MULTI-STEP TIME SERIES PREDICTION USING RECURRENT KERNEL ONLINE SEQUENTIAL EXTREME LEARNING MACHINE

LIU ZONGYING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

MULTI-STEP TIME SERIES PREDICTION USING RECURRENT KERNEL ONLINE SEQUENTIAL EXTREME LEARNING MACHINE

LIU ZONGYING

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: (I.C./Passport No.:)

Registration/Matric No.:

Name of Degree:

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Field of Study:

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:
Designation:

MULTI-STEP TIME SERIES PREDICTION USING RECURRENT KERNEL ONLINE SEQUENTIAL EXTREME LEARNING MACHINE

ABSTRACT

Multi-ahead time series prediction model is essential to human activities in the financial domain, electrical load, and natural disaster analysis/forecasting. Recent years, Machine learning algorithms play a significant role in time series prediction, by which inherent serial correlation and potential non-stationary of the data can be automatically analyzed. However, the problems with traditional offline and online learning algorithms in machine learning algorithms are usually faced with parameter dependency, concept drift handling problem, connectionless of neural net and unfixed reservoir. In this study, the proposed models consist of the off-line and on-line algorithm for time series prediction: Recurrent Kernel Extreme Reservoir Machine with Quantum Particle Swarm Optimization (RKERM-QPSO) and Meta-cognitive Recurrent Recursive Kernel On-line Sequential Extreme Learning Machine with the Drift Detection Machine (Meta-RRKOS-ELM-DDM) to solve above mentioned problems and improved overall prediction accuracy. In the forecasting process, the restriction of the prediction horizon is solved by the recurrent multi-steps-ahead algorithm for both off-line and on-line prediction models. The parameter dependency problem of the off-line and on-line model is handled with QPSO and new meta-cognitive learning strategy, respectively. Moreover, reservoir computing is applied to generate a fixed reservoir with high dimension information in the off-line prediction model in order to improve the forecasting accuracy. In the on-line learning model, the recursive kernel is also successfully used to generate a fixed reservoir with optimized information by dissipating and overwriting the information from the coming data in the learning part of the prediction model, which is helpful for improvement of forecasting performance. Besides, concept drift problem in on-line learning model is solved by Drift Detection Machine (DDM). In this study presents the theoretical insights which supporting the proposed arguments. The experiment is carried out by analyzed with benchmark data and a real-world example. In the proposed online learning model, the SMAPE result of 1-18 step periods in real-world data sets are 3.39% for S&P 500, 3.45% for Shanghai, and 5.48% for Ozone, respectively. It shown the super prediction ability compared to other state-of-the-art time-series-specific.

Keywords: Extreme Learning Machine, Time Series Prediction, Multiple Steps Prediction, Concept Drift, Kernel, Reservoir.

ABSTRAK

Model ramalan siri masa mult-ahead adalah penting untuk aktiviti manusia bagi sektor kewangan, medan elektrik, dan analisis bencana alam / peramalan. Algoritma pembelajaran mesin memainuan poranan penting dalam ramalan siri-masa, di mana korelasi bersiri dan ciri-ciri data yang tidak pegun dapat dianalisis secara automatik. Walau bagaimanapun, masalah dengan algoritma luar talian dan dalam talian dari segi pembelajaran secara tradisional lazimnya dihadapi dengan ketergantungan parameter, masalah pengendalian drift konsep, sambungan tanpa simpang neural dan takungan yang tidak tetap terhadap neural. Dalam kajian ini, model-model yang dicadangkan terdiri daripada algoritma luar talian dan dalam talian untuk ramalan siri masa: Mesin Takungan Keseimbangan Berulang Kernel dengan Pengoptimuman zarah Swarm Kuantum Quantum (RKERM-QPSO) dan Keseimbangan Rekursif Berulang Meta-kognitif dalam Talian Terperinci Sequential Mesin dengan Mesin Pengesan Drift (Meta-RRKOS-ELM-DDM) untuk menyelesaikan masalah yang disebutkan di atas dan meningkatkan ketepatan ramalan keseluruhan. Dalam proses peramalan, pembatasan cakap ramalan diselesaikan oleh algoritma pelbagai langkan ke nadapan yang berulang untuk kedua-dua model ramalan luar talian dan dalam talian. Masalah ketergantungan parameter bagi model luar talian dan dalam talian ditangani dengan strategi pembelajaran QPSO dan meta-kognitif yang baru. Selain itu, pengkomputeran takungan digunakan untuk menghasilkan takungan tetap dengan maklumat dimensi tinggi dalam model ramalan luar talian untuk meningkatkan ketepatan ramalan. Dalam model pembelajaran dalam talian, kernel rekursif juga berjaya digunakan untuk menghasilkan takungan tetap dengan maklumat yang dioptimumkan dengan menghilangkan dan menggantikan maklumat daripada data yang akan datang dalam bahagian pembelajaran model ramalan, yang membantu untuk meningkatkan prestasi ramalan. Selain itu, masalah drift konsep dalam model pembelajaran dalam talian diselesaikan oleh Drift Detection Machine (DDM). Dalam kajian ini, membentangkan pandangan teori yang menyokong hujah-hujah kami. Eksperimen ini dijalankan dengan dianalisis dengan data penanda aras ukur dan contoh dunia sebenar. Dalam model pembelajaran dalam talian yang dicadangkan, hasil SMAPE dari 1-18 tempoh langkah dalam set data dunia nyata ialah 3.39% untuk S & P 500, 3.45% untuk Shanghai, dan 5.48% untuk Ozon. Ia menunjukkan keupayaan ramalan super berbanding dengan siri-masa lain siri-masa khusus yang lain.

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my supervisor Professor Dr. Loo Chu Kiong at Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, for his expert guidance, encouragement and full support throughout this research and for allowing me to grow as a research scientist. I also would like to appreciate my lovely wife who advises me not only in daily life but also in research work, my parents and sisters who give me strong support.

During my stay at the Advanced Robotics Lab, Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, I had the chance to collaborate with other researchers. Finally, I would like to thanks all the laboratory members for their assistance when I first join the advanced robotic lab.

TABLE OF CONTENTS

Abstract iii

Abstrak v

Acknowledgements vii

Table of Contents viii

List of Figures xii

List of Tables xiv

List of Symbols and Abbreviations xvi

List of Appendices xx

CHAPTER 1: INTRODUCTION 1

1.1 Background 1

1.1.1 Offline and Online Learning Algorithms 4

1.1.2 Concept Drift Problem 6

1.1.3 Reservoir Computing 7

1.1.4 Multi-step Ahead Prediction 8

1.2 The Problem Statement 10

1.3 Research Objectives 11

1.4 Research Contributions 11

1.5 Thesis Outline 13

CHAPTER 2: LITERATURE REVIEW 14

2.1 Introduction 14

2.2 Prediction Models 14

2.2.1 Support Vector Regression 14

2.2.2 Extreme Learning Machine and Its Extended Algorithms..................... 17

2.2.2.1 Extreme Learning Machine 17

2.2.2.2 Kernel Extreme Learning Machine 19

2.2.2.3 Online Sequential Extreme Learning Machine 21

2.2.2.4 Online Sequential Extreme Learning Machine with Kernel ... 22

2.2.3 Strengths & Weaknesses of Prediction Models...................................... 23

2.3 Adaptive Kernel Filter.. 26

2.4 Recurrent Models for Time Series Prediction.. 28

2.5 Reservoir Computing 30

2.6 Concept Drift 33

2.7 Research Gap 36

2.8 Summary.. 38

CHAPTER 3: METHODOLOGY **40**

3.1 Data Transformation 40

3.2 Normalization 42

3.3 Offline Algorithms for Time Series Prediction.. 42

3.3.1 Recurrent Multi-step ahead Algorithm.. 43

3.3.2 Recurrent Kernel Extreme Learning Machine 45

3.3.3 Reservoir Computing... 46

3.3.4 Recurrent Kernel Extreme Reservoir Machine 48

3.3.5 Parameter Optimization by QPSO.. 51

3.4 Online Algorithms for Time Series Prediction 54

3.4.1 Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism 54

3.4.1.1 Concept Drift Detection 55

3.4.1.2 Combining Drift Detector Mechanism with RKOS-ELM 55

3.4.1.3 Combining ECDD with RKOS-ELM.................................... 57

3.4.1.4 New Meta-cognitive Learning Strategy 61

3.4.2 Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism................. 63

3.5 The Differences Between Reservoir in Offline Learning and Recursive Kernel in Online Learning Algorithm 69

3.6 Performance Measures... 71

3.7 Summary.. 72

CHAPTER 4: EXPERIMENTS DESIGN AND RESULTS FOR OFFLINE LEARNING MODEL.. 73

4.1 Experiment 1.. 73

4.1.1 Data Description and Processing.. 73

4.1.2 Evaluation of Synthetic Data.. 76

4.2 Experiment 2.. 79

4.2.1 Data Processing and Description.. 80

4.2.1.1 Palm oil.. 80

4.2.1.2 Ozone 81

4.2.1.3 Sunspot... 82

4.2.1.4 S&P500 82

4.2.1.5 PC Water Levels 83

4.2.2 Evaluation of Real-world Data 84

4.3 Statistical Comparison 90

4.4 Discussion 91

4.5 Summary 92

CHAPTER 5: EXPERIMENTS DESIGN AND RESULTS FOR ONLINE LEARNING MODEL 93

5.1 Experiments 1 93

5.1.1 Synthetic Data Experiments 94

5.1.2 Real-world Experiment 103

5.2 Experiment 2 108

5.2.1 Synthetic Data Experiment and Results 109

5.2.2 Real-world Experiment and Results 111

5.3 Statistical Comparison 113

5.4 Discussion 115

5.5 Summary 116

CHAPTER 6: CONCLUSIONS, CONTRIBUTIONS AND FUTURE WORKS 118

6.1 Conclusions 118

6.2 Contributions 121

6.3 Limitation of Works 122

6.4 Future Works 122

References 124

Appendices 136

LIST OF FIGURES

Figure 2.1: Construction of ELM (Huang et al., 2004).. 18

Figure 3.1: Architecture of Recurrent Model.. 44

Figure 3.2: Architecture of RKELM.. 50

Figure 3.3: Architecture of RKERM.. 51

Figure 3.4: Flowchart of QPSO algorithm.. 53

Figure 3.5: Meta-cognitive learning for time series prediction model...................... 61

Figure 3.6: Different architectures for three networks 70

Figure 4.1: Mackey-Glass Time Series Data.. 74

Figure 4.2: Lorenz Time Series Data 75

Figure 4.3: The convergent chart of RKERM based on QPSO.................................. 77

Figure 4.4: The comparison between actual values and prediction values in synthetic time series data sets in the 18-th step of RKERM. 78

Figure 4.5: Palm Oil Time Series Data 81

Figure 4.6: Ozone Time Series Data... 82

Figure 4.7: Sun Spot Time Series Data... 83

Figure 4.8: S&P500 Time Series Data.. 84

Figure 4.9: Location of PC station in the Chao Phraya River in Thailand................. 85

Figure 4.10: Water level time series data 86

Figure 4.11: The convergent chart of RKERM based on QPSO for real-world data.... 87

Figure 4.12: The comparison between actual values and prediction values in real-world time series data sets in the 18-th step of RKERM. 90

Figure 5.1: Synthetic time series datasets without concept drift (TS_1, TS_2, TS_3)........ 95

Figure 5.2: Synthetic time series datasets with concept drift (TS_4, TS_5, TS_6). 96

Figure 5.3: The comparison between actual values and prediction values in synthetic time series data sets in the 18-th step of Meta-RKOS-ELM-DDM. 103

Figure 5.4: The comparison between actual values and prediction values in real-world time series data sets in the 18-th step of Meta-RKOS-ELM-DDM. 109

Figure 5.5: The comparison between actual values and prediction values in synthetic time series data sets in the 18-th step of Meta-RRKOS-ELM-DDM. 112

Figure 5.6: The comparison between actual values and prediction values in real-world time series data sets in the 18-th step of Meta-RRKOS-ELM-DDM. 114

LIST OF TABLES

Table 2.1: The strengths and drawbacks of Prediction Models................................. 25

Table 2.2: The summary of models with reservoir computing 33

Table 2.3: The summary of models with concept drift detector 37

Table 3.1: Comparisons between RKERM and RKELM models............................. 51

Table 3.2: Comparisons between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM.. 67

Table 4.1: The parameters settings of all models for synthetic data sets 79

Table 4.2: The performance comparison of synthetic data for multi-step-ahead prediction. 79

Table 4.3: Parameter settings of all models for real-world data sets.......................... 88

Table 4.4: Performance comparison on real-world data for multi-step-ahead prediction. 89

Table 4.5: The comparison of RKELM and RKERM 91

Table 5.1: The performance comparison on synthetic data for multi-step-ahead prediction between ROS-ELM and recurrent kernel online sequential extreme learning machine with different kernel filters. The best performance is in boldface. 98

Table 5.2: The performance comparison on synthetic data for multi-step-ahead prediction by RKOS-ELM with different concept detectors. The best performance is in boldface. 100

Table 5.3: The learning time (second) of RKOS-ELM with DDM and Meta-RKOS-ELM with DDM 101

Table 5.4: The performance comparison on synthetic data for multi-step-ahead prediction by Meta-RKOS-ELM with DDM. 102

Table 5.5: The performance comparison on real-world data for multi-steps-ahead prediction between ROS-ELM and RKOS-ELM with different kernel filters. The best performance is in boldface. 105

Table 5.6: The performance comparison on real-world data for multi-steps-ahead prediction by RKOS-ELM with the different concept detectors. The best performance is in boldface. 107

Table 5.7: The performance comparison on real-world data for multi-step-ahead prediction by Meta-RKOS-ELM with DDM. 108

Table 5.8: The learning time (second) of RKOS-ELM with DDM and Meta-RKOS-ELM with DDM in real-world data. 108

Table 5.9: The performance comparison on synthetic data for multi-steps-ahead prediction by Meta-RRKOS-ELM-DDM 110

Table 5.10: The performance on real-world data for multi-steps-ahead prediction by Meta-RRKOS-ELM-DDM.. 113

Table 5.11: Comparison of meta-RKOS-ELM-DDM and meta-RRKOS-ELM-DDM. 115

LIST OF SYMBOLS AND ABBREVIATIONS

ER_{min}	:	the minimum error.
F	:	the maximum size of hidden neurons in KOS-ELM.
G	:	a mapping function.
H_0	:	the initial hidden layer output matrix.
L	:	the number of training data.
M	:	a time series matrix.
Q	:	the interval matrix.
S	:	the training input data.
SD_{min}	:	minimum standard deviation.
T	:	transpose operator.
TS	:	The testing input matrix.
W	:	a matrix that indicates reservoir neuron connectivity.
Y_p	:	the training target data in p step.
Δ	:	the ALD value.
β	:	the output weights.
β_0	:	The initial output weights in OS-ELM.
β_p	:	The output weight in the p step.
β_p^R	:	the output weight of reservoir in p step.
$\hat{y_p}$	:	the training prediction value in p step.
$\hat{y}$	:	the prediction value.
λ	:	the scale factor.
ν	:	the initial size of training data in OSELM.
ε	:	insensitive loss function.
φ	:	the threshold of ALD.
a	:	the input weights of ELM.
b	:	the threshold in SVR.
c	:	the bias in ELM.
e_pat	:	the minimum error pattern.
g	:	kernel matrix.
k	:	the kernel function.
w	:	the weight in SVR.
$ymem$	:	the memory label.
ABC	:	Artificial Bee Colony.
ADWIN	:	Adaptive Windowing.
ALD	:	Approximate Linear Dependency.

ALD-KOS-ELM	:	Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency Filter.
ALD-RKOS-ELM	:	Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency Filter.
ANN	:	Artificial Neural Networks.
AOS-ELM	:	Adaptive OS-ELM.
AR	:	Autoregressive.
ARIMA	:	autoregressive integrated moving average.
ARMA	:	Autoregressive Moving Average.
ARMAX	:	Autoregressive–moving-average model with exogenous inputs.
CA	:	Cellular Automata.
DDM	:	Drift Detection Method.
DOF	:	Degree of Drift.
ECDD	:	Exponentially Weighted Moving Average for Concept Drift Detection.
ELM	:	Extreme Learning Machine.
ESN	:	Echo State Network.
EWMA	:	Exponentially Weighted Moving Average.
FB	:	Fixed-Budget.
FB-KOS-ELM	:	Kernel Online Sequential Extreme Learning Machine with Fixed-Budget Filter.
FB-RKOS-ELM	:	Recurrent Kernel On-line Sequential Extreme Learning Machine with Fixed-Budget Filter.
GA	:	Genetic Algorithm.
GAP-RBF	:	Growing And Pruning Radial Basis Function Network.
KELM	:	Extreme Learning Machine with Kernel.
KOS-ELM	:	Online Sequential Extreme Learning Machine with Kernel.
LF	:	Linear Function.
LS-SVM	:	Least Squares Support Vector Machines.
LSM	:	Liquid-State Machine.
Meta-RKOS-ELM	:	Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency.
Meta-RKOS-ELM-DDM	:	Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism.

Meta-RRKOS-ELM-DDM	:	Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism.
Meta-RRKOS-ELM-DDM	:	Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism.
MK-ELM	:	Multiple Kernel Extreme Learning Machine.
MRAN	:	Minimal Resource Allocation Network.
MSE	:	Mean Squared Error.
NARX	:	Nonlinear Autoregressive Network with Exogenous Inputs.
OP-ELM	:	Optimally Pruned Extreme Learning Machine.
OS-ELM	:	On-line Sequential Extreme Learning Machine.
P	:	the prediction horizon.
PELM	:	Parallel Extreme Learning Machine.
PF	:	Polynomial Function.
PSO	:	Particle Swarm Optimization.
QPSO	:	Quantum Particle Swarm Optimization.
RAN	:	Resource Allocation Network.
RANEKF	:	Resource Allocation Network via Extended Kalman Filter.
RBF	:	Radial Basis Function.
RC	:	Reservoir Computing.
RELM	:	Recurrent Extreme Learning Machine.
RKELM	:	Recurrent Kernel Extreme Learning Machine.
RKERM	:	Recurrent Kernel Extreme Reservoir Machine.
RKOS-ELM	:	Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency.
RKOS-ELM-DDM	:	Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism.
RNNs	:	Recurrent Neural Networks.
ROS-ELM	:	Recurrent Online Sequential Extreme Learning Machine.
RRKOS-ELM-DDM	:	Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism.
RSVR	:	Recurrent Support Vector Regression.
SLFNs	:	Single-hidden Layer Feed-forward Networks.
SMAPE	:	Symmetric Mean Absolute Percentage Error.
SMOTE	:	Synthetic Minority Over-sampling Technique.

STEPD	:	Test of Equal Proportions.
SVM	:	Support Vector Machine.
SVR	:	Support Vector Regression.
WEMA	:	Exponentially Weighted Moving Average.

LIST OF APPENDICES

Appendix A: Publications and Papers Presented 136

CHAPTER 1: INTRODUCTION

There are many projects based on time series prediction that affect and service our daily life, such as prediction of weather, air quality, water level, electricity load, and so forth. The goal of time series prediction is to mine the inner regular patterns of data in order to predict future accurately. This is because forecasting future information or event is crucial for most organizations, such as government agencies, private companies or others. The future information is important in assisting the management in strategic planning and decision making. Therefore, time series prediction plays a vital role in real-world life. It is a challenge in many domains and this has attracted increasing number of researchers who are keen to create new models or improve various existing algorithms in order to enhance the performance of prediction.

1.1 Background

There are many prediction techniques that required researchers to obtain necessary information. Therefore, in the process of using models, there are some limitations or restriction conditions. Normally, the selection of techniques depends on the data type, quality of models available and the predefined assumption (Haykin & Simon, 2009). Taking Autoregressive Moving Average (ARMA) as an example, it requires data to follow a normal distribution. However, the majority of real-world data is not following normal distribution. Thus, data should be transformed to be a normal distribution before applying it in ARMA. Many statistical models require removing the outliers values from data set before using time series prediction, because it cannot handle the data that is not stationary. In order to select a suitable model for a real-world application, researchers are supposed to understand the different characteristics of models and determine which model is suitable for the application requirements.

Stationarity is usually a desirable assumption in the analysis and prediction of time series. Indeed, when one is interested in finding a relationship between the present and the past through prediction, the relationship must be stationary throughout the evolution of time; otherwise, the prediction would not be possible (Burgess & Refenes, 1999). Most statistical forecasting methods are based on the assumption that the time series is approximately stationary. A stationary series is relatively easy to predict by: you simply forecast that its statistical properties will be the same in the future as they have been in the past. Unfortunately, most of real-world data sets are non-stationary. The non-stationary datasets, such as Mackey-Glass, Lorenz, Sunspot and stock index, are usually used in research as benchmark datasets. Based on historical data, non-stationary time series data predicts the future values using by the related model. It is important to note that accuracy is the most important element that measures predictive ability of models. There are lots of many researchers who paid more attention to enhance accuracy of time series prediction in different models. These models can be classified as traditional statistical models and non-statistical models. For instance, Exponential Smoothing (Taylor, 2012), Multiple Regression (Aranda, Ferreira, Mainar-Toledo, Scarpellini, & Sastresa, 2012), Autoregresson Autoregressive (AR) (Pötzelberger, 1990) and so on belong to statistical method, while fuzzy logic (Miao, Potts, Huang, Elliott, & Rivett, 2012), Support Vector Machine (SVM) (Drucker, Burges, Kaufman, Smola, & Vapnik, 1997), Artificial Neural Networks (ANN) (Yadav & Sahu, 2017; Li & Chan, 2017) are non-statistical methods.

Statistical methods have been successfully employed in time series prediction in the past century. These include ARMA, autoregressive integrated moving average (ARIMA) and some statistical methods were applied to solve predicting problem on time series data. However, their predictive ability were constrained by their assumption of linear behavior. Furthermore, in the real-world practical applications, time series data is almost non-stationary, which has some limited conditions for above methods. For instance, the researchers must assume that the data was stationary or transform non-stationary data to be a stationary data before doing any analysis. Therefore, the results from these statistical methods are unsatisfactory because of the characteristic of real-world time series data.

In recent decades, artificial intelligence has became a popular topic and lots of algorithms that have been successfully applied in a variety of domains for time series prediction. The most efficient algorithm was Support Vector Regression (SVR) (Drucker et al., 1997). Since then, it has been widely studied by researchers. Outlier detection with SVR by robust regression (Colliez, Dufrenois, & Hamad, 2006), which significantly improved the robustness against outliers compared to traditional SVR. Least Squares Support Vector Machines (LS-SVM) (Van Gestel et al., 2001) with the evidence framework is the least squares version related to a cost function, which opens new perspectives with respect to time-series prediction. However, there are some limitations and drawbacks in the process of SVR and its extension algorithms. Firstly, it requires full information of input data, especially for target value, which determines the length of the predict horizon. Secondly, it has computational complexity when datasets become large, which leads to large time consumption. Lastly, parameter dependency is a vital problem in SVR. There are three main parameters in SVR that impact on the performance of prediction, including bound on the Lagrangian multipliers, conditioning parameter for quadratic programming method and e-tube.

However, in 2004, Huang et.al proposed a new algorithm called Extreme Learning Machine (ELM) (G.-B. Huang, Zhou, Ding, & Zhang, 2012), which was a single hidden layer feedforward neural network. When compared with traditional feedforward neural networks, ELM is remarkably fast and efficient to reach a global optimum. The learning speed of ELM can be thousands of times faster than traditional feedforward network learning algorithms, i.e. back-propagation algorithm (P. Werbos, 1974) while obtaining better generalization performance. In (G.-B. Huang et al., 2012) and (Fernández-Delgado, Cernadas, Barro, Ribeiro, & Neves, 2014), the authors have shown that the generalization ability of ELM was comparable or even better than that of SVM. Moreover, ELM requires less human intervention than SVM. There is only one parameter that needs to be specified by users in ELM, which is the number of hidden nodes. Furthermore, ELM was also applied in the on-line learning algorithm. On-line Sequential Extreme Learning Machine (OS-ELM), which combined an on-line sequential learning algorithm with ELM, was proposed by Liang et. al. (Liang, Huang, Saratchandran, & Sundararajan, 2006) in 2006. It concluded that the OS-ELM could update itself based on the information of the coming data and produce better performance in time series prediction than other sequential algorithms, such as Growing And Pruning Radial Basis Function Network (GAP-RBF), Minimal Resource Allocation Network (MRAN), Resource Allocation Network via Extended Kalman Filter (RANEKF) and Resource Allocation Network (RAN). No matter in aspect of offline or online learning ELM models, it has been proven that ELM and its extended models have good performance in time series prediction. Therefore, this study pays more attention to the ELM model based approaches.

1.1.1 Offline and Online Learning Algorithms

Although ELM has been proven that it had super forecasting ability and less parameter dependency compared with SVR, the number of hidden nodes like a double-edged sword that impacts on the accuracy of prediction. Besides, due to random weights and biases of input layers of ELM, the hidden layer mapping is not stable, which directly leads to the lack of prediction deterministic in ELM. On the other hand, OS-ELM also produced better performance in time series prediction than other sequential algorithms. However, the number of hidden nodes and the initial number of training data is required to be defined before training. These parameters have directly impacted on the result of prediction in OS-ELM. In addition, the selecting method of input weight in OS-ELM is the same as that of ELM, which also leads to the lack of deterministic prediction.

Scardapane et. al. proposed an on-line learning model for time series prediction called On-line Sequential Extreme Learning Machine with Kernel (KOS-ELM) (Scardapane, Comminiello, Scarpiniti, & Uncini, 2015), which was updated using implicit mappings, defined by the kernel method. Due to replacement of the kernel method, the implicit mappings of KOS-ELM become stable, solving the problem of prediction deterministic in OS-ELM. In 2017, Zhang et.al. proposed a modified online sequential learning algorithm with the forgetting factor (named WOS-ELM algorithm) that weighs the new observation (H. Zhang, Zhang, & Yin, 2018). However, it has more complex computation for seeking weight in learning part than KOS-ELM. Therefore, due to the stability by applying kernel method to replace random part in OS-ELM, less parameter dependency problem, and much better forecasting performance comparing with other online learning model, this study pays attention to KOS-ELM.

At the same time, due to the characteristics of kernel method, KOS-ELM has the heavy computation, especially for the large scale data set. In order to solve this problem, the research by paper (Scardapane et al., 2015) used two types of kernel filters, i.e., Approximate Linear Dependency (ALD) and Fixed-Budget (FB), in the learning which could filter the incoming data in order to decide which data could be updated. It is noted that KOS-ELM with ALD had better forecasting performance than KOS-ELM with FB and KOS-ELM also obtained better results than OS-ELM in aspect of time series prediction based on experiments. However, ALD has parameter dependency problem. It needs to define the threshold of ALD before training the model. At the same time, the concept drift problem also appears in the training process of on-line learning model, because of the characteristic of non-stationary data.

1.1.2 Concept Drift Problem

In non-stationary environments, the statistical characteristics of training data streams may change, resulting in concept drift. Concept drifts generally occur when the underlying distribution changes over time. Majority of research that deal with the concept drift problem are in the classification domain (Hammoodi, Stahl, & Tennant, 2016; Dehghan, Beigy, & ZareMoodi, 2016; Kulkarni & Ade, 2016). A multi-class ELM based meta-cognitive method, called Meta-cognitive ELM (McELM) was proposed in (Babu & Suresh, 2012), where it decided which samples to learn, when and how to learn and concluded that McELM had better performance in multi-class classification than ELM, SVM and other cognitive networks. In the aspect of time series prediction, work by (Cavalcante & Oliveira, 2015) combined two explicit drift detector mechanisms, namely the Drift Detection Mechanism (DDM) and the Exponentially Weighted Moving Average for Concept Drift Detection Mechanism (ECDD), with OS-ELM, in which the OS-ELM updated the decision model just in the presence of concept drift in data. It finally concluded that the drift detectors were able to speed up the prediction time of OS-ELM while maintaining equivalent accuracy.

In aspect of time series prediction, a feature extraction method for explicit concept drift detection was proposed by (Cavalcante, Minku, & Oliveira, 2016). This method was based on the time series features to monitor how concepts evolve over time. Various on-line methods of drift detection in data stream, including DDM, EDDM, Dynamic Weight Majority (DWM) (Kolter & Maloof, 2003), and Adaptive expert ensembles (AddExp) (Kolter & Maloof, 2005) were compared in (Mittal & Kashyap, 2015). The results of this study concluded that DDM performed well for detecting changes rather than other methods. Furthermore, ensemble learning techniques are very popular nowadays for detecting concept drift problem, such as Accuracy and Growth rate updated Ensemble

(AGE) (Liao & Dai, 2014) and On-line Bagging (Oza & Russell, 2001). Moreover, there are other popular methods dealing with concept drift, such as, data stream mining (Hammoodi et al., 2016), monitoring the distribution of ensemble‘s error (Dehghan et al., 2016), and logistic regression learning model (Kulkarni & Ade, 2016). However, these kinds of methods have high time consumption and are only used for classification.

Therefore, based on above reviewed studies and discussion, although the concept drift detections have good performance and DDM and ECDD successfully combined with OS-ELM for improving time series forecasting accuracy, the majority of researches on dealing with the concept drift problem are in the classification domain. The concept drifts generally occur when the underlying distribution changes over time. This thesis should pay more attention to the concept drift problem in the process of on-line learning time series prediction.

1.1.3 Reservoir Computing

Reservoir Computing (RC), which is a framework for computation that may be viewed as an extension of neural networks(Schrauwen, Verstraeten, & Van Campenhout, 2007), is also employed in time series prediction. Typically, the reservoir is a fixed dynamical system that is fed by input information and the dynamics of the reservoir map the input to a higher dimension. Then a mechanism is trained to read the state of the reservoir and map it to the desired output. There are two major types of RC, including Liquid-State Machine (LSM) and Echo State Network (ESN). They have been used in different domains successfully. For example, Verstraeten et al. (Verstraeten, Schrauwen, d’Haene, & Stroobandt, 2007) applied all the types of RC and allowed the simulation of a wide range of reservoir topologies for a number of benchmarks. Bianchi and his teammates applied a developed ESN for the prediction of telephone calls load in time series prediction contest in 2015, and the results obtained confirmed the high accuracy of ESN in the load prediction of short leading times

(Bianchi, Scardapane, Uncini, Rizzi, & Sadeghian, 2015).

Similar to ELM, ESN is a random projection algorithm with a fixed and large reservoir that contains randomly and sparsely connected neurons. Therefore, it shares the same problem with ELM, including unstable forecasting ability and global control parameters. However, ESN is not only a special class of Recurrent Neural Networks (RNNs) but also helps to avoid the possibility of vanishing gradient associated with RNNs. Nevertheless, it is also computed based on the current and previous input data information.

Furthermore, it has been described with several different interconnection structures and analog neurons. The main idea is to drive a random, large and fixed recurrent neural network with the input signals, thereby inducing in each neuron with this reservoir network and a mechanism is trained to read the state of the reservoir that is mapped for computing output signal. There are more features can be obtained by reservoir computing method in output weight of model. In a research by (Ortín et al., 2015), RC has successfully unified with ELM. This approach used intermediate reservoir space, where inputs are randomly selected by a nonlinear form, in order to create a new space. However, due to random selection of input weights, these spaces in reservoir are not stable, which have directly affected the stability of the predict performance.

1.1.4 Multi-step Ahead Prediction

Over the past twenty years, predictor design paid a lot of attention to the nonlinear model structures. Among others, generalizations of the traditional ARMA and Autoregressive–moving-average model with exogenous inputs (ARMAX) models to the nonlinear domain have been proposed, and Nonlinear Autoregressive Network with Exogenous Inputs (NARX) has proven to be successful in many applications (Billings, Chen, & Korenberg, 1989; S. Chen & Billings, 1989). Recently, neural networks have been proposed and extensively used in the identification and control of dynamic systems

(Juditsky, Zhang, Delyon, Glorennec, & Benveniste, 1994; Narendra & Parthasarathy, 1990; Prasad, Swidenbank, & Hogg, 1998). Most of the published literature, however, considers the application of neural networks for a single step ahead prediction (Parlos, Rais, & Atiya, 2000).

One of the earliest attempts in using neural networks for long range (or multi-step) prediction was reported by Su et. al. (Su, McAvoy, & Werbos, 1992). Following the early success of feedforward networks in approximating complex static mappings, researchers attempted use of the so-called recurrent networks for the same purpose. Recurrent networks are those that have at least one feedback loop in their structure (S. Haykin, 1999). Either feedforward or recurrent networks can be combined with a dynamic architecture to obtain dynamic (feedforward) or dynamic recurrent neural networks, respectively. There were earlier attempts by the authors to use recurrent networks for multiple steps ahead prediction. However, these predictors performed somewhat poorly during certain transients (Parlos, Chong, & Atiya, 1994; Atiya & Parlos, 2000).

Over the last decade, Extreme Learning Machine has become popular in machine learning community (Saxe et al., 2011; Widrow, Greenblatt, Kim, & Park, 2013; Le, Sarlós, & Smola, 2013). It does not only have outstanding performance in time series prediction, but it is also thousands times faster than the traditional neural networks like back-propagation algorithm (Munro, 2010) while obtaining better generalization performance. In the research done by (Huang, Zhu, & Siew, 2004) and (Ježowicz, Gajdoš, Uher, Mišák, & Snášel, 2018), they have shown that the generalization ability of ELM is comparable or even better than that of SVM. Moreover, there are many extension algorithms that have been proposed in recent years, including some approaches that are closely related to ELM. For example, Optimally Pruned Extreme Learning Machine (OP-ELM) (Miche et al., 2010) and Parallel Extreme Learning Machine (PELM) (He, Shang, Zhuang, & Shi, 2013). In

2005, Huang and Siew proposed Extreme Learning Machine with Kernel (KELM)1.1, which solved the prediction deterministic problem of ELM. There are many researches based on KELM that proposed related algorithms to improve forecasting performance. Su et al. (L.-j. Su & Yao, 2013) optimized the kernel function using a weighted sum of kernel functions and concluded that this approach was able to make neural networks robust and generate better generalization performance for regression prediction. Multiple Kernel Extreme Learning Machine (MK-ELM) (Liu, Wang, Huang, Zhang, & Yin, 2015) showed better results and less computational cost compared to other multiple kernel learning algorithms. However, these types of models have only been used to generate one-step-ahead prediction and are restricted by the size of target data.

1.2 The Problem Statement

From the discussions of time series prediction models, ELM and its extended algorithms are very powerful time series prediction model. However, it still has some limitations, especially for applying in the real-world applications. As mentioned in the discussions, lack of deterministic prediction from random selection of input weight, parameter dependency problem, computation complexity, unstable reservoir in the network, concept drift problem in on-line learning algorithm, and the restriction of prediction horizon appear in the process of time series prediction. Therefore, the problems of this thesis are defined as follows:

1. The prediction horizon in off-line and on-line time-series prediction models was restricted by the number of target data in the training part, or prediction models only have the one-step-prediction horizon, which limits the practicability in the real-world project of time series prediction.

2. There are no stable reservoir in the offline and online algorithms, which can save optimized information.

3. The parameter dependency problem appears in offline and online algorithms, such as

parameters of the kernel method, leaking rate of reservoir computing, and the threshold of adaptive kernel filter in the on-line learning part.

4. It is because concepts are often not stable and change with time, thus the concept drift problem generally appears in the on-line learning part of time series prediction.

1.3 Research Objectives

In order to solve the problems of existing algorithms and improve the performance in long-term time series prediction. The following listed below are the objectives of this research research:

1. To forecast multiple prediction horizons by a recurrent multi-step algorithm in off-line and on-line learning model.

2. To generate a stable reservoir in off-line learning algorithm by combining kernel extreme learning machine and echo state network and generate a fixed reservoir in on-line learning algorithm by recursive kernel in order to improve the forecasting performance.

3. To solve parameter dependency problem in the offline and online learning algorithm by quantum particle swarm optimization and a new meta-cognitive learning strategy, respectively.

4. To solve the concept drift problem in time series prediction for the online learning model.

1.4 Research Contributions

In this thesis, based on the KELM and Online Sequential Extreme Learning Machine with Kernel (KOS-ELM), an off-line and on-line learning time series prediction models have been successfully applied in non-stationary data sets for multi-step ahead time series prediction. These models apply recurrent multi-step ahead prediction algorithm to solve the restriction of prediction horizon. Parameter dependency problem is also solved by Quantum Particle Swarm Optimization (QPSO) in off-line learning model and meta-cognitive learning strategy in on-line learning model, respectively. Furthermore, both models with fixed reservoir, which can optimize information by dissipating and overwriting the information from the previous data in the training part of the prediction model, improve the predictive performance by reservoir computing and recursive kernel method for proposed online and offline learning models, respectively. In the online learning model, kernel adaptive filter and meta-cognitive learning strategy assist on reducing complex computation in the learning part. Lastly, concept drift problem is solved using drift detection mechanism. In summary, the contributions of these on-line and off-line learning models are characterized as follows:

• Offline learning model for time series prediction: Recurrent Kernel Extreme Reservoir Machine with QPSO, with the following capabilities.

