“ INTELLIGENT TECHNIQUES FOR ECONOMIC TIME-SERIES ”

A Thesis submitted in partial fulfilment of the requirements for the Degree of

Master of Technology in **‘Intelligent Automation and Robotics**’
in the
Faculty of Engineering & Technology

Jadavpur University
May 2024

By
Purnima Das

Registration No: 145247 of 2018-2019
Examination Roll No: M6IAR24001
Under the Guidance of **Prof. Amit Konar**
(Department of Electronics & Telecommunication Engineering)
Jadavpur University
Kolkata-700032
India

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE

This is to certify that the dissertation entitled "**INTELLIGENT TECHNIQUES FOR ECONOMIC TIME-SERIES"** has been carried out by **Purnima Das** (University Registration No:145247 of 2018-2019, Examination Roll No: M6IAR24001) under my guidance and supervision and accepted in partial fulfilment of the requirement for the Degree of Master of Technology in Intelligent Automation and Robotics. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree to any other University or institute.

Prof. Amit Konar (Supervisor)
Dept. of Electronics & Telecommunication Engineering
Jadavpur University

Prof. Manotosh Biswas
Head of Department
(Electronics & Telecommunication Engineering)
Jadavpur University

Prof. Dipak Laha
(Dean, Faculty Council of Engineering and Technology)
Jadavpur University

CERTIFICATE OF APPROVAL

The forgoing thesis is hereby approved as a creditable study of a Master of Technology subject and presented in a manner satisfactory to warrant acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for which it is submitted.
Final examination for evaluation of the thesis.

_____

Signature of Examiner

_____

Signature of Supervisor

DECLARATION OF ORIGINALITY AND COMPILANCE

OF

ACADEMIC THESIS

I hereby declare that this thesis entitled "**INTELLIGENT TECHNIQUES FOR ECONOMIC TIME-SERIES**" contains literature survey and original research work by the undersigned candidate, as part of her Degree of Master of Technology in Intelligent Automation & Robotics.

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: PURNIMA DAS

Examination Roll No: M6IAR24001

Thesis Title : Intelligent Techniques for Economic Time-Series

_____

Signature of the candidate

ACKNOWLEDGEMENTS

I would like to seize this moment to-extend my appreciation to all those who provided me with unwavering support during my Masters of technology studies. I am deeply grateful for their invaluable mentorship, constructive feedback, and kind counsel throughout this journey. Would like to express my deepest gratitude to my supervisor, **Prof. Amit Konar**, for his unwavering support, encouragement, and profound expertise. Throughout the journey of my thesis, he has been a constant source of inspiration, and guiding me to overcome obstacles and negativities of the challenges not only in my Masters also in my personal life. I consider myself extremely fortunate to have had him as a mentor, friend, and guiding light This work would not have been materialized, without his whole-hearted help and support. Working under him it was a great experience.

I am deeply honored to extend my heartfelt gratitude to Dr. Arnab Rakshit and Miss Sayantani Ghosh for their valuable advices and unconditional help, support.

I would like to express my sincere respect to Prof, Manotosh Biswas, the Head of the Department of Electronics &Telecommunication Engineering at Jadavpur University. Prof. Pratyusha Rakshit, Prof. Sheli Sinha Chaudhuri.

I am indebted to my classmates and friends for their constant support and good wishes who supported me in a number of ways.

Lastly, I would like to thank my Family members for their love, support and guidance through the course work.

PURNIMA DAS

Examination Roll No: M61AR24001

Date: Jadavpur University

Place: Kolkata - 700032

LIST OF CONTENT

CERTIFICATE **1**

CERTIFICATE OF APPROVAL **2**

DECLARATION OF ORIGINALITY AND COMPILANCE OF ACADEMIC THESIS **3**

ACKNOWLEDGEMENTS **1**

LIST OF CONTENT **3**

LIST OF FIGURES **4-5**

LIST OF TABLES **5**

ABSTRAC **11,38**

INTRODUCTION & OBJECTIVE **11,38**

RELATED WORK ..... **Error! Bookmark not defined.**

LITERATURE SURVEY **5-10**

LIST OF CONTENT

CHAPTER-1

1. **Introduction** 11-13
2. **Principles and Techniques** 13-17
   - 2.1 Prediction of a time series by Taylor series approximation
   - 2.2 Reinforcement Learning Inspired Prediction.
     - 2.2.1 Probabilistic Learning matrix (PLM
3. **Proposed Schemes for time-series prediction** 17-20
   - 3.1 Time-Series Filtering.
4. **Experiments and performance analysis** 20-32
   - 4.1 Performance metric used.
   - 4.2 Data sets used.
   - 4.3 State-of-the-Art Algorithms used for the present comparative study.
5. **Statistical validation of the Proposed Algorithms** 32-33
   - 5.1 Combined 5 by 2 paired t-test.
   - 5.2 Friedman Test.
6. **Discussion** 34
7. **Conclusion** 35
8. **Appendix** 35-36
9. **References** 36-37

