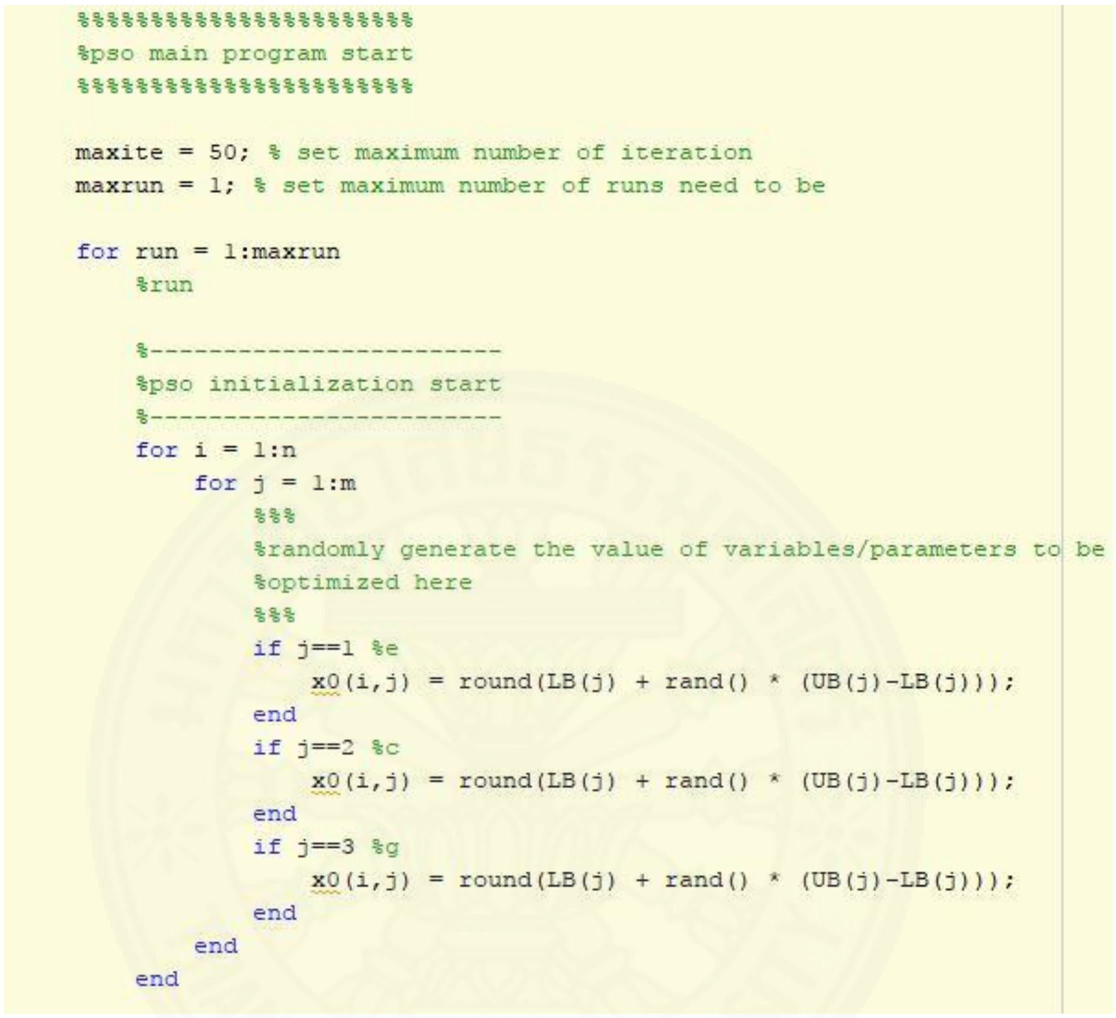
Constructing the SVR-PSO, Putting Data, Noting Outputs



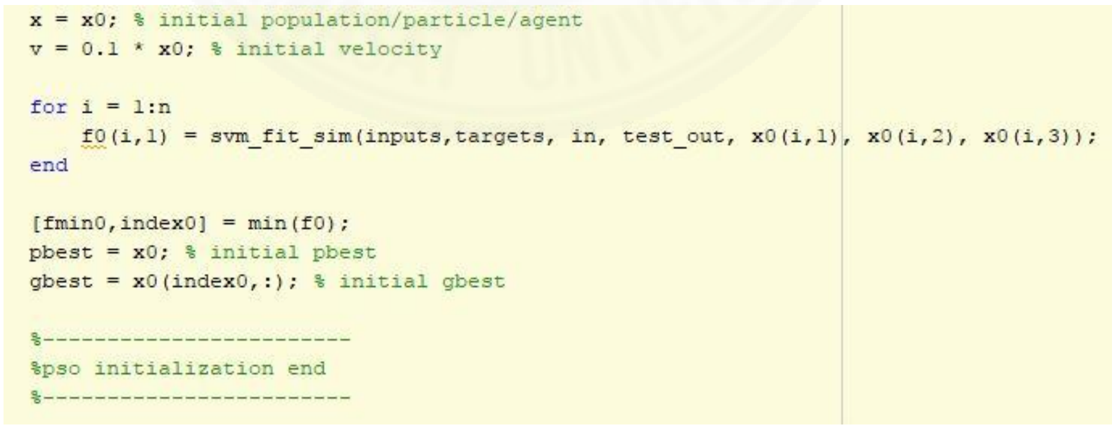
```
clear all;
clc;

        tic;
        p = xlsread('Example_for PSO.xlsx',1)';
        t = xlsread('Example_for PSO.xlsx',2)';
        IN = xlsread('Example_for PSO.xlsx',3)';
        c = xlsread('Example_for PSO.xlsx',4)';

        inputs = p;
        targets = t;
        in = IN;
        test_out= c;
        [I N ] = size(inputs)
        [O N ] = size(targets)

        %##################################################################

        rng default

        % pso parameters values
        m = 3; % number of variables/parameters to be optimized
        n = 100; % population/agent/particle size
        wmax = 0.9; % inertia weight
        wmin = 0.4; % inertia weight
        c1 = 2; % acceleration factor
        c2 = 2; % acceleration factor
        LB = zeros(1, m); %lower bounds of variables/parameters e,c,g
        UB = zeros(1, m); %upper bounds of variables/parameters e,c,g
```