- Releasing the restriction of prediction horizon.
- Generating a fixed reservoir with high dimension information.
- Solving the parameter dependency problem.
- Increasing the forecasting performance in multi-step ahead time series prediction.

• Online learning model for time series prediction: Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism.

- Solving the concept drift problem.
- Reducing the complex computation in the learning part.
- Generating a fixed reservoir that optimized information by dissipating and overwriting the information from the previous data in the learning part.

1.5 Thesis Outline

This thesis is organized into four parts: Introduction and Literature Review (Chapters 1 and 2), Methodology (Chapter 3), Experiments (Chapters 4 and 5), and Conclusions and Future Works (Chapter 6).

Chapter 2 reviews the traditional neural network and fundamentals of ELM in off-line and on-line learning algorithms, including SVR, ELM, KELM, OS-ELM and KOS-ELM. Furthermore, it discussed studies of reservoir computing and concept drift problem in on-line learning algorithm.

Chapter 3 formally defines the data transformation, recurrent multi-steps-ahead prediction algorithm. Furthermore, detailed a new off-line algorithm for time series prediction (Recurrent Kernel Extreme Reservoir Machine (RKERM) with QPSO) and on-line algorithm for time series prediction, including Meta-cognitive Recurrent Kernel On-line Sequential Extreme Learning Machine with Drift Detector Mechanism and Meta-cognitive Recurrent Recurrent Recursive Kernel On-line Sequential Extreme Learning Machine with Drift Detector Mechanism.

Chapter 4 presents an empirical performance analysis of off-learning algorithm using two synthetic and five real-world non-stationary time series data sets and compares fundamental neural networks, such as SVR, ESN, Recurrent Extreme Learning Machine (RELM), RKERM, Recurrent Kernel Extreme Learning Machine (RKELM) with QPSO with proposed off-line algorithm.

Chapter 5 presents an empirical performance analysis in two of online learning algorithms using six synthetic and three real-world non-stationary time series data sets, respectively and finds out the role of recursive kernel in the prediction model.

Chapter 6 presents the conclusions, contributions and future works.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In the past decades, numerous types of models were applied in the field of time series prediction, which was to mine the inner regular patterns of data in order to predict the future accurately. In particular, artificial intelligence has become a popular topic and lots of algorithms have been successfully applied in a variety of domains for time series prediction in recent years. This chapter mainly focused on the significant algorithms that made the contribution and played a vital role in the time series prediction, namely Support Vector Regression, Extreme Learning Machine, and its extended algorithms. Studies of reservoir computing, which is applied to generate a fixed reservoir in the training part, also is considered. Furthermore, studies on recurrent models for time series models, adaptive kernel filter, and the concept drift will be reviewed.

2.2 Prediction Models

Many traditional methods, such as autoregressive moving average and autoregressive integrated moving average and some statistical methods, are applied to solve predicting problem on stationary time series data. However, the majority of real-world data sets are non-stationary. In recent decade, artificial intelligence has become a popular topic and lots of algorithms have been successfully applied in a variety of domains for time series prediction.

2.2.1 Support Vector Regression

The Support Vector Machine, developed by Vapnik and others in 1995, is used for many machine learning tasks such as pattern recognition, and object classification. In the case of time series prediction, Drucker et al. proposed a supervised learning model for regression called SVR (Drucker et al., 1997), which was the methodology by which a function is estimated using observed data which in turn “trains” the SVM and could lead to great potential and superior performance in practical applications. It is a kernel method for regression based on the principle of structural risk minimization (Smola & Schölkopf, 2004).

In order to predict future situation by SVR, firstly, a set of time series data is assumed as x(t), where $t = 1, 2, \ldots, N$. If the data is not linear in its input space, the goal is to map the data $x(t)$ to a higher dimension feature space by a kernel function ($k(x)$), then perform a linear regression in the higher dimensional feature space by equation (2.1). The goal is to find weights (w) and the threshold (b) in SVR as well as to define the criteria for finding an optimal set of weights.

$$f(x) = sign(w^T x + b), \tag{2.1}$$

where the weight in SVR (w) is the weights and the threshold in SVR (b) represents the threshold. In this process of finding weights and threshold, the final goal is the minimization of the regularized risk $R_{reg}(f)$:

$$R_{reg}(f) = R_{emp}(f) + \frac{\lambda}{2}\|w\|^2, \tag{2.2}$$

where, the scale factor (λ) is commonly referred to as the regularization constant, and the empirical risk is defined as: $\frac{1}{N}\sum_{i=0}^{N=1} \psi(x(i), \gamma(i), f(x(i), w))$, where $y(i)$ is the target data and $\psi(\cdot, \cdot)$ is loss function. The most common loss function is the ε-insensitive loss function defined by (Vapnik, 2013). In order to solve the optimal weights and minimize the regularized risk, a quadratic programming problem is formed by the insensitive loss function (ε)-insensitive loss function. The mathematic formula is shown as follow:

$$\text{minimize:}\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \psi(\gamma(i), f(x(i), w)), \tag{2.3}$$

where n is the number of observation x.

In the process of minimization, $\psi(\gamma(i), f(x(i), w))$ depends on the parameter ε. It should be noted both ε and C are both user defined constants and are typically computed empirically. At the same time, in order to carry out the non-linear regression using SVR, it is necessary to map the input space $x(i)$ into $varph$. A kernel function that satisfies Mercer's conditions can be generated as:

$$k(x, x') = (\varphi(x), \varphi(x')), \tag{2.4}$$

Therefore, the optimal weights w can be computed in feature space. The use of kernels is the key in SVM/SVR applications. It provides the capability of mapping non-linear data into "feature" spaces that are essentially linear, where the optimization process can be duplicated as in the linear case. The use of Gaussian kernels appears to be the most prevalent choice, but typically empirical analyses are necessary in the selection of an appropriate kernel function.

Furthermore, the SVR has been widely studied by researchers and applied in a variety of real-world projects. Some modifications and extensions have been successfully applied to this standard SVR algorithm for time series prediction. For example, Least Squares SVM (Rubio, Pomares, Rojas, & Herrera, 2011) which is the state-of-the-art in kernel methods for regression; outlier detection with SVR by robust regression (Colliez et al., 2006), which significantly improved the robustness against outliers compared to traditional SVR; and Multi-kernel SVR using Linear Programming method (Z. Zhang, Gao, Tian, & Yue, 2016) have considerably improved predictive accuracy and interpret-ability of regression forecasting. However, there are some limitations and drawbacks in the process of SVR and its extended algorithms. These include high computational complexity, parameter dependency and the requirement of full information for input data.

2.2.2 Extreme Learning Machine and Its Extended Algorithms

During the past decade, ELM has been extensively studied in theory and application. The ELM learning frameworks' randomized strategies for nonlinear feature construction have drawn great interest in the computational intelligence and machine learning community (Saxe et al., 2011; Widrow et al., 2013; Le et al., 2013). There are many extension algorithms that have been proposed in recent years. This section, it focuses on introducing the theory, merits, and demerits in extreme learning machine and its extended algorithms.

2.2.2.1 Extreme Learning Machine

ELM was proposed for Single-hidden Layer Feed-forward Networks (SLFNs) where the hidden layer need not be neuron (G.-B. Huang & Chen, 2007, 2008). The output function of ELM for generalized SLFNs is shown as follow:

$$f(x) = \sum_{i=1}^{N} \beta_i H_i(x) = H(x)\beta, \tag{2.5}$$

where $\beta = [\beta_1, \beta_2, \ldots, \beta_N]^T$ is the output weight vector between the hidden layer of N nodes to the output node, and $H(x) = [H_1(x), H_2(x), \ldots, H_N(x)]$ is a nonlinear feature mapping, which represents the output vector of the hidden layer with respect to the input x. The $H_i(x)$ can be computed by the following equation:

$$H_i(x) = G(a_i, b_i, x), \tag{2.6}$$

where, $G(a, b, x)$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems (Huang et al., 2004; G.-B. Huang, Zhu, & Siew, 2006). The most common mapping functions (a mapping function (G)) include Sigmoid

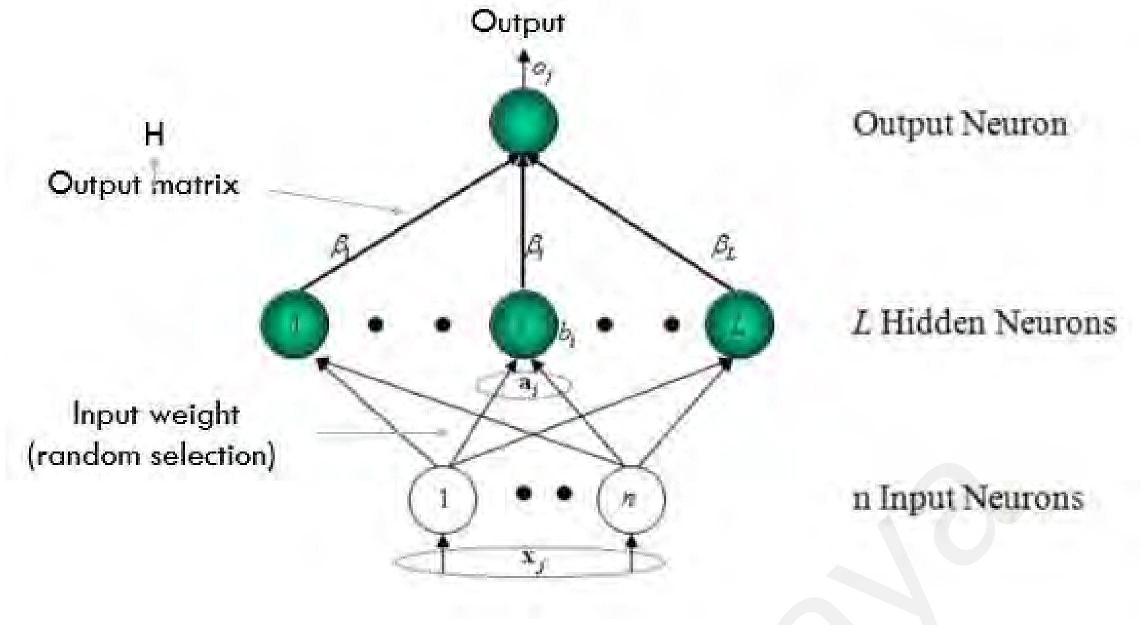


Figure 2.1: Construction of ELM (Huang et al., 2004).

function, Gaussian function, cosine function, tangent function, and Multi-quadric function. Furthermore, in the computing process of mapping function, the hidden node parameters (the input weights of ELM (a) and the bias of ELM (b)) are randomly selected. In the other word, ELM randomly initialized the hidden layer to map the input data into a feature space by a mapping function. The construction of ELM is shown in Figure 2.1.

In ELM, the hidden node parameters (the input weights of ELM (a) and the bias in ELM (c)) are randomly generated according to any continuous probability distribution instead of being explicitly trained, leading to remarkable efficiency compared to traditional neural networks, including Back-propagation algorithm (BP) and SVR (G. Huang, Huang, Song, & You, 2015). However, due to the random selection of input weights in ELM, the prediction results are different when ELM is used in the same time series data repeatedly. At the same time, the number of hidden nodes, which is a user-defined parameter in ELM, is like a double-edged sword that impacts the prediction accuracy. A large number of hidden nodes, to some extent, increases the forecasting accuracy for a huge data set, while it sometimes also leads to over-fitting for a small data set.

2.2.2.2 Kernel Extreme Learning Machine

In order to overcome the limitations of ELM, KELM, which was proposed by Huang, applied a kernel method to create a kernel matrix based on training input data (x) by kernel method to replace the hidden layer output matrix of ELM.

$$K = k\left(x, x\right), \tag{2.7}$$

where K is kernel matrix of the input data x, k stands for the function of kernel method. the output weights (β) of KELM in each step can be computed by (2.8).

$$\beta = \left(K + \frac{I}{C_r}\right)^{-1} Y, \tag{2.8}$$

where, β are an output weight of KELM, I is an identity matrix, Y represents the target data, and Cr is a regularization parameter. Comparing with ELM, the number of hidden layers in KELM need not be specified and its output weight does not randomly select. the prediction value ($\hat{y}$) can be computed by the kernel function (k) and β of the model, as shown in (2.9).

$$\hat{y} = K^T \beta, \tag{2.9}$$

where, transpose operator (T) denotes transpose operator.

It should be noted that, in the KELM, any kernel functions which satisfying Mercer's condition can be employed, e.y., Radial Basis Function (RBF), Linear Function (LF), Polynomial Function (PF), and Wave Function. There are different number of kernel parameters thats define in these kernel functions, such as RBF with one kernel parameter, and PF with two kernel parameters. Although KELM has solved some problems of ELM, including the definition of number of hidden nodes and a lack of deterministic prediction, there are two main limitations have appeared in KELM. Firstly, the kernel method generates the high dimension matrix for the input data, which leads to the complex computation in the learning part. Secondly, kernel parameter also plays a vital role in the forecasting result of KELM.

Furthermore, there are other several researches on offline time series prediction applying extended ELM algorithm in different datasets, which have good performance. For example, ensemble algorithm which also plays a significant role in time series prediction, the adaptive ensemble extreme learning machines achieved a test error comparable to the best method while keeping adaptivity (Van Heeswijk et al., 2009), and Lian et. al. proposed a novel neural network technique, which was called ensemble of extreme learning machine, performed well in landslide hazard. Moreover, the different extended algorithms were also applied in the various datasets. Such as, optimized extreme learning machine for financial time series prediction (Dash, Dash, & Bisoi, 2014), extreme learning machine based probabilistic forecasting method for wind power (Wan, Xu, Pinson, Dong, & Wong, 2014), the extreme learning machine improved price intervals forecast accuracy by incorporating bootstrapping method, (X. Chen et al., 2012) and extreme learning machine with adaptive metrics of inputs had good performance in fashion retailing (Xia, Zhang, Weng, & Ye, 2012).

Although ELM and its offline extended algorithms have demonstrated good performance in time series prediction as above discussion, the main limitations of ELM are, such as the lack of deterministic prediction and parameter dependency of hidden neurons. Therefore, the prediction performance still have room for improvement.

2.2.2.3 Online Sequential Extreme Learning Machine

With the wide application of ELM and its extended algorithms, the ability of self-renewal in prediction model becomes necessary, especially for real-world projects. In 2006, Liang et al. developed an online sequential learning algorithm for single hidden layer feedforeward networks with additive or RBF hidden nodes in a unified framework that was called Online Sequential ELM (Liang et al., 2006; G.-B. Huang & Siew, 2005). It does not require retraining whenever a new data is received. In OS-ELM, there are two main steps in the learning part. Firstly, in initialization phase, the initial size of training data in OSELM (ν) is define. Then based on these initial training data and the hidden node parameters (the weight vector connecting the input layer to the hidden nodes and bias of the hidden nodes) that are selected randomly, the initial hidden layer output matrix (H_0) is calculated. The initial output weights in OS-ELM (β_0) can be calculated by the following equation:

$$\beta_0 = V_0 H_0^T Y_0, \tag{2.10}$$

where T denotes transpose operator, $V_0 = (H_0^T H_0)^{-1}$, and Y_0 is corresponding to the target values in the initial training data.

The second step is sequential learning phase with $l = \{\nu + 1, \ldots, L\}$, L is the number of training data. The partial hidden layer output matrix (H_l) for the l-th training data is calculated based on the l-th new training input data. The l-th output weights (β_k) are updated by Equation (2.12).

$$V_{(l)} = V_{l-1} - V_{l-1} H_l^T (I + H_l V_{l-1} H_l^T)^{-1} H_l P_{l-1}, \tag{2.11}$$

$$\beta_l = \beta_{l-1} + V_l H_l^T (Y_l - H_l \beta_{l-1}), \tag{2.12}$$

where I is a sparse identity matrix.

Then set $l = l + 1$ and recalculate the hidden layer output matrix and update output weight by new received data until l is equal to the number of training data (L). Finally, Liang et al. proved that the OS-ELM was faster than the other sequential algorithms and produced better generalization performance in the time series prediction.

However, there are still some limitations impact on the performance of time series prediction. Firstly, the hidden node parameters are selected randomly, which directly causes the lack of prediction deterministic. Secondly, parameter dependency has also affected on the prediction results, including the initialized size of training data and the number of hidden nodes.

2.2.2.4 Online Sequential Extreme Learning Machine with Kernel

In order to solve the limitations of OS-ELM, Scardapane et al. (2015) proposed an algorithm for online learning with Kernel Extreme Learning Machine (Scardapane et al., 2015), which was called Online Sequential Extreme Learning Machine with Kernel. In KOS-ELM, there are two phases in the learning part, including initialization and updated phases. It is assumed that the training input data ($S = \{S_{(1,\cdot)}, S_{(2,\cdot)}, \ldots, S_{(L,\cdot)}\}$) and the corresponding target Y ($Y = \{Y_1, Y_2, \ldots, Y_L\}$). In the initialized phase, the size of initial training data is one ($s = 1$) and the initial output weight $\beta_1 = Q_1 Y_1$ with $Q_1 = (C^{-1} + k(S_1, S_1)^{-1})$ (function $k(\cdot,\cdot)$ is Gaussian kernel). In KOS-ELM, the size of output weight will increase when the new training samples are received. In the update phase, $l \in \{2, \ldots, L\}$, the output weights (β_l) with interval coefficient Q_l can be calculated by the following equations:

$$g_l = [k(S_l, S_1), \ldots, k(S_l, S_{l-1}))]^T, \tag{2.13}$$

$$z_l = Q_{l-1} g_l, \tag{2.14}$$

$$r_l = C^{-1} + k(S_l, S_l) - z_l^T g_l, \tag{2.15}$$

$$Q_l = \begin{bmatrix} Q_{l-1} r_l + z_l z_l^T & -z_l \\ -z_l^T & 1 \end{bmatrix} \tag{2.16}$$

$$e_l = Y_l - g_l^T \beta_{l-1}, \tag{2.17}$$

$$\beta_l = \begin{bmatrix} \beta_{l-1} - z_l r_l^{-1} e_l \\ r_l^{-1} e_l \end{bmatrix} \tag{2.18}$$

where, k is kernel function, C is a regularization parameter that generally is defined by 1, and e stands for prediction error.

However, the main problem in KOS-ELM is, due to the characteristic of kernel function, complex computation in the learning part. This is especially for the large scale data, the kernel matrix and interval matrix, such as kernel matrix (g) and the interval matrix (Q) has become huge in the process of computation, which directly leads to slow training speed.

2.2.3 Strengths & Weaknesses of Prediction Models

Based on above discussion on different kind of prediction models, it is found that ELM family played a significant role in time series prediction. They have shown the outstanding performance rather than other classic models, such as SVR and ESN. However, these models still have its exited drawbacks. Table 2.1 shows their advantages and disadvantages. The series ELM family models, whether in offline or online learning algorithms, have better forecasting performance than other classic models. In aspect of offline learning models, KELM overcomes the drawbacks of ELM and other classic models, such as ESN and ELM. It does not only have the deterministic prediction ability due to replacement of kernel method for random part of ELM, but it also has super outstanding prediction ability comparing with other neural networks. Thus, KELM based approach for improving prediction performance and solving existing problems in the offline learning model. On the other hand, according to the review of online learning models, KOS-ELM has unique advantages and outstanding prediction performance among other online learning algorithms. Thus, KOS-ELM will be applied in this thesis for improving prediction performance and solving existing problems in the online learning model.

Table 2.1: The strengths and drawbacks of Prediction Models

Algorithm	Author	Advantages	Disadvantages
SVR	(Drucker et al., 1997)	Wide applications Good generalization Good performance in time series and classification	Multiple parameters need to be defined Computation complexion Only short-term prediction horizon is considered
ESN	(Jaeger, 2001)	Generate a dynamical reservoir in the learning part Calculates current neuron based on current input information and previous neuron	Lack of deterministic prediction Only short-term prediction horizon is considered Multiple parameters need to be defined
ELM	(Huang et al., 2004)	Generalization ability better than SVR Less parameters than SVR Faster than SVR	Lack of deterministic prediction One parameter need to be defined Only short-term prediction horizon is considered
KELM	(G.-B. Huang & Siew, 2005)	Deterministic prediction Improved prediction performance comparing with ELM and SVR	Only short-term prediction horizon is considered Kernel parameter needs to be defined Complex computations No connection between hidden neurons
OS-ELM	(G.-B. Huang et al., 2006)	Updating model when new data coming Good performance rather than other online learning algorithms	Parameter dependency Lack of deterministic prediction Only short-term prediction horizon is considered Does not solve concept drift
KOS-ELM	(Scardapane et al., 2015)	Stable prediction Good prediction performance rather than OSELM	Parameter dependency Complex computation Only short-term prediction horizon is considered Does not solve concept drift

2.3 Adaptive Kernel Filter

In the paper by (Scardapane et al., 2015), Fixed-Budget (FB) and Approximate Linear Dependency (ALD) as sparsification criteria are applied to discard nodes and select input data for processing in the the updated phase of KOS-ELM. The main aim of FB criterion in KOS-ELM is to avoid complex computation and keep the reasonable size of hidden nodes in the learning part. It assumes that the number of training data is L and the maximum size of hidden neurons in KOS-ELM is F. The pruning sparsification criterion is to decide if the size of the hidden nodes (l) in the learning process of KOS-ELM are larger than the maximum size of hidden neurons in KOS-ELM (F) or not. If the hidden nodes is larger than the maximum number of hidden nodes, the least significant pattern is discarded from the dictionary based on the minimum error pattern (e_pat).

$$e\_pat = \frac{absolute(\beta)}{diag(K\_inv)}, \tag{2.19}$$

where, $diag$ represents diagonal matrix, $Kinv$ is an extend kernel matrix that is based on the value of K_inv in previous input data and kernel matrix. After discarding, the output weight (β) is computed by the inverse of a matrix in which column and row are removed, and the memory label ($ymem$) whose is discarded.

$$\beta = K\_inv \;\; ymem, \tag{2.20}$$

where K_inv and $ymem$ are the value after discarding the row that has minimum error pattern. The pruning process is shown in the Algorithm 1.

This algorithm is called Kernel Online Sequential Extreme Learning Machine with Fixed-Budget Filter (FB-KOS-ELM) when FB is employed in the learning part of KOS-ELM. In the process of training part, the output weights will be updated when the size of

Algorithm 1 Pruning criterion of FB

```
1: if $l > F$ then
2:     Calculate error pattern ($e\_pat$) by (2.19);
3:     Find the minimum error pattern and label the row of dictionary that has minimum
       error pattern;
4:     Discard all row of dictionary that has minimum error pattern in the memory label
       ($ymem$) and corresponding input data in the memory ($mem$);
5:     Update the output weight ($\beta$) by (2.20)
6: end if
```

the hidden nodes are larger than the defined maximum hidden nodes of KOS-ELM. After training, the number of hidden nodes will be set in the fixed number and discarded those samples with reduplicative or redundant information of the training samples.

On the other hand, KOS-ELM was combined with ALD in order to select training data in the second phase of KOS-ELM and reduce the computation of kernel. It was called Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency Filter (ALD-KOS-ELM). Using ALD, the input data is rejected if its distance (Δ) to the linear span of the current dictionary (*mem*) in the feature space is less than the threshold of ALD (φ). If the ALD value (Δ) is more than φ, the corresponding input data will put into the memory data (*mem*). The distance of input data (S_{k+1}) to the linear span of the current dictionary in the feature space can be calculated by equation (2.21).

$$\Delta_l = ktt_l - kt_l^T Q_l kt_l, \tag{2.21}$$

where, Ql is the interval coefficient with the information of previous input data in KOS-ELM, $ktt_l = k(S_l, S_l)$ that is kernel matrix of S_l, and $kt_l = k(mem, S_l)$ that is kernel matrix of current dictionary (*mem*) and input data (S_l).

In ALD-KOS-ELM, the initialization phase is similar to KOS-ELM. In the updating phase, ALD-KOS-ELM employs ALD criteria to decide whether the input data need to update or retrain. The process of ALD criteria in KOS-ELM is shown in Algorithm 2.

Algorithm 2 ALD criterion in KOS-ELM

```
1:  for $l \in \{2, ..., L\}$ do
2:      if $\Delta \geq \varphi$ then
3:          Calculate $Q_l$ by equation (2.16);
4:          Update outputweight $\beta_l$ by equation (2.18);
5:          Update dictionary $mem$;
6:      else
7:          Update $V_l$ by equation (3.18);
8:          Update $\beta_k$ by equation (3.19);
9:      end if
10: end for
```

Compared with the results of the experiment in (Scardapane et al., 2015), ALD has superior performance in time series prediction than FB and solved the problem of expensive computation in KOS-ELM. At the same time, although KOS-ELM obtained progress in time series prediction, it still has some limitations. For example, KOS-ELM only predicts in one step. Moreover, it keeps the old information and cannot be overwritten by the information of neurons when time progresses.

2.4 Recurrent Models for Time Series Prediction

In the time series prediction purposes, recurrent neural networks have gained a lot of attention. The majority of studies focused on supervised learning models, which seemed to be the most popular ones. For example, Pearlmutter (Pearlmutter, 1989) in 1990 presented the basic contributions on the subject by distinguishing algorithms for full recurrent networks from those used for local feedback networks. Moreover, Support Vector Machine, as a popular supervised learning model, was used in solving nonlinear regression and time series problems. In 2005, Pai (Pai & Hong, 2005) proposed a recurrent support vector machines with genetic algorithms (RSVMG) for forecasting electricity load. The empirical results proved that the RSVMG outperformed the SVM model, artificial neural network (ANN) model and regression model. The other outstanding model named Recurrent Extreme Learning Machine, which was adapted to train a single hidden layer

Jordan recurrent neural network, was proposed as a novel approach to forecast electricity load more accurately by Ertugrul (Ertugrul, 2016).

Generally, temporal relationship is often captured using recurrent in neural networks, which has lead to the different architectures. The three main architectures include globally recurrent networks, locally recurrent networks and nonlinear autoregressive with exogenous input networks (Bouchachia, 2009). In the globally recurrent networks, hidden nodes provides a context and are globally fed back as a new input. In locally recurrent networks, the feedback connections are allowed only from neurons to themselves. In the nonlinear autoregressive with exogenous input networks architecture, the output of the network is fed back to the input layer.

Firstly, in aspect of globally recurrent networks, the network outputs are fed back to the network inputs through time delay units. Generally, it was used for nonlinear multi-step ahead prediction models. For example, Su and McAvoy, by compared recurrent networks and feedforward networks for identification of chemical processes, proved recurrent networks were demonstrated to be superior to feedforward networks for long-term predictions and multi-step predictions (Su et al., 1992). The other one is a simple method that was based on back-propagation proposed by Werbos (P. J. Werbos, 1990), which could be applied to neural networks of varying degrees of complexity. Secondly, in the aspect of locally recurrent networks, the output of a hidden neuron is fed back to its input through one or several time delay units. There are many studies used locally network for different applications (Frasconi, Gori, & Soda, 1992; Zhang & Morris, 1995; Barbounis & Theocharis, 2006; Mostafa, Teich, & Lindner, 2011). For example, Zhang et. al proposed a sequential orthogonal training strategy which allows hidden neurons being gradually added to avoid unnecessarily large network structure (Zhang & Morris, 1995). Experimental results in research by (Barbounis & Theocharis, 2006) on the wind prediction problem demonstrated that the proposed algorithms exhibited enhanced performance, in terms of the accuracy of the attained solutions, compared to conventional gradient-based methods and the recurrent forecast models outperformed the atmospheric and time-series models. The main advantage of these models is that the neurons of models can interaction and obtain more information for training. Although recurrent networks can theoretically store information from arbitrarily long time age in the context layer, it also appears gradient exploding problem in the practice application when it is applied in time series prediction. The last architecture is the Nonlinear Autoregressive with Exogenous Input (NARX). Bouchachia proposed a novel ensemble learning approach based on recurrent radial basis function networks for multi-step ahead time series prediction(Bouchachia, 2009). The experimental results showed that the model improved the prediction accuracy and had successfully applied NARX to solve multi-step ahead prediction problem in time series data.

In the process of time series prediction in real-world applications, the most important things about prediction model are that the model can forecast multiple steps that are easy to track the tendency of observed target and high prediction accuracy in the different prediction steps. Therefore, in order to improve the forecasting ability of proposed model, this study employs the advantages of different recurrent architectures to solve restriction of prediction horizon of prediction model by the idea of NARX and improve the prediction performance by connecting neurons of model.

2.5 Reservoir Computing

In the offline and online learning of extended ELM algorithms, the neurons of learning part keep the old information and cannot be overwritten with time progresses. In the early work in (Buonomano & Merzenich, 1995), a random network of spiking neurons with short-term plasticity (dynamic synapses) was used. It used a supervised correlation-based learning rule to train a separate output layer. The idea of using a random recurrent network, which is left untrained by a simple regression/classification technique, was later independently reinvented in research by (Jaeger, 2001) as the Echo State Network. It is a most common implementation of reservoir computing.

The Echo State Network (ESN) consists of a random, recurrent network of analog neurons that is driven by a high-dimensional map, and the activations of the neurons are used to do linear regression or classification. Generally, in order to build interconnection for the hidden nodes in the neural network, Reservoir Computing is used input information to fed into a dynamical system that maps the input data to a high dimension with the information of input data. ESN is one of the major types of RC, which is a new random projection and was created by (Jaeger, 2001). The RC often focuses on a specific type of reservoir interconnection and node type (Verstraeten et al., 2007). Typically, input information is fed into a dynamic system called a reservoir. It maps the input data into a high dimensional space. It has been described as having several different interconnection structures and analog neurons. The main idea is to drive a random, large and fixed recurrent neural network with the input signals, thereby inducing this *reservoir* network in neurons–a mechanism is trained to read the state of the reservoir that is mapped for computing output signals. If calculating output weights in the training part by applying the reservoir of ESN that contains input data, then more features can be obtained in the output weight by this method.

In recent years, the two machine learning approaches, Extreme Learning Machines and Reservoir Computing, in particular Echo State Networks, have attracted a great interest for information processing because of their simplifying training process. For example, in the paper by (Ortín et al., 2015), RC was successfully unified with ELM. This approach is the use of intermediate reservoir space, where inputs are randomly selected in a nonlinear form, in order to create a new space $r(n+1)$, which can be computed by the following equation:

$$r(n+1) = F\left(\gamma W_{\text{in}} S(n) + \zeta W r(n)\right), n = 1, \ldots, L \tag{2.22}$$

where $r(n)$ is the state of the reservoir neurons, F is an activation function in ELM, γ and ζ are input scaling factor and connecting scaling factor, respectively, $S(n)$ is the n-th row in the training data, W is a matrix that indicates reservoir neuron connectivity, and W_{in} represents the input weight of the matrix. For the current experiments, W_{in} and W were randomly selected in the initialization, and W was impacted by the magnitude of the largest eigenvalue of W and the spectral radius. It presented an unified photonic implementation of ELM and ESN, which was based on random nonlinear projections of data into a high-dimensional network using an intermediate single layer of neurons.

The neurons in ELM are not inter-connected, while in ESN connectivity provides the fading memory suitable for time-dependent data. In 2017, (McDonald, 2017) presented a framework for implementing reservoir computing and extreme learning machines based on 1D elementary Cellular Automate (CA) is presented, in which two separate Cellular Automata (CA) rules explicitly implement the minimum computational requirements of the reservoir layer: hyper-dimensional projection and short-term memory. It applied ESN to connect neurons of ELM, which was also a kind of recurrent network in order to obtain more information in neurons. Finally, it was concluded that the general results combining the shift rules with the memory rule are encouraging. However, due to random selection of input weights, this space in the reservoir is not stable, which directly impacts the stability of predicting performance. Therefore, a fixed reservoir, which can connect the neurons of prediction model, is supposed to be generated in the model in order to improve the forecasting performance and store more information in neurons from training data.

Therefore, based on the discussion about solving concept drift problem in the models, Table 2.2 summarizes the disadvantages and advantages of main models that solved concept drift problem.

Table 2.2: The summary of models with reservoir computing

Algorithm	Author	Advantages and Disadvantages
ESN	(Jaeger, 2001)	Advantages
		Generate a dynamical reservoir in the learning part Calculates current neuron
		Disadvantages
		Lack of deterministic prediction Only short-term prediction horizon is considered
ELM+ESN	(Ortín et al., 2015)	Advantages
		The reservoir remembers past input information Improves prediction performance rather than ESN and ELM
		Disadvantages
		Lack of deterministic prediction Only short-term prediction horizon is considered
ELM+CA	(McDonald, 2017)	Advantages
		Obtaining more information in neurons Generating a reservoir
		Disadvantages
		Lack of deterministic prediction Only focus on short-term prediction

2.6 Concept Drift

The majority of data sets are non-stationary in the real world. In the non-stationary environments, the data distribution can change over time, which directly results into the phenomenon of concept drift. Concept drift problem, which consists a change in the relationship between input data and target variable over time (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014), also generally appears in the process of learning part in KOS-ELM. A machine learning method also needs to adapt to over time changes in the environment.

Furthermore, concept drift problem, which consists a change in the relationship between input data and target variable over time (Gama et al., 2014), also generally appears in the process of learning part. In the last decade, although there has been expanded the significance of concept drift in machine learning and additionally information mining tasks, lots of researchers pay attention to solve the concept drift problem in the classification.

Generally, there are several concept drift detectors, which have been successfully applied in projects. Taking some typical detectors as example, such as, Drift Detection Method (DDM), which proposed by (Gama, Medas, Castillo, & Rodrigues, 2004), used a binomial distribution and has a good behavior detecting abrupt or gradual changes when the gradual change is not very slow, but it has difficulties when the change is slowly gradual. In that case, the examples will be stored for long time, the drift level can take too much time to trigger and the examples memory can be exceeded. Early Drift Detection Method (EDDM) (Baena-García et al., 2006) has been developed to improve the detection in presence of gradual concept drift. It keeps a good performance with abrupt concept drift. The basic idea is to consider the distance between two errors classification instead of considering only the number of errors. While the learning method is learning, it will improve the predictions and the distance between two errors will increase. Next one is glsPHT (Page, 1954), which is a sequential adaptation of the detection of an abrupt change of the average of a Gaussian signal and the detection problem consists of running two tests in parallel. Adaptive Windowing (ADWIN) (Bifet & Gavalda, 2007) automatically adjusts its window size to the optimum balance point between reaction time and small variance. Since ADWIN keeps bits or real numbers, it can be put to monitor the error rate of the current model. The other popular detector is ECDD (Ross, Adams, Tasoulis, & Hand, 2012). It is a proposal for a drift detection method based on Exponentially Weighted Moving Average (EWMA), used for identifying an increase in the mean of a sequence of random variables. Degree of Drift (DOF) can detect drifts by processing data chunk by chunk, computing the nearest neighbor in the previous batch for each instance in the current batch and comparing their corresponding labels. The last one is Test of Equal Proportions (STEPD). It assumes that 'the accuracy of a classifier for recent *W* examples will be equal to the overall accuracy from the beginning of the learning if the target concept is stationary; and a significant decrease of recent accuracy suggests that the concept is changing' (Nishida & Yamauchi, 2007).

There are various papers that have been proposed methods for learning from drifting concepts, such as Synthetic Minority Over-sampling Technique (SMOTE) in (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), (Moreno-Torres, Raeder, Alaiz-RodríGuez, Chawla, & Herrera, 2012) that presented a unifying framework for concept drift and an ensemble algorithm detecting concept drift in incremental learning in (Elwell & Polikar, 2011). For the online learning of ELM algorithm that the study focuses on, (Budiman, Fanany, & Basaruddin, 2016) proposed an enhancement of OS-ELM and its variant constructive enhancement OS-ELM by adding an adaptive capability for classification, which was named Adaptive OS-ELM (AOS-ELM). It was a single classifier scheme that worked well to handle concept drift problem. It concluded that the proposed model gave better adaptive capability than non adaptive OS-ELM in term of retaining the recognition performance when handing concept drifts. However, in some specific data sets the AOS-ELM accuracy may not exceed the non adaptive sequential ELM. Recently, Mirza and Lin proposed a online learning model named Meta-cognitive Online Sequential Extreme Learning Machine (MOS-ELM) for class imbalance and concept drift learning (Mirza & Lin, 2016) cognition was used to self-regulate the learning by selecting suitable learning strategies.