CHAPTER -2

1. **Introduction** 38
2. **An Overview of the Proposed Structure Prediction Technique** 38-43
3. **Slope-Sensitive Natural Segmentation** 43-47
4. **Clustering of Segmented Structures** 47-51
5. **Knowledge Encoding and Prediction by Dynamic Stochastic Automaton** 51-55
6. **Prediction Experiments** 55-56
7. **Performance Analysis** 56-60
   - *7.1 Performance Analysis by RMSE and Prediction Accuracy.*
   - *7.2 Sensitivity Analysis.*
   - *7.3 Statistical t-test.*
   - *7.4 Performance of SSNS Algorithm.*
   - *7.5 Performance of the Extended DBSCAN Clustering Algorithm.*
8. **Conclusions** 61
9. **Appendix** 62-63
10. **REFERENCES** 64-66

LIST OF FIGURES

Chapter -1

Fig. 1 Structure of the PLM as the state-action matrix.....**16**

Fig. 2 TAIEX Time-series prediction (plotted from 200th data point to 450th data point in the selected year) representing the actual time series, and predicted time series obtained by Taylor series approximation, Pseudo codes 1, 2 and 3..... **25**

Fig. 3 The actual TAIEX test data and the predicted time-series using the proposed scheme-3..... **26**

Fig. 4 Comparative analysis of the proposed algorithm with the existing algorithms for time-series prediction on TAIEX test data..... **27**

Fig. 5 The actual Gold price test data and the predicted time-series using the proposed Taylor-series approximation..... **27**

Fig. 6 Comparative analysis of the proposed Taylor-series approximation with the existing ones for time-series prediction on Gold price test data..... **27**

Fig. 7 The actual Brent oil test data and the predicted time-series using the proposed algorithm 3..... **28**

Fig. 8 Comparative analysis of the proposed scheme 3 with the existing algorithms for time-series prediction on Brent oil test data..... **28**

Fig. 9 The actual SP500 test data and the predicted time-series using the proposed scheme-3.....**28**

Fig. 10 Comparative analysis of the proposed scheme-3 with the existing algorithms for time-series prediction on SP500 test data.....**29**

Fig. 11 The actual N225 test data and the predicted time-series using the proposed scheme-3..... **29**

Fig. 12 Comparative analysis of the proposed scheme-3 with the existing algorithms for time-series prediction on N225 test data **29**

Fig. 13 The actual Facebook test data and the predicted time-series using the proposed algorithm 3.....**30**

Fig. 14 Comparative analysis of the proposed scheme-3 with the existing algorithms for time-series prediction on Facebook test data.....**30**

Fig. 15 The actual FTSE-100 test data and the predicted time-series using the proposed scheme-2.....**30**

Fig. 16 Comparative analysis of the proposed scheme-2with the existing algorithms for time-series prediction on FTSE-100 test data.....**31**

Fig. 17 Identification of possible failures in prediction of TAIEX time series from 300th to 400th test data points in presence of different types of non-stationarity..... **32**

CHAPTER -2

Fig. 1. Basic steps of knowledge encoding in a DSA from a given time-series during the training phase **41**

Fig. 2. Construction of an automaton (c) for a segmented and partitioned time-series (a) with discovered primitive structures (cluster centers) p_1, p_2, p_3 and p_4, shown in (b).....**41**

Fig. 3. Illustrating a Dynamic Stochastic Automation..... **42**

Fig. 4. Basic steps of structure prediction in a time-series using a DSA during the application phase.....**42**

Fig. 5. Illustrating the SSNS segmentation: (a) time-series, (b) partitioning and T- SOT construction, (c) windowing, (d) freq. of window labels, (e) segmentation **45**

Fig. 6. The mechanism of multi-level clustering: Original data set is clustered at different levels based on local neighborhood data density. Points of higher density are clustered at higher level, allowing the low- d e n s i t y points to get clustered at subsequent lower levels.....**50**
Fig. 7. Linked list representation of the PIT.....**50**
Fig. 8. The resulting cluster centers obtained by segmentation followed by multi-level DBSCAN clustering of the TAIEX 1990-2017 time-series.....**51**
Fig. 9. A DSA to illustrate its parameters, the labels in each arc are of the form *"a/r,u"* where *a* is the input symbol, *r* is the probability of occurrence of the transition and *u* is the expected duration of the transition. The groups G_1, G_2 and G_3 corresponding to state s_1 are shown using closed dotted curves.....**53**
Fig. 10. Example of a time-labeled dynamic stochastic automaton.....**54**

Fig. 11. Results of structure prediction for TAIEX 2017, November 1. to December 31 time-series by three methods: Proposed, FSW + Fuzzy C-means (FCM) and SWAB + k-means clustering **56**