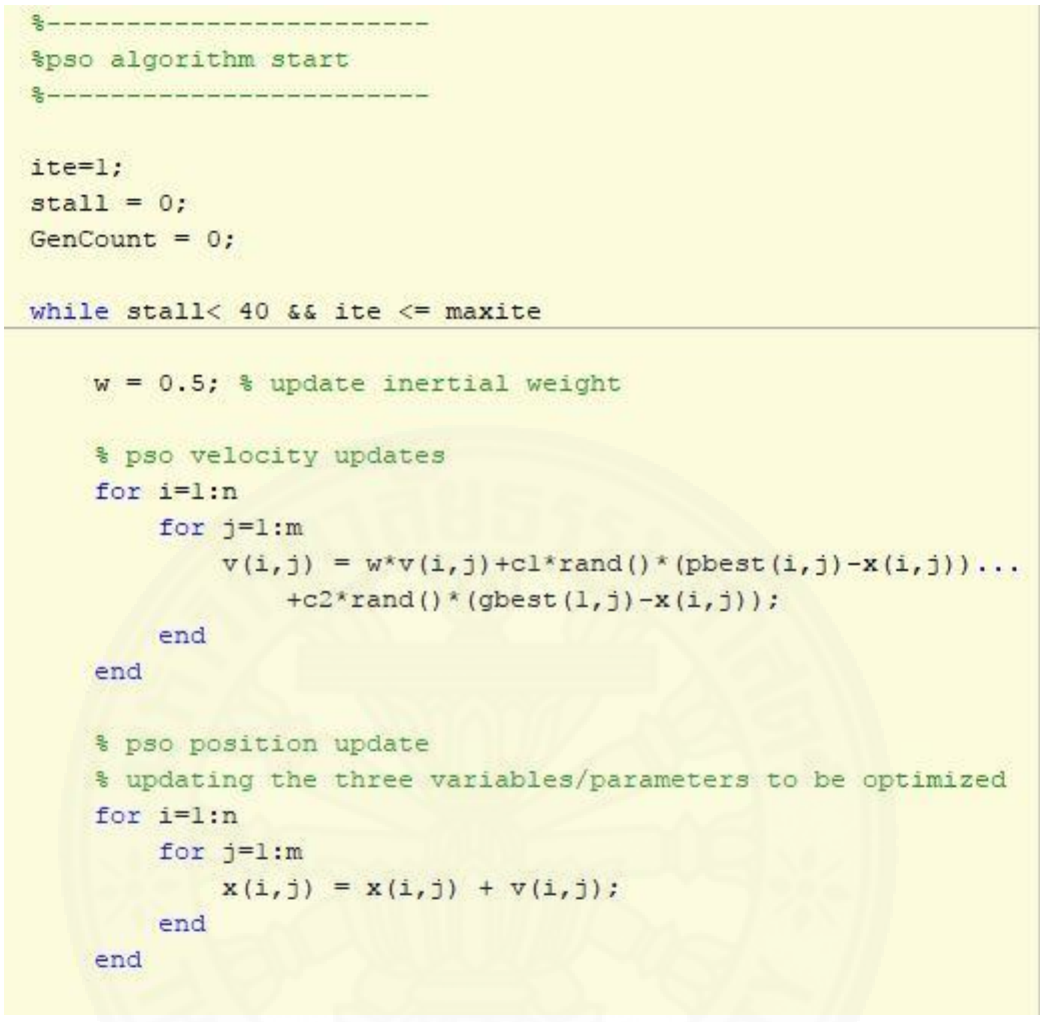
```
%set lower bound
for ll = 1:m
    if ll==1 %e
        LB(ll) = 0.001;
    end
    if ll==2 %c
        LB(ll) = 0.01;
    end
    if ll==3 %g
        LB(ll) = 0.00001;
    end
end

%set upper bound
for ll = 1:m
    if ll==1 %e
        UB(ll) = 10;
    end
    if ll==2 %c
        UB(ll) = 1000;
    end
    if ll==3 %g
        UB(ll) = 200;
    end
end
```



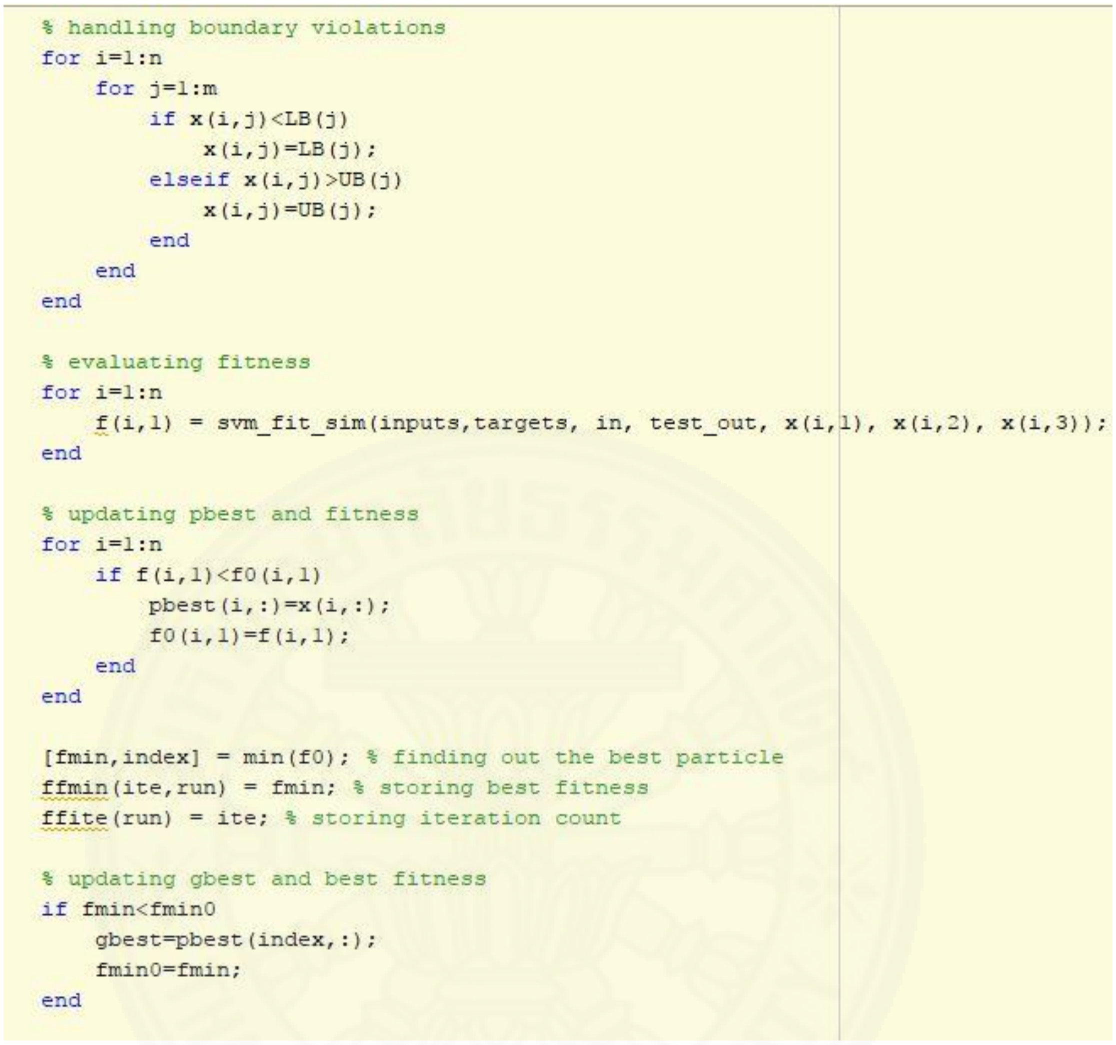
```
%%%%%%%%%%%%%%%%%%%%%%%
%pso main program start
%%%%%%%%%%%%%%%%%%%%%%%

maxite = 50; % set maximum number of iteration
maxrun = 1; % set maximum number of runs need to be

for run = 1:maxrun
    %run

    %-----------------------
    %pso initialization start
    %-----------------------
    for i = 1:n
        for j = 1:m
            %%%
            %randomly generate the value of variables/parameters to be
            %optimized here
            %%%
            if j==1 %e
                x0(i,j) = round(LB(j) + rand() * (UB(j)-LB(j)));
            end
            if j==2 %c
                x0(i,j) = round(LB(j) + rand() * (UB(j)-LB(j)));
            end
            if j==3 %g
                x0(i,j) = round(LB(j) + rand() * (UB(j)-LB(j)));
            end
        end
    end
```