Furthermore, A feature extraction method for explicit concept drift detection in time series was proposed in (Cavalcante et al., 2016). This method based on the time series features to monitor how concepts evolve over time. Various online methods of drift detection in data stream, including DDM, EDDM, Dynamic Weight Majority (DWM)

(Kolter & Maloof, 2003), and Adaptive expert ensembles (AddExp) (Kolter & Maloof, 2005) were compared in the research by (Mittal & Kashyap, 2015). Results in (Mittal & Kashyap, 2015) show that DDM performed well for detecting changes rather than other methods. Furthermore, ensemble learning techniques are very popular nowadays for detecting concept drift problem, such as Accuracy and Growth rate updated Ensemble (AGE) (Liao & Dai, 2014) and Online Bagging (Oza & Russell, 2001). Both of the algorithms were used for the classification of streaming data with concept drift. However, these kinds of methods spend a lot of time and only used for the classification.

Therefore, based on the above discussion on concept drift detector, however, some types of methods have been proven were not efficient or increasing the training time, including methods by preparing data, detecting and choosing model. Moreover, the majority of methods were applied in solving concept drift problem for classification rather than time series prediction. Table 2.3 summarizes the disadvantages and advantages of all models that solved concept drift problem.

2.7 Research Gap

The main aim of time-series prediction is seeking the trend or situation of the certain object in the future. The time series prediction affects on our daily life, such as weather forecasting, pollution prediction. It has even been applied in economics, water level, financial market, and prediction of earthquake.

According to the review, it found that ELM and its extended algorithms have superior ability for time series prediction. However, they still have limitations in the process of prediction. Firstly, these algorithms have the restriction of prediction horizon, which cannot predict in multiple steps without the corresponding target data. In the other word, the one-step prediction algorithm cannot determine the future trend. Secondly, there are no a stable reservoir in both offline and online learning time series prediction, which

Table 2.3: The summary of models with concept drift detector

Algorithm	Author	Advantages and Disadvantages
AOS-ELM	(Budiman et al., 2016)	Advantages To handle concept drift problem in classification Good performance in classification Disadvantages Lack of deterministic prediction Parameter dependency Only focus on classification
AGE	(Liao & Dai, 2014)	Advantages To handle concept drift problem in classification Improved classification performance Disadvantages Only focus on classification Taking long time for training part
Online bagging	(Oza & Russell, 2001)	Advantages To handle concept drift problem in classification Improved classification performance Disadvantages Only focus on classification Heavy computation
SMOTE	(Chawla et al., 2002)	Advantages To handle concept drift problem in classification Incremental learning Increasing accuracy in classification Disadvantages Only focus on classification Heavy computation
MOS-ELM	(Mirza & Lin, 2016)	Advantages To handle concept drift problem in classification self-regulate learning by strategies Increasing accuracy in classification Disadvantages Only focus on classification lack of deterministic prediction
Feature extraction method	(Cavalcante et al., 2016)	Advantages To handle concept drift problem in time series prediction Increasing accuracy in time series prediction Disadvantages Taking long time to extract features

cannot overwrite the old information in the process of learning part. It directly leads to decrease the learning efficiency in the model, even results to over-fitting problem. More importantly, there is computation complexion in the learning part, especially for the large scale dataset. Thirdly, almost all algorithms have the parameter dependency problem. For example, ELM requires the definition of number of hidden nodes, KELM and KOS-ELM require to define the kernel parameter, and adaptive kernel filters also require set the threshold of ALD and the maximum number of hidden nodes for FB. This problem directly impacts o the performance of prediction. Lastly, the majority of real-world datasets are non-stationary. Due to this characteristic of dataset, the concept drift generally appears in the machine learning application, which greatly impacts on the performance and reliability of time-series prediction in the real-world applications of machine learning (Webb, Lee, Petitjean, & Goethals, 2017). In the process of online learning part, the concept drift problem generally appears.

Therefore, the research gaps of this study can be briefly summarized as follows:

i) There is restriction of prediction horizon problem in the offline and online learning model.

ii) There is no a fixed reservoir that can dissipate and overwrite the old information of neurons in the offline and online learning model.

iii) There is the parameter dependency problem in the process of building offline and online model.

iv) There is the concept drift problem in online learning model for time series prediction and it is rarely took consideration in time series prediction.

2.8 Summary

In this chapter, the fundamentals of offline and online learning models are introduced, and the two kernel adaptive filters are presented. The recurrent models for time series prediction are introduced as well. Furthermore, the conventional studies for reservoir computing are reviewed, especially for the most common reservoir computing model-ESN. Besides, the concept drift which affects the performance of time series prediction is introduced. Finally, the research gaps on time series prediction are highlighted.

As summarized in this chapter, several studies on offline and online learning models for time series prediction have introduced. Although these models have certain advantages, the other problems appear, such as restriction of prediction horizon, no fixed reservoir, parameter dependency problem, complex computation, and concept drift problem. In this thesis, the solutions of the research problems are carried out by corresponding methods, such as multi-step prediction algorithm, new reservoir computing methods, suitable concept drift detector and so on.

CHAPTER 3: METHODOLOGY

In this chapter, in order to bridge the research gap that the last chapter mentioned, there are one offline and two online algorithms that will be introduced in this chapter. They include RKERM with QPSO for offline algorithm, Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency (Meta-RKOS-ELM), and Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism (Meta-RRKOS-ELM-DDM) for online learning algorithm. Before introducing these algorithms, the data type should give some consideration.

3.1 Data Transformation

For all proposed algorithms, the time series data is transformed into a matrix form. It is assumed that the time series data X=[X_1,. . .,X_N] is given, where N is the number of samples. In real-world application of time series prediction, the reasonable period of prediction plays a vital role in trend analysis and reports forecasting information, including stock analysis, water level prediction and weather forecast. In (Bao, Xiong, & Hu, 2014), they proposed that 18 steps ahead is an appropriate long prediction period. Therefore, the size of the prediction horizon and the time window are defined as eighteen in this thesis. In order to generate a intact matrix with eighteen time window and eighteen prediction horizon, we obtain from the first observation (X_1) to the thirty-sixth observation (X_{36}) from the simple time series data (X) to be the first row of matrix, and then the second row of matrix can be generated from the second observation (X_2) to the thirty-seventh observation ((X_{37})) from data (X) until generate the whole matrix. The time series X is transformed to a matrix a time series matrix (M) with dimension of $(N - D - Z + 1) \times (D + Z)$ as shown in (3.1).

$$M = \begin{bmatrix} X_{1,1} & \cdots & X_{1,D} & X_{1,D+1} & \cdots & X_{1,D+Z} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{L,1} & \ldots & X_{L,D} & X_{L,D+1} & \cdots & X_{L,D+Z} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{N-D-Z+1,1} & \ldots & X_{N-D-Z+1,D} & X_{N-D-Z+1,D+1} & \cdots & X_{N-D-Z+1,D+Z} \end{bmatrix} \quad (3.1)$$

where D is the size of the time window, Z is the size of prediction horizon ($Z = \{1, 2, \ldots, Z\}$), and L stands for the number of training data.

According to the size of time window and prediction horizon, the training input data(the training input data (S)) and the training target data in p step (Y_p) are shown in (3.2) and (3.3), respectively.

$$S = \begin{bmatrix} X_{1,1} & \cdots & X_{1,D} \\ \vdots & \ddots & \vdots \\ X_{L,1} & \ldots & X_{L,D} \end{bmatrix} \quad (3.2)$$

$$Y_p = \begin{bmatrix} X_{1,D+p} \\ \vdots \\ X_{L,D+p} \end{bmatrix} \quad (3.3)$$

where p is the p-step in the prediction ($p = 1, 2, \ldots, P$). The testing input matrix (TS)

and TY_p can be defined as follows:

$$TS = \begin{bmatrix} X_{L+1,1} & \cdots & X_{L+1,D} \\ \vdots & \ddots & \vdots \\ X_{N-D-P+1,1} & \cdots & X_{N-D-P+1,D} \end{bmatrix} \tag{3.4}$$

$$TY_p = \begin{bmatrix} X_{L+1,D+p} \\ \vdots \\ X_{N-D-P+1,D+p} \end{bmatrix} \tag{3.5}$$

3.2 Normalization

The data should be normalized before training in both offline and online models. The following equation is applied to normalize the matrix M_{ij} in offline learning algorithm.

$$\hat{X}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \tag{3.6}$$

where x_{max} and x_{min} are the maximum and minimum values in the data matrix, respectively.

Furthermore, in the online learning algorithm, the normalized sample ($\hat{X}_i$) can be set as Algorithm 3.

3.3 Offline Algorithms for Time Series Prediction

In this section, a novel recurrent multi-step ahead prediction model called Recurrent Kernel Extreme Reservoir Machine with Quantum Particle Swarm Optimization is proposed. This model combines the strengths of Recurrent Kernel Extreme Learning Machine and modified Reservoir Computing in order to overcome the restriction of prediction horizon and instable reservoir based on existed Reservoir Computing theory. Furthermore, QPSO

Algorithm 3 Data normalization for online learning algorithm

```
Require: - Size of time series data ($N$);
         - Time series data ($X$);
         - The maximum of time series data ($X_{max}$);
         - The number of digit in $X_{max}$ ($z$);
Ensure:  - Normalized samples ($\hat{X}$);
 1: Find out the maximum value of time series data $X_{max}$;
 2: Based on $X_{max}$, find out the number of digit in $X_{max}$;
 3: Main loop:
 4: for $n \in \{1, \ldots, N\}$ do
 5:     Compute normalized sample $\hat{X}_n = X_n/10^z$;
 6:     if $\hat{X}_n > 1$ then
 7:         $X_{max} = X_n$;
 8:         Let z = the new number of digit in $X_{max}$;
 9:         $\hat{X}_n = X_n/10^z$;
10:     end if
11: end for
```

is used to optimize parameters of kernel method and leaking rate of reservoir computing in RKERM, which can solve the problem of parameter dependency.

In literature review, it was mentioned that KELM replaced the hidden layer mapping in ELM with a kernel function, which solved the problem of deterministic prediction in ELM (Gurpinar, Kaya, Dibeklioglu, & Salah, 2016). However, the size of prediction horizon in KELM for time series prediction is still restricted by the size of target data, which limits the generalization of prediction in real world applications. Therefore, the recurrent multi-step-ahead-prediction is applied to solve this kind of problem in this study.

3.3.1 Recurrent Multi-step ahead Algorithm

The recurrent multi-step ahead algorithm is a feedback network which backward from the output of current step to the input of the next step. Although ELM and its extended algorithms can be applied to predict multiple targets in regression, the number of target data determines the size of prediction horizon. It restricts the generalization of prediction in the application of the real world. In order to extend the prediction horizon, the recurrent multi-step ahead algorithm is applied in the prediction model. Figure 3.1 shows the architecture of recurrent multi-step ahead prediction algorithm.

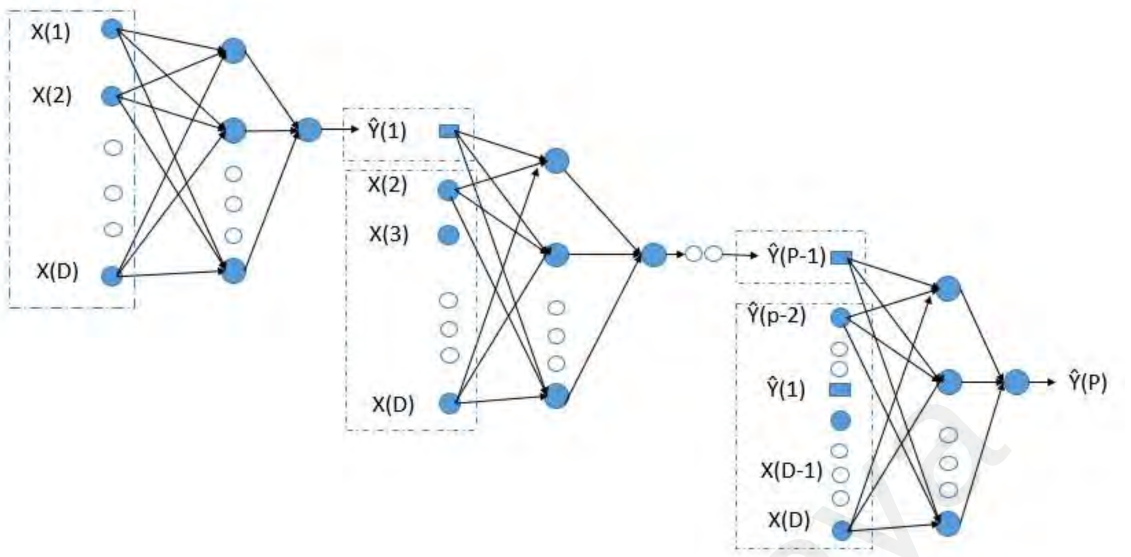


Figure 3.1: Architecture of Recurrent Model

In the Figure 3.1, D is the size of input data, P is the prediction horizon, and $(D + 1) \geq P$. It is assumed that the input data and forecasting values of the prediction model are S and $\hat{y}$ in the first step, respectively. In order to reduce the computation in the learning, recurrent multi-step ahead algorithm maintains a constant size of time window. The training input data in the $(p + 1)$-th step can be represented as

$$S_{p+1} = \begin{bmatrix} X_{1,1+p} & \cdots & X_{1,D} & \hat{y}_{1,1} & \cdots & \hat{y}_{1,p} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{L,1+p} & \ldots & X_{L,D} & \hat{y}_{L,1} & \cdots & \hat{y}_{L,p} \end{bmatrix} \tag{3.7}$$

where $\hat{y}_p$ represents the predicted value of training data in the p-th step.

The recurrent multi-step-ahead-prediction algorithm is applied with KELM. It is applied in prediction model as a external recurrent algorithm, which feeds the current prediction result to the input of next step in order to solve the restriction of prediction horizon. The following subsection will introduce the details of a new algorithm called RKELM.

3.3.2 Recurrent Kernel Extreme Learning Machine

In RKELM, a kernel matrix based on training input data (S) is created by kernel method to replace the hidden layer output matrix of ELM.

$$K = k(S, S), \tag{3.8}$$

where K is a kernel matrix of S and k stands for the kernel function. The output weight in the p step (β_p) of RKELM can be computed by (3.9).

$$\beta_p = \left(K + \frac{I}{C_r}\right)^{-1} Y_p, \tag{3.9}$$

where β_p is an output weight of RKELM in the p-th step; I is an identity matrix; Y_p represents the target data (Y) in the p-th step, and Cr is a regularization parameter.

The predicted values in the p-th step can be computed by (3.10).

$$\hat{y_p} = K^T \beta_p, \tag{3.10}$$

where T denotes transpose operator.

The recurrent multi-steps-ahead algorithm in RKELM is defined for a single hidden-layer feed-forward neural network. It aims at providing new training data in the next step for the prediction. In our proposed model, in order to keep the same size of time window and reduce training computation overhead, we generate training input data (S_{p+1}) in the ($p + 1$)-th step by (3.7). Then, the kernel matrix and output weight in the ($p + 1$)-th step can be calculated by (3.8) and (3.9), respectively.

To summarize the procedure of our proposed method, training input data is S and the target value in the p-th step is Y_p. The kernel matrix of training input data is computed by a

```
Algorithm 4 Recurrent Kernel Extreme Learning Machine
Require: - Size of prediction horizon $P$;
         - Time window size $D$;
         - Number of training data $L$;
         - Training data $S$ by (3.2);
         - Target data $Y_p$ by (3.3);
         - Kernel parameter $\delta$;
         - Output weight of RKELM in the $p$-th step $\beta_p$;
         - Prediction value in the $p$-th step $\hat{y}_p$;
Ensure: - Output weight $\beta$;
        - Prediction value $\hat{y}_p$;
1: Normalize data between zero and one by (3.6);
2: Set initial training data $S$ by (3.2), and target data $Y_{.,1}$ by (3.3);
3: Main loop:
4: for $p \in \{1, \dots, P\}$ do
5:     Compute kernel matrix based on input in each step
6:     Compute output weight $\beta_p$; by (3.9);
7:     Compute prediction value $\hat{y}_p$ by (3.10);
8:     Add prediction value into training data as a new
           training input data for next step by (3.7);
9: end for
```

widely used kernel called RBF kernel. Then the output weight of each step in RKELM can be calculated by (3.9). Finally, the training prediction value in p step ($\hat{y}_p$) are computed by (3.10). Based on the current forecasting value and training input data, the training input data of the next step (S_{p+1}) can be calculated by (3.7). The pseudo-code of RKELM is shown in Algorithm 4.

However, the neurons of RKELM is computed by the kernel matrix from the training data. The information of neurons is directly from current training data. It does not impact or update by the previous neurons.

3.3.3 Reservoir Computing

The RC often focuses on a specific type of reservoir interconnection and node type (Verstraeten et al., 2007). Typically, input information is fed into a dynamic system called a reservoir. It maps the input data into a high dimensional space. ESN is one of the major types of RC, which is a new neural network that was created by (Jaeger, 2001).

It has been described as having several different interconnection structures and analog neurons. The main idea is to drive a random, large and fixed recurrent neural network with the input signals, thereby inducing this *reservoir* network in neurons–a mechanism is trained to read the state of the reservoir that is mapped for computing output signals. We apply the reservoir of ESN that contains input data to calculate output weights in the training part. Many more features can be obtained in the output weight by this method. In (Ortín et al., 2015), RC was successfully unified with ELM. This approach is a use of intermediate reservoir space, where inputs are randomly selected in a nonlinear form, in order to create a new space $r(n+1)$, which can be computed by the following equation:

$$r(n+1) = F\left(\gamma W_{\text{in}} S(n) + \partial W r(n)\right), n = 1, \ldots, L \tag{3.11}$$

where r is the state of the reservoir neurons, F is an activation function in ELM, γ and ∂ are input and connecting scaling factors, respectively, $S(n)$ is the n-th row in the training data, W is a matrix that indicates reservoir neuron connectivity, and W_{in} represents the input weight of the matrix. For the current experiments, W_{in} and W were selected randomly in the initialization, and a matrix that indicates reservoir neuron connectivity (W) was impacted by the magnitude of the largest eigenvalue of W and the spectral radius.

However, due to random selection of input weights, this space in the reservoir is not stable, which directly impacts the stability of prediction performance. Therefore, we apply kernel method to replace the random weight in (3.11) in order to solve the problem of stability in the reservoir. In this study, the modified ESN is used to make a stable reservoir that contains the information of all input data before calculating output weights in the training part of RKELM.

3.3.4 Recurrent Kernel Extreme Reservoir Machine

RC was successfully unified with ELM based on a single time-delayed neuron in (Ortín et al., 2015). However, due to the characteristic of random projection algorithm (ESN and ELM), the input weights and bias in their algorithms are randomly selected. This leads to unstable prediction result. Furthermore, these algorithms only focus on the one-step prediction, which cannot be used in real-world project. Therefore, this study unifies a modified ESN with RKELM in order to solve these two limitations and make a stable reservoir with high amounts of short-term memory. This reservoir can be made to remember past input information by forcing it to reproduce the information in the current input data. In other words, the current state of the reservoir can be calculated based on the past state of the reservoir and finally the reservoir is stable.

Here, we assume that the training data and its corresponding target in p-th step are S and Y_p, respectively. The state of reservoir neurons r_i in the training data of RKERM can be computed by the following equation:

$$r_i = \alpha F\left(S(i), S\right) + (1 - \alpha) r_{\text{i-1}}, \tag{3.12}$$

where $r_1 = \alpha F(S(1), S)$, F is a Gaussian function, the coefficient α is the leaking rate of RC ($\alpha \in [0, 1]$), and i is the index of the training data, $i = (2, \ldots, L)$. The output weight of RKERM in the p-th step can calculated by the state of reservoir and target values of the p-th step in the following equation.

$$\beta_p^R = Y_p r (r r^T + I)^{-1}, \tag{3.13}$$

where β_p^R is the output weight of reservoir computed in the p-th step, Y_p is the target value of the training data in the p-th step.

Then, the prediction output values in the p-th step can be computed by the following equation:

$$\hat{y}_p = \beta_p^R r, \tag{3.14}$$

In the first step, we set the number of training samples as the size of the reservoir, where input values from training data are transformed into L-dimensional r space as in (3.12). Referring to the states of the reservoir as neurons, the output weight of reservoir in p step (β_p^R) in the first step can be computed by (3.13). Therefore, the training output values of the first step, $\hat{y}_1$, are calculated by (3.14). In the test part, the state of reservoir neurons of test data can be calculated by (3.12) with test input (TS) and target (TY). Then, the first-step output values of the test data ($\hat{y}_1$) are computed by the output weight β_p^R, and the state of reservoir neurons of test data by (3.14). In the following step ($p + 1$), we update training input data by (3.7) and test input data by the following equation:

$$TS_{p+1} = \begin{bmatrix} X_{L+1,1+p} & \cdots & X_{L+1,D} & \hat{y}_{1,1} & \cdots & \hat{y}_{1,p} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{N-D-P+1,1+p} & \ldots & X_{N-D-P+1,D} & \hat{y}_{N-D-P-L+1,1} & \cdots & \hat{y}_{N-D-P-L+1,p} \end{bmatrix} \tag{3.15}$$

where $\hat{Y}_p$ represents the predicted values of test data in the p-th step, $(N - D - P - L + 1)$ is the number of test samples.

The main difference between RKELM and RKERM is the architecture of the neural network. Figure (3.3.4) and (3.3.4) show the architecture of RKERM and RKELM, respectively. They indicate that each of the neurons in RKERM is computed by kernel method and previous neuron information, which not only obtains more information from input data than the method of RKELM, but also obtains stable output weights based on the reservoir. In the meantime, the process of computing output weights and predicted values

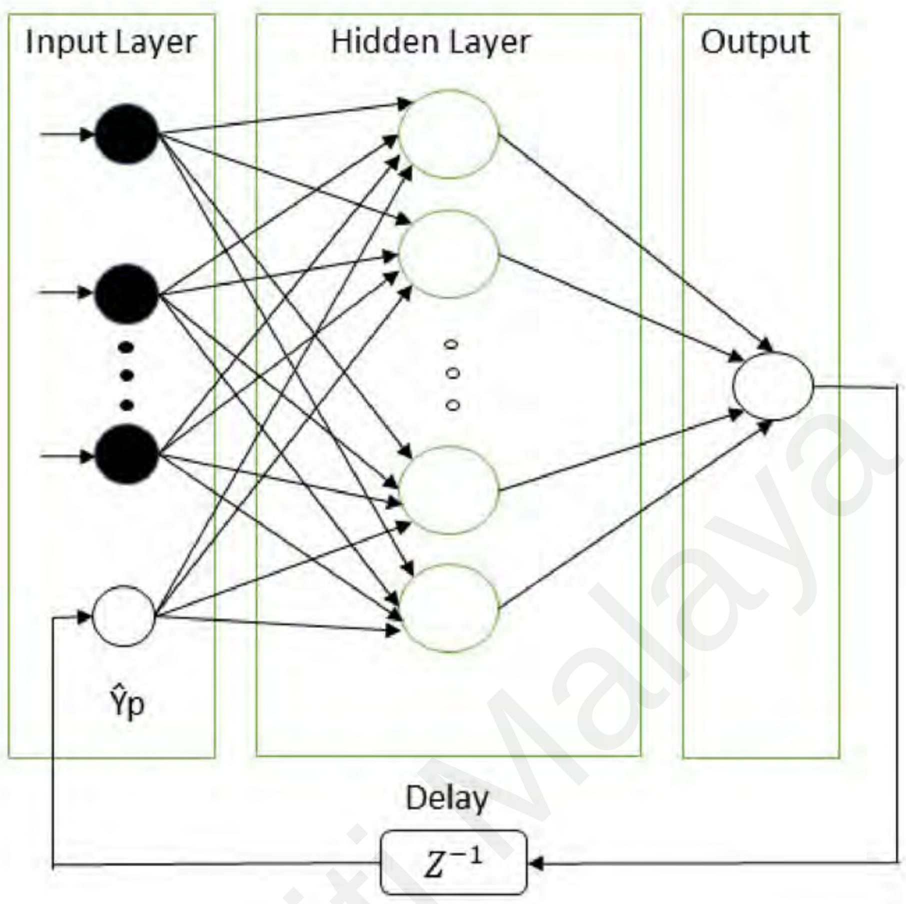


Figure 3.2: Architecture of RKELM

also show the difference between RKERM and RKELM. Table 3.1 shows the different training and forecasting processes in RKERM and RKELM. In the training process, the kernel matrix of RKELM in the training part is computed by batch. However, due to the relationship of the neurons, the reservoir of RKERM in the training part is computed one by one. The difference is also presented in the process of computing output weights between the two algorithms. The direct evidence is that RKERM has fewer parameters compared to RKELM.

In this section, the unify modified RC and RKELM is combined to enhance the stability of the reservoir and the ability of prediction for time series. Equation (3.12) can be utilized

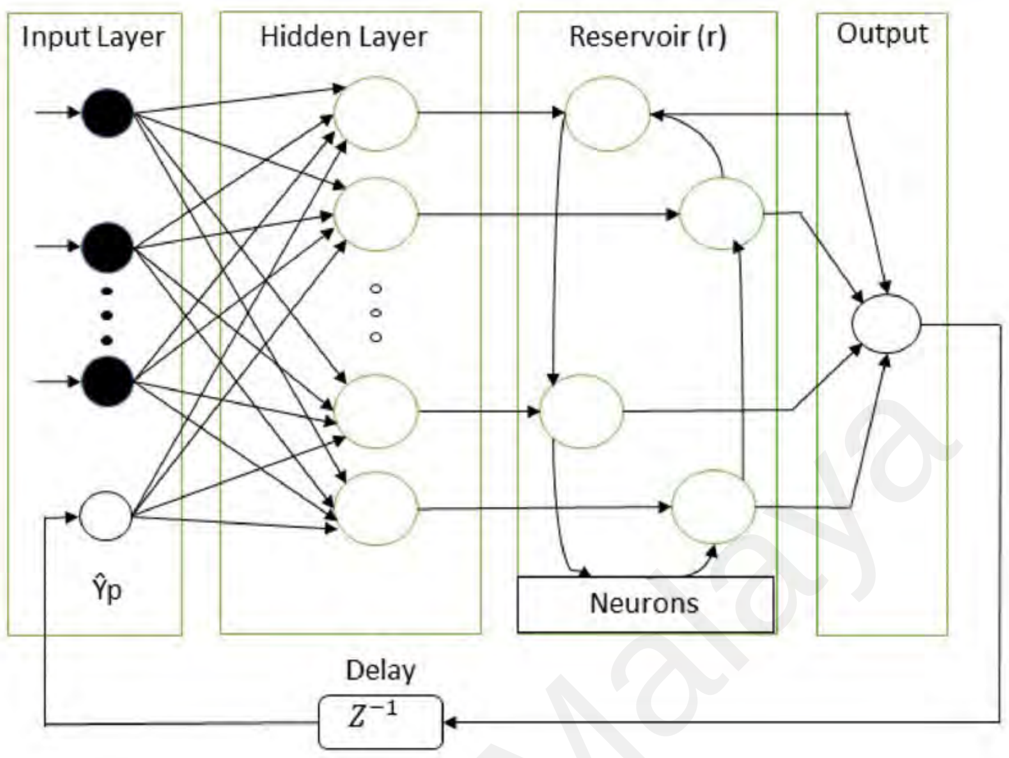


Figure 3.3: Architecture of RKERM

Table 3.1: Comparisons between RKERM and RKELM models.

RKERM		RKELM	
Reservoir (r_i)	$\alpha k\left(S(i), S\right) + (1-\alpha) r_{\text{i-1}}$	Kernel matrix (K)	$k\,(S, S)$
Output weight (β)	$r(rr^T + I)^{-1} Y_p$	Output weight (β)	$\left(K + \frac{I}{C_r}\right)^{-1} Y_p$
Prediction ($\hat{y}$)	Wr	Prediction ($\hat{y}$)	WK

to create a new and stable reservoir. Then Eq. (3.14) can be used to obtain predicted values of RKERM in both training and test data. In the recurrent part, (3.7) and (3.15) update the current training and test input data to provide sufficient input data in training and test part for the next step, respectively. The pseudo-code of RKERM is presented in Algorithm 5.

3.3.5 Parameter Optimization by QPSO

Generally, Genetic Algorithm (GA) (Austin, 1990), Artificial Bee Colony (ABC) (Karaboga, 2005) and Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995) are common

Algorithm 5 Recurrent Kernel Extreme Reservoir Machine

```
Require: - Size of prediction horizon $P$;
         - Time window size $D$;
         - Number of training data $L$;
         - Input data $S$ by (3.2);
         - Target data $Y$ by (3.3);
         - Kernel parameter $\delta$;
         - Output weight of KELM $\beta$;
         - Prediction value of training/test data $\hat{y}$;
         - Leaking rate $\alpha$;
Ensure:  - Output weight $W_{\text{out}}$;
         - Prediction value $\hat{y}$.
 1: Normalize data by (3.6).
 2: Set initial training data $S$ by (3.2), and target data $Y_{.,1}$ by (3.3);
 3: Compute $r_1$ by (3.12).
 4: Main loop:
 5: for $p \in \{1, \dots, P\}$ do
 6:     for $i \in \{2, \dots, L\}$ do
 7:         Use RBF kernel to compute kernel matrix;
 8:         Compute reservoir $r_i$ by (3.12);
 9:     end for
10:     Compute output weight $\beta_p^R$ by (3.13);
11:     Compute prediction value of training data ($\hat{y}_p$) in the
            $p$-th step by (3.14);
12:     Compute reservoir $r$ based on test input data ($\hat{S}$);
13:     Compute prediction value of test data ($\hat{y}_p$) in the
            $p$-th step by (3.14);
14:     Add training prediction values in the $p$-th step into
            training input data of the $p$-th step as new training
            input data for next step by (3.7);
15:     Add testing prediction values in the $p$-th step into
            test input data of the $p$-th step as new test input data
            for the next step by (3.15);
16: end for
```

optimization algorithms that have been applied in several fields. QPSO is proposed as a probabilistic variant of PSO and is motivated by concepts from quantum mechanics and PSO (S. Yang, Wang, et al., 2004). Numerical experiments show that the QPSO algorithm offers outstanding performance in solving a wide range of continuous optimization problems (Zhou, Yang, & Liu, 2008; Bao et al., 2014; Al Baity, 2015).

In RKELM and RKERM, some parameters directly impact the forecasting result, including the kernel parameter of RBF kernel method and leaking rate in (3.12). Therefore,

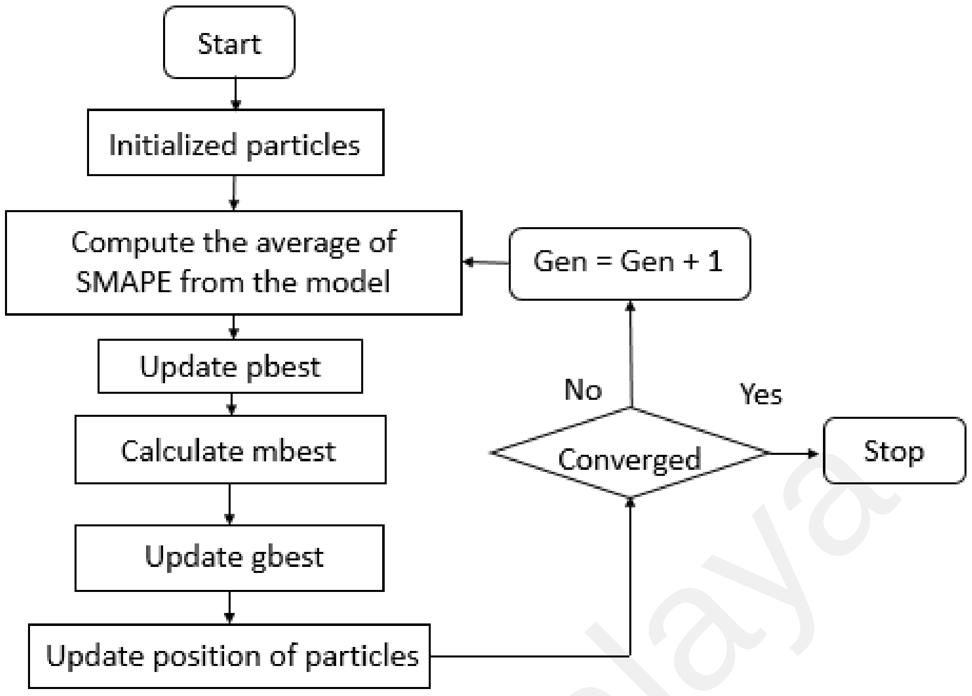


Figure 3.4: Flowchart of QPSO algorithm.

we utilize an optimization algorithm to search for a set of optimal parameters for our models.

The QPSO algorithm starts with an initial population with the best initial position ($pbest$), the best global position of the population ($gbest$) and the best mean position ($mbest$). In order to update the position of particles, we first update the $pbest$ and $gbest$, and then calculate a local attractor. Lastly, the distance from the best mean position to the particle's current position can also be calculated, which impacts the particle's new position distribution. Therefore, according to the current position, the local attractor and mean position of $pbest$ positions of all particles, and the position of particles in the new iteration can be updated. Each iteration has one $gbest$ of the population until the iteration criterion is met. The QPSO algorithm flowchart is shown in Figure 3.4.

3.4 Online Algorithms for Time Series Prediction

In this section, firstly, a meta-cognitive recurrent multi-step prediction model called Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism will be proposed. It aims on improving the prediction accuracy, dealing with concept drift problem in time series data, solving parameter dependency, and reducing learning computation in the prediction model. Secondly, in order to generates a fixed reservoir with optimized information for enhancing the forecasting performance in the prediction model, Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism will be introduced. In this model, recursive kernel method successfully replaces the first model. The proposed models are complex, the capability of each constituent method may be compromised. However, each method are responsible to their own problems. In the chapter experiments, it will prove the abilities of solving specific problem for different methods and select the most suitable method to solve its corresponding problem.

3.4.1 Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism

In order to solve disadvantages of OS-ELM, an algorithm that updates works with implicit mappings defined by kernels, called KOS-ELM, was proposed in (Scardapane et al., 2015). Due to the heavy computation during learning in KOS-ELM, the authors used two types of kernel filters, i.e. ALD and FB in the learning which could filter the incoming data in order to decide which data could be updated. KOS-ELM with ALD filter (ALD-KOS-ELM) has superior performance in time series prediction. Furthermore, recurrent multi-step ahead algorithm is also applied in ALD-KOS-ELM for multi-step time series prediction, which is called Recurrent Kernel Online Sequential Extreme Learning Machine with ALD Adaptive Kernel Filter, which can be shorten by RKOS-ELM in order to distinguish with ALD-KOS-ELM.

3.4.1.1 Concept Drift Detection

The majority of data sets are non-stationary that means the characteristics of data streams may change. It directly causes concept drift problem. Although there are many methods for dealing with concept drift, such as data stream mining (Hammoodi et al., 2016), the method from (Dehghan et al., 2016) that monitored the distribution of ensemble's error in order to detect probable drifts, and logistic regression learning model (Kulkarni & Ade, 2016), the majority of researches on dealing with concept drift problem are in classification domain. In the dynamically changing or non-stationary environments, the data distribution can change over time results in to the phenomenon of concept drift. In the learning part of prediction model, the phenomenon of concept drift needs to be taken into consideration. According to the comparison of the eight different concept drift detectors in (Gonçalves, de Carvalho Santos, Barros, & Vieira, 2014), Drift Detector Mechanism (Drift Detection Method (DDM)) and Exponentially Weighted Moving Average for Concept Drift Detection (ECDD) has the outstanding performance in drift detection for the experiments of classification. Therefore, in order to detect the occurrence of concept drift in the process of time series prediction, the study will apply modified DDM and ECDD in Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency (RKOS-ELM), respectively.

3.4.1.2 Combining Drift Detector Mechanism with RKOS-ELM

In (Gama et al., 2004), DDM combined with three learning algorithms (perception, neural network and decision tree) is proposed in order to learn the new concept for classification. The experiments showed a good performance detection drift in the process of classification. We modify DDM in order to apply it in the time series prediction,

$$ER_{l,p} = \frac{\left|Y_{l,p} - \hat{Y}_{l,p}\right|}{Y_{l,p}}, \tag{3.16}$$

$$SD_{l,p} = \sqrt{\frac{ER_{l,p} \times (1 - ER_{l,p})}{l}}, \tag{3.17}$$

where, SD is standard deviation, ER is error rate, Y is target values, and $\hat{Y}$ is prediction values.