Fig. 12. The DSA obtained from TAIEX time-series for the periods: January to October in 2017 for Fig. 11 ..... **56**

LIST OF TABLES

CHAPTER -1

Table1: Data composition of different time series.....**21**
Table 2: Parameter settings for the best performance of different time-series along with obtained RMSE **22-24**
Table 3: Results of Experiment -1..... **25**
Table 4. Statistical Validation using Combined 5 by 2 paired t-test with 7 individual datasets.....**33**

Table 5. Statistical Validation using Friedman Test **33**

CHAPTER -2

Table 1: Pseudo code of the SSNS Algorithm **46**
Table 2: Frequency counts of transitions from state s_1 to other states in the DSA of Fig. 9 ..... **53**
Table 3: Pseudo code of the DSA Construction Algorithm **53**
Table 4: MPS/MPSS with their probability of occurrence and expected duration for each pair of states in the DSA of Fig. 10 .....**54**
Table 5: Computational and Runtime Complexities of Segmentation, Clustering and DSA Construction Algorithms..... **55**
Table 6: Percentage accuracy and RMSE averaged over 3 time-series **57**
Table 7: Evaluation metrics for assessing the performance of the proposed model on TAIEX, NASDAQ and DJIA time-series.....**58**
Table 8: Optimal parameter setting by sensitivity analysis **58**
Table 9: Results of 2-Sample t^2 test, considering the proposed technique as the Reference Algorithm **59**
Table 10: Comparison of time-series segmentation algorithms b matching of generated segments with hand-crafted ground-truth segments **60**
Table 11: Comparison of existing clustering algorithms with the proposed technique based on the SSID and SICD metrics **60**

LITERATURE SURVEY

A literature survey on intelligent techniques for economic time-series analysis would delve into various methodologies, algorithms, and studies aimed at leveraging artificial intelligence and machine learning to analyze and forecast economic data. Here's how you might structure such a survey:

1. **Introduction to Economic Time-Series Analysis:**
    - Define economic time-series data and its significance in understanding economic phenomena and making informed decisions.
    - Discuss the challenges associated with economic time-series analysis, such as non-linearity, volatility, and the presence of multiple interacting factors.
2. **Traditional Approaches vs. Intelligent Techniques:**
    - Review traditional statistical methods commonly used for economic time-series analysis, such as autoregressive models, ARIMA, and exponential smoothing.
    - Introduce intelligent techniques, including machine learning and artificial intelligence algorithms, as alternatives or complements to traditional approaches.
3. **Machine Learning Algorithms for Economic Time-Series Analysis:**
    - Explore various machine learning algorithms applied to economic time-series analysis, such as:
        - Regression models (linear regression, polynomial regression).
        - Time-series forecasting models (LSTM, GRU, Prophet).
        - Clustering algorithms (k-means, hierarchical clustering) for pattern recognition.
    - Discuss the strengths and limitations of each algorithm in the context of economic data analysis.
4. **Feature Engineering and Data Preprocessing:**
    - Discuss the importance of feature engineering and data preprocessing in economic time-series analysis.
    - Review techniques for extracting relevant features from economic data, handling missing values, and scaling data for modeling.
5. **Deep Learning Architectures for Economic Time-Series Analysis:**
    - Explore deep learning architectures tailored for time-series forecasting and analysis, such as:
        - Recurrent neural networks (RNNs), including LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit).
        - Convolutional neural networks (CNNs) for image-based time-series analysis.
        - Transformer models for sequence-to-sequence learning.
    - Discuss the advantages and challenges of using deep learning architectures for economic time-series analysis.
6. **Ensemble Techniques and Hybrid Models:**
    - Introduce ensemble techniques, such as bagging, boosting, and stacking, for combining multiple models in economic time-series forecasting.
    - Discuss the concept of hybrid models that integrate traditional statistical methods with machine learning or deep learning approaches.
7. **Applications and Case Studies:**

- Present real-world applications and case studies where intelligent techniques have been applied to economic time-series analysis, such as:
  - Stock market prediction.
  - Macroeconomic forecasting (GDP, inflation, unemployment).
  - Financial risk management.
- Discuss the performance of intelligent techniques in different economic domains and scenarios.

8. **Evaluation Metrics and Performance Benchmarking:**
   - Describe common evaluation metrics used to assess the performance of models in economic time-series analysis, including accuracy, precision, recall, and Mean Absolute Error (MAE).
   - Compare the performance of different techniques and models using appropriate benchmarks and datasets.
9. **Challenges and Future Directions:**
   - Identify challenges and limitations in applying intelligent techniques to economic time-series analysis, such as data scarcity, model interpretability, and generalization to unseen data.
   - Propose potential research directions for addressing these challenges and advancing the field of intelligent economic time-series analysis.
10. **Conclusion:**
    - Summarize key findings from the literature survey.
    - Highlight the significance of intelligent techniques in enhancing the accuracy and efficiency of economic time-series analysis.
    - Provide insights into future opportunities for research and innovation in this field.
11. **References:**
    - Compile a comprehensive list of references cited throughout the survey, including academic papers, books, and relevant online resources.