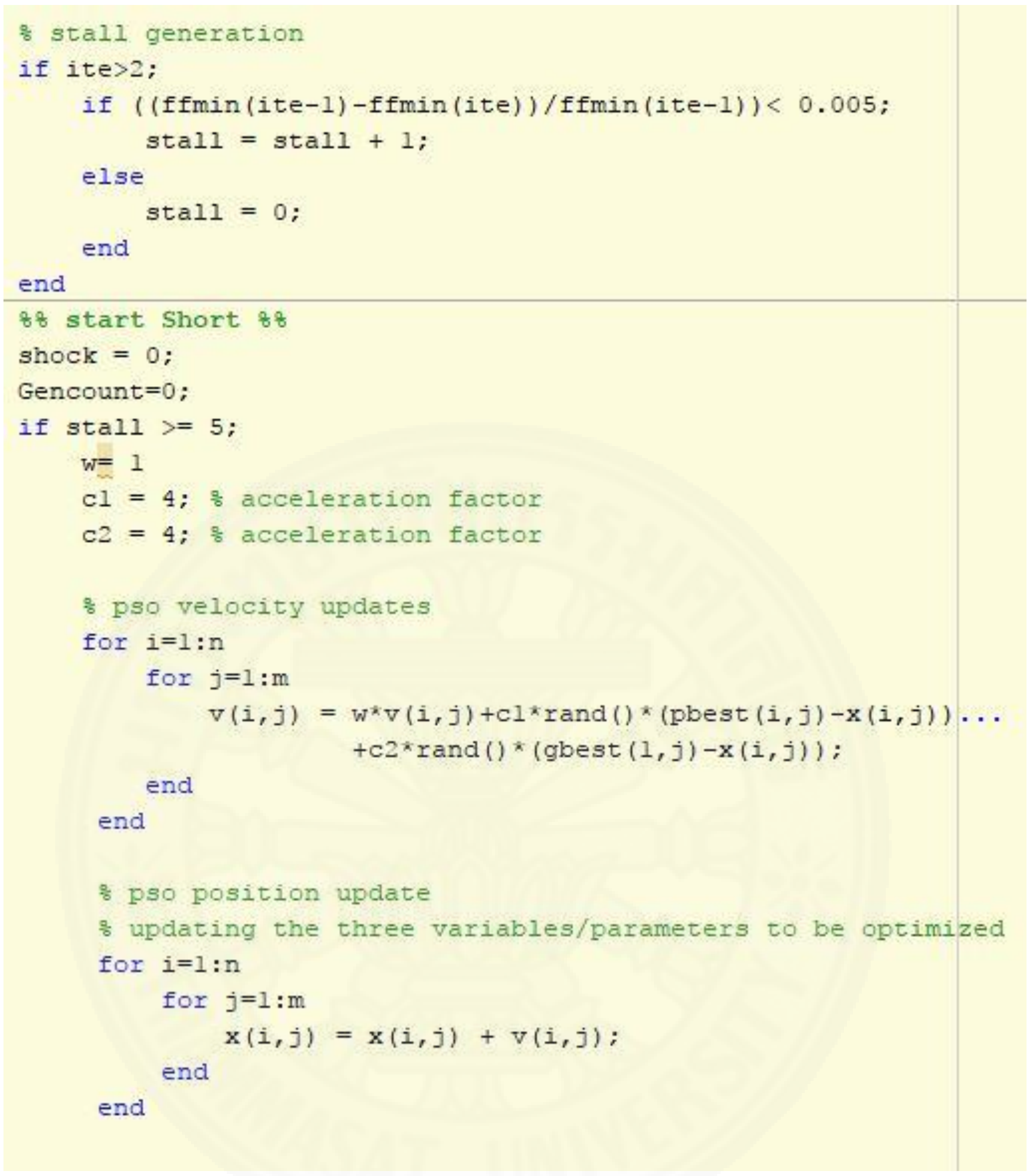
```
x = x0; % initial population/particle/agent
v = 0.1 * x0; % initial velocity

for i = 1:n
    f0(i,1) = svm_fit_sim(inputs,targets, in, test_out, x0(i,1), x0(i,2), x0(i,3));
end

[fmin0,index0] = min(f0);
pbest = x0; % initial pbest
gbest = x0(index0,:); % initial gbest

%-----------------------
%pso initialization end
%-----------------------
```