For a sufficient large number of example, the binomial distribution is closely approximated by a normal distribution with the same mean and variance (Gama et al., 2004). Therefore, this study pay attention to the probability distribution that is the key flag for representing the change of context. In the learning of RKOS-ELM, the l-th error-rate and the standard deviation in the p step is defined with the Eqs. (3.16) and (3.17), respectively. Then, the error-rate ($ER_{l,p}$) in the $1 - \alpha/2$ confidence interval is approximately $ER_{l,p} \pm \alpha \times SD_{l,p}$ when there are a large of examples ($L \geq 30$). The parameter α depends on the confidence level. At the same time, when the first data coming to the model, the initial minimum value of error rate ER_{min} and standard deviation SD_{min} are defined as $ER_{1,p}$ and $SD_{1,p}$, respectively. The following new data of p step coming in the learning of RKOS-ELM is processed and updated ER_{min} and SD_{min}. For instance, in the sequence of learning of RKOS-ELM in the first step, there is a sample $S_{l,\cdot}$ with correspondent $ER_{l,1}$ and $SD_{l,1}$. In the experiments, described below the confidence level for warning has been set to 95%, that is, the warning level is reached if $ER_i + SD_i \geq ER_{min} + 2 \times SD_{min}$. The confidence level for concept drift is set as 99%, that is, the concept drift problem appears if $ER_i + SD_i \geq ER_{min} + 3 \times SD_{min}$. Otherwise, under the condition ($ER_i + SD_i \leq ER_{min} + SD_{min}$), no concept drift happens. At the same time, the minimum of error rate and standard deviation will be updated. DDM is used in the learning of RKOS-ELM as with ALD in order to decide the new incoming data whether add new information into the output weight or update the output weight. The updating process of output weight is shown in the following equations:

$$V_{(l)} = V_{l-1} - V_{l-1}H_l^T(I + H_l V_{l-1} H_l^T)^{-1} H_l P_{l-1}, \tag{3.18}$$

$$\beta_{(l)} = \beta_{l-1} + V_l H_l^T (Y_l - H_l \beta_{l-1}), \tag{3.19}$$

where V is interval parameter for output weight, and I is a sparse identity matrix.

The pseudo-code of RKOS-ELM with DDM is shown in Algorithm 6.

3.4.1.3 Combining ECDD with RKOS-ELM

A new method for detecting concept drift which used an Exponentially Weighted Moving Average (WEMA) chart was proposed in (Ross et al., 2012) to monitor the misclassification rate of an steaming classifier. However, the algorithm was only applied for classification. In this study, we employ error percentage of regression to replace the error rate of classification and modify the setting method of control limit (B) in ECDD criteria for the algorithm to be suited to time series prediction.

This section presents an algorithm for detecting concept drift for time series prediction, called ECDD. During initialization, when the first input data coming, the value of ECDD criteria (U_1) and estimator of percentage error (pt_1) are defined as zero. In detection, ER_l is updated using

$$pt_l = \frac{l-1}{l} pt_{l-1} + \frac{1}{l} ER_l, \tag{3.20}$$

where l is number of training input data ($l = [2, \ldots, L]$), and ER_l is the percentage error in the l-th observation. The pt does not give more weight to recent observations from the

Algorithm 6 Learning of RKOS-ELM with DDM

```
Require: - Size of prediction horizon $P$;
         - Time window size $D$;
         - Number of training data $L$;
         - Training data: $S$ by (3.2);
         - Target data: $Y$ by (3.3);
         - Kernel parameter $\delta$;
         - Output weight of RKELM in $p$ step $\beta_p$;
         - Prediction value in p-step $\hat{y_p}$;
         - The threshold of ALD $\varphi$;
Ensure:  - Output weight $\beta_p$;
         - Prediction value $\hat{y_p}$.
1:  for $p \in \{1, \ldots, P\}$ do
2:      Initialize $Q_{1,p}$ and $\beta_{1,p}$ based on data ($[S_{1,p}, Y_{1,p}]$);
3:      $mem_p = S_p(1)$;
4:      $V_{1,p} = 1$;
5:      Calculate $ER_{1,p}$, $SD_{1,p}$, $ER_{min}$ and $SD_{min}$;
6:      for $l \in \{2, \ldots, L\}$ do
7:          Calculate $\Delta_{l,p}$ by equation (2.21);
8:          Compute $ER_{l,p}$, $SD_{l,p}$ by equation (3.16) and (3.17);
9:          Based on DDM algorithm, determine
            whether the coming data has concept drift problem;
10:         if The coming data has concept drift problem then
11:             CD = 1;
12:         else
13:             CD = 0;
14:         end if
15:         if $CD = 1$ $and$ $\Delta_{l,p} \geq \varphi_l$ then
16:             Update $Q_{l,p}$ by (2.16);
17:             Update output weight ($\beta_{l,p}$) by (2.18);
18:             Add the current data into the memory
                dictionary ($mem_p$);
19:         else
20:             Update $V_p$ by (3.18);
21:             Update output weight ($\beta_{p,out}$) by (3.19);
22:             $ER_{min} = ER_{l,p}$;
23:             $SD_{min} = SD_{l,p}$
24:         end if
25:     end for
26:     Add prediction value into training data as new training
        input data for next step by (3.7);
27: end for
```

streaming data and is less sensitive to changes in ER. It is in tended to be an estimate of its pre-change value. When a change in the value of pt occurs, the ECDD criteria U_l should react fast. The algorithm detects for the occurrence of concept drift whenever the value of

U_l in the l-th observation is more than the addition of these two estimators ($pt_l + B_l\sigma_{U_l}$), where B_l is control limit ($B_l = 1/pt_l$). Based on the equation (3.20), we have simply computed the value pt_l. The standard deviation of EWMA (σ_{U_l}) can be computed using

$$\sigma_{U_l} = pt_l(1 - pt_l)\sqrt{\frac{\alpha}{2-\alpha}(1 - (1 - \alpha))^{2l}}, \tag{3.21}$$

According to the previous value of ECDD criteria, the current ER_l and the value of π that is assumed as 0.2, the ECDD criteria U_l is computed by the following equation and compares with the value of $pt_l + B_l\sigma_{U_l}$ in order to decide whether the current input data has concept drift problem.

$$U_l = (1 - \pi)U_{l-1} + \pi ER_l \tag{3.22}$$

Hence, the algorithm called RKOS-ELM with ECDD that combines ECDD with RKOS-ELM in order to solve the concept drift and reduce computation in the learning. The detail of pseudo-code in the learning process is show in the Algorithm 7.

Algorithm 7 Learning of RKOS-ELM with ECDD

```
Require:  - Size of prediction horizon $P$;
          - Time window size $D$;
          - Number of training data $L$;
          - Training data: $S$ by (3.2);
          - Target data: $Y$ by (3.3);
          - Kernel parameter $\delta$;
          - Output weight of RKELM in $p$ step $\beta_p$;
          - Prediction value in p-step $\hat{y_p}$;
          - The threshold of ALD $\varphi$;
Ensure:   - Output weight $\beta_p$;
          - Prediction value $\hat{y_p}$.
 1: for $p \in \{1, \ldots, P\}$ do
 2:     Initialize $Q_{1,p}$ and $\beta_{1,p}$ based on data ($[S_{1,p}, Y_{1,p}]$);
 3:     Initialize $U_{1,p} = 0$ and $pt_{1,p} = 0$;
 4:     $mem_p = S_{1,p}$;
 5:     $V_{1,p} = 1$;
 6:     Calculate $ER_{1,p}$, ;
 7:     for $l \in \{2, \ldots, L\}$ do
 8:         Calculate $\Delta_{l,p}$ by equation (2.21);
 9:         Compute $ER_{l,p}$ in the $l$-th coming input data of $p$
            step;
10:         Compute $pt_{l,p}$ by equation (3.20);
11:         Compute $B_{l,p} = 1/ER_{l,p}$;
12:         Compute $\sigma_{U_{l,p}}$ by equation (3.21);
13:         Compute ECDD criteria $U_{l,p}$ by equation (3.22);
14:         According to the algorithm of ECDD, determine
            whether the coming data has concept drift problem;
15:         if $U_{l,p} \geq (pt_{l,p} + B_{l,p}\sigma_{U_{l,p}})$ then
16:             CD = 1;
17:         else
18:             CD = 0;
19:         end if
20:         if $CD = 1$ $or$ $\Delta \geq \varphi_l$ then
21:             Update $Q_{l,p}$ by (2.16);
22:             Update output weight ($\beta_{l,p}$) by (2.18);
23:             Add the current data into the memory
                dictionary ($mem_p$);
24:         else
25:             Update $V_p$ by (3.18);
26:             Update output weight ($\beta_{p,out}$) by (3.19);
27:         end if
28:     end for
29:     Add prediction value into training data as new training
        input data for next step by (3.7);
30: end for
```

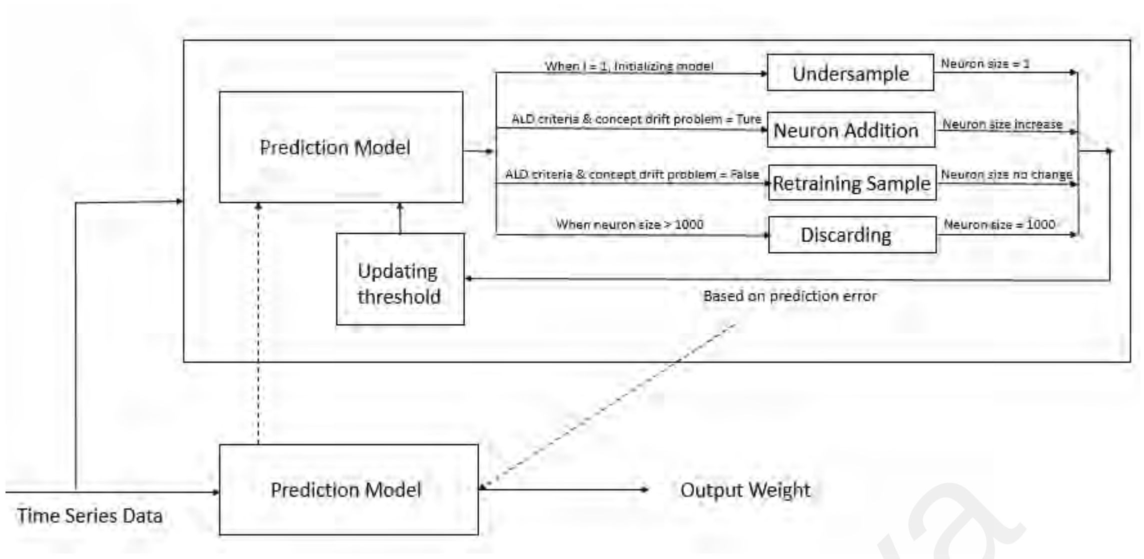


Figure 3.5: Meta-cognitive learning for time series prediction model

3.4.1.4 New Meta-cognitive Learning Strategy

A new learning strategy is applied to reduce the learning time and decide when the coming data need to add neuron, retain or discard in learning of the prediction model. The strategy includes four parts, including undersampling, neuron addition, retain sample and discard sample. The flow chart of meta-cognitive learning for the time series prediction model is shown in Figure 3.5.

The first phase, undersampling, used to initialize the prediction model, requires the minimum number of hidden neurons for prediction. In our online learning prediction model, the minimum number of hidden neurons is one. At the same time, the initial threshold of ALD (φ) is equal to the current prediction error (e_1). The initial value of ALD threshold can be defined as the current prediction error.

The second phase is neuron addition, which requires the incoming data to fulfill the ALD kernel filter condition and concept drift detector criteria. It means that the hidden neuron will increase by (2.18) and the incoming data will add into dictionary memory when the coming data fulfills ALD condition and concept drift criteria. Then the current

Algorithm 8 The strategy of meta-cognitive learning that is applied in prediction model

```
    Undersampling:
        Initialize the output weight of model $\beta_1$;
        Calculate the prediction error $e_1$;
        Initialize the threshold of ALD ($\varphi_1 = e_1$);
 1: for $p \in \{2, \dots, P\}$ do
    Determine whether concept drift happen:
 2:     if happened then
          CD = 1;
 3:     else
          CD = 0;
 4:     end if
          Calculate $\Delta$ by equation 2.21
    Neuron Addition:
 5:     if $CD = 1 and \ \Delta \geq \varphi_{p-1}$ then
 6:         The hidden neuron increases;
 7:         The incoming data will add into dictionary memory;
 8:     end if
 9:     Update the threshold of ALD by equation 3.23;
    Retraining Phase:
10:     if $CD = 0 and \ \Delta \leq \varphi_{p-1}$ then
11:         Update output weight by the new coming data
12:         Update the threshold of ALD by equation 3.23
13:     end if
    Discarding Phase:
14:     if $HN \geq 1000$ then
15:         By Algorithm 1 to update output weight
16:         Update the threshold of ALD by equation 3.23
17:     end if
18: end for
```

threshold of ALD can be defined by the following equation:

$$\varphi_l = \lambda(\varphi_{l-1}) + (1 - \lambda)e_l, \tag{3.23}$$

where λ is the slope that controls the rate of self-adaptation and is set close to 1.

If the incoming data does not have concept drift problem or has the similar characteristics with dictionary memory by detecting from ALD filter, the incoming data will go to the retraining phase. The output weight will be retained by (3.19). The last phase is discarding phase, which defines the maximum number of hidden neurons is 1000. If the current number of hidden neurons (HN) is more than 1000, the corresponding sample with the minimum error pattern that is determined by FB will be discarded from dictionary memory. The algorithm of meta-cognitive learning strategy shows in the Algorithm 8.

3.4.2 Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Drift Detector Mechanism

As same as RKERM, online learning prediction model also needs a fixed reservoir that can keep optimized information in the learning part. In this section, the reservoir of prediction model is generated by recursive kernels, which can update memory using old information. The old information can be dissipated and overwritten with time. The reservoir finally becomes a fixed dynamic system that only contains filtered and usable information.

Prior to introducing recursive kernel, the definition of kernel function need to be considered. It is assumed that there is kernel function $k(x, x\prime)$ with data points x and $x\prime$. The kernel function can be defined using

$$k(x, x') = F(x)F(x'), \tag{3.24}$$

where F is feature map. If x and x' are concatenations of two vectors, such as $x = [x_1|x_2]$ and $x' = [x'_1|x'_2]$. The kernel function can be written as follow:

$$k(x, x') = f(||x_1 - x'_1||^2 + ||x_2 - x'_2||), \tag{3.25}$$

When x_1 and x'_1 correspond to the current inputs x(n) and $x'(n)$, and x_2 and x'_2 to the recursive maps, the recurrent kernel with input data x and x' can be written as

$$k_n(x, x') = f(||x(n) - x'||^2 + k_{n-1}(x, x) + K_{n-1}(x', x') - 2k_{n-1}(x, x')), \tag{3.26}$$

A recurrent formula that can be applied for any kernel function of the form specified in equation (3.25) (Hermans & Schrauwen, 2012). T he corresponding recurrent kernel can be simplified to

$$r_n(x, x') = f(x(n)x'(n) + k_{n-1}(x, x')), \tag{3.27}$$

In Meta-cognitive Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism (Meta-RKOS-ELM-DDM), Radial Basis Function is used as kernel method in order to calculate kernel matrix of input data. The size of kernel matrix increases with the increasing number of samples in the learning of Meta-RKOS-ELM-DDM. The kernel matrix (g_l) is computed by the equation 2.13. When we assume that there are two samples x and x', the RBF kernel represents as a matrix of the input space that is defined as

$$k(x, x') = exp(-\frac{||x - x'||^2}{2\sigma^2}), \tag{3.28}$$

where $||x - x'||^2$ is represented by the squared Euclidean distance between the two samples, and σ is kernel parameter.

Although ALD plays a significant role in filtering input data and reduce the number of hidden nodes in the model of Meta-RKOS-ELM with DDM, the computing process of kernel matrix (g) in Eq. (2.13) only pays attention to the kernel matrix between the current input data and the current memory. In order to deal with the finitely large hidden states of Meta-RKOS-ELM with DDM, this study applies recursive kernel, the inner product of the hidden states of infinite networks, to replace RBF kernel method during learning. Besides, the old information of kernel matrix can be dissipated and overwritten by the recursive kernel over the time passes. This means that the reservoir of predicting model that is made by the recursive kernel has the more efficient, completed, and variety of information than by the common kernel method. Therefore, the recursive RBF kernel method is employed to calculate kernel matrix in the prediction model, called Meta-cognitive Recurrent Recursive RBF Kernel Online Sequential Extreme Learning Machine with DDM (Meta-cognitive Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism (Meta-RRKOS-ELM-DDM)).

$$rk(x, x') = k(k_{t-1}(x, x') + x(t)x'(t-1)), \tag{3.29}$$

The recursive kernel (rk) of $k_t(x, x')$ and $k_{t-1}(x, x')$ can be simplified to Eq. (3.29). Using RBF kernel in Eq. (3.28) and applying the rules of Eq. (3.29), the study finds the formula of the recursive RBF as

$$rk_{t+1}(x, x') = exp(-\frac{||x(t+1) - x(t+1)'||^2}{2\sigma^2})exp(-\frac{rk_t(x(t), x(t)) - 1}{\sigma'^2}), \tag{3.30}$$

where $sigma$ is the recursive kernel parameter. Therefore, according to Eq. (3.30), the kernel matrix (rg) in prediction model can be modified from the previous kernel matrix (g) in Meta-RKOS-ELM as

$$rg_l = [rk(S_l, S_1), \ldots, rk(S_l, S_{l-1}))]^T, \tag{3.31}$$

In Meta-RRKOS-ELM-DDM, meta-cognitive learning strategy is as same as the first online learning time series prediction model. Furthermore, it assumes the training input data S ($S = \{S_{(1,\cdot)}, S_{(2,\cdot)}, \ldots, S_{(L,\cdot)}\}$) and the corresponding target Y ($Y = \{Y_1, Y_2, \ldots, Y_L\}$). During initialization, the kernel matrix for the first input sample is $rg_1 = exp(-\frac{||x(t+1) - x(t+1)'||^2}{2\sigma^2})$, $ER_{1,p} = 0$, $SD_{1,p} = \sqrt{ER_{1,p}(1 - ER_{1,p})}$. The minimum error (the minimum error (ER_{min}

)) and minimum standard deviation (minimum standard deviation (SD_{min})) are defined as same as $ER_{1,p}$ and $SD_{1,p}$, respectively, and the initial output weight $\beta_1 = Q_1 Y_1$ with $Q_1 = 1$. In the updating phase, $l \in \{2, \ldots, L\}$, the kernel matrix has been replaced by the equation (3.30) and the output weight (β_l) with interval coefficient Q_l can be calculated using

$$z_l = Q_{l-1} r g_l, \tag{3.32}$$

$$r_l = C^{-1} + 1 - z_l^T r g_l, \tag{3.33}$$

$$Q_l = \begin{bmatrix} Q_{l-1} r_l + z_l z_l^T & -z_l \\ -z_l^T & 1 \end{bmatrix} \tag{3.34}$$

$$e_l = Y_l - r g_l^T \beta_{l-1}, \tag{3.35}$$

$$\beta_l = \begin{bmatrix} \beta_{l-1} - z_l r_l^{-1} e_l \\ r_l^{-1} e_l \end{bmatrix} \tag{3.36}$$

The main difference between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM is the method of computing kernel matrix. The Eq. (3.28) indicates that each of kernel neurons of Meta-RRKOS-ELM-DDM is computed by the current kernel method and previous kernel matrix, which not only obtains more information from the coming data rather than RKOS-ELM-DDM, but also generate a fixed reservoir saving optimized information based on the characteristic of kernel. At the same time, the process of computing out weights and predicted values also show the difference between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM. Table 3.2 shows the difference in the training and testing part for Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM. In the training process, recursive kernel of Meta-RRKOS-ELM-DDM is computed by current kernel and the previous kernel matrix, but the kernel of Meta-RKOS-ELM-DDM only is computed by the current kernel method. Moreover, the recursive kernel matrix and prediction values in Meta-RRKOS-ELM-DDM is computed by the corresponding result of recursive kernel method.

Table 3.2: Comparisons between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM.

Meta-RRKOS-ELM-DDM		Meta-RKOS-ELM-DDM	
Recursive kernel method ($rk_{t+1}(x, x')$)	$exp(-\frac{\|\|x(t+1)-x(t+1)'\|\|^2}{2\sigma^2})exp(-\frac{rk_t(x(t),x(t))-1}{\sigma'^2})$	Kernel matrix ($k(x, x')$)	$exp(-\frac{\|\|x-x'\|\|^2}{2\sigma^2})$
Recursive kernel matrix (rg_l)	$[rk(S_l, S_1), \dots, rk(S_l, S_{l-1}))]^T$	Kernel matrix (K)	$[k(S_l, S_1), \dots, k(S_l, S_{l-1}))]^T$
Prediction ($\hat{y}$)	βr	Prediction ($\hat{y}$)	WK

Learning of pseudo-code in Recurrent Recursive Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism (RRKOS-ELM-DDM) is given in Algorithm 9. The meta-cognitive learning strategy shown in the section 3.4.1.4.

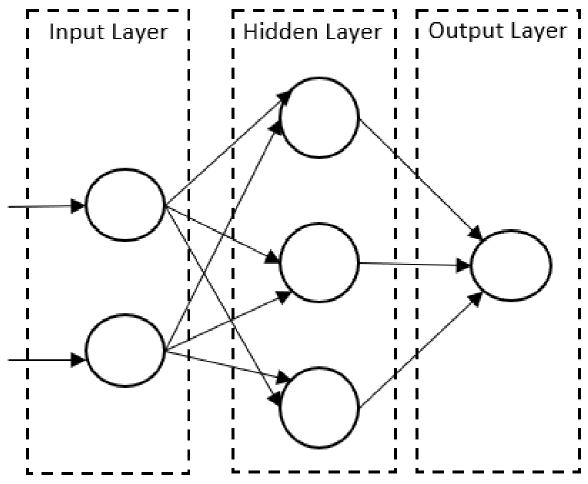
Algorithm 9 Learning of RRKOS-ELM with DDM

```
Require: - Size of prediction horizon $P$;
         - Time window size $D$;
         - Number of training data $L$;
         - Training data: $S$ by (3.2);
         - Target data: $Y$ by (3.3);
         - Kernel parameter $\delta = 0.7$;
         - recursive kernel parameter $\delta'$;
         - Output weight in $p$ step $\beta_p$;
         - Prediction value in p-step $\hat{y_p}$;
         - The threshold of ALD ($\varphi$);
Ensure:  - Output weight $\beta_p$;
         - Prediction value $\hat{y_p}$.
 1: for $p \in \{1, \ldots, P\}$ do
 2:     Initialize:
 3:     $rg_1 = exp(-\frac{||x(t+1)-x(t+1)'||^2}{2\sigma^2})$;
 4:     $ER_{1,p} = 0, SD_{1,p} = \sqrt{ER_{1,p}(1 - ER_{1,p})}$;
 5:     $Q_{1,p} = 1 and \beta_{1,p} = Q_1 Y_1$;
 6:     $mem_p = S_p(1)$;
 7:     $V_{1,p} = 1$;
 8:     $ER_{min} = ER_{1,p} and SD_{min} = SD_{1,p}$;
 9:     for $l \in \{2, \ldots, L\}$ do
10:         Calculate $\Delta_{l,p}$ by equation (2.21);
11:         Compute $ER_{l,p}$, $SD_{l,p}$ by equation (3.16) and (3.17);
12:         Based on DDM algorithm, determine
            whether the incoming data has concept drift problem;
13:         if The incoming data has concept drift problem then
14:             CD = 1;
15:         else
16:             CD = 0;
17:         end if
18:         if $CD = 1 \ and \ \Delta_{l,p} \geq \varphi_l$ then
19:             Update $rg_{l,p}$ by (3.31);
20:             Update $Q_{l,p}$ by (3.34);
21:             Update output weight ($\beta_{l,p}$) by (3.36);
22:             Add the current data into the memory
                dictionary ($mem_p$);
23:         else
24:             Update $V_p$ by (3.18);
25:             Update output weight ($\beta_{p,out}$) by (3.19);
26:             $ER_{min} = ER_{l,p}$;
27:             $SD_{min} = SD_{l,p}$
28:         end if
29:     end for
30:     Add prediction value into training data as new training
        input data for next step by (3.7);
31: end for
```

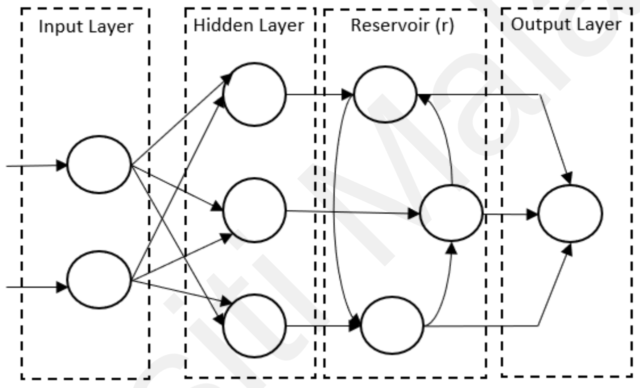
3.5 The Differences Between Reservoir in Offline Learning and Recursive Kernel in Online Learning Algorithm

To generate a fixed reservoir with optimized information, the new reservoir computing that employs kernel method to develop ESN and recursive kernel method is used in the offline and online learning model, respectively. Our proposed reservoir computing connects each hidden neurons in the hidden layer of offline prediction model, which saves and optimized all information in the reservoir. And recursive kernel method, which is associated with an infinite recurrent network, connects the 'input' of the network in the current time step with the previous hidden state.

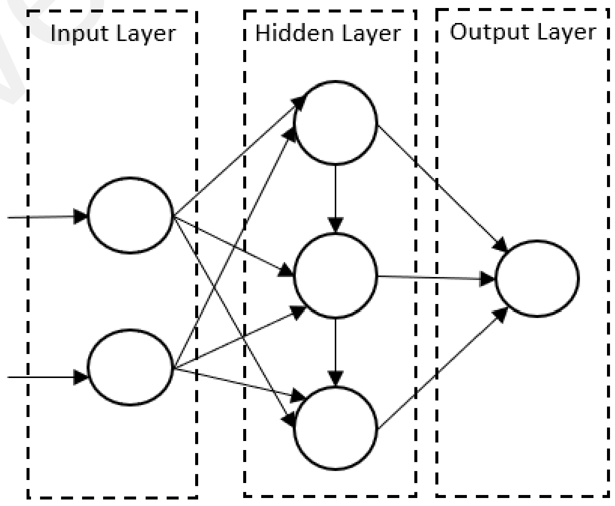
Although the theory of construction in these methods are associated with recurrent algorithm, there is one main difference with the recurrent multi-steps-ahead algorithm that apply in the offline and online learning model. The recurrent multi-steps-ahead algorithm adds the current step prediction result into the input data of the next step as new input data, which solves the restriction of prediction horizon. It also appears in offline and online learning models as external recurrent architecture, which is similar with NARX architecture that feeds the output of network to the input layer. On the other hand, our proposed new reservoir computing method and recursive kernel method apply their theories to connect the hidden neurons in the network. The new reservoir computing method maps a high-dimensional space in the offline learning model and recursive kernel method optimizes information by dissipating and overwriting the current neuron information from previous information in the online learning model, which generate a fixed reservoir and improve the prediction performance. They appear in the offline and online learning model as internal recurrent architecture.



Normal Network



New Reservoir Computing Method



Recursive Kernel Method

Figure 3.6: Different architectures for three networks

Furthermore, there are some differences between new reservoir computing method and recursive kernel method. Firstly, the construction is different in the network. The Figure 3.6 shows the difference between new reservoir computing method and recursive kernel method in construction. It indicates that the new reservoir computing method connects all neurons as the closed circle in the hidden layer of the offline model and the recursive kernel only make the connection between the t-th and $t-1$-th neuron in the hidden layer of online learning model. Secondly, due to the above characteristic of architecture, the new reservoir computing method is only applied in offline learning method and recursive kernel can be applied in online learning model, because the fixed reservoir that is generated by new reservoir computing method cannot be updated immediately when the new data coming. The only condition is retraining the whole dataset, which takes lots of time. Therefore, there are the reasons why this study employs two different methods to generate a fixed reservoir to improve the forecasting performance.

3.6 Performance Measures

For the purpose of comparison in the experiments, Symmetric Mean Absolute Percentage Error (SMAPE) (Armstrong & Forecasting, 1985) and Mean Squared Error (MSE) (Makridakis, 1993) are employed to measure the performance of models. SMAPE is an accuracy measure based on percentage error. It is defined as,

$$SMAPE = \frac{100\%}{n} \sum_{t=1}^{T} \frac{\left|\hat{Y}_t - Y_t\right|}{[\left(\left|\hat{Y}_t\right| + |Y_t|\right)/2]}, \tag{3.37}$$

where T is the number of predictions ($t = 1, 2, \ldots, T$), $\hat{Y}_t$ is the prediction value in the t-th time, and Y_t is the t-th actual target value.

MSE measures the average of squared errors, that is, the difference between the estimated and the target values. The MSE measure is defined by the following equation,

$$MSE = \frac{1}{n}\sum_{t=1}^{T}(Y_t - \hat{Y}_t)^2, \tag{3.38}$$

3.7 Summary

This chapter discusses the method of data transformation, one offline learning model and two online learning models. Firstly, a simple time series data can be transformed into a matrix for further prediction models. Secondly, it introduces an offline learning time series prediction model named RKERM with QPSO, which extracts the strengths of RKELM and ability of modified Reservoir Computing. It overcomes the restriction of prediction horizon and the limitations of neurons construction in RKELM. Furthermore, QPSO is used to optimize the parameters of kernel method and leaking rate of reservoir computing in RKERM. Thirdly, Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM are introduced in the part of online learning time series prediction model. They aim on generating a fixed dynamic reservoir by recursive kernel method, dealing with concept drift problem in time series data by concept drift detectors, overcoming the restriction of prediction horizon, reducing learning time and solving the parameter dependency by new meta-cognitive learning strategy. Lastly, the differences between the new reservoir computing method and recursive kernel method, which is applied in offline and online learning model respectively, are introduced.

CHAPTER 4: EXPERIMENTS DESIGN AND RESULTS FOR OFFLINE LEARNING MODEL

In the experimental work, we apply the proposed techniques for offline prediction on two synthetic benchmark data sets (Mackey-Glass and Lorenz) and five real-world time series data sets including Malaysia palm oil price, ozone concentration of Toronto, sunspot Standard & Poor 500 and water level dataset of Phra Chulachomklao Fort (PC) in Thailand. Experiment 1 shows the detail and results in synthetic data sets and Experiment 2 pays attention to the real-world data sets.

4.1 Experiment 1

In this section, experiments using synthetic data sets are carried out to prove the efficiency of the proposed RKERM model. The synthetic data includes the Mackey-Glass and Lorenz time series data sets. The accuracy of RKERM with QPSO is compared with not only the RKELM and RKELM with QPSO in order to prove the efficiency of RC and QPSO in our proposed algorithm, but also with other popular and classic machine learning algorithms that are applied in solving time series problems such as Recurrent glsESN (Jaeger & Haas, 2004), Recurrent Support Vector Regression (RSVR) (Pai & Hong, 2005) and RELM (Ertugrul, 2016).

4.1.1 Data Description and Processing

In the synthetic data sets, Mackey-Glass equation had originally been proposed as a model of blood cell regulation (Mackey, Glass, et al., 1977). It has been used in the literature as a benchmark model (Martınez-Rego, Fontenla-Romero, & Alonso-Betanzos, 2008) due to its chaotic characteristics. It is created by the following equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{(1+10(t-\tau))-bx(t)}, \tag{4.1}$$

In the Mackey-Glass equation, the delay parameter, τ, determines the characteristics of (4.1). For example, $\tau < 4.43$ produces a fixed point attractor, $4.43 < \tau < 13.3$ produces a stable limit cycle attractor, $13.3 < \tau < 16.8$ gives a double limit cycle attractor, and $\tau > 16.8$ is chaotic. The parameters are selected according to a previous study by (Wang & Mendel, 1992), where the constants are set to be $a = 0.1$, $b = 0.2$, $\tau = 30$, and $x(t)$ represents the value of the variable at t time ($t = 1, 2, \ldots, T$). A Mackey-Glass time series sample set with length 1,035 is generated by (4.1). The following Figure shows the time series data of Mackey-Glass.

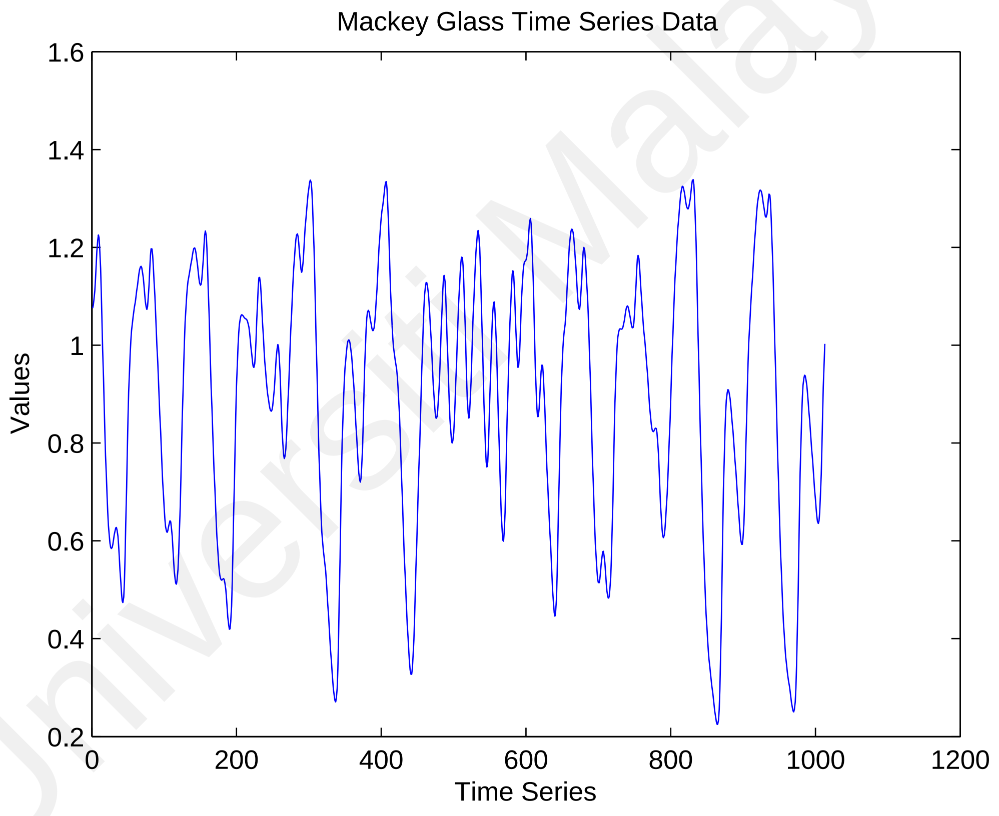


Figure 4.1: Mackey-Glass Time Series Data

Lorenz time series data set is a weather example of chaotic systems, which has extensively contributed to the establishment of chaos theory (Kantz & Schreiber, 2004). Lorenz equations are a simplified version of Navier-Stokes equations which are used in fluid mechanics. Lorenz equations are defined as follows:

$$
\begin{aligned}
\frac{dy}{dt} &= \sigma(y(t) - x(t)) \\
\frac{dx}{dt} &= ry(t) - x(t) - y(t)z(t) \\
\frac{dz}{dt} &= y(t)x(t) - bz(t),
\end{aligned}
\tag{4.2}
$$

The following dimensionless parameters are applied: $\sigma = 10$, $b = 8/3$, $r = 28$. The x-coordinate of the Lorenz time series is considered for forecasting and a time series with a length of 2,535 is generated as described in the research work by Rojas in 2002. Figure 4.2 shows the Lorenz time series data.