By following this structured approach, you can provide a comprehensive overview of the state-of-the-art techniques and methodologies for analyzing economic time-series data using intelligent techniques, along with insights into current challenges and future research directions.

LITERATURE SURVEY on CHAPTER-1

A literature survey on Taylor Series and reinforcement learning-inspired adaptive prediction of non-stationary time-series would be a comprehensive exploration of existing research, methodologies, and approaches in this domain. Here's a structured overview of what such a survey might include-

1. **Introduction to Taylor Series and Reinforcement Learning:**
   - Briefly introduce Taylor Series and its application in mathematical analysis.
   - Provide an overview of reinforcement learning (RL) and its use in adaptive prediction tasks.
2. **Non-Stationary Time-Series Analysis:**
   - Define non-stationary time-series and discuss their characteristics.

- Review traditional methods for analyzing and predicting non-stationary time-series, such as autoregressive models, moving averages, and exponential smoothing.

3. **Taylor Series in Time-Series Prediction:**
   - Explore how Taylor Series can be applied to time-series prediction tasks.
   - Discuss the advantages and limitations of using Taylor Series in this context.
   - Review relevant studies that have utilized Taylor Series for time-series prediction.
4. **Reinforcement Learning for Adaptive Prediction:**
   - Explain how reinforcement learning techniques can be used for adaptive prediction in non-stationary environments.
   - Discuss different RL algorithms suitable for time-series prediction tasks, such as Q-learning, Deep Q-Networks (DQN), and Policy Gradient methods.
   - Provide examples of applications where RL has been successfully applied to time-series prediction.
5. **Integration of Taylor Series and Reinforcement Learning:**
   - Review existing research that combines Taylor Series and reinforcement learning for adaptive prediction of non-stationary time-series.
   - Discuss the theoretical foundations and practical implications of integrating these two approaches.
   - Highlight any challenges or limitations associated with this integration.
6. **Case Studies and Applications:**
   - Present case studies or real-world applications where the combined approach of Taylor Series and reinforcement learning has been implemented.
   - Analyze the performance of these methods compared to traditional approaches or standalone RL methods.
7. **Evaluation Metrics and Performance Benchmarks:**
   - Discuss common evaluation metrics used to assess the performance of predictive models in non-stationary time-series scenarios.
   - Compare the performance of Taylor Series-based methods, RL-based methods, and their combination using appropriate benchmarks.
8. **Challenges and Future Directions:**
   - Identify challenges and open research questions in the field.
   - Suggest potential avenues for future research, such as improving model robustness, scalability, and interpretability.
9. **Conclusion:**
   - Summarize key findings from the literature survey.
   - Highlight the significance of integrating Taylor Series and reinforcement learning for adaptive prediction of non-stationary time-series.
   - Provide insights into potential applications and implications of this research.
10. **References:**
    - Compile a comprehensive list of references cited throughout the survey, including academic papers, books, and relevant online resources.

Each section should delve into relevant studies, methodologies, and findings to provide a thorough understanding of the topic. Additionally, it's essential to critically analyze the strengths

and weaknesses of existing approaches and identify opportunities for innovation and improvement.

LITERATURE SURVEY on CHAPTER-2

A literature survey on learning regularity in an economic time-series for structure prediction would involve examining various approaches, methodologies, and findings in the intersection of machine learning, time-series analysis, and economics. Here's how you might structure such a survey-

1. **Introduction to Economic Time-Series Analysis:**
    - Define economic time-series data and its importance in understanding market behavior and making forecasts.
    - Discuss the challenges associated with analyzing economic time-series data, such as non-linearity, seasonality, and structural changes.
2. **Importance of Learning Regularity for Structure Prediction:**
    - Explain why learning regularity within economic time-series data is crucial for accurate structure prediction.
    - Highlight the benefits of identifying and modeling regular patterns or trends in economic data for decision-making and forecasting.
3. **Traditional Approaches to Economic Time-Series Analysis:**
    - Review traditional statistical methods used for analyzing economic time-series data, such as autoregressive models, ARIMA (Autoregressive Integrated Moving Average), and exponential smoothing techniques.
    - Discuss the limitations of these traditional approaches in capturing complex patterns and dynamics within economic time-series data.
4. **Machine Learning Techniques for Time-Series Analysis:**
    - Explore machine learning algorithms commonly applied to economic time-series analysis, including but not limited to:
        - Regression models (linear regression, logistic regression).
        - Time-series forecasting models (ARIMA, SARIMA, Prophet).
        - Deep learning architectures (recurrent neural networks, convolutional neural networks, transformers).
    - Discuss the advantages and limitations of using machine learning techniques for modeling economic time-series data.
5. **Learning Regularity in Economic Time-Series:**
    - Survey existing research on methods for learning regularity in economic time-series data.
    - Explore techniques for trend detection, seasonality decomposition, and anomaly detection in economic time-series.
    - Discuss approaches for identifying and modeling structural breaks or regime changes in economic data.
6. **Structure Prediction Models:**

- Review approaches and models specifically designed for predicting the structure of economic time-series data.
- Discuss methods for identifying and forecasting cyclical patterns, long-term trends, and other structural components in economic data.
- Highlight studies that have successfully applied structure prediction models to economic forecasting tasks.