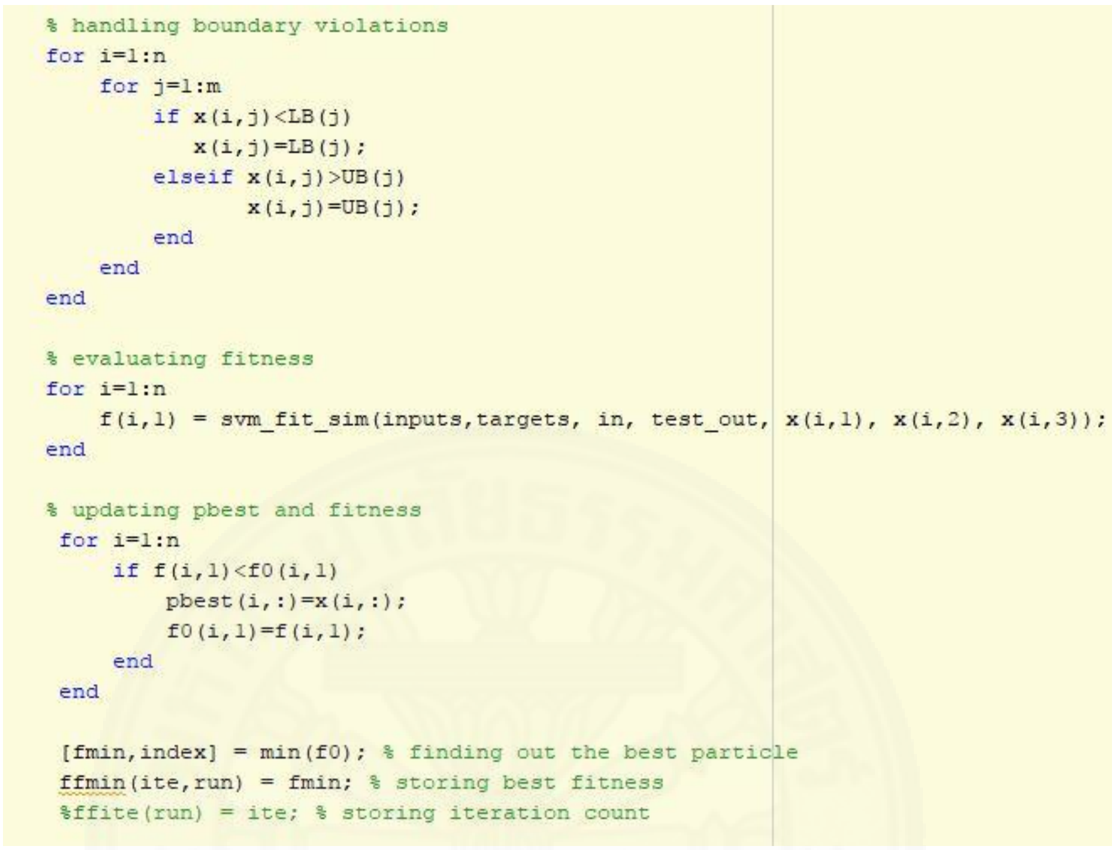
```
%------------------------
%pso algorithm start
%------------------------

ite=1;
stall = 0;
GenCount = 0;

while stall< 40 && ite <= maxite

    w = 0.5; % update inertial weight

    % pso velocity updates
    for i=1:n
        for j=1:m
            v(i,j) = w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))...
                +c2*rand()*(gbest(1,j)-x(i,j));
        end
    end

    % pso position update
    % updating the three variables/parameters to be optimized
    for i=1:n
        for j=1:m
            x(i,j) = x(i,j) + v(i,j);
        end
    end
```



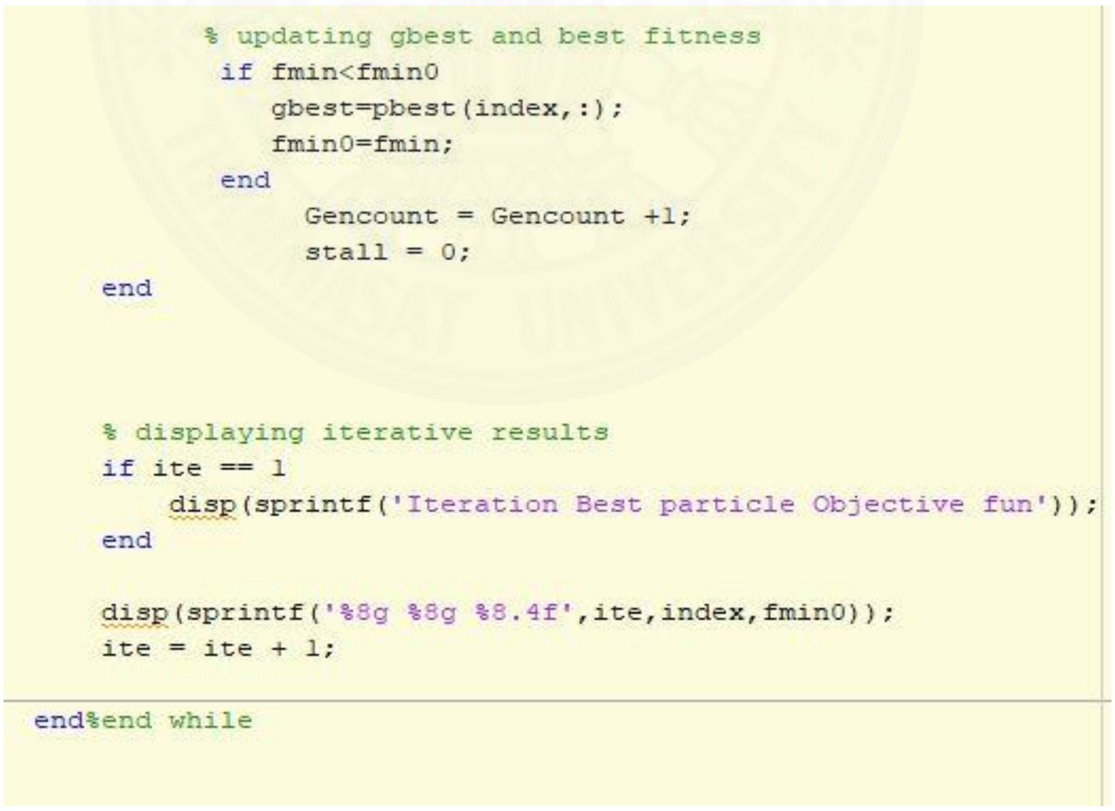
```
% handling boundary violations
for i=1:n
    for j=1:m
        if x(i,j)<LB(j)
            x(i,j)=LB(j);
        elseif x(i,j)>UB(j)
            x(i,j)=UB(j);
        end
    end
end

% evaluating fitness
for i=1:n
    f(i,1) = svm_fit_sim(inputs,targets, in, test_out, x(i,1), x(i,2), x(i,3));
end

% updating pbest and fitness
for i=1:n
    if f(i,1)<f0(i,1)
        pbest(i,:)=x(i,:);
        f0(i,1)=f(i,1);
    end
end

[fmin,index] = min(f0); % finding out the best particle
ffmin(ite,run) = fmin; % storing best fitness
ffite(run) = ite; % storing iteration count

% updating gbest and best fitness
if fmin<fmin0
    gbest=pbest(index,:);
    fmin0=fmin;
end
```