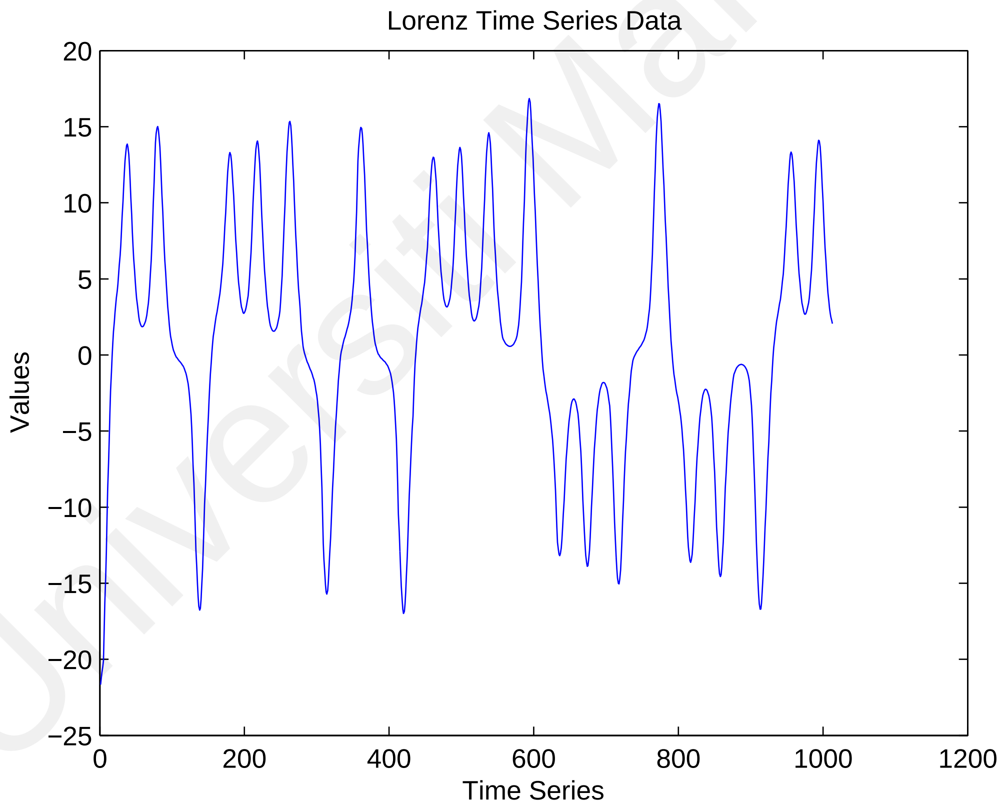


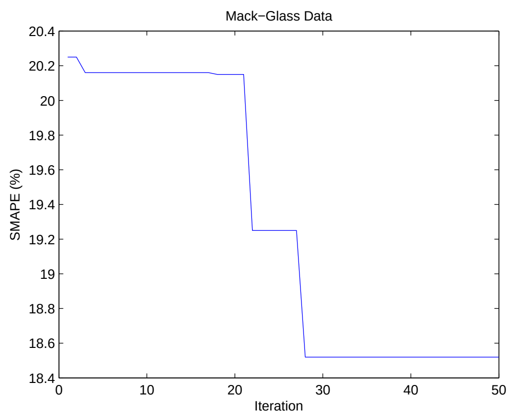
Figure 4.2: Lorenz Time Series Data

Due to the condition of the recurrent multi-step-ahead algorithm, the prediction horizon must be less than or equal to the time window. We employ eighteen historical data to predict the situation of eighteen future steps, which means that the size of the time window

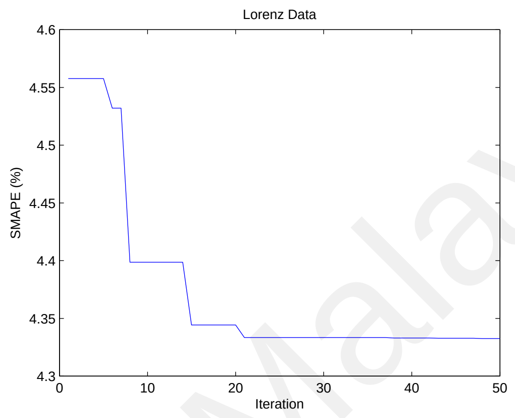
(D) and the prediction horizon (P) are eighteen. Therefore, these two data sets can be transformed by (3.1). The time series data of Mackey-Glass and Lorenz become matrices with size (1000×36) and (2500×36), respectively.

4.1.2 Evaluation of Synthetic Data

These two synthetic data sets were usually used as benchmark data for evaluating the models in the previous studies (Gholipour, Araabi, & Lucas, 2006; Ardalani-Farsa & Zolfaghari, 2010). The most common percentages of separation in training and test data are 50 % and 50 % for Mackey-Glass. Lorenz data is divided into 1,500 for training and 1,000 for test sets, respectively. In order to obtain the best performance for all models, we referred to previous research to select the parameters of RELM, ESN and RSVR, as shown in Table 4.1. In the other novel models, we apply QPSO to define the parameters in RKELM and RKERM for Mackey-Glass and Lorenz data sets. This thesis sets the parameters of QPSO as follows, including maximum number of iterations is 50, population size is 20, the value of alpha if the fixed value is used is 0.6, and the lower and upper bound of the search scope are set as 0.6E-10 and 100. The reasons of setting above parameters of QPSO are as follows: Firstly, due to finding out the suitable kernel parameter and linking rate, they do not have too long range of parameter searching scope. Secondly, the 50 number of iterations is enough for searching the best results. It generally convergences around 20 iterations. Thirdly, alpha value is generally set as 0.6 (Al Baity, 2015; Bao et al., 2014). The minimum average SMAPE of the prediction horizons in both data sets is achieved by QPSO. The optimal parameters in RKELM and RKERM are shown in Table 4.1. The convergence charts of RKERM with QPSO are shown in Figure 4.3 for data of Mackey-Glass and Lorenz. Based on the Figure 4.3, the parameter setting of QPSO is acceptable. The iteration of convergence for different artificial data are 29 for Mack-Glass and 20 for Lorenz. The convergent speed for both artificial data sets are fast.



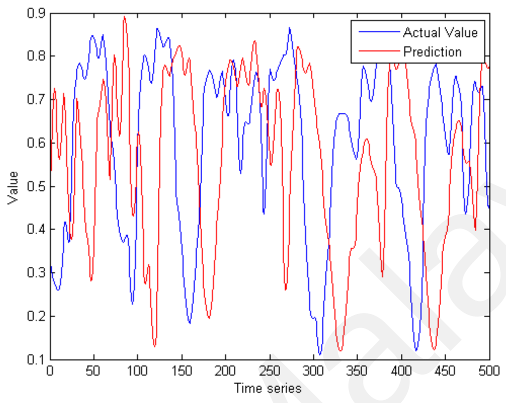
$[MG]$



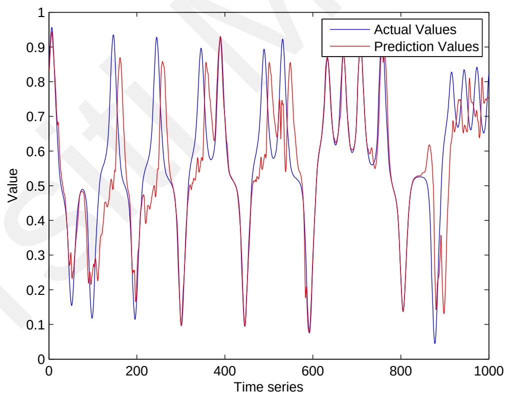
$[Lorenz]$

Figure 4.3: The convergent chart of RKERM based on QPSO

According to these parameter settings, the results of multiple step ahead prediction with two types of measurements are shown in Table 4.2. This indicates that RKERM with QPSO has the best performance among all algorithms in multi-step-ahead prediction under measurement of SMAPE and MSE, except the case of the average prediction horizon $(1-7)$. In a short prediction horizon, such as in the first and fourth step, the rank of the proposed model in predicting performance is second in all models. The comparative results between RKELM with QPSO and our proposed model indicate that RC plays a vital role in the accuracy of time-series prediction and QPSO is an efficient algorithm for defining parameters. RKERM with QPSO provides much better predicting performance in both data sets than other types of recurrent networks due to the inclusion of RC. Furthermore, recurrent multiple steps prediction also plays a significant role in time series prediction. Compared with the results of different prediction horizons, although the accuracy of prediction decreased as the number of prediction horizons increased, the decrease in error rate is not rapid which means that the accuracy of multi-step ahead-prediction is still acceptable. Therefore, the recurrent multi-step ahead-prediction algorithm can be applied to time series data not only to solve the limitation of the prediction horizon, but also to obtain acceptable forecasting results in long-term predictions.



$[MG]$



$[Lorenz]$

Figure 4.4: The comparison between actual values and prediction values in synthetic time series data sets in the 18-th step of RKERM.

Based on the comparison of results in different prediction horizons, RKERM has super prediction ability rather than other models. Figure 4.4 shows the situation between actual values and prediction values in the 18-th prediction horizon in order to represent the final forecasting ability of RKERM for two artificial data.

Table 4.1: The parameters settings of all models for synthetic data sets

Model	Parameter	Data Set	
		Mackey-Glass	Lorenz
ESN		(Venayagamoorthy & Shishir, 2009)	(L. Yang, Liang, Ma, & Liu, 2013)
	N_r	400	124
	ρ	0.8	0.76
	δ	0.1	0.07
RSVM		(Thissen, Van Brakel, De Weijer, Melssen, & Buydens, 2003)	(Hou & Li, 2009)
	C_s	15	1000
	λ	0.001	0.0001
	δ	0.1	0.1
RELM		(Singh & Balasundaram, 2007)	(G.-B. Huang et al., 2012)
	L	37	20
RKELM	δ	0.6454	0.1896
RKERM	δ	17.6357	0.0619
	α	0.0300	0.3300

Table 4.2: The performance comparison of synthetic data for multi-step-ahead prediction.

Data sets	Algorithms	Predicting horizon (p)						Average 1-p			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
Mack-Glass		MSE									
	ESN	1.63E-02	1.63E-01	3.70E-01	2.48-01	1.50E-01	5.60E-02	1.72E-01	3.61E-01	1.17E-01	2.06E-01
	RSVM	6.37E-03	6.89E-03	1.61E-02	4.47E-02	4.84E-02	4.03E-02	8.90E-03	3.31E-02	4.43E-02	2.74E-02
	RELM	**1.39E-05**	**6.71E-04**	6.87E-03	2.74E-02	3.94E-02	9.12E-02	**1.83E-03**	2.01E-02	5.78E-02	2.56E-02
	RKERM	7.30E-04	4.20E-03	1.70E-02	4.58E-02	4.99E-02	5.24E-02	6.32E-03	3.55E-02	5.02E-02	2.90E-02
	RKELM+QPSO	4.35E-03	1.16E-02	2.55E-02	4.82E-02	5.27E-02	5.48E-02	1.31E-02	4.01E-02	5.37E-02	3.41E-02
	RKERM+QPSO	6.95E-04	2.05E-03	**6.52E-03**	**1.60E-02**	**1.84E-02**	**1.86E-02**	2.84E-03	**1.27E-02**	**1.85E-02**	**1.08E-02**
		SMAPE									
	ESN	26.95	57.51	95.18	80.58	65.82	43.74	59.43	92.61	58.14	68.22
	RSVM	18.55	18.07	23.97	35.74	36.28	34.13	19.70	31.85	35.31	28.28
	RELM	**2.19**	**7.12**	17.28	29.16	32.34	40.24	**8.50**	25.45	35.31	22.15
	RKERM	7.60	14.38	23.93	34.86	36.40	38.33	15.04	31.47	36.91	26.90
	RKELM+QPSO	12.36	20.22	28.64	37.68	39.37	40.97	20.33	34.78	40.04	30.92
	RKERM+QPSO	5.95	10.36	**17.19**	**23.69**	**24.74**	**24.85**	10.87	**21.72**	**24.78**	**18.52**
Lorenz		MSE									
	ESN	3.79E-04	5.03E-04	9.27E-04	3.96E-03	6.40E-03	1.24E-02	5.67E-04	2.41E-03	8.64E-03	3.77E-03
	RSVM	1.43E-03	1.42E-03	2.10E-03	2.41E-03	3.06E-03	3.38E-03	1.54E-03	2.62E-03	2.98E-03	2.32E-03
	RELM	**2.22E-05**	2.87E-04	1.12E-03	1.91E-03	2.76E-03	4.12E-03	4.06E-04	1.64E-03	3.26E-03	1.70E-03
	RKERM	3.84E-04	5.28E-04	7.61E-04	1.25E-03	1.52E-03	2.21E-03	5.48E-04	1.03E-03	1.77E-03	1.09E-03
	RKELM+QPSO	1.35E-04	**2.52E-04**	**4.88E-04**	1.12E-03	1.48E-03	2.36E-03	**2.78E-04**	8.41E-04	1.81E-03	9.45E-04
	RKERM+QPSO	4.09E-04	4.84E-04	5.43E-04	**5.93E-04**	**6.31E-04**	**8.47E-04**	4.80E-04	**5.75E-04**	**7.07E-04**	**5.82E-04**
		SMAPE									
	ESN	4.03	4.75	6.71	11.51	13.80	18.05	5.04	9.55	15.43	9.76
	RSVM	7.49	7.30	8.63	9.21	9.95	10.18	7.50	9.58	9.77	8.84
	RELM	**0.87**	3.19	6.66	7.48	9.29	10.88	3.39	7.30	9.88	6.64
	RKERM	4.20	4.98	5.42	6.56	7.10	8.02	4.89	6.00	7.43	6.05
	RKELM+QPSO	1.54	**2.39**	**3.48**	5.37	6.12	7.42	**2.44**	4.64	6.63	4.45
	RKERM+QPSO	4.03	4.14	4.09	**4.38**	**4.58**	**4.91**	4.10	**4.22**	**4.71**	**4.33**

Note: The best performance is in boldface.

4.2 Experiment 2

There are five time-series data sets obtained from real-world applications. These are the daily palm oil transaction price in Malaysia (Palm oil), the ozone concentration of Toronto (Ozone), sunspot series data (Sunspot), Standard & Poor's 500 index (S&P500), and the water level of Phra Chulachomklao Fort at Chao Phraya river in Thailand (PC water level). Similar to synthetic data, real-world data sets also apply the same measurement and compared models to evaluate the performance of RKERM in time series prediction.

4.2.1 Data Processing and Description

Missing values are a common problem that exists in the process of analyzing data and building a model. In real-world data sets, missing values directly affect prediction results of algorithms. Generally, there are three methods that can be used to deal with the missing values problem–discarding, replacement by linear model, and replacement by the mean of the data set. In this study, the missing values in the first four real-world data sets are replaced by the mean of existing values. Missing values in the water levels forecast in Thailand data were handled similarly to the paper (Pasupa & Jungjareantrat, 2016) did. The next step is data transformation where all data are transformed into matrices by (3.1). Similar to synthetic data experiments, real-world data experiments also set the size of the time window and prediction horizons to eighteen. The percentage of training and test data in each real-world data set is allocated as 70 % and 30 %, respectively.

4.2.1.1 Palm oil

Malaysia currently accounts for 39 % of the world's palm oil production and 44 % of world exports. Being one of the biggest producers and exporters of palm oil and its products, Malaysia has an important role to play in fulfilling the growing global need for oil and fat sustainability. Therefore, the price of palm oil in Malaysia impacts global prices. We will apply 14 years of daily palm oil transaction prices (RM), which are calculated by the mean of opening and closing prices on each day, from 2002 to 2016 in the Commodity Futures Market of Malaysia. The time series data of palm oil is shown in Figure 4.5. After data transformation, the palm oil time series data forms a matrix of $(4,991 \times 36)$.

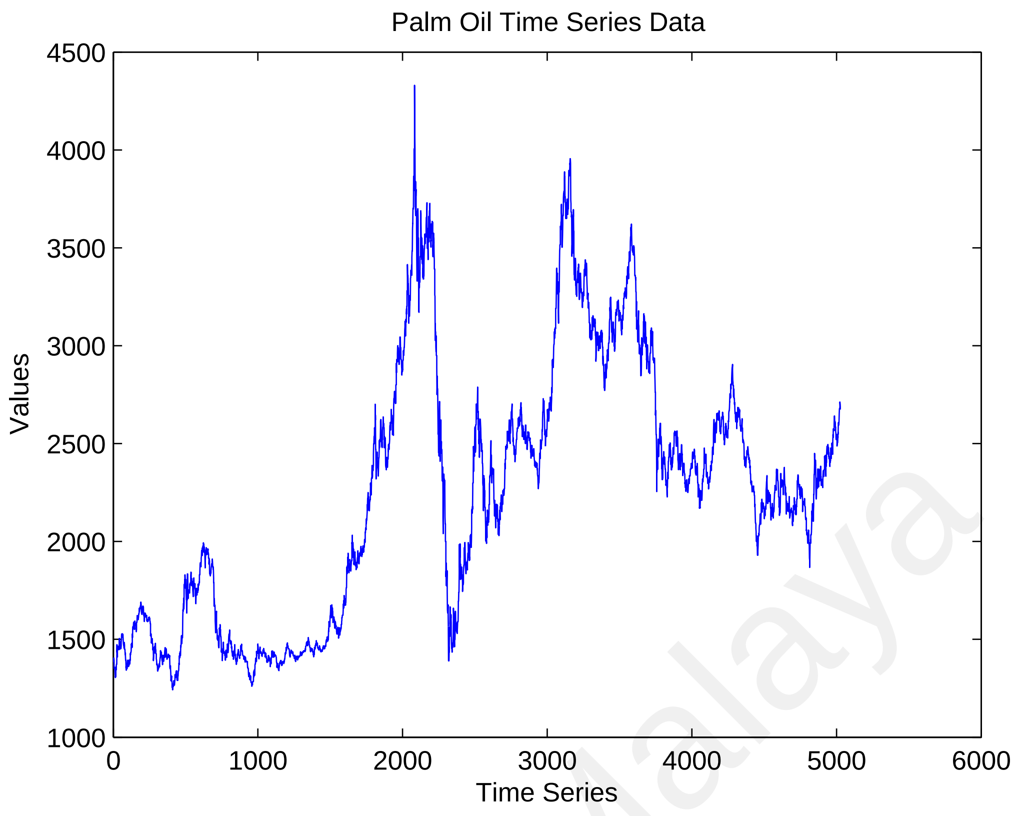


Figure 4.5: Palm Oil Time Series Data

4.2.1.2 Ozone

In recent decades, people have realized that air pollution plays a significant role in the ozone layer. In the stratosphere, the ozone layer can reflect ultraviolet rays and protect people. However, near the ground, the high ozone concentration can damage people's eyes and respiratory system. A high prediction accuracy can help people to arrange their outdoor activities to avoid the effects of air pollution. Therefore, air quality data and daily ozone concentration (parts per billion) in Toronto from 1/1/2003 to 12/31/2010 is also used in the experiment. The Ozone time series data is shown in Figure 4.6. It is available from the database of the Ministry of the Environment in Ontario. The ozone time series data is transformed to a matrix of $(2,887 \times 36)$.

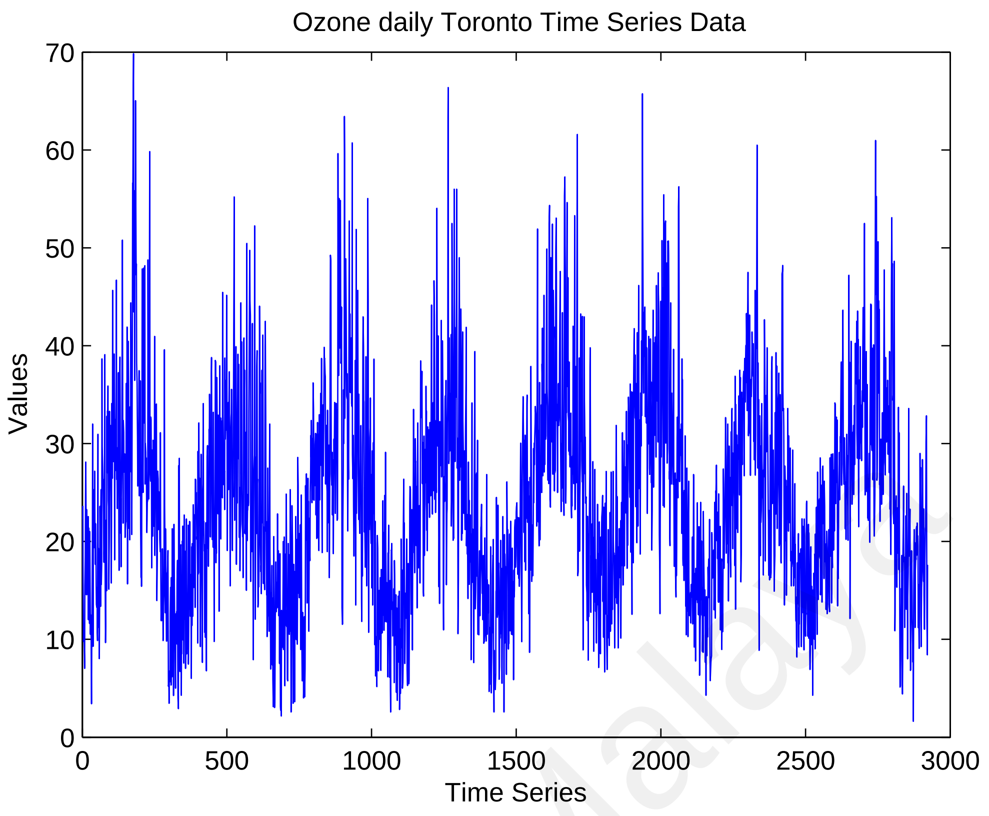


Figure 4.6: Ozone Time Series Data

4.2.1.3 Sunspot

Predicting solar activity is a challenging area and critical topic for researchers and industries. There is a variety of activities that are impacted by solar activity, such as climate, satellites, and so forth. However, because of the complexity of the system and the lack of a mathematical model, predicting solar activity is extremely challenging. Therefore, the third real-life case study is performed using a sunspot time series which is an indication of solar activity. Monthly sunspot series from January 1838 to February 2005 are selected for evaluating the model. Sunspot time series data is shown in Figure 4.7. The dimension of the matrix of the sunspot data is $(3,142 \times 36)$.

4.2.1.4 S&P500

The next real-life case study is performed using S&P500. It is a stock market index based on the market capitalization of 500 large companies listed on the New York Stock

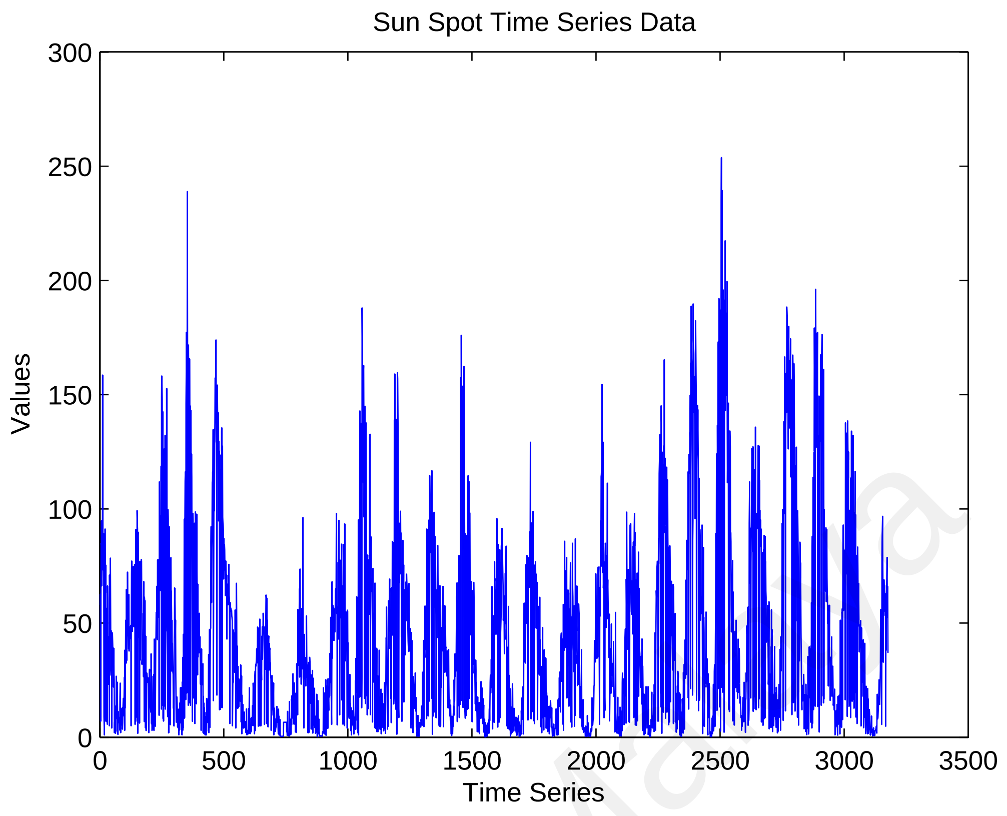


Figure 4.7: Sun Spot Time Series Data

Exchange. It is one of the most commonly followed equity indices, and many consider it to be one of the best representations of the U.S. stock market. The index components and their weightings are determined by S&P Dow Jones Indices and classified as a leading indicator of business cycles. We selected 10 years daily financial time series data in S&P 500 from 12/20/2006 to 12/19/2016. The S&P 500 time series data is shown in Figure 4.8. By (3.1), the S&P500 time series data becomes a matrix of $(2,574 \times 36)$.

4.2.1.5 PC Water Levels

Water resource management for maintaining efficient agricultural production is nationally important. The water level (mm) at Phra Chulachomklao (PC) in the Chao Phraya River of Thailand was collected from 1/1/2012 to 12/31/2012. The data was sampled and stored every hour. The PC station is located near the Gulf of Thailand as shown in Figure 4.9. In this study, we apply historical water level data from PC to predict future water levels. The

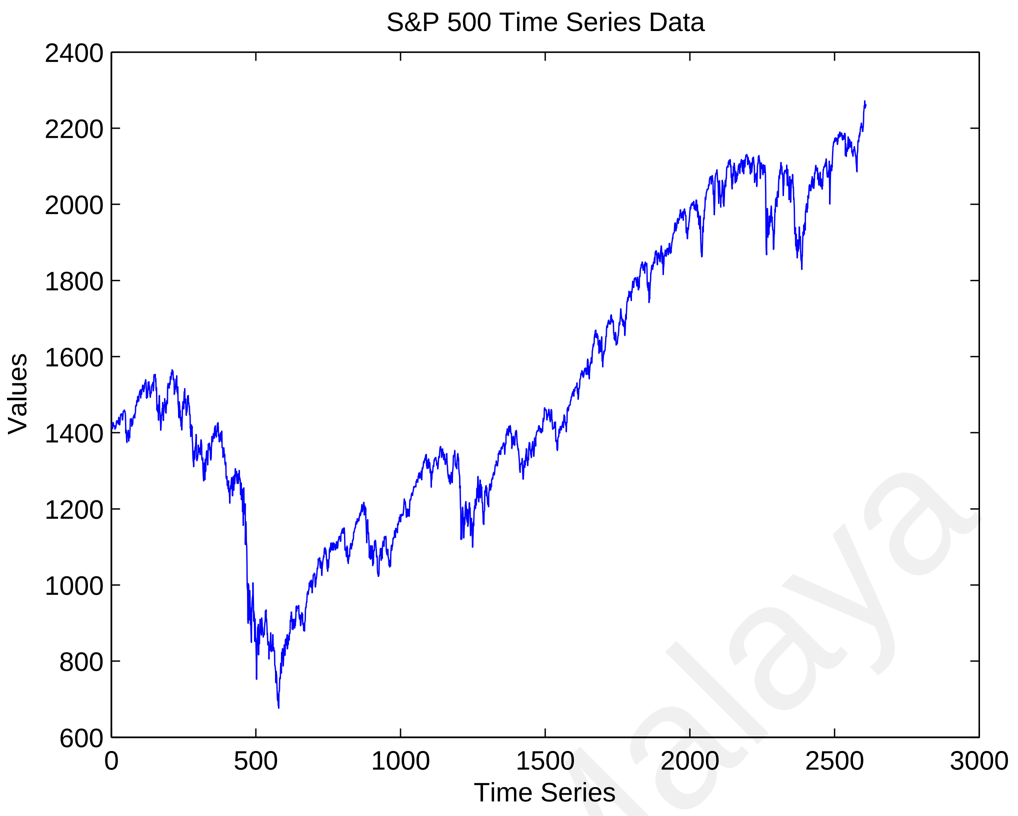


Figure 4.8: S&P500 Time Series Data

water level time series data is shown in Figure 4.10. Finally, the water level time series data is transformed into a matrix of $(8,710 \times 36)$.

4.2.2 Evaluation of Real-world Data

This study evaluates the performance of RKERM with QPSO on the five different real-world data sets mentioned earlier, which include palm oil price, ozone concentration, sunspots, S&P500 and PC water levels. ESN, RSVR, RELM and RKELM are also compared with our proposed model on these data sets.

In order to compare these models fairly, the most suitable parameters need to set for each models. Firstly, according to the default values reported by Huang, the parameters of RKERM are defined as 1.0 and 0.7 for kernel parameter and leaking rate, respectively. Secondly, we pay attention to the algorithms that are optimized by QPSO. QPSO is used to define parameters in RKELM and RKERM in order to ensure that all models have good

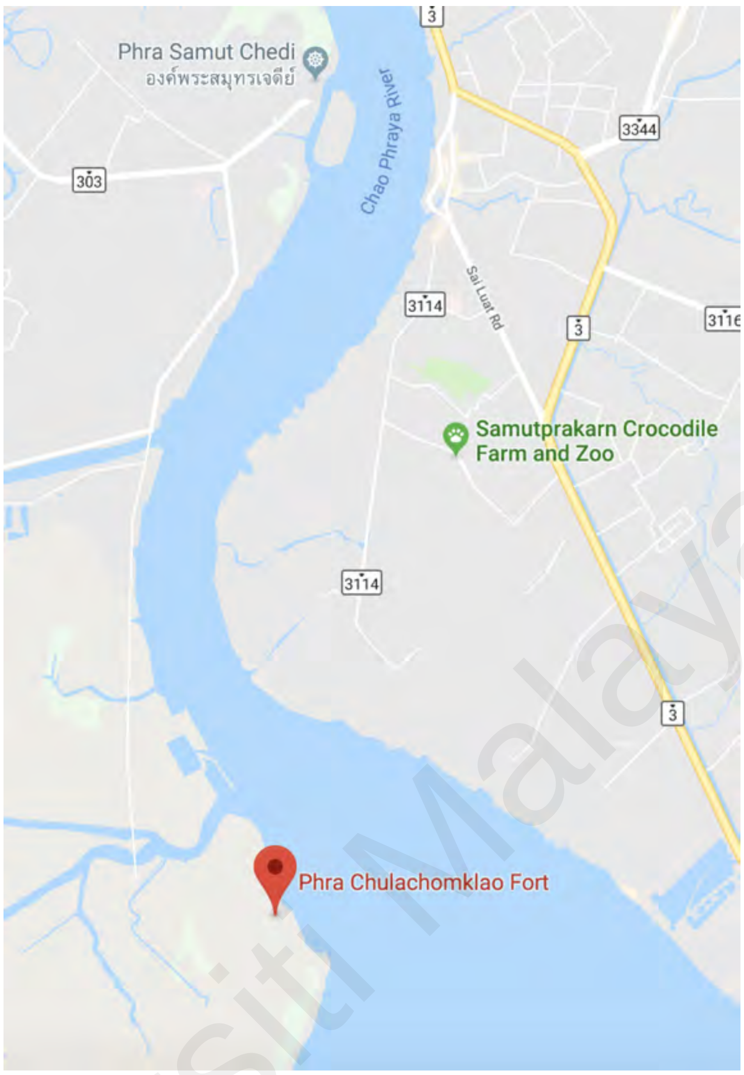


Figure 4.9: Location of PC station in the Chao Phraya River in Thailand

predictive performance. In QPSO, we set 50 iterations and 20 particles in order to find the minimum error rate. The convergent charts for all real-world data sets are shown in Figure 4.11.

The condition of convergence in the real-world data are similar with artificial data sets. The majority of data sets converge around 30 iteration, including 25 iteration for palm oil, 27 iteration for S&P500, 28 iteration for Sunspot, 30 iteration for water level and 38 iteration for Ozone. This means that 50 as iteration of QPSO is accepted. QPSO parameters setting may results the change of optimization time, but there is no affection on the optimizing result.

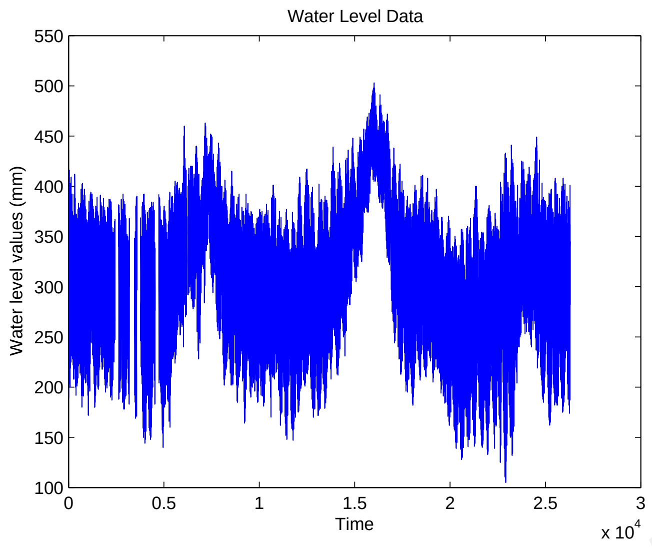
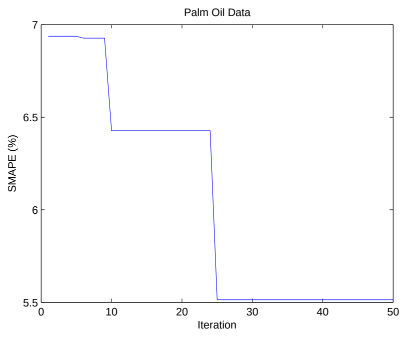


Figure 4.10: Water level time series data

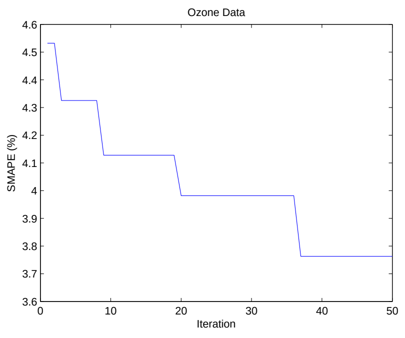
Table 4.3 shows the definition of parameters in all models, including the parameters in RKELM and RKERM obtained by QPSO.

Based on these parameters defined by QPSO and parameter settings based on references in Table 4.3, the multi-step-ahead prediction performances of all five models with their accuracy measures (SMAPE and MSE) are shown in Table 4.4. The columns labeled as 'Average' show the average accuracy measure over the prediction horizon. According to the results of evaluation in synthetic and real-world datasets, although the predicting performance of RKELM with QPSO in the $(1-7)$ step period is not the best comparing with other models, RKERM with QPSO has outstanding predicting performance in the rest of average periods or even majority of specific steps show in the result tables, including 12nd, 14th and 18th step. Especially in the real-world experiment, there is only PC water level that does not show the good performance in every average periods and steps. However, the rest of datasets in RKERM with QPSO show the outstanding predicting ability in different prediction periods for MSE and SMAPE. And RKERM still has the best prediction result in the average period $(8-12)$ and 1-18, compared with other prediction models.

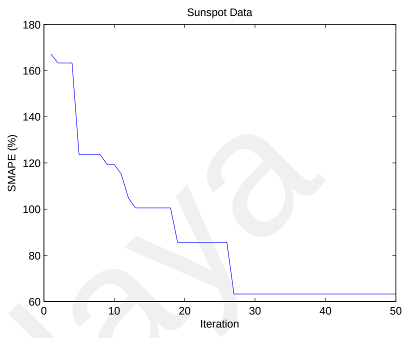
Comparing RKELM with RKERM, the performance of our proposed model in the



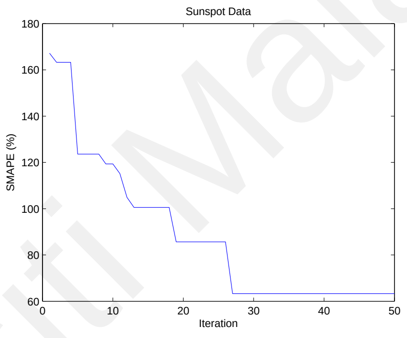
[*PalmOil*]



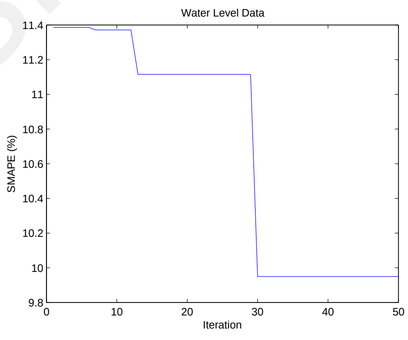
[*Ozone*]



[*SP*500]



[*Sunspot*]



[*WaterLevel*]

Figure 4.11: The convergent chart of RKERM based on QPSO for real-world data

average accuracy (1 − 18) is better than that of RKELM with QPSO. This proves that RC has the ability to enhance the performance of prediction in different prediction horizons. Furthermore, the average results of different prediction steps in the test data sets in Table 4.4 reveals that RKERM with QPSO has the best performance in most of the cases in the five real-world time series data sets, except for PC water level. Although not all the average prediction horizon results of PC water level in our proposed model are the best, the overall average accuracy–between $1 - 18$ step–yield the best performance in the prediction. This reveals that the proposed model has a stable forecasting ability in each step. According to the prediction performance of all data sets in our proposed model, most of the data sets have the highest accuracies in different prediction horizons, except for sunspots with $p = 1$ and PC water level with $p = \{1, 4, 14, 18\}$. As previously mentioned, it is still the best performance on average prediction accuracy. Therefore, unified modified RC with RKELM successfully plays a vital role in time-series prediction. Recurrent multi-step-ahead prediction is not only used to unlock the prediction horizon, but it also can be applied to forecast multiple steps ahead or even for long-term prediction. QPSO also has the ability to efficiently define the parameters of the proposed model.