7. **Applications and Case Studies:**
   - Present real-world applications and case studies where learning regularity in economic time-series data has been instrumental in making predictions or informing decision-making processes.
   - Discuss the performance of structure prediction models in various economic domains, such as finance, macroeconomics, and business analytics.
8. **Evaluation Metrics and Performance Benchmarking:**
   - Describe common evaluation metrics used to assess the performance of structure prediction models in economic time-series analysis.
   - Compare the performance of different models using appropriate benchmarks and datasets.
9. **Challenges and Future Directions:**
   - Identify challenges and limitations in learning regularity for structure prediction in economic time-series data.
   - Propose potential research directions for improving the accuracy, robustness, and interpretability of structure prediction models in economic analysis.
10. **Conclusion:**
    - Summarize key findings from the literature survey.
    - Highlight the significance of learning regularity in economic time-series data for structure prediction and its implications for decision-making and forecasting.
    - Provide insights into future avenues for research and innovation in this field.
11. **References:**
    - Compile a comprehensive list of references cited throughout the survey, including academic papers, books, and relevant online resources.

This structured survey should provide a comprehensive overview of the state-of-the-art methods and approaches for learning regularity in economic time-series data for structure prediction, as well as insights into current challenges and future research directions.

CHAPTER -1

Taylor Series and Reinforcement Learning Inspired Adaptive Prediction of a Non-stationary Time-Series

Abstract *- Prediction of a non-stationary time-series is hard as the frequency components and their amplitudes in the series vary randomly over time. This paper proposes a two-step approach for prediction of such non-stationary time-series. The first step employs Taylor series to approximately predict the next time-point value in the series from its current and last two preceding sample values. The Taylor series-based prediction, however, presumes that the time-series is locally stationary. The second step includes reinforcement learning inspired error correction in the Taylor series-based prediction results using an adaptive probabilistic learning matrix. The said matrix is used to encode the scale factors (error coefficients) in the learning equation from an estimate of errors incurred in Taylor series-based prediction. The learning is performed by a reward/penalty type adaptation of conditional probabilities involved in the probabilistic learning matrix. On convergence of the matrix, the saved probabilities are used to determine the appropriate scale factor needed to scale the estimated error in Taylor series-based prediction with an aim to rectify the prediction result by adding the scaled error to the predicted time-series value. Experiments undertaken confirm that the proposed technique outperforms the state-of-the-art prediction models by a significant margin of prediction accuracy.*

Keywords: Time-series prediction; Taylor series; Reinforcement learning; Prediction model.

1. Introduction

A time-series represents a discrete sequence of a time-valued function ordered over time. Generally, the samples of the real-world entities, such as temperature, rainfall, population-growth etc., are measured over a regular interval of time. The unprocessed sample values of the above entities presented in the sequential order of their occurrence over a given interval of time together constitutes a time-series [1]. Time-series prediction refers to forecasting the future values of the series from its present and past values. Prediction of a time-series, in general, is hard because of the inherent non-determinism, non-linearity, non-stationarity and non-Gaussian characteristics of the series [2].

Certain real-world signals, such as electrocardiogram (ECG) that maintains a fixed wave-shape, usually are deterministic. Prediction of such deterministic signals is relatively easier as the users have knowledge about their wave-shape. Forecasting of a non-deterministic time-series, such as an electroencephalogram (EEG), is hard as the signal at a given time-point may take up an arbitrary value in an expected interval [3]. Non-linearity refers to nonlinear shape of the signal, and generally requires a non-linear regression model for forecasting of such time-series. The Gaussian characteristic of a signal requires that the sample values of the signal should lie within a range of $[(m-3\times\sigma), (m+3\times\sigma)]$, where m and σ denote mean and standard deviation of the signal. In many real world time-series, such as an economic time-series, the above characteristic is not maintained, and the series is referred to as non-Gaussian.