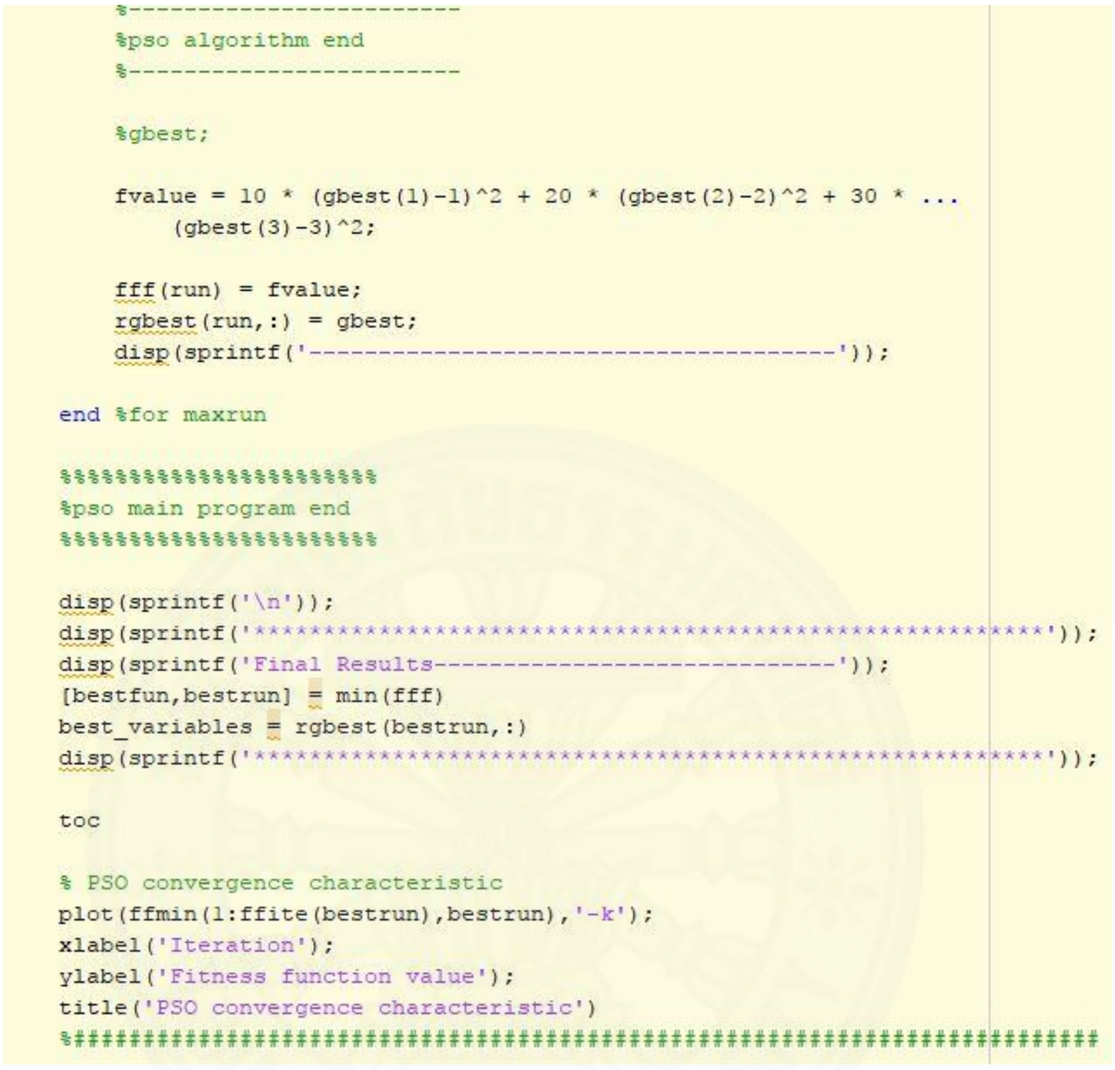
```
% stall generation
if ite>2;
    if ((ffmin(ite-1)-ffmin(ite))/ffmin(ite-1))< 0.005;
        stall = stall + 1;
    else
        stall = 0;
    end
end
%% start Short %%
shock = 0;
Gencount=0;
if stall >= 5;
    w= 1
    c1 = 4; % acceleration factor
    c2 = 4; % acceleration factor

    % pso velocity updates
    for i=1:n
        for j=1:m
            v(i,j) = w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))...
                     +c2*rand()*(gbest(1,j)-x(i,j));
        end
     end

     % pso position update
     % updating the three variables/parameters to be optimized
     for i=1:n
         for j=1:m
             x(i,j) = x(i,j) + v(i,j);
         end
     end
```



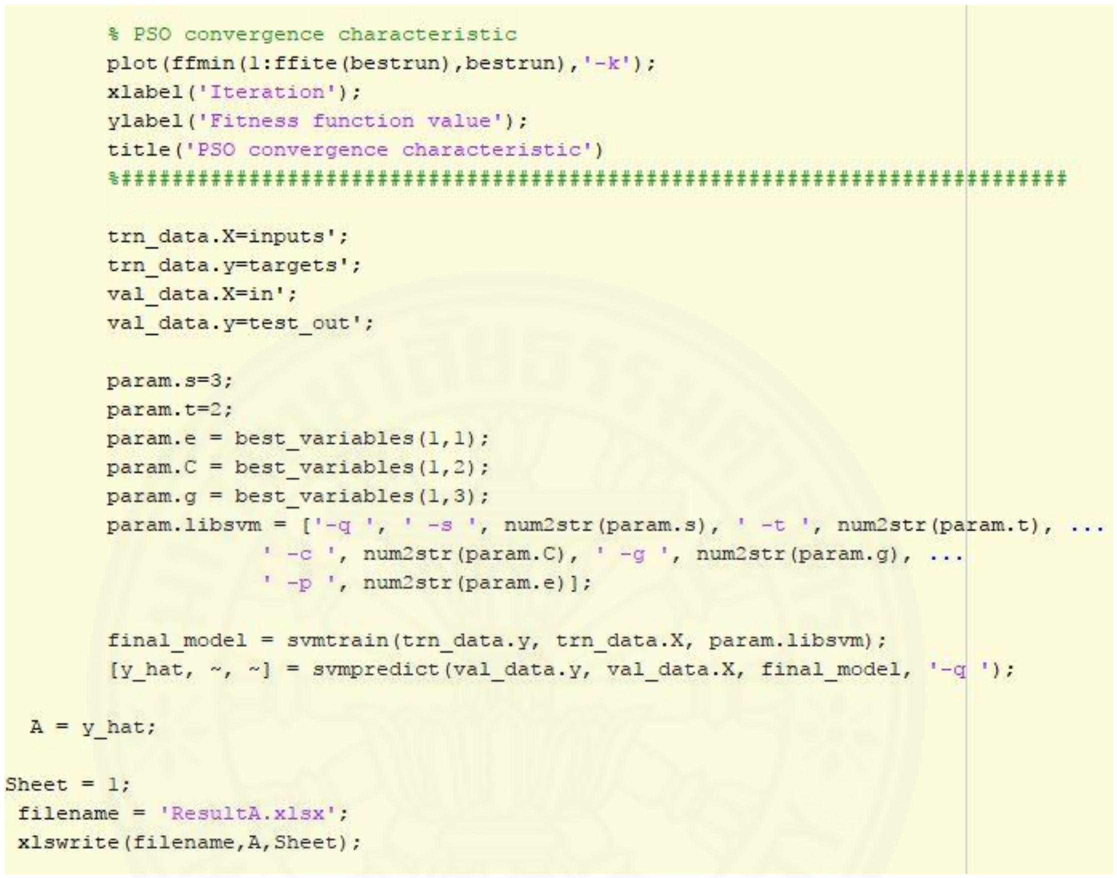
```
% handling boundary violations
for i=1:n
    for j=1:m
        if x(i,j)<LB(j)
           x(i,j)=LB(j);
        elseif x(i,j)>UB(j)
               x(i,j)=UB(j);
        end
    end
end

% evaluating fitness
for i=1:n
    f(i,1) = svm_fit_sim(inputs,targets, in, test_out, x(i,1), x(i,2), x(i,3));
end

% updating pbest and fitness
 for i=1:n
     if f(i,1)<f0(i,1)
         pbest(i,:)=x(i,:);
         f0(i,1)=f(i,1);
     end
 end

 [fmin,index] = min(f0); % finding out the best particle
 ffmin(ite,run) = fmin; % storing best fitness
 %ffite(run) = ite; % storing iteration count
```



```
          % updating gbest and best fitness
           if fmin<fmin0
              gbest=pbest(index,:);
              fmin0=fmin;
           end
                Gencount = Gencount +1;
                stall = 0;
    end



    % displaying iterative results
    if ite == 1
        disp(sprintf('Iteration Best particle Objective fun'));
    end

    disp(sprintf('%8g %8g %8.4f',ite,index,fmin0));
    ite = ite + 1;

end%end while
```



```
    %------------------------
    %pso algorithm end
    %------------------------

    %gbest;

    fvalue = 10 * (gbest(1)-1)^2 + 20 * (gbest(2)-2)^2 + 30 * ...
        (gbest(3)-3)^2;

    fff(run) = fvalue;
    rgbest(run,:) = gbest;
    disp(sprintf('--------------------------------------'));

end %for maxrun

%%%%%%%%%%%%%%%%%%%%%%%
%pso main program end
%%%%%%%%%%%%%%%%%%%%%%%

disp(sprintf('\n'));
disp(sprintf('*************************************************************'));
disp(sprintf('Final Results-----------------------------'));
[bestfun,bestrun] = min(fff)
best_variables = rgbest(bestrun,:)
disp(sprintf('*************************************************************'));

toc

% PSO convergence characteristic
plot(ffmin(1:ffite(bestrun),bestrun),'-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
%#######################################################################
```