Table 4.3: Parameter settings of all models for real-world data sets.

Model	Parameter	Data Set				
		Palm oil	Ozone	Sunspots	S&P 500	Water level
ESN		-	(Hung, Dung, et al., 2016)	(Schwenker & Labib, 2009)	(Lin, Yang, & Song, 2008)	-
	N_r	200	50	1000	600	200
	ρ	1.25	0.1	0.01	0.1	1.25
	δ	0.3	0.3	0.79	0.05	0.3
RSVM		-	(Lu & Wang, 2005)	(Samsudin, Shabri, & Saad, 2010)	(Wen, Yang, Song, & Jia, 2010)	-
	C_s	500	100	10	550	500
	λ	7.5E-07	0.001	0.1	2.13E-05	7.5E-07
	δ	0.05	0.1	0.5	2.13E-05	0.05
RELM		-	(Salcedo-Sanz, Casanova-Mateo, Pastor-Sánchez, & Sánchez-Girón, 2014)	(Pan, Zhang, & Xia, 2009)	(Cavalcante & Oliveira, 2015)	-
	L	20	20	30	10	20
RKELM	δ	1.2937	45.6096	1.5381	94.0246	0.6770
RKERM	δ	40.8770	15.4611	0.8629	42.1816	0.1925
	α	0.12E-06	0.9400	0.8900	0.1800	0.7200

Based on the above comparison of results in different prediction horizons, RKERM has super prediction ability rather than other models. Figure 4.12 shows the situation between actual values and prediction values in the 18-th prediction horizon in order to represent the final forecasting ability of RKERM for the different real-world datasets.

Table 4.4: Performance comparison on real-world data for multi-step-ahead prediction.

Data sets	Algorithms	Predicting horizon (*p*)						Average			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
		MSE									
	ESN	9.01E-04	3.81E-04	6.52E-04	1.34E-03	1.64E-03	1.02E-02	2.31E-03	5.85E-03	8.57E-03	5.38E-03
	RSVM	1.89E-04	3.81E-04	6.52E-04	1.34E-03	1.64E-03	2.31E-03	4.10E-04	1.04E-03	1.88E-03	1.08E-03
	RELM	8.35E-05	3.59E-04	6.15E-04	1.10E-03	1.32E-03	1.75E-03	3.51E-04	9.03E-04	1.49E-03	8.83E-04
	RKERM	8.40E-05	3.45E-04	6.23E-04	1.15E-03	1.41E-03	2.08E-03	3.49E-04	9.29E-04	1.65E-03	9.45E-04
	RKELM+QPSO	1.32E-04	3.79E-04	6.45E-04	1.11E-03	1.31E-03	1.71E-03	3.83E-04	9.21E-04	1.46E-03	8.93E-04
	RKERM+QPSO	**8.10E-05**	**3.29E-04**	**5.90E-04**	**1.04E-03**	**1.24E-03**	**1.63E-03**	**3.32E-04**	**8.57E-04**	**1.38E-03**	**8.29E-04**
Palm oil		SMAPE									
	ESN	6.09	9.61	13.66	18.01	18.96	20.33	9.86	16.51	19.33	14.86
	RSVM	3.03	3.90	5.18	7.36	8.04	9.47	4.04	6.52	8.53	6.23
	RELM	1.64	3.75	4.98	6.71	7.31	8.42	3.55	6.05	7.75	5.64
	RKERM	1.64	3.67	5.01	6.81	7.53	9.09	3.53	6.12	8.11	5.78
	RKELM+QPSO	2.17	3.85	5.08	6.69	7.28	8.23	3.75	6.07	7.64	5.69
	RKERM+QPSO	**1.58**	**3.61**	**4.91**	**6.58**	**7.18**	**8.15**	**3.46**	**5.95**	**7.55**	**5.51**
		MSE									
	ESN	3.12E-01	8.03E-02	1.18E-01	2.50E-01	2.66E-01	4.12E-01	1.36E-01	1.55E-01	3.35E-01	2.08E-01
	RSVM	6.22E-03	4.57E-03	4.66E-03	4.47E-03	4.27E-03	4.46E-03	4.94E-03	4.58E-03	4.32E-03	4.63E-03
	RELM	3.64E-03	5.40E-03	3.22E-02	2.28E-01	1.36E+00	2.42E-01	9.78E-03	6.39E-02	4.49E-01	1.71E-01
	RKERM	3.70E-03	4.04E-03	4.16E-03	4.18E-03	4.13E-03	4.37E-03	4.03-03	4.19E-03	4.19E-03	4.13E-03
	RKELM+QPSO	3.52E-03	4.24E-03	4.27E-03	4.31E-03	4.31E-03	4.30E-03	4.09E-03	4.30E-03	4.31E-03	4.22E-03
	RKERM+QPSO	**3.34E-03**	**3.78E-03**	**3.81E-03**	**3.88E-03**	**3.90E-03**	**3.95E-03**	**3.69E-03**	**3.85E-03**	**3.92E-03**	**3.81E-03**
Ozone		SMAPE									
	ESN	9.43	8.85	9.05	8.72	8.45	8.99	8.58	8.92	8.80	8.75
	RSVM	5.18	5.38	5.56	5.36	5.12	5.24	5.40	5.49	5.13	5.34
	RELM	3.51	4.16	5.14	5.28	4.99	5.08	4.44	4.96	5.04	4.78
	RKERM	3.60	3.83	3.93	4.04	4.01	4.12	3.81	4.01	4.04	3.94
	RKELM+QPSO	3.81	3.99	4.04	4.12	4.14	4.19	3.96	4.10	4.15	4.06
	RKERM+QPSO	**3.48**	**3.66**	**3.74**	**3.83**	**3.85**	**3.88**	**3.64**	**3.81**	**3.86**	**3.76**
		MSE									
	ESN	1.35E+00	1.94E+00	2.56E+00	2.51E+00	4.03E+00	2.20E+00	2.14E+00	2.51E+00	2.73E+00	2.44E+00
	RSVM	2.09E-02	2.26E-02	2.42E-02	2.82E-02	2.94E-02	3.27E-02	2.27E-02	2.64E-02	3.10E-02	2.65E-02
	RELM	**2.07E-02**	2.27E-02	2.69E-02	2.91E-02	3.10E-02	3.57E-02	2.37E-02	2.71E-02	3.25E-02	2.75E-02
	RKERM	2.26E-02	2.30E-02	2.48E-02	2.81E-02	2.99E-02	3.19E-02	2.35E-02	2.63E-02	3.05E-02	2.65E-02
	RKELM+QPSO	2.60E-02	2.70E-02	2.80E-02	3.09E-02	3.17E-02	3.36E-02	2.72E-02	2.96E-02	3.28E-02	2.97E-02
	RKERM+QPSO	2.23E-02	**2.27E-02**	**2.42E-02**	**2.72E-02**	**2.91E-02**	**3.07E-02**	**2.33E-02**	**2.62E-02**	**2.95E-02**	**2.63E-02**
Sunspot		SMAPE									
	ESN	104.22	112.06	112.53	125.00	130.10	126.74	110.14	121.57	126.78	118.86
	RSVM	56.27	59.45	62.41	66.74	68.46	71.68	59.71	65.12	70.05	64.66
	RELM	**55.99**	60.30	64.61	69.27	72.07	72.63	60.73	67.27	72.41	66.44
	RKERM	57.42	58.96	61.95	65.44	67.03	69.57	59.59	64.27	67.84	63.62
	RKELM+QPSO	58.87	62.04	64.73	67.33	68.86	70.75	62.37	66.54	69.90	66.04
	RKERM+QPSO	57.11	**58.84**	**61.68**	**64.50**	**66.46**	**68.64**	**59.53**	**63.73**	**67.11**	**63.25**
		MSE									
	ESN	2.81E-04	8.21E-03	1.23E-02	6.48E-03	1.59E-02	9.91E-02	7.89E-03	9.02E-03	4.41E-02	2.03E-02
	RSVM	6.03E-02	5.32E-02	5.10E-02	4.09E-02	3.56E-02	3.37E-02	5.43E-02	4.55E-02	3.49E-02	4.54E-02
	RELM	1.35E-03	3.31E-03	6.10E-03	1.46E-03	3.40E-03	3.26E-03	3.54E-03	4.18E-03	2.29E-03	3.30E-03
	RKERM	1.23E-03	1.17E-02	3.03E-02	5.82E-02	6.14E-02	7.15E-02	1.37E-02	5.09E-02	6.71E-02	4.19E-02
	RKELM+QPSO	3.96E-03	4.65E-03	5.41E-03	6.95E-03	7.72E-03	9.83E-03	4.66E-03	6.30E-03	8.48E-03	6.39E-03
	RKERM+QPSO	**1.20E-04**	**4.16E-04**	**6.66E-04**	**1.03E-03**	**1.16E-03**	**1.39E-03**	**4.04E-04**	**8.86E-04**	**1.25E-03**	**8.19E-04**
S&P500		SMAPE									
	ESN	1.51	9.10	11.21	7.63	12.32	37.41	7.99	9.05	21.24	12.70
	RSVM	30.82	28.62	27.98	24.59	22.72	22.80	28.96	26.18	22.50	26.03
	RELM	3.83	6.22	8.63	3.75	6.14	5.94	6.31	6.20	4.74	5.76
	RKERM	3.44	11.39	19.36	28.08	28.77	31.20	11.49	26.02	30.20	21.76
	RKELM+QPSO	7.13	7.72	8.35	9.48	10.00	11.30	7.73	9.02	10.47	9.00
	RKERM+QPSO	**0.95**	**1.80**	**2.31**	**2.87**	**3.02**	**3.31**	**1.72**	**2.66**	**3.12**	**2.45**
		MSE									
	ESN	8.64E-02	6.57E-02	8.03E-02	6.58E-02	4.38E-02	3.82E-02	8.79E-02	9.87E-02	5.58E-02	8.02E-02
	RSVM	5.98E-04	**3.49E-03**	5.94E-03	7.68E-03	9.55E-03	1.14E-02	**3.47E-03**	7.99E-03	9.77E-03	7.55E-03
	RELM	**1.70E-03**	7.97E-03	1.03E-02	4.00E-03	3.78E-03	**3.33E-03**	7.02E-03	8.24E-03	4.12E-03	6.39E-03
	RKERM	3.66E-03	6.15E-03	6.94E-03	7.22E-02	7.47E-03	6.96E-03	5.73E-03	7.22E-03	7.41E-03	6.71E-03
	RKELM+QPSO	3.19E-03	1.07E-02	1.76E-02	5.96E-03	**4.34E-03**	3.73E-03	1.06E-02	1.12E-02	**4.14E-03**	8.61E-03
	RKERM+QPSO	2.59E-03	4.87E-03	**3.96E-03**	**3.34E-03**	4.69E-03	4.90E-03	4.21E-03	**4.73E-03**	4.73E-03	**4.53E-03**
PC Water Level		SMAPE									
	ESN	39.89	37.68	44.51	35.20	30.44	31.12	41.72	44.42	32.71	39.47
	RSVM	3.17	**7.52**	9.72	10.41	11.21	14.98	**7.11**	10.12	13.47	10.24
	RELM	**5.91**	11.84	13.80	8.80	8.68	**8.13**	11.05	12.75	8.91	10.81
	RKERM	9.15	11.13	11.70	12.07	12.37	12.46	10.79	12.03	12.53	11.72
	RKELM+QPSO	6.17	12.84	16.74	9.86	**8.55**	8.28	12.24	13.17	**8.49**	11.25
	RKERM+QPSO	8.29	10.29	**9.59**	**8.66**	9.93	10.11	9.84	**10.05**	9.98	**9.95**

Note: The best performance is in boldface.

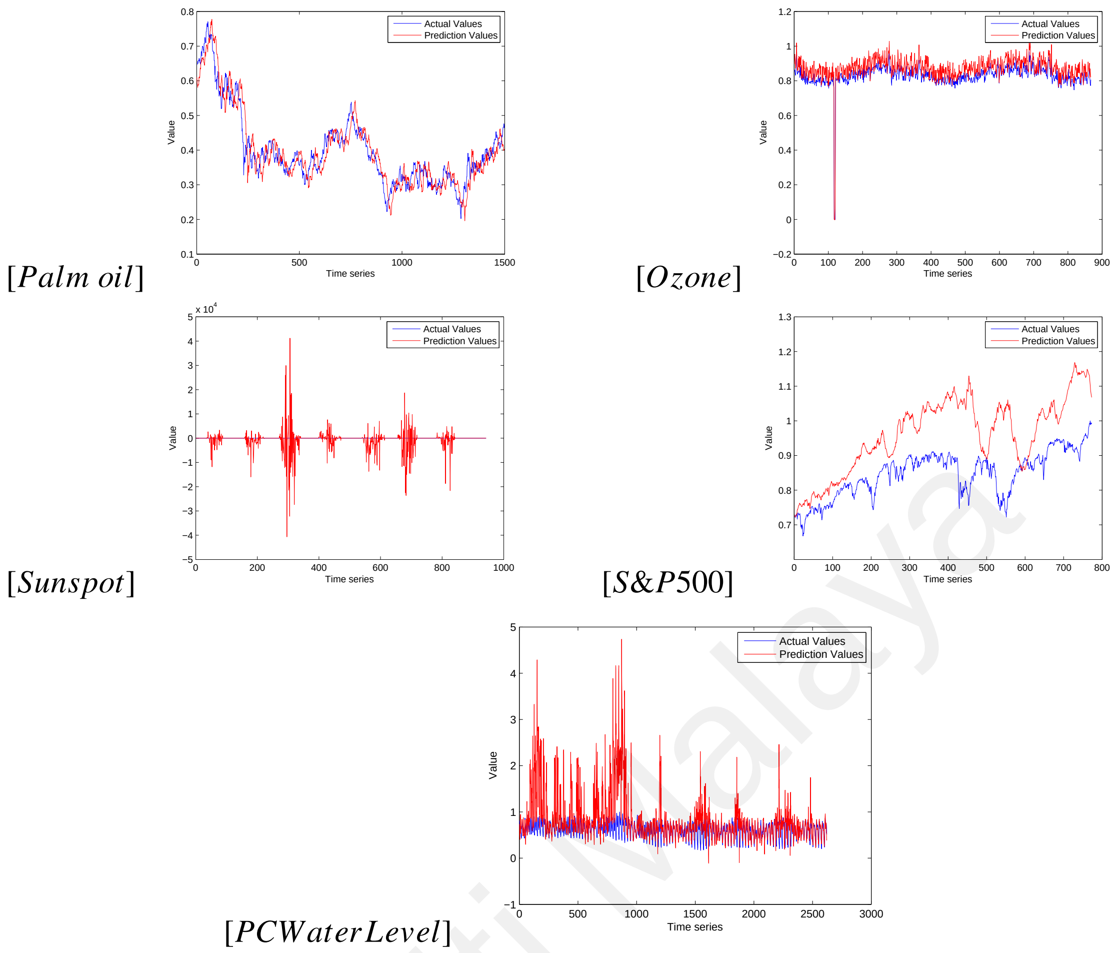


Figure 4.12: The comparison between actual values and prediction values in real-world time series data sets in the 18-th step of RKERM.

4.3 Statistical Comparison

In this section, we applied Wilcoxon Signed Rank Test (Siegal, 1956), which is a popular nonparametric test for matched or paired data, to prove RKERM with QPSO is superior in time series prediction than RKELM with QPSO. This test is mainly based on difference scores, in addition to analyzing the signs of the differences. It also takes into account the magnitude of the observed differences. Firstly, the difference d_i between the accuracy in SMAPE of the two algorithms on seven testing real-world data is computed. Secondly, these differences are ranked according to their absolute values. Table 5.11 shows the different values of (1-SMAPE) in seven testing data and its rank corresponding to two models. Then R^+ and R^- are computed, respectively. R^+ is the sum of ranks for the positive and R^- is the sum of ranks for the negative. Finally, the value of R^+ and R^- are 28 and 0, respectively. According to the table of exact critical values for the two-tails Wilcoxon's test, for a confidence level of $\alpha = 0.05$, the difference between the algorithms is significant if the smaller of the sums is equal to or less than 1. Therefore, we can conclude that RKERM is statistically superior to the RKELM.

Table 4.5: The comparison of RKELM and RKERM

Data set	RKERM with QPSO	RKELM with QPSO	Difference		Rank
Mack-Glass	81.48	69.08	12.40	+	7
Lorenz	95.67	95.55	0.12	+	1
Palm oil	94.49	94.31	0.18	+	2
Ozone	96.24	95.94	0.30	+	3
Sunspot	36.75	33.96	2.79	+	5
S&P500	97.55	91.00	6.55	+	6
PC water level	90.05	88.75	1.30	+	4

4.4 Discussion

In this chapter, RKELM-QPSO refers to Recurrent Kernel Extreme Reservoir Machine with QPSO, which was based on reservoir computing to create a reservoir that contains the information of all input data before calculating output weight, is evaluated by MSE and SMAPE for benchmark and real-world datasets. According to the results from experiment 1 and 2, due to recurrent multi-step ahead prediction algorithm, the proposed model can predict in different prediction horizons. It not only deal with time series prediction, but also releases the restriction of prediction horizon. Furthermore, recursive computing also plays a vital role in generating a fixed reservoir in the learning part, which connects each neuron of the model and enhances the prediction ability in time series prediction, especially for long-term prediction.

According to the results of the experiments, RKERM-QPSO can deal with different kinds of time-series data in different prediction horizons and obtains the best predictive performance compared with other explored models. The major benefits of RKERM-QPSO are good generalization model for chaotic data can be produced, the limitation of prediction horizon is released, and the reservoir computing can be applied in time series prediction with improved prediction accuracy in multi-steps prediction. QPSO also can define the parameters of prediction model and enhance forecasting accuracy. Therefore, the proposed offline learning model has a good generalization capacity for time series data, and it is the first time that recurrent multi-step-ahead prediction can handle time series prediction and overcome the limitations of prediction horizon. Furthermore, RC is applied into the proposed model, which can generate a fixed reservoir with high dimension information for improving the forecasting accuracy. Lastly, this proposed offline model solves the parameter dependency by QPSO.

4.5 Summary

This chapter discussed and explained the implementation and evaluation of the proposed recurrent offline time series prediction model (RKERM-QPSO). Two different measurements are employed to evaluate the model's predictive performance against the popular benchmark models using two synthetic and five real-world from different fields time series data sets. The outcomes from these evaluations showed that RKERM-QPSO had a more adaptable forecasting results in different prediction periods rather than other models. Based on the experimental results, Recurrent multi-step ahead algorithm solves the restriction of prediction horizon. Recursive computing, which generates a fixed reservoir that can save more information and connect each neurons of model, increases the forecasting performance in different prediction horizons. The parameter dependency problem is solved by QPSO, which found the most suitable kernel parameters and leaking rate in the proposed model. Therefore, RKERM-QPSO not only solved the parameter dependency problem and restriction of prediction horizon, but it also generated a fixed reservoir for enhancing the prediction performance.

CHAPTER 5: EXPERIMENTS DESIGN AND RESULTS FOR ONLINE LEARNING MODEL

In this chapter, in order to compare the performance of two online learning models in time series prediction, six synthetic data sets and three real-world time series data sets are employed to evaluate the ability of kernel filters, the performance of concept drift detectors and meta-cognitive learning strategy in time series prediction, including Standard & Pool 500, Shanghai SSE Composite Index, and the ozone concentration of Toronto. There are two experiments in the following sections. The first experiment consists of three parts. Firstly, in order to prove that the kernel method and kernel filter play a vital role in the prediction model, it compares Recurrent Online Sequential Extreme Learning Machine (ROS-ELM) and Recurrent Kernel Online Sequential Extreme Learning Machine with two different kernel adaptive filters. Then it compares two different concept drift detectors in the prediction model for finding the most suitable method that can detect concept drift problem in the process of time series prediction. Finally, a new meta-cognitive learning strategy is applied in our proposed model to check its role in our proposed model. In the second experiment, the study compares the result of the prediction models that have been proved in the first experiment with that of model with recursive kernel method.

5.1 Experiments 1

This section not only compares the basic ROS-ELM with Recurrent Kernel Online Sequential Extreme Learning Machine with different Kernel adaptive filters (FB and ALD), but also applies DDM and ECDD into the model of Recurrent Kernel Online Sequential Extreme Learning Machine to check the ability to solve concept drift problem. Although concept drift has been studied by many researchers, the effects of concept drift in the time series have not been studied widely. Due to the lack of source which simulates the change of environment in time series and invisible characteristic of concept drift in time series, the synthetic datasets can show the different pattern in time series and evaluate the ability of model to deal with concept drift. On the other hand, three different real-world time series datasets also are used in the same experiment, which include Standard&Poor‘s 500 ($S\&P500$), Shanghai Stock Exchange Composite Index (Shanghai) and Ozone Concentration of Toronto (Ozone).

5.1.1 Synthetic Data Experiments

In the synthetic experiments, we generate six time series datasets, including three artificial time series without concept drift and three time series with concept drift. The time series without concept drift is used to check the generalization of the model. Firstly, the autoregressive process is used to generate 20,035 values for each time series data. The three time series without concept drift, including Time Series 1 (TS_1), Time Series 2 (TS_2) and Time Series 3 (TS_3), are generated by 5.1–5.3, respectively.

$$X_t = 1.5X_{t-1} - 0.4X_{t-2} - 0.3X_{t-3} + 0.2X_{t-4} + w_t \tag{5.1}$$

$$X_t = -0.1X_{t-1} + 1.2X_{t-2} + 0.4X_{t-3} - 0.5X_{t-4} + w_t \tag{5.2}$$

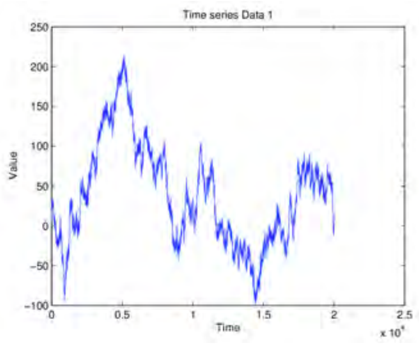
$$\begin{aligned} X_t &= 0.9X_{t-1} + 0.8X_{t-2} - 0.6X_{t-3} + 0.2X_{t-4} \\ &\quad -0.5X_{t-5} - 0.2X_{t-6} + 0.4X_{t-7} + w_t \end{aligned} \tag{5.3}$$

where $w_(t)$ is white noise in the t-th time series.

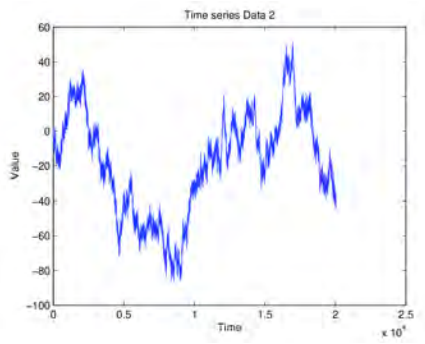
These six time series datasets are shown in Figure **??**.

Then, the combinations of different autoregressive processes are generated as synthetic data with concept drift. Each dataset has 20,035 values and the concept drift appears in the 10,001-st value in the time series. Time Series 4 (TS_4) is a combination of (5.1) and (5.2); Time Series 5 (TS_5) is a combination of (5.3) and (5.1); and Time Series 6 (TS_6)

$[TS_1]$



$[TS_2]$



$[TS_3]$

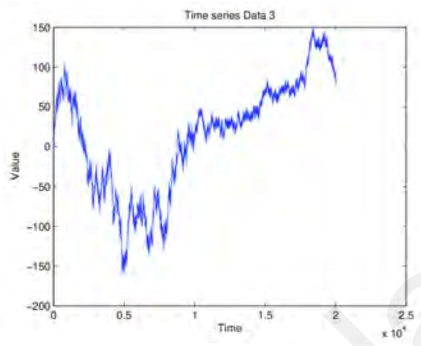
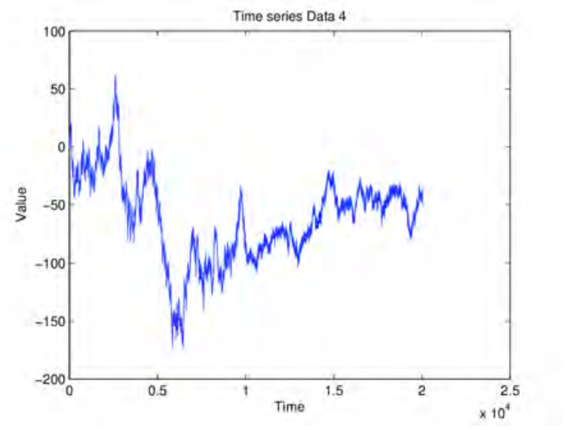


Figure 5.1: Synthetic time series datasets without concept drift (TS_1, TS_2, TS_3).

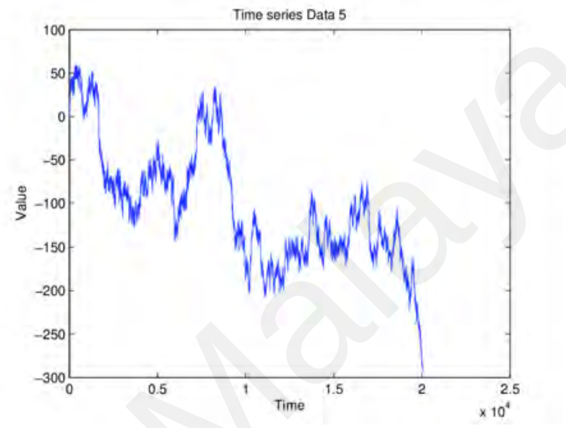
that was built by combining (5.3) and (5.2). These synthetic datasets with concept drift are represented in Figure 5.2.

The performances of all experiments are recorded using an Intel Core i7 processor with 16 GB of RAM. In the application of time series prediction, the reasonable period of prediction plays a vital role in analyzing trend and report forecasting information. In (Bao et al., 2014), the authors proposed that 18 steps ahead is a long prediction period. Therefore, the size of time window and prediction horizon in the following experiments are defined as 18. In the other words, we use 18 steps historical data to predict the next 18 steps. The synthetic time series datasets need to be transformed to adapt to the assumption of the experiment. After data transformation, the six synthetic time series datasets become six matrices with 20,000 rows and 36 columns. We employ 14,000 rows of the matrix in each artificial data to train the models and the remaining part for testing.

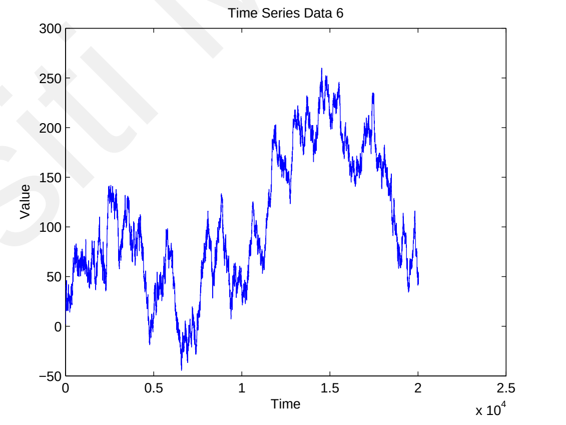
To evaluate the performance of first model in online learning algorithm, the study employs ROS-ELM, recurrent kernel online sequential extreme learning machine with



$[TS_4 = TS_1 + TS_2]$



$[TS_5 = TS_1 + TS_3]$



$[TS_6 = TS_2 + TS_3]$

Figure 5.2: Synthetic time series datasets with concept drift (TS_4, TS_5, TS_6).

different adaptive kernel filters and with different concept drift detectors to prove that our proposed model has the best performance in time series prediction and can deal with concept drift problem in time series stream.

Firstly, three models, including ROS-ELM and recurrent kernel online sequential extreme learning machine with two different adaptive kernel filters (FB and ALD), are compared in the first comparison in order to prove whether the kernel method and adaptive kernel filter can improve the forecasting performance. Due to the limitation of ROS-ELM, we define the size of initialized training data is 200 and select the best number of hidden neurons from 14 different hidden neurons (5, 10, 15, 20, 30, 40, 50, 70, 100, 150, 200, 300, 500, 1000). Furthermore, the threshold (ξ) of ALD was searched uniformly in $\{0.00010, 0.00011, \ldots, 0.001\}$, while the maximum of hidden neurons in recurrent kernel online sequential extreme learning machine with FB is defined as 200 in order to compare with ROS-ELM fairly. The reason of select 1000 as the maximum number of hidden neurons is thinking about over-fitting problem that impacts on the performance of prediction.

First, by searching, the number of hidden neurons and threshold of ALD that leads to the best performance in the learning are found in different six synthetic datasets, respectively. Table 5.1 shows the performance in six artificial datasets of ROS-ELM and recurrent kernel online sequential extreme learning machine with two types of filters in the different prediction horizons and the average values of different periods of prediction horizons. There are six varieties of prediction horizons (p) and the average value over the four different certain periods of prediction horizon. The performance of model is measured by MSE and SMAPE in the experiments of synthetic datasets. Although the performance of recurrent kernel online sequential extreme learning machine with ALD in synthetic TS_5 is not the best in every prediction horizons, it still has the best performance in the average value of SMAPE over prediction horizon $1-18$.

The SMAPE results of Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency Filter (ALD-RKOS-ELM) in synthetic TS_4 has different performance in several prediction horizons, such as the fourth, seventh and fourteenth prediction horizon. These results show that Recurrent Kernel On-line Sequential Extreme Learning Machine with Fixed-Budget Filter (FB-RKOS-ELM) has the best performance in certain specific prediction horizons. However, the MSE results of synthetic

Table 5.1: The performance comparison on synthetic data for multi-step-ahead prediction between ROS-ELM and recurrent kernel online sequential extreme learning machine with different kernel filters. The best performance is in boldface.

Datasets	Algorithms	Predicting horizon (p)						Average predicting horizon			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
		MSE									
	ROS-ELM	1.72E-04	4.16E-04	2.08E-01	9.22E-03	4.48E-02	3.81E+00	3.01E-02	3.23E-02	1.25E+01	4.17E+00
	FB-RKOS-ELM	1.81E-04	4.15E-04	8.58E-04	1.24E-03	1.51E-03	1.40E-03	4.64E-04	1.09E-03	1.42E-03	9.58E-04
	ALD-RKOS-ELM	**2.14E-05**	**1.39E-04**	**3.52E-04**	**6.23E-04**	**8.89E-04**	**1.07E-03**	**1.57E-04**	**5.35E-04**	**9.36E-04**	**5.22E-04**
TS_1		SMAPE									
	ROS-ELM	5.64	6.81	40.81	18.96	23.90	87.18	11.25	17.61	46.72	24.81
	FB-RKOS-ELM	4.19	5.96	7.88	10.98	11.35	11.44	5.89	9.54	11.30	8.71
	ALD-RKOS-ELM	**1.56**	**3.72**	**5.86**	**7.57**	**9.10**	**9.84**	**3.61**	**7.13**	**9.21**	**6.45**
		MSE									
	ROS-ELM	8.72E-05	2.69E-04	2.35E-04	4.23E-04	4.66E-04	5.64E-04	1.96E-04	3.58E-04	5.38E-04	3.55E-04
	FB-RKOS-ELM	1.83E-04	2.44E-04	3.06E-04	4.27E-04	4.77E-04	5.96E-04	2.33E-04	3.77E-04	5.15E-04	3.67E-04
	ALD-RKOS-ELM	**5.74E-05**	**1.45E-04**	**2.17E-04**	**3.67E-04**	**4.30E-04**	**5.47E-04**	**1.25E-04**	**3.08E-04**	**4.70E-04**	**2.91E-04**
TS_2		SMAPE									
	ROS-ELM	1.14	2.01	1.89	2.52	2.66	2.95	1.67	2.33	2.86	2.25
	FB-RKOS-ELM	1.66	1.92	2.15	2.56	2.71	3.03	1.87	2.39	2.81	2.33
	ALD-RKOS-ELM	**0.94**	**1.49**	**1.81**	**2.36**	**2.55**	**2.91**	**1.33**	**2.16**	**2.68**	**2.01**
		MSE									
	ROS-ELM	1.45E-01	5.48E+01	1.34E+02	8.68E+00	1.40E-01	3.29E+00	4.42E+01	3.70E+00	1.36E+00	1.87E+01
	FB-RKOS-ELM	3.13E-02	4.31E-02	5.77E-02	7.55E-02	7.72E-02	6.75E-02	4.35E-02	6.95E-02	7.44E-02	6.11E-02
	ALD-RKOS-ELM	**6.81E-03**	**2.93E-02**	**4.33E-02**	**5.34E-02**	**5.36E-02**	**6.32E-02**	**2.18E-02**	**4.62E-02**	**4.78E-02**	**3.72E-02**
TS_3		SMAPE									
	ROS-ELM	21.64	43.29	57.70	36.01	44.02	44.85	38.23	44.83	37.86	39.94
	FB-RKOS-ELM	13.68	16.96	21.18	26.32	27.18	24.93	17.09	24.63	26.56	22.34
	ALD-RKOS-ELM	**4.15**	**10.78**	**14.89**	**18.65**	**19.11**	**22.36**	**8.86**	**16.47**	**18.26**	**14.11**
		MSE									
	ROS-ELM	2.34E-04	4.41E-05	5.91E-05	1.25E-04	1.36E-04	3.42E-04	8.74E-05	1.15E-04	5.50E-04	2.49E-04
	FB-RKOS-ELM	5.55E-05	6.34E-05	8.83E-05	1.26E-04	1.66E-04	2.31E-04	6.55E-05	1.07E-04	1.88E-04	1.18E-04
	ALD-RKOS-ELM	**1.96E-05**	**4.17E-05**	**6.57E-05**	**1.06E-04**	**1.60E-04**	**1.67E-04**	**3.65E-05**	**9.23E-05**	**1.48E-04**	**8.91E-05**
TS_4		SMAPE									
	ROS-ELM	8.79	3.35	**3.85**	5.55	**5.75**	7.26	4.76	5.32	8.06	6.02
	FB-RKOS-ELM	3.60	3.95	4.85	5.73	6.53	7.72	4.03	5.28	6.99	5.36
	ALD-RKOS-ELM	**2.13**	**3.15**	4.20	**5.08**	6.21	**6.33**	**2.94**	**4.84**	**6.00**	**4.49**
		MSE									
	ROS-ELM	2.16E-03	**1.46E-04**	1.24E-03	8.67E-04	8.40E-04	7.90E-02	3.18E-03	5.72E-03	1.72E-02	8.57E-03
	FB-RKOS-ELM	6.36E-04	8.34E-04	5.06E-04	**7.46E-04**	**6.01E-04**	**1.22E-03**	6.91E-04	**7.13E-04**	**7.99E-04**	**7.33E-04**
	ALD-RKOS-ELM	**5.48E-05**	2.10E-04	**1.24E-03**	2.88E-03	2.39E-03	1.59E-03	**4.80E-04**	1.56E-03	1.60E-03	1.15E-03
TS_5		SMAPE									
	ROS-ELM	6.40	**2.91**	6.38	9.13	7.58	17.19	6.44	8.34	10.01	8.16
	FB-RKOS-ELM	6.93	7.78	**5.72**	**6.85**	**6.11**	9.27	6.92	**6.77**	**7.05**	6.92
	ALD-RKOS-ELM	**1.96**	3.72	6.66	8.57	8.59	**8.77**	**4.15**	7.26	8.03	**6.31**
		MSE									
	ROS-ELM	2.12E-01	3.53E-02	1.81E-02	4.65E-02	7.95E-02	7.11E-02	1.02E-01	9.99E-02	3.91E-01	1.98E-01
	FB-RKOS-ELM	1.91E-02	2.47E-02	3.34E-02	5.49E-02	6.25E-02	5.88E-02	2.52E-02	4.62E-02	6.21E-02	4.34E-02
	ALD-RKOS-ELM	**2.98E-04**	**1.89E-03**	**1.50E-03**	**2.44E-02**	**4.28E-02**	**3.46E-02**	**1.88E-03**	**2.45E-02**	**6.18E-02**	**3.15E-02**
TS_6		SMAPE									
	ROS-ELM	42.45	18.06	13.80	26.11	34.81	31.07	30.76	41.18	34.66	34.96
	FB-RKOS-ELM	9.56	10.91	12.81	17.09	18.67	18.14	10.98	15.38	18.67	14.77
	ALD-RKOS-ELM	**1.27**	**2.93**	**2.80**	**10.33**	**13.25**	**12.56**	**2.74**	**10.27**	**16.84**	**9.54**

TS_4 indicates that ALD-RKOS-ELM still has the best performance in all prediction horizon. It also has the best performance in all average values of different periods of prediction horizon under the measuring criteria MSE and SMAPE. Therefore, based on the results of Table 5.1, ALD-RKOS-ELM in the most of the synthetic datasets has the best predicting performance in not only every prediction horizon but also in the average values of different periods of prediction horizon. The kernel method and ALD kernel filter play a significant role in the time series prediction. As compared with these combined datasets (TS_4, TS_5, and TS_6) that have concept drift problem, ALD-RKOS-ELM has the more stable predicting performance in two different measuring criteria for the three datasets (TS_1, TS_2 and TS_3). ALD-RKOS-ELM is shorted for RKOS-ELM in the following content.