The most fundamental hindrance in time-series prediction, perhaps, is the non-stationarity of the signal. A stationary signal usually maintains a uniform value in its statistical characteristics, including mean, variance and auto-correlation over the non-overlapped discrete time-windows of the signal. An economic time-series or an EEG signal, hardly maintains the above characteristics, and so is called non-stationary. Besides variation in statistical attributes mentioned above over the time-windows, a non-stationary signal also has temporal fluctuation in frequency contents and their amplitudes, and sometimes in trend, such as steep rise (bullish) or fall (bearish) in an economic time-series [2]. Handling non-stationarity in time-series prediction is challenging as there is hardly any generic solution to this problem. However, there are approaches to handle individual types of non-stationarity. For example, trend removal is often carried out by taking first or higher order differences between successive sample values of the time-series [4]. Random fluctuation in signal amplitude over time-windows is taken care of by partitioning the amplitude of the signal into intervals of fixed width, and then the prediction problem is re-casted as to find the interval containing the next-time point value [5]. In this paper, we would classify non-stationarity of a time-series from the points of view of large variation in signal i) mean, ii) variance, iii) both mean and variance jointly, and iv) trends.1Early works on time-series, including Newton-Cotes and Lagrange's extrapolation

techniques, dealt with deterministic models [6], [7] for stationary time-series prediction. These models date back around 350 years. Only in last three decades, significant research outcomes in time-series modelling and forecasting is noticed. Among the popular and significant contributions to time-series research, the linear prediction models, such as the Autoregressive (AR) model [8] and the Moving Average (MA) model [8] need special mention. An AR model of order p aims at forecasting the future value of the time-series as a linear function of past p observations along with a random error and a constant term [9]. The MA model of order q attempts to predict the future time-series value from the prediction errors obtained at last q time-points along with the mean of the series and a random error term. The Autoregressive Moving Average (ARMA) model utilizes the composite benefits of AR and MA models by linearly combining their mathematical structures [8]. The ARMA model can be used for prediction of a stationary time-series. Autoregressive Integrated Moving Average (ARIMA) model [9] is an extension of ARMA model to transform a non-stationary time-series into an equivalent stationary one by first applying "finite differencing of successive data point values" [10], and then by employing an ARMA model for prediction. Box and Jenkins [4] extended the ARIMA model to take into account of seasonality along with non-stationarity, and called it Seasonal ARIMA (SARIMA). Significant extension of the basic ARIMA model is undertaken over the last one decade to reduce its computational overhead for real-time prediction [11], and also to hybridize with other regression/function approximation models [12], [13] to improve its performance.

Among the well-known non-linear time-series models, the Non-linear Moving Average (NMA) model [9] needs special mention. Engle's ARCH model [14] takes into account of prediction error at time point t and its squared value (or square-foot value) to predict the next time point value. Artificial Neural Network (ANN) models have shown promising results [15]-[18] in the prediction of time-series, particularly for their inherent characteristic of universal function approximation by cascading nonlinearities in their functional architectures. Among the well-known ANN models, feed-forward neural network models, trained with Back-propagation algorithm and Time-lagged Neural Network (TLNN) models need special mention.

Vapnik proposed two distinct formulations of the well-known Support Vector Machines (SVM) [19] classifiers and SVM for regression (SVR) applications [20] in the settings of quadratic programming problem (QPP). Among the formulation of SVM for regression applications, the Least Square SVM (LS-SVM) [21] is most popular. Here, the QPP formulation is bypassed by a relatively simplified formulation to minimize a cost function of sum of squared weights (w_i) and sum of squared prediction errors (e_i) over last N time-points, which is subjected to a regression function between x_i and y_i, where y_i represents the weighted sum of kernelized x_i plus error sample value e_i plus a constant b. The merit of LS-SVM lies in the phenomenon that it transforms the traditional QPP into a simultaneous linear system problem. This results in simplicity in computation, fast convergence and high precision. There are variants of LS-SVM, called Dynamic LS-SVM (DLS-SVM), and Recurrent LS-SVM (RLS-SVM), which are not discussed here for lack of space. A few interesting works extending classical SVM based prediction include [22]-[25].

Fuzzy time-series are currently gaining importance for its inherent characteristics of handling uncertainty in certain time-series, for instance economic time-series. Amplitude of such time-series is first partitioned into finite (typically < 10) number of partitions, and one or more fuzzy sets are incorporated to represent a partition [5]. Fuzzy implication rules are designed to describe the transitions of the series from one partition to the next. These rules are used later for prediction of the given time-series at a given time-point. Both classical and type-2 fuzzy sets have been explored to handle the problem, where type-2 fuzzy sets show promising results than their classical counterpart [1] with respect to prediction accuracy.