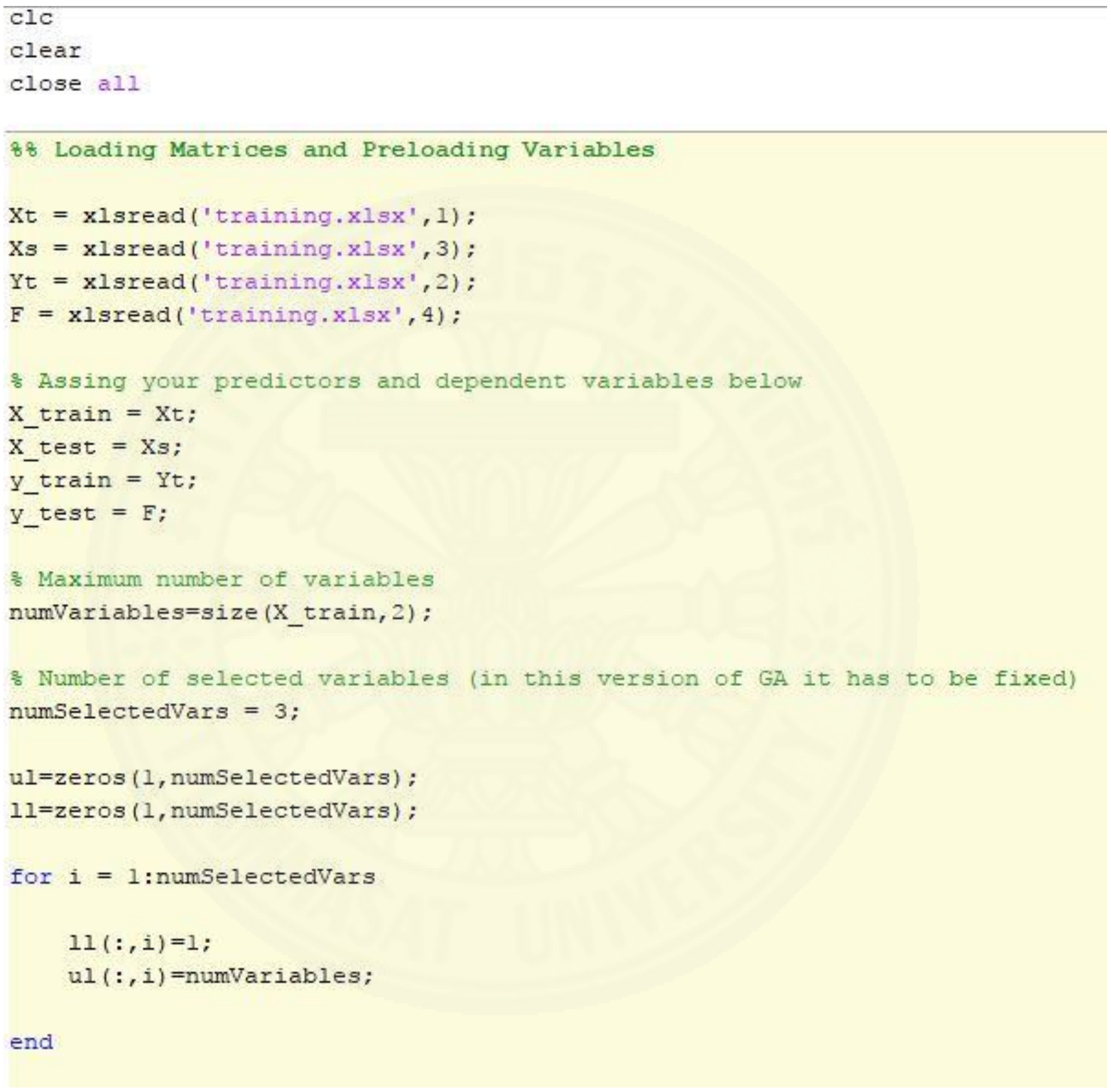


```
        % PSO convergence characteristic
        plot(ffmin(1:ffite(bestrun),bestrun),'-k');
        xlabel('Iteration');
        ylabel('Fitness function value');
        title('PSO convergence characteristic')
        %##########################################################################

        trn_data.X=inputs';
        trn_data.y=targets';
        val_data.X=in';
        val_data.y=test_out';

        param.s=3;
        param.t=2;
        param.e = best_variables(1,1);
        param.C = best_variables(1,2);
        param.g = best_variables(1,3);
        param.libsvm = ['-q ', ' -s ', num2str(param.s), ' -t ', num2str(param.t), ...
                    ' -c ', num2str(param.C), ' -g ', num2str(param.g), ...
                    ' -p ', num2str(param.e)];

        final_model = svmtrain(trn_data.y, trn_data.X, param.libsvm);
        [y_hat, ~, ~] = svmpredict(val_data.y, val_data.X, final_model, '-q ');

  A = y_hat;

Sheet = 1;
 filename = 'ResultA.xlsx';
 xlswrite(filename,A,Sheet);
```