Next, the two types of methods, including DDM and ECDD, are applied to solve concept drift problem in the experiments. RKOS-ELM is combined with DDM and ECDD, respectively. Table 5.9 compares the different performance between DDM and ECDD that apply in ALD-RKOS-ELM. The performance of model is represented by the three measuring criteria in each dataset, including MSE and SMAPE. As the threshold of ALD that impacts on the results of prediction, we seek the suitable threshold of RKOS-ELM with DDM and ECDD in {0.00010, 0.00011, . . . , 0.001}, respectively. The performance of RKOS-ELM with different concept drift detectors are shown in Table 5.9. Comparing with the result of ALD-RKOS-ELM in Table 5.1, both of concept drift detectors impact on multi-steps prediction in positive way. However, DDM plays a more significant role in the model of RKOS-ELM for enhancing the performance of prediction than ECDD, especially for the combined datasets that have concept drift problem.

Though the performance of ECDD in TS_1 is better than that of DDM, comparing with the results of RKOS-ELM, the results of Recurrent Kernel Online Sequential Extreme Learning Machine with Approximate Linear Dependency and Drift Detector Mechanism

Table 5.2: The performance comparison on synthetic data for multi-step-ahead prediction by RKOS-ELM with different concept detectors. The best performance is in boldface.

Datasets	Algorithms	Predicting horizon (p)						Average predicting horizon			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
		MSE									
	RKOS-ELM with ECDD	**2.15E-05**	**1.36E-04**	3.41E-04	**6.03E-04**	**7.26E-04**	**1.02E-03**	1.54E-04	**5.00E-04**	**8.27E-04**	**4.74E-04**
	RKOS-ELM with DDM	2.32E-05	1.37E-04	**3.41E-04**	6.32E-04	7.30E-04	1.04E-03	**1.53E-04**	5.05E-04	8.46E-04	4.82E-04
TS_1		SMAPE									
	RKOS-ELM with ECDD	**1.54**	**3.59**	5.74	**7.37**	**8.13**	**9.66**	**3.53**	**6.72**	**8.61**	**6.11**
	RKOS-ELM with DDM	1.73	3.69	**5.71**	7.78	8.19	9.73	3.56	6.83	8.77	6.20
		MSE									
	RKOS-ELM with ECDD	**5.69E-05**	1.43E-04	**2.15E-04**	**3.62E-04**	**4.25E-04**	5.44E-04	1.24E-04	**3.04E-04**	4.66E-04	2.88E-04
	RKOS-ELM with DDM	5.71E-05	**1.43E-04**	2.17E-04	3.63E-04	4.29E-04	**5.39E-04**	**1.24E-04**	3.05E-04	**4.66E-04**	**2.88E-04**
TS_2		SMAPE									
	RKOS-ELM with ECDD	0.93	1.48	1.81	**2.35**	**2.54**	2.90	**1.32**	2.15	**2.66**	2.00
	RKOS-ELM with DDM	**0.93**	**1.48**	**1.81**	2.36	b2.55	**2.88**	1.33	**2.15**	2.67	**2.00**
		MSE									
	RKOS-ELM with ECDD	8.59E-03	1.78E-02	4.28E-02	6.08E-02	6.11E-02	2.55E-02	2.34E-02	4.90E-02	5.58E-02	4.13E-02
	RKOS-ELM with DDM	**7.41E-03**	**1.48E-02**	**3.05E-02**	**5.82E-02**	**2.67E-02**	**2.46E-02**	**1.83E-02**	**4.66E-02**	**3.36E-02**	**3.13E-02**
TS_3		SMAPE									
	RKOS-ELM with ECDD	4.84	7.97	15.21	20.79	20.56	12.35	9.35	17.31	19.34	14.89
	RKOS-ELM with DDM	**4.42**	**7.12**	**12.04**	**19.75**	**11.69**	**11.67**	**8.03**	**15.23**	**14.09**	**12.05**
		MSE									
	RKOS-ELM with ECDD	2.01E-05	4.05E-05	**5.71E-05**	9.59E-05	1.17E-04	1.82E-04	**3.37E-05**	8.27E-05	1.40E-04	8.29E-05
	RKOS-ELM with DDM	**1.99E-05**	**4.03E-05**	5.78E-05	**9.51E-05**	**1.17E-04**	**1.71E-04**	3.41E-05	**8.17E-05**	**1.35E-04**	**8.10E-05**
TS_4		SMAPE									
	RKOS-ELM with ECDD	2.15	3.06	**3.66**	4.87	**5.20**	6.65	**2.74**	4.55	5.88	4.29
	RKOS-ELM with DDM	**2.15**	**3.02**	3.79	**4.78**	5.40	**6.61**	2.78	**4.45**	**5.82**	**4.26**
		MSE									
	RKOS-ELM with ECDD	**2.28E-05**	2.75E-04	6.43E-04	**5.06E-04**	**7.80E-04**	9.53E-03	**3.02E-04**	6.28E-04	7.78E-04	5.51E-04
	RKOS-ELM with DDM	3.56E-05	**2.19E-04**	**2.81E-04**	6.04E-04	7.92E-04	8.25E-04	**3.22E-04**	**6.21E-04**	**7.23E-04**	**5.39E-04**
TS_5		SMAPE									
	RKOS-ELM with ECDD	2.97	3.99	5.67	**5.95**	**6.88**	7.58	3.77	**5.99**	6.97	5.46
	RKOS-ELM with DDM	**1.68**	**3.75**	**4.50**	6.30	6.93	**7.33**	**3.69**	6.02	**6.80**	**5.38**
		MSE									
	RKOS-ELM with ECDD	2.22E-02	1.09E-02	1.72E-02	4.35E-02	5.59E-02	2.70E-01	1.06E-02	4.74E-02	1.08E-01	5.32E-02
	RKOS-ELM with DDM	**1.99E-03**	**7.28E-03**	**1.66E-02**	**1.94E-02**	**3.08E-02**	**4.65E-02**	**8.20E-03**	**2.13E-02**	**4.34E-02**	**2.36E-02**
TS_6		SMAPE									
	RKOS-ELM with ECDD	2.81	6.48	8.51	14.67	16.61	30.13	5.98	15.03	22.05	13.85
	RKOS-ELM with DDM	**2.63**	**5.17**	**7.91**	**8.34**	**10.44**	**13.77**	**5.25**	**8.97**	**13.39**	**9.00**

(RKOS-ELM-DDM) in two different measuring criteria increased by 0.48E-04 and 1.11 %, respectively. In the combined datasets with concept drift problem, including TS_4, TS_5 and TS_6, DDM has superior ability to solve concept drift problem in the multi-steps time series prediction rather than ECDD, based on the results of Table 5.1 and 5.9. Only the performance of DDM in the early prediction horizon in TS_4 is worse than that of ECDD, while it still decreases the error percentage of prediction. Therefore, DDM is not only a concept drift detector that can deal with concept drift problem in time series prediction, but it also a kernel filter that assists at filtering training data and reducing the computation of learning.

The process of searching suitable threshold of ALD takes lots of time during learning of RKOS-ELM with DDM. We apply modified meta-cognitive method to find out the threshold of ALD automatically in the process of learning. The efficiency of computation in the learning of Meta-RKOS-ELM with DDM is improved tremendously comparing with that of RKOS-ELM with DDM. Table 5.3 shows the different learning time between Meta-RKOS-ELM with DDM and RKOS-ELM with DDM in six artificial datasets. It shows the learning speed of Meta-RKOS-ELM with DDM in each synthetic datasets is more than a hundred times faster than that of RKOS-ELM with DDM. At the same time, the average performance over $1 - 18$ prediction horizon of Meta-RKOS-ELM with DDM in the most of datasets has been improved or at least are equal to the performance of RKOS-ELM with DDM.

Table 5.3: The learning time (second) of RKOS-ELM with DDM and Meta-RKOS-ELM with DDM

Datasets	RKOS-ELM with DDM	Meta-RKOS-ELM with DDM
TS_1	2868.45	20.16
TS_2	7189.70	45.92
TS_3	3833.59	21.73
TS_4	3906.06	21.59
TS_5	3017.27	20.14
TS_6	3487.31	22.14

Table 5.4 shows the performance of Meta-RKOS-ELM with DDM in six synthetic data by three measuring criteria. Comparing with results of RKOS-ELM with DDM, Meta-RKOS-ELM with DDM improves the performance of different periods of prediction horizon in TS_1, TS_4 and TS_5. In TS_2, the average value of prediction horizon $(1 - 7)$

and (1 − 18) in Meta-RKOS-ELM keeps a constant with the values in RKOS-ELM with DDM, and the rest of periods are improved slightly. In addition, in TS_3 and TS_6, the performance of Meta-RKOS-ELM with DDM in 1 − 18 prediction horizons is slightly worse than that of RKOS-ELM with DDM, decreasing by 1.08E-02 and 0.61E-02 in MSE, respectively. Therefore, meta-cognitive is not only a method that finds the threshold of ALD automatically in the model, but it also reduces the computation of learning.

Table 5.4: The performance comparison on synthetic data for multi-step-ahead prediction by Meta-RKOS-ELM with DDM.

Datasets	Measurements	Predicting horizon (p)						Average predicting horizon			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
TS_1	MSE	2.38E-05	1.39E-04	3.31E-04	5.99E-04	6.99E-04	9.48E-04	1.53E-04	5.01E-04	8.03E-04	4.66E-04
	SMAPE	1.69	3.67	5.45	7.23	7.73	9.11	3.46	6.73	8.35	6.01
TS_2	MSE	5.81E-05	1.43E-04	2.17E-04	3.63E-04	4.25E-04	5.36E-04	1.24E-04	3.04E-04	4.64E-04	2.88E-04
	SMAPE	0.94	1.48	1.82	2.35	b2.54	2.88	1.33	2.14	2.66	2.00
TS_3	MSE	7.66E-03	2.72E-02	4.68E-02	4.91E-02	3.78E-02	5.12E-02	2.57E-02	5.16E-02	5.32E-02	4.21E-02
	SMAPE	4.52	10.65	16.42	17.86	15.73	19.17	10.10	18.27	19.46	15.49
TS_4	MSE	2.15E-05	4.10E-05	5.44E-05	8.83E-05	1.07E-04	1.39E-04	3.37E-05	7.52E-05	1.17E-04	7.29E-05
	SMAPE	2.29	3.12	3.49	4.49	5.07	5.80	2.75	4.16	5.23	3.97
TS_5	MSE	2.76E-05	2.04E-04	3.81E-04	4.89E-04	6.09E-04	9.33E-03	1.99E-04	5.21E-04	8.07E-04	4.91E-04
	SMAPE	1.57	3.72	5.02	5.83	6.39	7.53	3.45	5.63	6.99	5.24
TS_6	MSE	2.07E-03	8.05E-03	1.20E-02	1.82E-02	4.53E-02	5.71E-01	9.71E-03	2.51E-02	5.69E-02	2.97E-02
	SMAPE	2.70	5.48	6.79	8.17	14.18	16.84	5.79	9.92	16.55	10.52

Due to the impaction of prediction horizon in time series prediction that represents forecasting ability of model, the situation of artificial data sets between actual values and prediction values in the $18 - th$ step are shown in Figure 5.3.

Based on the results in Figure 5.3, Tables 5.1 and 5.4, we can conclude that RKOS-ELM with DDM that obtains better results and detects the concept drift problem than with ECDD, and DDM is a filter like ALD to check whether the new data in the learning needs to be updated or not. Furthermore, meta-cognitive plays a vital role in increasing learning efficiency. It automatically finds out the suitable threshold of ALD in the learning when the new data coming. Thus, based on the experiments of synthetic data, Meta-RKOS-ELM

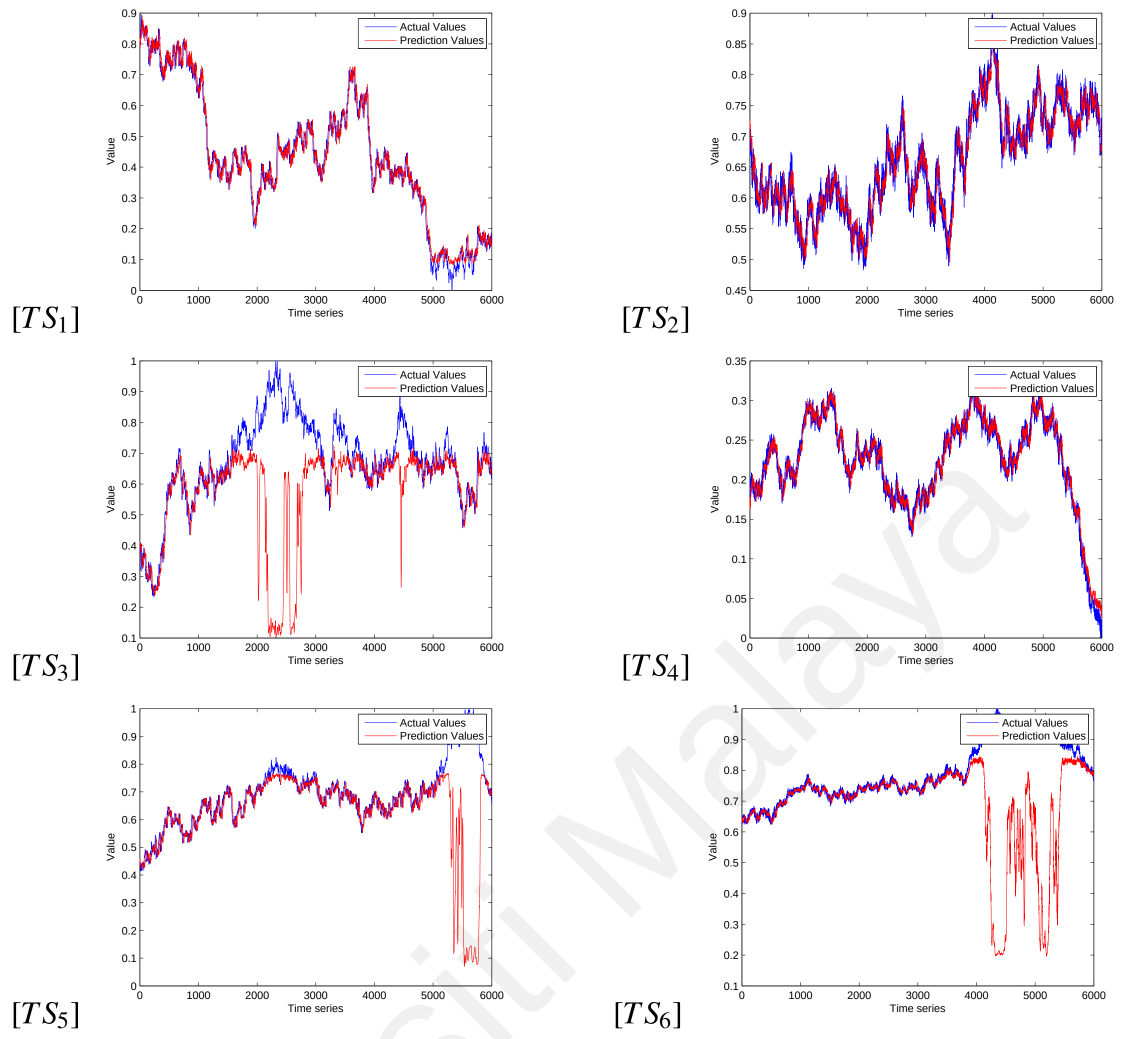


Figure 5.3: The comparison between actual values and prediction values in synthetic time series data sets in the 18-th step of Meta-RKOS-ELM-DDM.

with DDM obtains the best performance in multi-step-ahead time series prediction.

5.1.2 Real-world Experiment

Three different time series datasets are used in the real-world experiments, which include Standard&Poor‘s 500 ($S\&P500$), Shanghai Stock Exchange Composite Index (Shanghai) and Ozone Concentration of Toronto (Ozone). Experiments are first conducted using the S&P500 data. The index components and their weightings are determined by S&P Dow Jones Indices and classified as a leading indicator of business cycles (Levanon, Manini, Ozyildirim, Schaitkin, & Tanchua, 2015). We selected 10 years daily financial time series data in S&P 500 from 12/20/2006 to 12/19/2016. Then, Shanghai dataset from 1 January 1991 to 6 January 2017 is used to evaluate the performance of the models. This dataset is obtained from Yahoo Finance (Yahoo, 2017). The third case study is performed using daily ozone concentration in Toronto from 1/1/2003 to 12/31/2010 (of the environment in Ontario, 2016). It is taken from the database in the website of The Ministry of the Environment and Climate Change in Ontario. The missing value problem also appears in our real-world data, we use the mean value of missing value's nearest values to replace missing value in all experiments. Using data transformation, we transform the format of real-world time series data to a matrix, including (1270×36) for $S\&P$ 500, (6558×36) for Shanghai and (2887×36) for Ozone.

The same method as synthetic experiments is applied to evaluate the real-world datasets. This experiment compares recurrent kernel online sequential extreme learning machine with different kernel adaptive filters in order to test the effect of kernel adaptive filters and then finding out the impaction of time series prediction in the different concept drift detectors after adding DDM and ECDD based on the result of the previous comparison. Lastly, we employ meta-cognitive component to address computation efficiency problem. The parameters setting is as same as in the synthetic experiments.

Table 5.5 compares the performance of ROS-ELM in 18-step time series prediction with that of FB-RKOS-ELM and ALD-RKOS-ELM. It explains the effect of kernel method and Kernel adaptive filters (FB and ALD) in recurrent kernel online sequential extreme learning machine. From the average value of prediction horizon $1 - 18$, the performance of recurrent kernel online sequential extreme learning machine with the different filters in all real-world data are superior to ROS-ELM, which proves kernel method and ALD kernel adaptive filter play a significant role in the learning process of time series prediction. The ALD filter has much better predicting performance in recurrent kernel online sequential extreme learning machine than that of FB filter. When comparing FB-RKOS-ELM with

Table 5.5: The performance comparison on real-world data for multi-steps-ahead prediction between ROS-ELM and RKOS-ELM with different kernel filters. The best performance is in boldface.

Datasets	Algorithms	Predicting horizon (p)						Average predicting horizon			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
		MSE									
	ROS-ELM	**5.58E-06**	**2.28E-05**	**5.85E-05**	**1.05E-04**	2.11E-04	5.44E-04	**2.35E-05**	**8.30E-05**	4.23E-04	1.73E-04
	FB-RKOS-ELM	1.77E-04	1.96E-04	2.30E-04	2.85E-04	3.21E-04	4.01E-04	1.99E-04	2.63E-04	3.50E-04	2.67E-04
	ALD-RKOS-ELM	1.68E-05	4.98E-05	8.19E-05	1.14E-04	**1.25E-04**	**1.65E-04**	5.07E-05	9.58E-05	**1.53E-04**	**9.74E-05**
S&P500		SMAPE									
	ROS-ELM	**0.84**	**1.78**	**2.83**	**3.86**	5.31	8.33	**1.63**	**3.39**	6.96	3.90
	FB-RKOS-ELM	5.08	5.36	5.89	6.64	7.10	8.02	5.41	6.35	7.43	6.34
	ALD-RKOS-ELM	1.39	2.47	3.24	3.94	**4.17**	**4.84**	2.44	3.56	**4.62**	**3.48**
		MSE									
	ROS-ELM	2.48E-04	8.94E-04	7.46E-04	**1.00E-03**	**1.38E-03**	**2.44E-03**	8.00E-04	1.37E-03	**1.73E-03**	**1.27E-03**
	FB-RKOS-ELM	2.29E-04	3.95E-04	8.62E-04	1.73E-03	2.10E-03	2.93E-03	4.51E-04	1.42E-03	2.38E-03	1.36E-03
	ALD-RKOS-ELM	**1.10E-04**	**3.33E-04**	**6.25E-04**	1.43E-03	5.07E-03	7.13E-03	**3.46E-04**	**1.13E-03**	5.01E-03	2.12E-03
Shanghai		SMAPE									
	ROS-ELM	2.20	4.46	3.91	**4.57**	**5.80**	**8.02**	4.11	5.62	**6.39**	5.29
	FB-RKOS-ELM	2.36	3.10	4.71	6.83	7.48	9.15	3.23	6.09	8.15	5.66
	ALD-RKOS-ELM	**1.37**	**2.44**	**3.46**	4.98	5.91	8.05	**2.39**	**4.43**	6.74	**4.41**
		MSE									
	ROS-ELM	2.82E-01	**5.95E-03**	**7.66E-03**	**6.96E-03**	6.02E-02	**8.07E-03**	2.27E-01	9.04E-03	1.74E-02	9.65E-02
	FB-RKOS-ELM	8.16E-03	9.56E-03	1.01E-02	1.14E-02	**1.22E-02**	1.33E-02	9.20E-03	**1.08E-02**	**1.27E-02**	1.18E-02
	ALD-RKOS-ELM	**6.62E-03**	8.72E-03	1.02E-02	1.34E-02	1.46E-02	1.50E-02	**8.81E-03**	1.26E-02	1.48E-02	**1.19E-02**
Ozone		SMAPE									
	ROS-ELM	5.01	**4.41**	**5.14**	**5.69**	19.79	**5.27**	5.76	5.91	7.96	6.54
	FB-RKOS-ELM	6.45	5.93	5.80	6.17	**6.37**	6.76	5.95	**5.98**	**6.52**	6.15
	ALD-RKOS-ELM	**4.26**	5.24	5.65	6.63	6.93	7.07	**5.21**	6.38	6.98	**6.12**

ALD-RKOS-ELM, there are varying degrees of growth in the average prediction horizon over 1 – 18 of three real-world data, including the increase of S&P 500 by 2.86 % in SMAPE, 1.25 % for Shanghai and 0.03 % for Ozone. However, in the other average prediction periods, the performance of ALD-RKOS-ELM is not always the best among three models. At the same time, ALD-RKOS-ELM also does not have the best predicting performance in the prediction horizon 4, 7, 12 and 14. Therefore, in order to improve the performance in the mid-late prediction horizon period, we employ concept drift detectors, including DDM and ECDD, to enhance the predicting performance in ALD-RKOS-ELM that is the best time series prediction model in Table 5.5.

Based on the method of searching threshold of ALD, the values of ALD that lead to the best training performance are applied in RKOS-ELM with DDM and ECDD respectively, including 3.1E-04 and 2.7E-04 for S&P 500, 1.1E-03 and 9.5E-04 for Shanghai, 9.7E-04 and 4.8E-04 for Ozone. Table 5.10 shows the predicting performance of different concept drift detectors that are applied into RKOS-ELM, including DDM and ECDD. As compared with the two measuring criteria results in Table 5.5, DDM assists at improving the predicting performance in all average prediction horizon periods for S&P 500 and Shanghai dataset. In Ozone, although there is a small difference (0.04E-03 in MSE) between RKOS-ELM with DDM and RKOS-ELM in the average of prediction horizon $1-7$, the value of the rest of prediction horizon periods are improved in different degrees by RKOS-ELM with DDM. For the mid-late prediction horizon periods, including $8-12$ and $13-18$, the performance of RKOS-ELM with DDM in the prediction is better than that of RKOS-ELM with ECDD. It means that DDM has the superior ability to deal with concept drift problem and improve the performance of the mid-late prediction horizon periods rather than ECDD.

meta-cognitive learning strategy also is component into RKOS-ELM with DDM in order to reduce the computation of learning and search the suitable value of ALD threshold. Table 5.8 shows the difference of learning time between RKOS-ELM with DDM and Meta-RKOS-ELM with DDM. Meta-cognitive component sets the threshold of ALD automatically when the new data coming in the learning process, which saves lots of time in computation. For instance, Meta-RKOS-ELM with DDM takes 21.73s for Ozone, and the same model without meta-cognitive component spends 1935.98s. The rest of datasets also appear the same situation as Ozone. On the other hand, the performance of Meta-RKOS-ELM with DDM is shown in Table 5.7. It shows that Meta-RKOS-ELM with DDM has better predicting performance in Shanghai and Ozone dataset for all

Table 5.6: The performance comparison on real-world data for multi-steps-ahead prediction by RKOS-ELM with the different concept detectors. The best performance is in boldface.

Datasets	Algorithms	Predicting horizon (p)						Average predicting horizon			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
	RKOS-ELM with ECDD	2.31E-05	**5.27E-05**	**1.08E-04**	**1.19E-04**	**1.17E-04**	1.72E-04	6.09E-05	1.17E-04	1.45E-04	1.04E-04
	RKOS-ELM with DDM	**1.65E-05**	5.89E-05	1.08E-04	1.19E-04	1.37E-04	**1.69E-04**	**4.90E-05**	**1.13E-04**	**1.42E-04**	**9.77E-05**
*S&P*500		SMAPE									
	RKOS-ELM with ECDD	1.65	**2.54**	**3.66**	**4.02**	**4.06**	**4.95**	2.65	3.94	4.50	3.62
	RKOS-ELM with DDM	**1.41**	2.72	3.70	4.09	4.34	**4.88**	**2.33**	**3.87**	**4.44**	**3.46**
		MSE									
	RKOS-ELM with ECDD	9.10E-05	2.80E-04	6.26E-04	1.30E-03	1.58E-03	2.49E-03	3.01E-04	1.02E-03	1.92E-03	1.04E-03
	RKOS-ELM with DDM	**7.85E-05**	**2.47E-04**	**5.61E-04**	**1.07E-03**	**1.44E-03**	**2.14E-03**	**2.81E-04**	**8.89E-04**	**1.69E-03**	**9.20E-04**
Shanghai		SMAPE									
	RKOS-ELM with ECDD	1.34	2.35	3.44	5.06	5.52	7.05	2.28	4.40	6.11	4.14
	RKOS-ELM with DDM	**1.27**	**2.25**	**3.33**	**4.59**	**5.41**	**6.71**	**2.25**	**4.10**	**5.89**	**3.98**
		MSE									
	RKOS-ELM with ECDD	7.60E-03	**8.02E-03**	1.22E-02	**1.18E-02**	1.62E-02	1.73E-02	9.75E-02	1.27E-02	1.78E-02	1.32E-02
	RKOS-ELM with DDM	**5.53E-03**	9.61E-02	**1.18E-02**	1.40E-02	**1.36E-02**	**1.46E-02**	**9.16E-03**	**1.27E-02**	**1.42E-02**	**1.18E-02**
Ozone		SMAPE									
	RKOS-ELM with ECDD	4.47	**5.23**	6.12	**6.22**	7.43	7.76	5.41	6.42	7.78	6.48
	RKOS-ELM with DDM	**4.27**	5.49	**6.12**	6.75	**6.63**	**6.97**	**5.31**	**6.39**	**6.82**	**6.11**

prediction horizon periods than RKOS-ELM with DDM. Although the results of S&P 500 in Meta-RKOS-ELM with DDM is not better than that of RKOS-ELM with DDM, there are no huge differences in all prediction horizon periods between this two models. For example, the difference of SMAPE is 0.30E-04 in prediction horizon 1-18, 0.32E-04 for prediction horizon $13 - 18$, 0.31E-04 for prediction horizon $8 - 12$ and 2.77E-05 for prediction horizon $1 - 7$. Therefore, meta-cognitive component has superior ability to set suitable threshold of ALD in the learning process and increase the computation efficiency of RKOS-ELM with DDM.

Furthermore, as same as artificial data sets, real-world data sets also showed the situation of actual values and prediction values for the $18 - th$ step in Figure 5.4. It showed the prediction values from Meta-ALD-RKOS-ELM with DDM are matchable with the actual values for the testing part of real-world data sets.

Table 5.7: The performance comparison on real-world data for multi-step-ahead prediction by Meta-RKOS-ELM with DDM.

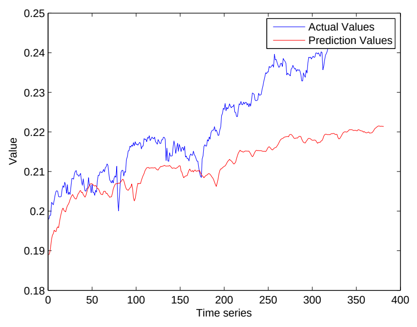
Datasets	Measurements	Predicting horizon (p)						Average predicting horizon			
		1	4	7	12	14	18	1-7	8-12	13-18	1-18
S&P500	MSE	7.58E-05	8.31E-05	8.62E-05	1.40E-04	1.44E-04	2.17E-04	7.67E-05	1.44E-04	1.74E-04	1.28E-04
	SMAPE	3.06	3.25	3.34	4.32	4.46	5.57	3.10	4.33	4.92	4.05
Shanghai	MSE	8.47E-05	2.49E-04	5.90E-04	1.27E-03	1.47E-03	2.19E-03	2.85E-04	9.91E-04	1.76E-03	9.72E-04
	SMAPE	1.29	2.25	3.36	4.73	5.40	6.42	2.27	4.13	5.86	3.98
Ozone	MSE	5.85E-03	8.36E-03	1.05E-02	1.45E-02	1.34E-02	1.45E-02	8.40E-03	1.25E-02	1.41E-02	1.14E-02
	SMAPE	4.15	5.19	5.74	6.92	6.61	6.92	5.10	6.38	6.80	6.02

Table 5.8: The learning time (second) of RKOS-ELM with DDM and Meta-RKOS-ELM with DDM in real-world data.

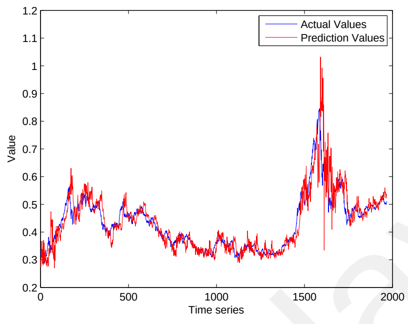
Datasets	RKOS-ELM with DDM	Meta-RKOS-ELM with DDM
S&P500	105.66	2.22
Shanghai	1417.05	12.70
Ozone	1935.98	21.73

5.2 Experiment 2

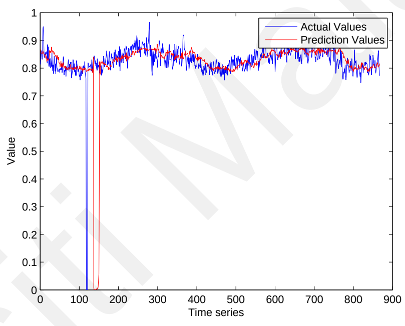
In this section, based on the result of experiment 1, the study focuses on the role of recursive kernel in Meta-RKOS-ELM. The synthetic and real-world data sets are used as same as Experiment 1. Firstly, comparing the performance between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM in synthetic data sets, this comparative results reveal the role of recursive kernel plays in the prediction model. Secondly, the three real-world is applied in the experiments, including Standard&Poor‘s 500 (S&P500), Shanghai SSE Composite Index (Shanghai) and ozone concentration of Toronto (Ozone). They also take the same measurement to evaluate the performance models. Therefore, the synthetic and real-world data sets can show the different pattern in time series and then evaluate the predicting performance with and without recursive kernel method in the prediction model.



$[S\&P500]$



$[Shanghai]$



$[Ozone]$

Figure 5.4: The comparison between actual values and prediction values in real-world time series data sets in the 18-th step of Meta-RKOS-ELM-DDM.

5.2.1 Synthetic Data Experiment and Results

In the synthetic data sets, the six time series data sets are generated as same as Experiment 1. the synthetic time series data sets need to transform their format in order to adapt the assumption of the experiment. After data transformation, the six synthetic time series data sets become six matrices with 20000 rows and 36 columns. We employ 14000 rows of the matrix in each artificial data to train models and the remaining part as testing part.

In order to further increase the performance of prediction and generate a fixed reservoir that is the finite dimensional equivalent of the recurrent Gaussian kernel, we apply recursive

Table 5.9: The performance comparison on synthetic data for multi-steps-ahead prediction by Meta-RRKOS-ELM-DDM

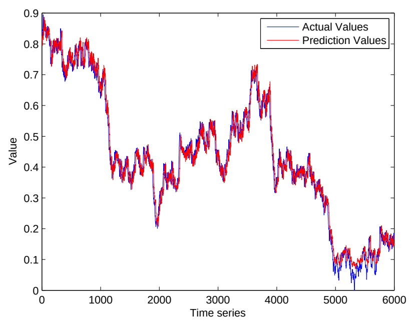
Data sets	Measurements	Predicting horizon (p)				Average predicting horizon			
		1	7	14	18	1-7	8-12	13-18	1-18
TS_1	MSE	2.32E-05	3.41E-04	7.08E-04	9.64E-04	1.55E-04	4.95E-04	8.00E-04	4.65E-04
	SMAPE	1.66	5.74	7.70	9.01	3.61	6.57	8.25	5.98
TS_2	MSE	5.95E-05	2.20E-04	4.27E-04	5.37E-04	1.27E-04	3.08E-04	4.68E-04	2.91E-04
	SMAPE	0.95	1.83	b2.54	2.88	1.34	2.16	2.67	2.01
TS_3	MSE	1.16E-02	3.99E-02	4.85E-02	3.80E-02	2.35E-02	4.55E-02	5.08E-02	3.87E-02
	SMAPE	6.07	14.39	18.31	16.10	9.60	16.76	18.89	14.68
TS_4	MSE	1.89E-05	5.36E-05	1.04E-04	1.38E-04	3.23E-05	7.43E-05	1.14E-04	7.12E-05
	SMAPE	2.05	3.47	4.78	5.69	2.65	4.09	5.07	3.86
TS_5	MSE	4.07E-05	1.92E-04	6.36E-04	7.82E-04	1.58E-04	3.77E-04	7.62E-04	4.20E-04
	SMAPE	1.87	3.27	6.45	7.21	2.80	5.12	6.81	4.78
TS_6	MSE	3.52E-03	1.08E-02	3.53E-02	3.18E-01	8.83E-03	1.83E-02	3.06E-02	1.87E-02
	SMAPE	3.52	6.38	12.18	11.38	5.59	8.47	10.92	8.17

kernel to replace the kernel method of Meta-RKOS-ELM-DDM. However, recursive kernel parameter is an influential factor in the generation of the reservoir. Furthermore, recursive kernel parameter generally is not too large number. Therefore, in order to obtain the best prediction model by training, the most suitable recursive kernel parameter is searched at the range between 1.0 and 30.0 with 0.1 intervals in the learning part of RRKOS-ELM-DDM for the different datasets. Finally, we obtain six different recursive kernel parameters for TS_{1-6}, including 9.0, 18.4, 15.0, 14.0, 10.0, and 17.0. Table 5.9 shows the performance of Meta-RRKOS-ELM-DDM in the six synthetic data by these recursive kernel parameters. Except for the data TS_2, the predicting ability of Meta-RRKOS-ELM-DDM has been improved in the result of synthetic data sets over different periods of prediction horizons. Although the forecasting performance of Meta-RRKOS-ELM-DDM in data TS_2 is slight worse than that of Meta-RKOS-ELM-DDM, the difference measurement values between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM in the 1-18 period of prediction horizons are only 0.03E-04 for MSE and 0.01% for SMAPE, respectively. In the other periods of prediction horizons, the results of Meta-RRKOS-ELM-DDM in the six synthetic data sets also prove the vital role of the recursive kernel in RRKOS-ELM-DDM. Furthermore, compared the results of the normal time series data TS_{1-3} with that of combined data TS_{4-6} in the different models (Meta-RRKOS-ELM-DDM, Meta-RKOS-ELM-DDM), the extent of growth of the combined data in the former model is much better than that of normal data, which means the reservoir that is generated by the recursive kernel is benefited to renew and save information. For example, under SMAPE criteria, Meta-RRKOS-ELM-DDM enhances the predicting performance in TS_4 by 0.11%, TS_5 by0.64%, and TS_6 by 2.35%. The corresponding values of normal datasets only are 0.03%, -0.01%, and 0.81%. Therefore, recursive RBF kernel method generates a fixed reservoir with optimized information, which plays a vital role in the time series prediction and improves the predicting performance in concept drift data.

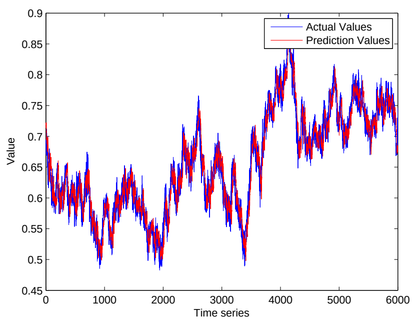
Furthermore, due to the impaction of prediction horizon in time series prediction that represents forecasting ability of model, the situation of artificial data sets between actual values and prediction values in the $18-th$ step are shown in Figure 5.5. There are matchable trend between actual values and prediction values in the $18-th$ step for the model Meta-RRKOS-ELM-DDM.