In recent times, researchers are paying attention to synergistically employ two or more prediction models to intelligently overcome the four basic hindrances of time-series prediction. For example, in [1], the authors employ fuzzy logic to handle non-stationary characteristic and neural nets to model the nonlinear regressive characteristic of the time-series. Recently, Al-hnaity et al. used Genetic Algorithm (GA) as an optimizer to determine the optimal weights of the responses of 3 prediction models, including Back-Propagation Neural Network (BPNN), SVR and LS-SVM, indicating better prediction result of the composite model in comparison to that of the individual models [26]. In another recent work [27], the authors employ Fuzzy C-means (FCM) clustering algorithm to cluster the data points in a time-series and then utilize the membership values of the data-points to determine the fuzzy prediction rules using a feed-forward neural network. Aladag et al. proposed an alternative approach to utilize both FCM for clustering of the data points and Particle Swam Optimization (PSO) technique in the next phase to determine the optimal fuzzy prediction rules for the subsequent prediction phase [28]. In [29], Abdulkarim et al. proposed a neural network (NN) based time-series forecasting in the settings of a dynamic optimization problem, where a NN is trained with a dynamic PSO algorithm. They compared the performances of feed-forward NN with different recurrent NNs (Jordan NNs, Elman NNS, multi-recurrent NNs and time delayed NNs) based architecture on eight different time-series forecasting problems, and they ultimately arrived at the conclusion that recurrent

connections are not necessary in the NN for time-series forecasting problem, if the network is trained with dynamic PSO algorithm. Among other interesting and useful works on time-series prediction, [30]-[37] need special mention. In particular, Simple Averaging (SA) technique [37], Bat Neural Network (BNN) [35] based prediction and Sparse Bayesian Learning (SBL) [36] offer attractive approaches to handle the prediction problem with appreciably high prediction accuracy.

The aim of this paper is to utilize Taylor series as a prediction model for time-series forecasting. Although there has been an attempt to employ Taylor series for the same prediction problem [38], the said method suffers from two fundamental limitations. First, it has high computational overhead as it requires expanding the Taylor series up to 8 coefficients without significant improvement in performance. Second, it trains a feed-forward neural network to reduce the residual error emerged due to truncation of the Taylor series after 8 terms. However, the residual error might grow in case the prediction point falls at the juncture of 2 consecutive structures [1] of the time-series, as the higher order terms in the Taylor series may involve sample points in the series falling in previous structure. In the present context, we employ a second order expansion of the series only, with approximation of the first and second order time-derivatives at current time-point by the current and last 2 sample values in the series. Effectively, such formulation has 2 merits. First, it saves significant computational overhead. Second and most importantly, it increases the probability that the previous 2 sample points used for prediction lie on the same structure of the series under prediction. The residual error in Taylor series prediction, here is corrected by reinforcement learning inspired probabilistic learning scheme. The learning scheme employs an adaptation law to reduce the prediction error over the learning epochs with the help of an error coefficient F, capable of controlling the learning speed. The choice of F is determined with an aim to reduce the error in the next iteration. Although F could have different values for different error levels, we quantize F into a few finite levels, and assign a suitable quantized value to F, so as to reduce the error in the next learning epoch. The learning stops when the error comes below a pre-defined finite limit. On convergence of the learning algorithm, we use it for predictive application. Experiments undertaken on different time-series databases [39]-[43] reveal that the proposed learning algorithm can significantly reduce the prediction error obtained from Taylor series prediction, and thus serves an important role in the proposed prediction technique.

The remainder of this paper is classified as follow. In Section 2, we introduce the principles and techniques involved in the prediction procedure of the proposed algorithm. Section 3 demonstrates different schemes for time-series prediction following the principles narrated in Section 2. The experiments and performance analysis of different time-series are also taken under consideration in Section 3. Section 4 presents the statistical tests to validate the prediction performance of the proposed techniques with its competitors. A discussion on the pros and cons of the proposed techniques with possible justification is introduced in Section 5. Finally, concluding remarks are summarized in Section 6.

2. Principles and Techniques

This section proposes a novel method to predict a time series from its last three sample values using Taylor series approximation, followed by reinforcement learning.

2.1 Prediction of a time series by Taylor series approximation

Let $f(t)$ be a continuous time-function, available in discrete form: $f(kT)$ for $k = 1,2,3,..,\alpha$ where T denotes the sampling interval. In certain real-world problems, such as economic time-series, daily rainfall, daily temperature, monthly sunspot number etc., the time-series data is readily available in discrete form: $f(kT)$, $k = 1,2,3,..,n$ for some finite integer n.

The Taylor series expansion of a continuous function: $f(t+h)$ around a time point $t = t^*$ for small h is given in (1) below.

$$f(t^* + h) = f(t^*) + hf'(t^*) + \frac{h^2}{2!} f''(t^*) + ... \quad (1)$$

where, $f'(t^*) = \left[\frac{df}{dt} \right]_{*}$

$\left\lfloor \quad \right\rfloor_{t=t}$

$$= \lim_{h\to 0}\left[\frac{f(t^*)-f(t^*-h)}{h}\right] \quad (2)$$

and, $f''(t^*) = \left[\frac{d^2 f}{dt^2}\right]_{t=t^*}$

$$= \lim_{h\to 0}\left[\frac{f'(t^*)-f'(t^*-h)}{h}\right]$$

$$= \lim_{h\to 0}\left[\frac{f(t^*)-f(t^*-h)}{h^2} - \frac{f(t^*-h)-f(t^*-2h)}{h^2}\right]. \text{ [by (2)]} \quad (3)$$