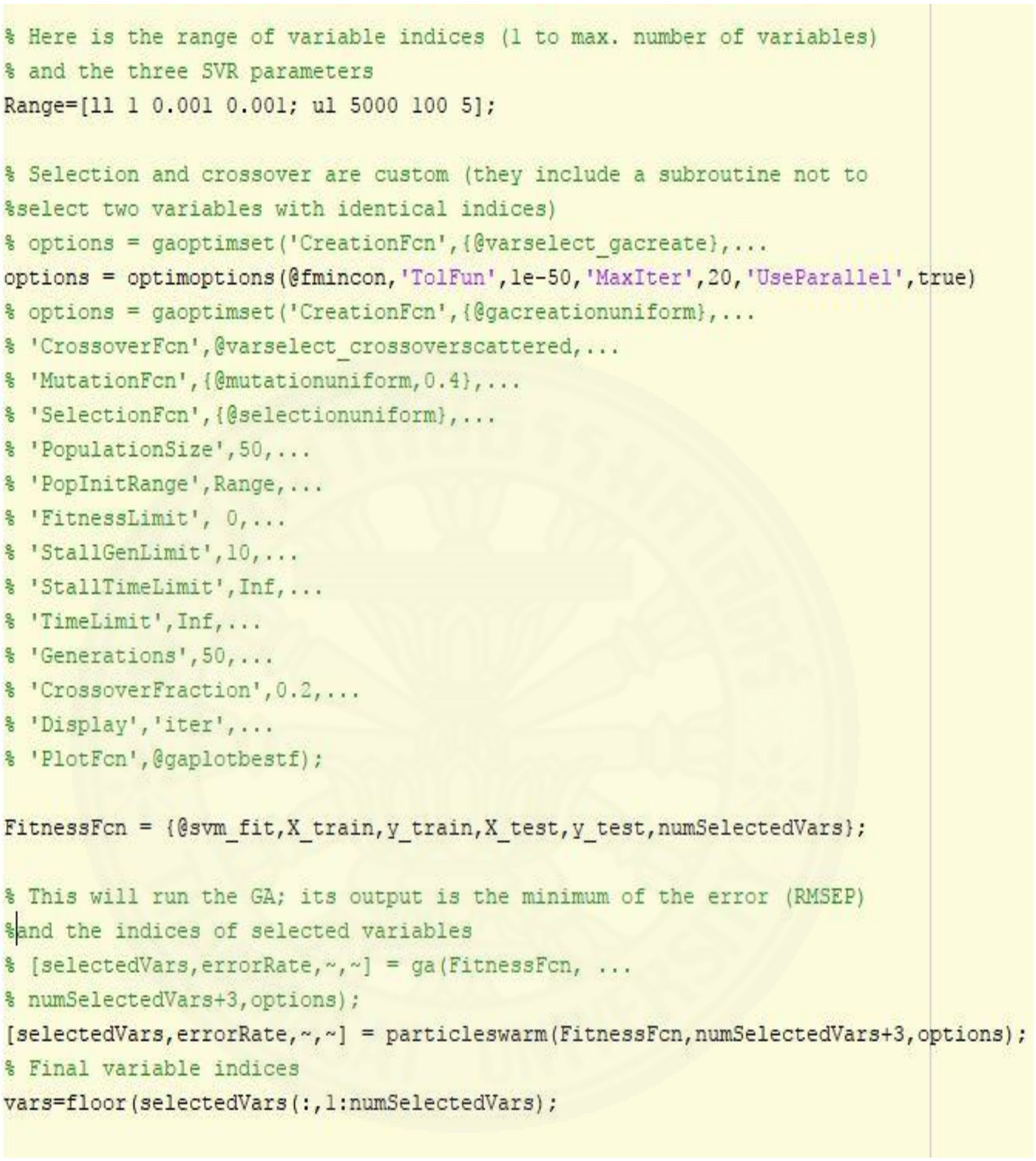
Appendix D

MATLAB code: Training SVR with GA

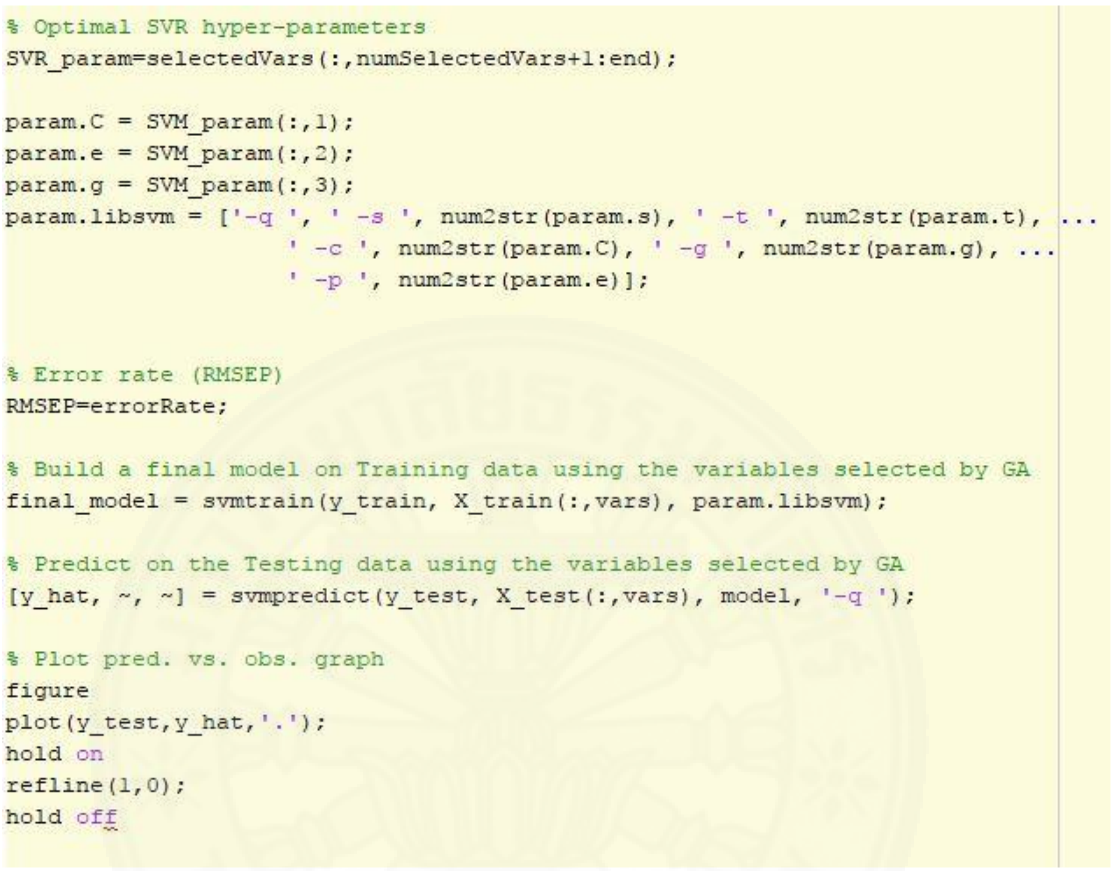
Constructing the SVR-GA, Putting Data, Noting Outputs



```
clc
clear
close all

%% Loading Matrices and Preloading Variables

Xt = xlsread('training.xlsx',1);
Xs = xlsread('training.xlsx',3);
Yt = xlsread('training.xlsx',2);
F = xlsread('training.xlsx',4);

% Assing your predictors and dependent variables below
X_train = Xt;
X_test = Xs;
y_train = Yt;
y_test = F;

% Maximum number of variables
numVariables=size(X_train,2);

% Number of selected variables (in this version of GA it has to be fixed)
numSelectedVars = 3;

ul=zeros(1,numSelectedVars);
ll=zeros(1,numSelectedVars);

for i = 1:numSelectedVars

    ll(:,i)=1;
    ul(:,i)=numVariables;

end
```



```
% Here is the range of variable indices (1 to max. number of variables)
% and the three SVR parameters
Range=[ll 1 0.001 0.001; ul 5000 100 5];

% Selection and crossover are custom (they include a subroutine not to
%select two variables with identical indices)
% options = gaoptimset('CreationFcn',{@varselect_gacreate},...
options = optimoptions(@fmincon,'TolFun',1e-50,'MaxIter',20,'UseParallel',true)
% options = gaoptimset('CreationFcn',{@gacreationuniform},...
% 'CrossoverFcn',@varselect_crossoverscattered,...
% 'MutationFcn',{@mutationuniform,0.4},...
% 'SelectionFcn',{@selectionuniform},...
% 'PopulationSize',50,...
% 'PopInitRange',Range,...
% 'FitnessLimit', 0,...
% 'StallGenLimit',10,...
% 'StallTimeLimit',Inf,...
% 'TimeLimit',Inf,...
% 'Generations',50,...
% 'CrossoverFraction',0.2,...
% 'Display','iter',...
% 'PlotFcn',@gaplotbestf);

FitnessFcn = {@svm_fit,X_train,y_train,X_test,y_test,numSelectedVars};

% This will run the GA; its output is the minimum of the error (RMSEP)
%and the indices of selected variables
% [selectedVars,errorRate,~,~] = ga(FitnessFcn, ...
% numSelectedVars+3,options);
[selectedVars,errorRate,~,~] = particleswarm(FitnessFcn,numSelectedVars+3,options);
% Final variable indices
vars=floor(selectedVars(:,1:numSelectedVars);
```