5.2.2 Real-world Experiment and Results

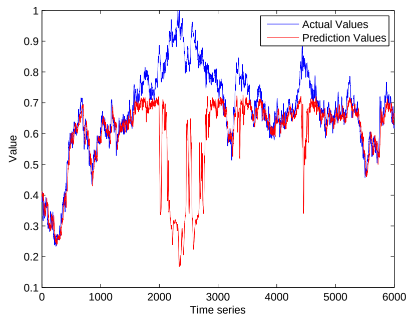
In this section, the three real-world is applied in this experiments, including S&P500, Shanghai and Ozone. As same as experiment 1, the missing value problem also appears in our real-world data, the mean value of missing value‘s nearest values are used to replace missing value in all experiments. By data transformation, the format of real-world time series data sets are (1270×36) for $S\&P$ 500, (6558×36) for Shanghai and (2887×36) for ozone.



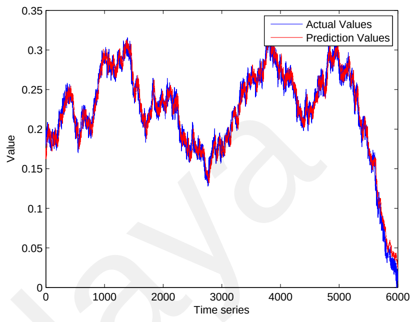
$[TS_1]$



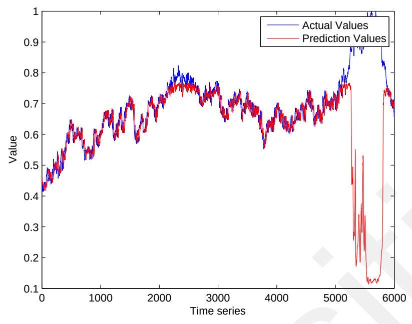
$[TS_2]$



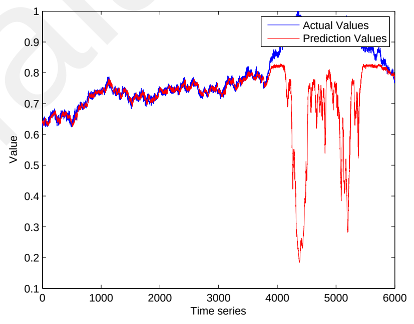
$[TS_3]$



$[TS_4]$



$[TS_5]$



$[TS_6]$

Figure 5.5: The comparison between actual values and prediction values in synthetic time series data sets in the 18-th step of Meta-RRKOS-ELM-DDM.

In order to check the role of recursive kernel method play in the learning process of Meta-RKOS-ELM-DDM, the results between Meta-RKOS-ELM-DDM in Table 5.7and Meta-RRKOS-ELM-DDM in Table 5.10 is compared. Due to kernel parameter (δ') of recursive kernel method that directly impacts on the result of prediction result, the study seeks the δ' parameter at the range between 1.0 and 30.0 with 0.1 intervals in the learning part of RRKOS-ELM-DDM for the different datasets, including 3.0 in $S\&P500$, 20.5 in $Shanghai$, and 7.1 in $Ozone$, respectively. Table 5.10 shows the predicting results of Meta-RRKOS-ELM-DDM in the different prediction horizons and periods of prediction horizons for the real-world datasets with the obtained recursive kernel parameters. It

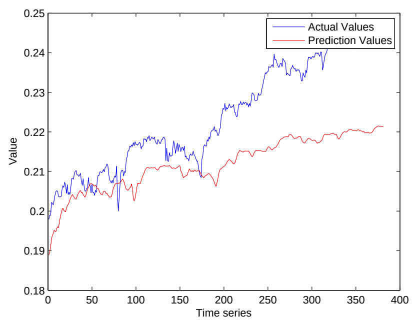
Table 5.10: The performance on real-world data for multi-steps-ahead prediction by Meta-RRKOS-ELM-DDM

Data sets	Measurements	Predicting horizon (p)				Average predicting horizon			
		1	12	14	18	1-7	8-12	13-18	1-18
*S&P*500	MSE	3.55E-05	1.58E-04	1.59E-04	2.13E-04	7.79E-05	1.14E-04	1.79E-04	1.22E-04
	SMAPE	2.06	4.60	4.69	5.51	3.04	3.90	5.00	3.93
Shanghai	MSE	2.64E-05	1.68E-04	3.79E-04	5.39E-04	8.57E-05	2.60E-04	4.38E-04	2.52E-04
	SMAPE	1.17	3.02	4.59	5.45	2.07	3.69	4.87	3.45
Ozone	MSE	5.70E-03	8.36E-03	1.13E-02	1.22E-02	7.27E-03	9.50E-03	1.19E-02	9.42E-03
	SMAPE	4.15	5.18	6.02	6.30	4.87	6.48	6.18	5.48

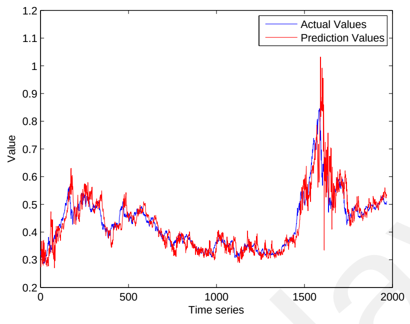
indicates that recursive kernel method improve the predicting performance in almost all periods of prediction horizons for all real-world datasets. Some average values of $8-12$ periods of prediction horizons, including the values in $S\&P500$ and $Shanghai$, are not increased under SMAPE. However, there is only slight difference in this periods of prediction horizons between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM. Therefore, recursive kernel method is a good way to improve the performance in multi-steps time series prediction. The situations between actual values and prediction values in the $18-th$ step for real-world data sets are shown in Figure 5.6. There are matchable trend between actual values and prediction values in the $18-th$ step for the model Meta-RRKOS-ELM-DDM.

5.3 Statistical Comparison

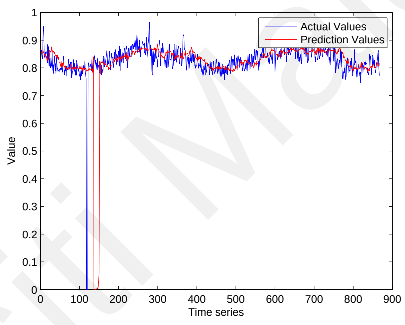
In this section, we used the Wilcoxon Signed-rank Test, a popular nonparametric test for matched or paired data, to prove that meta-RRKOS-ELM-DDM is superior in time series prediction than meta-RKOS-ELM-DDM. This test is mainly based on difference scores, in addition to analyzing the signs of the differences. It also takes into account the magnitude of the observed differences. Firstly, the difference d_i between the accuracy in SMAPE of the two algorithms on seven data sets is computed. Secondly, these differences are ranked



[$S\&P500$]



[$Shanghai$]



[$Ozone$]

Figure 5.6: The comparison between actual values and prediction values in real-world time series data sets in the 18-th step of Meta-RRKOS-ELM-DDM.

according to their absolute values. The table 5.11 shows the different values of (1-SMAPE) in the data sets and their rank corresponding to two models. Then R^{+} and R^{-} are computed. R^{+} is the sum of ranks for the positive and R^{-} is the sum of ranks for the negative. Finally, the values of R^{+} and R^{-} are 35 and 4, respectively. According to the table of exact critical values for Wilcoxon's test, for a confidence level of $\alpha = 0.10$, the difference between the algorithms is significant if the smaller of the sums is less than or equal to 8. At the same time, based on critical values for Wilcoxon's test with confidence level of $\alpha = 0.05$, the difference between the algorithms are considered significant if the smaller of the sums is less than or equal to 6. Therefore, we can conclude that meta-RRKOS-ELM-DDM is statistically superior to meta-RKOS-ELM-DDM.

Table 5.11: Comparison of meta-RKOS-ELM-DDM and meta-RRKOS-ELM-DDM.

Data set	meta-RRKOS-ELM-DDM	meta-RKOS-ELM-DDM	Difference	Rank
TS_1	92.46	92.32	0.14	4
TS_2	97.98	98.00	-0.02	1
TS_3	89.02	88.65	0.37	6
TS_4	92.76	92.89	-0.13	3
TS_5	91.24	89.79	1.45	9
TS_6	93.19	91.84	1.35	8
$S\&P500$	96.18	96.07	0.11	2
$Shanghai$	96.52	96.00	0.52	7
$Ozone$	96.18	95.95	0.23	5

5.4 Discussion

In experiment 1, ALD-RKOS-ELM has much better performance in different prediction horizons than that of FB-KOS-ELM. This shows that ALD can filter the incoming data successfully in the learning process of RKOS-ELM for different types of time series data and obtain the best prediction performance than FB filter. Furthermore, based on the performance of DDM and ECDD in ALD-RKOS-ELM, we can conclude that DDM has the superior ability to detect concept drift problem in the learning process of ALD-RKOS-ELM compared with ECDD. In addition, meta-cognitive play a vital role in reducing learning time and solving parameter dependency. From the meta-cognitive learning strategy, RKOS-ELM with DDM can decide when the incoming data in the learning process need to add neuron, to retrain sample, or to discard sample. At the same time, the threshold of ALD can be defined by meta-cognitive automatically, which not only saves learning time, but also solves the dependency of threshold of ALD. Therefore, Meta-RKOS-ELM-DDM has outstanding performance in time series prediction rather than other compared models. The major benefits of Meta-RKOS-ELM-DDM are that the good generalization model for solving concept drift problem can be achieved, the limitation of prediction horizon are released, and DDM can be applied in time series prediction that solves concept drift problem in time series prediction and improves the predicting performance in multi-steps prediction. Meta-cognitive and ALD filter plays a significant role in the learning time reduction and dealing with parameter dependency.

In the experiment 2, based on the conclusion of experiment 1, it try to find how recursive kernel method impacts on the predictive ability of Meta-RKOS-ELM-DDM. There are two different results between Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM in synthetic and real-world data sets. The comparable result shows that Meta-RRKOS-ELM-DDM has outstanding performance in the majority of average prediction periods and prediction horizons rather than Meta-RKOS-ELM-DDM. It can conclude that recursive kernel method can successfully replace kernel method in Meta-RKOS-ELM-DDM and enhances the predicting performance of different periods of prediction horizons. The results of Meta-RRKOS-ELM-DDM in the synthetic and real-world datasets have proved this phenomenon. Therefore, recursive kernel method generates a fixed reservoir with optimized information by dissipating and overwriting the information from the coming data in the learning part of the prediction model, which is helpful for improvement of forecasting performance.

Finally, the statistical comparison also proved that Meta-RRKOS-ELM-DDM has superior prediction ability rather than Meta-RKOS-ELM-DDM.

5.5 Summary

This chapter discussed and explained the implementation and evaluation of two models, including Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM. There are six synthetic and three real-world data employed to evaluate the forecasting performance by two different measurements. The outcomes from these experiments showed that Meta-RRKOS-ELM-DDM was a general time series prediction model and had the outstanding forecasting performance for different kinds of data sets. It not only successfully solves parameter dependency and concept drift problem, but it also increases the working efficiency and performance in multi-step ahead prediction horizon.

CHAPTER 6: CONCLUSIONS, CONTRIBUTIONS AND FUTURE WORKS

In this chapter, the comprehensive summary of this thesis is presented, and the contributions are discussed. Furthermore, the future works are discussed from the viewpoint of limitations of the models in this thesis.

6.1 Conclusions

In this thesis, the extreme learning machine, which is a recurrent neural network, has been improved from the perspective of offline and online learning. For the improvement of offline learning algorithm, the fixed reservoir helps the proposed model improving connection between neurons and generating a fixed memory in the process of learning part of models by a modified echo state network. Furthermore, the restriction of prediction horizon is solved by multi-step ahead prediction algorithm. Finally, the most suitable parameters are optimized by QPSO. The experimental results showed that the offline learning proposed models have outstanding performance in different prediction horizons compared to the conventional models. In the case of online learning improvement, except for solving the restriction of prediction horizon by the same method as online learning model, the concept drift problem is solved by DDM, and parameter dependency and complex computation of learning part is solved by a new meta-cognitive learning strategy. Furthermore, based on the experimental results, recursive kernel can generate a fixed reservoir that optimizes information by dissipating and overwriting the information from the coming data in the learning part of the prediction model. The following descriptions summarize the knowledge obtained in each chapter:

In Chapter 1, the background of time series prediction, and the typical algorithms in the different periods were reviewed. The several studies on offline and online learning models in time series prediction were introduced. The concept drift problem was taken into consideration as well. In addition, the popular topic reservoir computing and its extended algorithms, which were used to generate a dynamic reservoir in the training part, was reviewed. Moreover, based on discussions in the chapter, multi-steps prediction plays a vital role in applying in the real-world project of the time series prediction. Finally, the research problems, objectives and contributions in this thesis were defined.

In Chapter 2, the fundamentals of offline and online learning models were introduced, and the two kernel adaptive filters are presented. Furthermore, the conventional studies for reservoir computing were reviewed, especially for the most common reservoir computing model-ESN. Besides, the concept drift which affected the performance of time series prediction was introduced. Finally, the research gaps on time series prediction were presented.

In Chapter 3, it discussed the method of data transformation, and proposed one offline learning and two online learning time series prediction models. Firstly, a simple time series data could be transformed into a matrix for further prediction models. Secondly, it introduced an offline learning time series prediction model named RKERM with QPSO. It overcame the restriction of prediction horizon and generated a fixed reservoir in RKELM. Furthermore, parameter dependency problem was also solved by QPSO. Thirdly, Meta-RKOS-ELM-DDM and Meta-RRKOS-ELM-DDM were introduced in the part of online learning time series prediction model. They aimed at generating a fixed dynamic reservoir by recursive kernel method, dealing with concept drift problem in time series data by concept drift detectors, overcoming the restriction of prediction horizon, reducing learning time and solving the parameter dependency by a new meta-cognitive learning strategy.

In Chapter 4 and 5, the different synthetic and real-world datasets for offline and online learning models were evaluated, respectively. The MSE and SMAPE was used to measure the forecasting performance in the different prediction horizon periods and specific prediction horizons.

From the results of Chapters 4 and 5, it is evident that the proposed models have superior forecasting performance, fixed reservoir in the training part and solved corresponding problems mentioned in literature review, when compared with the conventional models. Due to these advantages, it is possible to expect more stable and accuracy than in the conventional models if the models are applied to the time series prediction in the real-world applications.

6.2 Contributions

The research objectives of this thesis are achieved by corresponding research contributions, which are defined in Chapter 1, as the following viewpoints:

In the Chapter 4, the modified ESN was applied to in KELM, which generated a fixed reservoir in the learning part. It does not only have stable and adaptable forecasting performance, but it also improved the time series prediction accuracy compared with conventional models. At the same time, multi-step ahead prediction algorithm assisted on overcoming the restriction of prediction horizon. QPSO helped the proposed offline prediction model optimizing the parameters. Therefore, research objectives 1, 2 and 3 have been addressed in the offline learning model.

In Chapter 5, multi-step ahead prediction algorithm in online learning models played the same role of the offline learning model. The modified DDM is the most efficient concept drift detector in this proposed on-learning models. It was successfully applied in the time series prediction model in order to detect concept drift problem. Furthermore, in the training process, meta-cognitive learning strategy plays a significant role in reducing the complex computation and parameter dependency in the online learning time series prediction models. Finally, recursive kernel method successfully replaced the kernel method in online learning model. It generated a fixed reservoir with optimized information for enhancing the forecasting performance. Thus, all research objectives have been achieved for online learning models.

From the results and discussions in Chapters 4, and 5, the research objectives, namely, restriction of prediction horizon, parameter dependency, unstable reservoir, and concept drift problem, were addressed, thus the research problems in this thesis have been solved.

6.3 Limitation of Works

Although our proposed offline and online models have fulfilled the research objectives, they still have limitations. For example, our proposed offline model applied QPSO to solve parameter dependency problem, but it takes a long time for seeking suitable parameters in the process of QPSO. On the other hand, although online learning model solved ALD threshold definition, recursive kernel parameter setting is also a key influential element for prediction performance. Therefore, this study still needs to improve in aspect of training efficiency, parameter settings and so on.

6.4 Future Works

This research presented a novel offline and online model to predict non-stationary time series more effectively and accurately. Experimental results confirmed the forecasting ability of prediction for the proposed models. They performed better in multi-step ahead prediction than other conventional models. However, in order to further enhance quality of the current study, the following propositions can be investigated as an extension to this research work.

The recursive kernel parameter in the proposed online learning model is researched by the specific range with interval 0.1. It impacts on the training efficiency and real-world implementation. The unsuitable parameter impacts on the performance of model. The fisher information matrix may be a good solution to seeking good parameter.

The fixed reservoir was generated in both offline and online learning models by reservoir computing. One method is by applying the modified ESN to connect each neurons of RKELM and other method is by employing recursive kernel method for updating the information of states in the online learning model. Although reservoir computing improved the forecasting performance based on optimized information from fixed reservoir of both models, this fixed reservoir still has potential space to mine. For example, the weights of reservoir are supposed to be dynamic, because of the too early information of state is too old to have any impaction on the long-term future prediction result. In the future, reduced kernel method may be a good solution that not only can reduce heavy computation but also simplify the size of reservoir.

The multi-step ahead prediction is applied for predicting 18 time steps of time series data. The performance of the proposed offline and online learning models have the outstanding performance in the long-term prediction. However, the accuracy of prediction slightly decreases with growing of prediction horizon. In the future, the study should focus on improving the long-term forecasting ability and extending the prediction horizon by using other type of recurrent algorithm.

As a conclusion, the research objectives have been achieved by the offline and online learning models. The investigated problems in literature review have been solved by this proposed models. By considering results of imperilments, the proposed models are the currently the best alternative for multi-steps-ahead time series prediction. It is hoped that the contributions and findings from this research will lead to new explorations and contributions in the domain of extreme learning machine and its application in the time series prediction.

REFERENCES

Al Baity, H. (2015). *A quantum behaved particle swarm approach to multi-objective optimization* (Unpublished doctoral dissertation). University of Birmingham.

Aranda, A., Ferreira, G., Mainar-Toledo, M., Scarpellini, S., & Sastresa, E. L. (2012). Multiple regression models to predict the annual energy consumption in the spanish banking sector. *Energy and Buildings*, *49*, 380–387.

Ardalani-Farsa, M., & Zolfaghari, S. (2010). Chaotic time series prediction with residual analysis method using hybrid elman–narx neural networks. *Neurocomputing*, *73*(13), 2540–2553.

Armstrong, J. S., & Forecasting, L.-R. (1985). From crystal ball to computer. *New York ua*.

Atiya, A. F., & Parlos, A. G. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE transactions on neural networks*, *11*(3), 697–709.

Austin, S. (1990). An introduction to genetic algorithms. *AI expert*, *5*(3), 48–53.

Babu, G. S., & Suresh, S. (2012). Meta-cognitive neural network for classification problems in a sequential learning framework. *Neurocomputing*, *81*, 86–96.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method.

Bao, Y., Xiong, T., & Hu, Z. (2014). Pso-mismo modeling strategy for multistep-ahead time series prediction. *IEEE transactions on cybernetics*, *44*(5), 655–668.

Barbounis, T., & Theocharis, J. (2006). Locally recurrent neural networks for long-term wind speed and power prediction. *Neurocomputing*, *69*(4-6), 466–496.

Bianchi, F. M., Scardapane, S., Uncini, A., Rizzi, A., & Sadeghian, A. (2015). Prediction of telephone calls load using echo state network with exogenous variables. *Neural Networks*, *71*, 204–213.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 siam international conference on data mining* (pp. 443–448).

Billings, S., Chen, S., & Korenberg, M. (1989). Identification of mimo non-linear systems using a forward-regression orthogonal estimator. *International journal of control*, *49*(6), 2157–2189.

Bouchachia, A. (2009). Radial basis function nets for time series prediction. *International Journal of Computational Intelligence Systems*, *2*(2), 147–157.

Budiman, A., Fanany, M. I., & Basaruddin, C. (2016). Adaptive online sequential elm for concept drift tackling. *Computational intelligence and neuroscience*, *2016*.

Buonomano, D. V., & Merzenich, M. M. (1995). Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, *267*(5200), 1028–1030.

Burgess, A. N., & Refenes, A. N. (1999). Modelling non-linear moving average processes using neural networks with error feedback: An application to implied volatility forecasting. *Signal Processing*, *74*(1), 89–99.

Cavalcante, R. C., Minku, L. L., & Oliveira, A. L. (2016). Fedd: Feature extraction for explicit concept drift detection in time series. In *Neural networks (ijcnn), 2016 international joint conference on* (pp. 740–747).

Cavalcante, R. C., & Oliveira, A. L. (2015). An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection. In *Neural networks (ijcnn), 2015 international joint conference on* (pp. 1–8).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Chen, S., & Billings, S. A. (1989). Representations of non-linear systems: the narmax model. *International Journal of Control*, *49*(3), 1013–1032.

Chen, X., Dong, Z. Y., Meng, K., Xu, Y., Wong, K. P., & Ngan, H. (2012). Electricity price forecasting with extreme learning machine and bootstrapping. *IEEE Transactions* *on Power Systems*, *27*(4), 2055–2062.

Colliez, J., Dufrenois, F., & Hamad, D. (2006). Robust regression and outlier detection with svr: Application to optic flow estimation. In *Bmvc* (pp. 1229–1238).

Dash, R., Dash, P. K., & Bisoi, R. (2014). A self adaptive differential harmony search based optimized extreme learning machine for financial time series prediction. *Swarm and Evolutionary Computation*, *19*, 25–42.

Dehghan, M., Beigy, H., & ZareMoodi, P. (2016). A novel concept drift detection method in data streams using ensemble classifiers. *Intelligent Data Analysis*, *20*(6), 1329–1350.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems* (pp. 155–161).

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, *22*(10), 1517–1531.

Ertugrul, Ö. F. (2016). Forecasting electricity load by a novel recurrent extreme learning machines approach. *International Journal of Electrical Power & Energy Systems*, *78*, 429–435.

Fernández-Delgado, M., Cernadas, E., Barro, S., Ribeiro, J., & Neves, J. (2014). Direct kernel perceptron (dkp): Ultra-fast kernel elm-based classification with non-iterative closed-form weight calculation. *Neural Networks*, *50*, 60–71.

Frasconi, P., Gori, M., & Soda, G. (1992). Local feedback multilayered networks. *Neural computation*, *4*(1), 120–130.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence* (pp. 286–295).

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44.

Gholipour, A., Araabi, B. N., & Lucas, C. (2006). Predicting chaotic time series using neural and neurofuzzy models: a comparative study. *neural processing letters*, *24*(3), 217–239.

Gonçalves, P. M., de Carvalho Santos, S. G., Barros, R. S., & Vieira, D. C. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, *41*(18), 8144–8156.

Gurpinar, F., Kaya, H., Dibeklioglu, H., & Salah, A. (2016). Kernel elm and cnn based facial age estimation. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 80–86).

Hammoodi, M., Stahl, F., & Tennant, M. (2016). Towards online concept drift detection with feature selection for data stream classification.

Haykin, & Simon. (2009). *Neural networks and learning machines* (Vol. 3). Pearson Upper Saddle River, NJ, USA:.

Haykin, S. (1999). Neural networks: A comprehensive foundation, vol. 2. *Segundo, Pentrice Hall. España*.

He, Q., Shang, T., Zhuang, F., & Shi, Z. (2013). Parallel extreme learning machine for regression based on mapreduce. *Neurocomputing*, *102*, 52–58.

Hermans, M., & Schrauwen, B. (2012). Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, *24*(1), 104–133.

Hou, S., & Li, Y. (2009). Short-term fault prediction based on support vector machines with parameter optimization by evolution strategy. *Expert Systems with Applications*, *36*(10), 12383–12391.

Huang, Zhu, Q.-Y., & Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural networks, 2004. proceedings. 2004 ieee international joint conference on* (Vol. 2, pp. 985–990).

Huang, G., Huang, G.-B., Song, S., & You, K. (2015). Trends in extreme learning machines: A review. *Neural Networks*, *61*, 32–48.

Huang, G.-B., & Chen, L. (2007). Convex incremental extreme learning machine.

Neurocomputing, *70*(16), 3056–3062.

Huang, G.-B., & Chen, L. (2008). Enhanced random search based incremental extreme learning machine. *Neurocomputing*, *71*(16), 3460–3468.

Huang, G.-B., & Siew, C.-K. (2005). Extreme learning machine with randomly assigned rbf kernels. *International Journal of Information Technology*, *11*(1), 16–24.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(2), 513–529.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*(1), 489–501.

Hung, M. D., Dung, N. T., et al. (2016). Application of echo state network for the forecast of air quality. *Journal of Science and Technology*, *54*(1), 54.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, *148*(34), 13.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, *304*(5667), 78–80.

Ježowicz, T., Gajdoš, P., Uher, V., Mišák, S., & Snášel, V. (2018, January). Improving the speed and quality of extreme learning machine by conjugate gradient method. In *Proceedings of the third international afro-european conference for industrial advancement — aecia 2016.* Springer. Retrieved from `http://dx.doi.org/10.1007/978-3-319-60834-1_14` doi: 10.1007/978-3-319-60834-1_14

Juditsky, A., Zhang, Q., Delyon, B., Glorennec, P.-Y., & Benveniste, A. (1994). *Wavelets in identification wavelets, splines, neurons, fuzzies: how good for identification* (Unpublished doctoral dissertation). INRIA.

Kantz, H., & Schreiber, T. (2004). *Nonlinear time series analysis* (Vol. 7). Cambridge university press.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization* (Tech. Rep.). Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proc. conf. ieee int neural networks.* (Vol. 4, pp. 1942–1948 vol.4). doi: 10.1109/ICNN.1995.488968

Kolter, J. Z., & Maloof, M. A. (2003). Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Data mining, 2003. icdm 2003. third ieee international conference on* (pp. 123–130).

Kolter, J. Z., & Maloof, M. A. (2005). Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd international conference on machine learning* (pp. 449–456).

Kulkarni, P., & Ade, R. (2016). Logistic regression learning model for handling concept drift with unbalanced data in credit card fraud detection system. In *Proceedings of the second international conference on computer and communication technologies* (pp. 681–689).

Le, Q., Sarlós, T., & Smola, A. (2013). Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning.*

Levanon, G., Manini, J.-C., Ozyildirim, A., Schaitkin, B., & Tanchua, J. (2015). Using financial indicators to predict turning points in the business cycle: The case of the leading economic index for the united states. *International Journal of Forecasting*, *31*(2), 426–445.

Li, Q., & Chan, M. F. (2017). Predictive time-series modeling using artificial neural networks for linac beam symmetry: an empirical study. *Annals of the New York Academy of Sciences*, *1387*(1), 84–94.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, *17*(6), 1411–1423.

Liao, J.-W., & Dai, B.-R. (2014). An ensemble learning approach for concept drift. In *Information science and applications (icisa), 2014 international conference on* (pp. 1–4).

Lin, X., Yang, Z., & Song, Y. (2008). The application of echo state network in stock data mining. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 932–937).

Liu, X., Wang, L., Huang, G.-B., Zhang, J., & Yin, J. (2015). Multiple kernel extreme learning machine. *Neurocomputing*, *149*, 253–264.

Lu, W.-Z., & Wang, W.-J. (2005). Potential assessment of the "support vector machine" method in forecasting ambient air pollutant trends. *Chemosphere*, *59*(5), 693–701.

Mackey, M. C., Glass, L., et al. (1977). Oscillation and chaos in physiological control systems. *Science*, *197*(4300), 287–289.

Makridakis, S. (1993). Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, *9*(4), 527–529.

Martınez-Rego, D., Fontenla-Romero, O., & Alonso-Betanzos, A. (2008). A method for time series prediction using a combination of linear models. *Advances in Computational Intelligence and Learning*.

McDonald, N. (2017). Reservoir computing & extreme learning machines using pairs of cellular automata rules. In *Neural networks (ijcnn), 2017 international joint conference on* (pp. 2429–2436).

Miao, Y., Potts, R., Huang, X., Elliott, G., & Rivett, R. (2012). A fuzzy logic fog forecasting model for perth airport. *Pure and applied geophysics*, *169*(5-6), 1107–1119.

Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., & Lendasse, A. (2010). Op-elm: optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, *21*(1), 158–162.

Mirza, B., & Lin, Z. (2016). Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification. *Neural Networks*, *80*, 79–94.

Mittal, V., & Kashyap, I. (2015). Online methods of learning in occurrence of concept drift. *International Journal of Computer Applications*, *117*(13).

Moreno-Torres, J. G., Raeder, T., Alaiz-RodríGuez, R., Chawla, N. V., & Herrera, F.

(2012). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530.

Mostafa, M., Teich, W. G., & Lindner, J. (2011). Global vs. local stability of recurrent neural networks as vector equalizer. In *Signal processing and communication systems (icspcs), 2011 5th international conference on* (pp. 1–5).

Munro, P. (2010, January). Backpropagation. *Encyclopedia of Machine Learning*. Retrieved from http://dx.doi.org/10.1007/978-0-387-30164-8_51 doi: 10.1007/978-0-387-30164-8_51

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, *1*(1), 4–27.

Nishida, K., & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In *International conference on discovery science* (pp. 264–269).

of the environment in Ontario, M. (2016, November). Environment and climate change (Vol. 146) [Computer software manual]. Retrieved from http://www.airqualityontario.com/history/index.php doi: 10.1016/j.gloplacha.2016.09.011

Ortín, S., Soriano, M. C., Pesquera, L., Brunner, D., San-Martín, D., Fischer, I., ... Gutiérrez, J. M. (2015). A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron. *Scientific reports*, *5*, 14945.

Oza, N. C., & Russell, S. (2001). Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings of the seventh acm sigkdd international conference on knowledge discovery and data mining* (pp. 359–364).

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, *41*(1/2), 100–115.

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.

Pan, F., Zhang, H., & Xia, M. (2009). A hybrid time-series forecasting model using extreme learning machines. In *Intelligent computation technology and automation,* *2009. icicta'09. second international conference on* (Vol. 1, pp. 933–936).

Parlos, A. G., Chong, K. T., & Atiya, A. F. (1994). Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks*, *5*(2), 255–266.

Parlos, A. G., Rais, O. T., & Atiya, A. F. (2000). Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural networks*, *13*(7), 765–786.

Pasupa, K., & Jungjareantrat, S. (2016). Water levels forecast in thailand: A case study of chao phraya river. In *Control, automation, robotics and vision (icarcv), 2016 14th international conference on* (pp. 1–6).

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, *1*(2), 263–269.

Pötzelberger, K. (1990). A characterization of random-coefficient ar (1) models. *Stochastic processes and their applications*, *34*(1), 171–180.

Prasad, G., Swidenbank, E., & Hogg, B. (1998). A neural net model-based multivariable long-range predictive control strategy applied in thermal power plant control. *IEEE Transactions on Energy Conversion*, *13*(2), 176–182.

Ross, G. J., Adams, N. M., Tasoulis, D. K., & Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, *33*(2), 191–198.

Rubio, G., Pomares, H., Rojas, I., & Herrera, L. J. (2011). A heuristic method for parameter selection in ls-svm: Application to time series prediction. *International Journal of Forecasting*, *27*(3), 725–739.

Salcedo-Sanz, S., Casanova-Mateo, C., Pastor-Sánchez, A., & Sánchez-Girón, M. (2014). Daily global solar radiation prediction based on a hybrid coral reefs optimization–extreme learning machine approach. *Solar Energy*, *105*, 91–98.

Samsudin, R., Shabri, A., & Saad, P. (2010). A comparison of time series forecasting using support vector machine and artificial neural network model. *Journal of Applied Sciences*, *10*, 950–958.

Saxe, A., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., & Ng, A. Y. (2011). On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 1089–1096).

Scardapane, S., Comminiello, D., Scarpiniti, M., & Uncini, A. (2015). Online sequential extreme learning machine with kernels. *IEEE transactions on neural networks and learning systems*, *26*(9), 2214–2220.

Schrauwen, B., Verstraeten, D., & Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007* (pp. 471–482).

Schwenker, F., & Labib, A. (2009). Echo state networks and neural network ensembles to predict sunspots activity. *network*, *3*, 2.

Siegal, S. (1956). *Nonparametric statistics for the behavioral sciences*. McGraw-hill.

Singh, R., & Balasundaram, S. (2007). Application of extreme learning machine method for time series analysis. *International Journal of Intelligent Technology*, *2*(4), 256–262.

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

Su, McAvoy, T. J., & Werbos, P. (1992). Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial & engineering chemistry research*, *31*(5), 1338–1352.

Su, L.-j., & Yao, M. (2013). Extreme learning machine with multiple kernels. In *Control and automation (icca), 2013 10th ieee international conference on* (pp. 424–429).

Taylor, J. W. (2012). Short-term load forecasting with exponentially weighted methods. *IEEE Transactions on Power Systems*, *27*(1), 458–464.

Thissen, U., Van Brakel, R., De Weijer, A., Melssen, W., & Buydens, L. (2003). Using support vector machines for time series prediction. *Chemometrics and intelligent laboratory systems*, *69*(1), 35–49.

Van Gestel, T., Suykens, J. A., Baestaens, D.-E., Lambrechts, A., Lanckriet, G., Vandaele, B., . . . Vandewalle, J. (2001). Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on neural networks*, *12*(4), 809–821.

Van Heeswijk, M., Miche, Y., Lindh-Knuutila, T., Hilbers, P. A., Honkela, T., Oja, E., & Lendasse, A. (2009). Adaptive ensemble models of extreme learning machines for time series prediction. In *International conference on artificial neural networks* (pp. 305–314).

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Venayagamoorthy, G. K., & Shishir, B. (2009). Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks*, *22*(7), 861–863.

Verstraeten, D., Schrauwen, B., d'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural networks*, *20*(3), 391–403.

Wan, C., Xu, Z., Pinson, P., Dong, Z. Y., & Wong, K. P. (2014). Probabilistic forecasting of wind power generation using extreme learning machine. *IEEE Transactions on Power Systems*, *29*(3), 1033–1044.

Wang, L.-X., & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on systems, man, and cybernetics*, *22*(6), 1414–1427.

Webb, G. I., Lee, L. K., Petitjean, F., & Goethals, B. (2017). Understanding concept drift. *arXiv preprint arXiv:1704.00362*.

Wen, Q., Yang, Z., Song, Y., & Jia, P. (2010). Automatic stock decision support system based on box theory and svm algorithm. *Expert Systems with Applications*, *37*(2), 1015–1022.

Werbos, P. (1974). New tools for prediction and analysis in the behavioral science. *Ph. D. Dissertation, Harvard University*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Widrow, B., Greenblatt, A., Kim, Y., & Park, D. (2013). The no-prop algorithm: A new learning algorithm for multilayer neural networks. *Neural Networks*, *37*, 182–188.

Xia, M., Zhang, Y., Weng, L., & Ye, X. (2012). Fashion retailing forecasting based on extreme learning machine with adaptive metrics of inputs. *Knowledge-Based Systems*, *36*, 253–259.

Yadav, A., & Sahu, K. (2017). Wind forecasting using artificial neural networks: A survey and taxonomy. *International Journal of Research In Science & Engineering*, *3*.

Yahoo. (2017). *Shanghaisse.* Retrieved 20170-08-30, from `https://finance.yahoo.com/quote/000001.SS?p=000001.SS`

Yang, L., Liang, X., Ma, T., & Liu, K. (2013). Spectrum prediction based on echo state network and its improved form. In *Intelligent human-machine systems and cybernetics (ihmsc), 2013 5th international conference on* (Vol. 1, pp. 172–176).

Yang, S., Wang, M., et al. (2004). A quantum particle swarm optimization. In *Evolutionary computation, 2004. cec2004. congress on* (Vol. 1, pp. 320–324).

Zhang, & Morris, A. J. (1995). Dynamic process modelling using locally recurrent neural networks. In *American control conference, proceedings of the 1995* (Vol. 4, pp. 2767–2771).

Zhang, H., Zhang, S., & Yin, Y. (2018). Kernel online sequential elm algorithm with sliding window subject to time-varying environments. *Memetic Computing*, *10*(1), 43–52.

Zhang, Z., Gao, G., Tian, Y., & Yue, J. (2016). Two-phase multi-kernel lp-svr for feature sparsification and forecasting. *Neurocomputing*, *214*, 594–606.

Zhou, L.-c., Yang, H.-z., & Liu, C.-b. (2008). Qpso-based hyper-parameters selection for ls-svm regression. In *Natural computation, 2008. icnc'08. fourth international conference on* (Vol. 2, pp. 130–133).