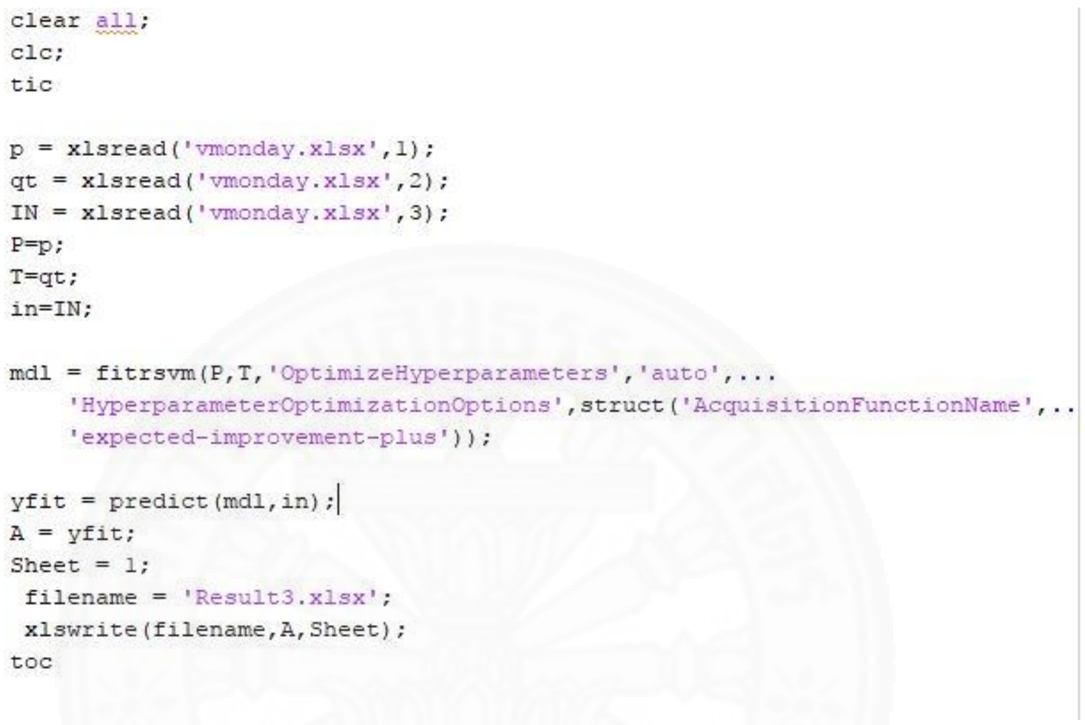


```
% Optimal SVR hyper-parameters
SVR_param=selectedVars(:,numSelectedVars+1:end);

param.C = SVM_param(:,1);
param.e = SVM_param(:,2);
param.g = SVM_param(:,3);
param.libsvm = ['-q ', ' -s ', num2str(param.s), ' -t ', num2str(param.t), ...
                    ' -c ', num2str(param.C), ' -g ', num2str(param.g), ...
                    ' -p ', num2str(param.e)];


% Error rate (RMSEP)
RMSEP=errorRate;

% Build a final model on Training data using the variables selected by GA
final_model = svmtrain(y_train, X_train(:,vars), param.libsvm);

% Predict on the Testing data using the variables selected by GA
[y_hat, ~, ~] = svmpredict(y_test, X_test(:,vars), model, '-q ');

% Plot pred. vs. obs. graph
figure
plot(y_test,y_hat,'.');
hold on
refline(1,0);
hold off
```

Appendix E

MATLAB code: Training SVR with BO

Constructing the SVR-BO, Putting Data, Noting Outputs



```matlab
clear all;
clc;
tic

p = xlsread('vmonday.xlsx',1);
qt = xlsread('vmonday.xlsx',2);
IN = xlsread('vmonday.xlsx',3);
P=p;
T=qt;
in=IN;

mdl = fitrsvm(P,T,'OptimizeHyperparameters','auto',...
    'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',..
    'expected-improvement-plus'));

yfit = predict(mdl,in);
A = yfit;
Sheet = 1;
 filename = 'Result3.xlsx';
 xlswrite(filename,A,Sheet);
toc
```