Plugging (2) and (3) into (1), we obtain:

$$f(t^*+h) \approx f(t^*) + \left[f(t^*)-f(t^*-h)\right] + \frac{1}{2}\times\left[f(t^*)-2f(t^*-h)+f(t^*-2h)\right].$$

$$f(t^*+h) = 2.5\,f(t^*) - 2\,f(t^*-h) + 0.5\,f(t^*-2h). \quad (4)$$

Defining $t^* = k^*T, h = T$, we rewrite (4) by (5)

$$f((k^*+1)T) = 2.5f(k^*T) - 2f(k^*T-T) + 0.5f(k^*T-2T) \quad (5)$$

$$= 2.5f(k^*T) - 2f((k^*-1)T) + 0.5f((k^*-2)T). \quad (6)$$

In most of the real-world applications, $h = T = 1$, where 1 denotes unit time, such as one day in daily time-series prediction. Thus, setting the above in (6), we obtain:

$$f(k^*+1) = 2.5f(k^*) - 2f(k^*-1) + 0.5f(k^*-2). \quad (7)$$

2.2 Reinforcement Learning Inspired Prediction

The Taylor series expansion provides only the approximate value of the predicted series because of incorporation of fewer terms (first phase) of the series. The additional terms of the series are intentionally dropped out as those terms involving higher derivatives would require previous values in the series, which may fall in a different structure (consecutive sequence of points having uniform slopes) than the one currently being used for the present prediction.

By (7), we predict the time series data at time-point (k^*+1) based on the original data available at the time-points: k^*, k^*-1, k^*-2, for $k^* = 3$ to l-1,where l is the length of the time-series. We denote the predicted data at time-point (k^*+1) as $\hat{f}(k^*+1)$ .One way to reduce the error in the prediction algorithm is to learn the prediction steps from the historical information of the series. The learning is required to reduce the prediction error at each of the predicted points of $k^* = 4$ to$l$-1, defined by equation (8),

$$e(k^*) = f(k^*) - \hat{f}(k^*) \quad (8)$$

where, $f(k^*)$ and $\hat{f}(k^*)$ respectively refer to the original and predicted time-series values atk*.We introduce a parameter $F(k^*)$ to scale the error so as to reduce prediction error in the second phase. Let

$$f(k^*+1) = \hat{f}(k^*+1) + F(k^*).e(k^*) \quad (9)$$

for $k^* = 4$ to $(l-1)$

where,

$f(k^*+1)$ = the original time series at time-point k^*+1

$\hat{f}(k^*+1)$ = the value predicted by Taylor series at time-point k^*+1

$e(k^*)$ = error obtained by Taylor series prediction at time-point k^*

$F(k^*)$ = scale-factor at time-point k^*

Here, $F(k^*)$ minimizes the difference between $f(k^*+1)$, $\hat{f}(k^*+1)$ for a given error-value $e(k^*)$.

2.2.1 Probabilistic Learning matrix (PLM)

The aim of introducing PLM is to understand the stochasticity in an observed series and to predict or forecast the future values of a series based on the history of that series. So, we come up with a machine learning model that learns from the past mistakes to improve the future prophecy. We prepare a probabilistic learning matrix that captures the best suitable action to be taken based on theTaylor series prediction error at the time-point k^*, in order to improve the Taylor series prediction at time k^*+1 for $k^*=4$ to $(l-1)$. In the PLM, the rows are represented by error bands and the columns are represented by the scale-factor(SF) bands. For the rows, the list of error is quantized into equal-width partitions, and for the columns the list of scale factor values is quantized in a similar fashion.

A. Quantization of error: The range between the approximate minimum $(e_{\min})$ and maximum value $(e_{\max})$ of Taylor series prediction error is divided into equal width error bands. The error bands are quantized intervals $q_1(e), q_2(e), q_3(e)$ such that the following conditions hold:

Condition: 1

$\lceil | q_1(e) \rceil | < | \lfloor q_2(e) | \rfloor$

$\lceil | q_2(e) \rceil | < | \lfloor q_3(e) | \rfloor$

.....

$\lceil | q_{n-1}(e) \rceil | < | \lfloor q_n(e) | \rfloor$

where, $\lceil | . \rceil |$ and $| \lfloor . | \rfloor$ respectively are Ceiling and floor functions.

Condition: 2

$e_{\min} \approx | \lfloor q_1(e) | \rfloor$

and,

$$e_{\max} \approx \lceil | q_n(e) \rceil |$$

where, error $e(k^*)$ lies in $[e_{\min}, e_{\max}]$.

Condition: 3

$q_i(e)$ and $q_{i+1}(e)$, for $i = 1$ to $(n-1)$ are contiguous

i.e., $| \lfloor q_{i+1}(e) | \rfloor - \lceil | q_i(e) \rceil | \leq \varepsilon$, where ε is a small predefined positive number.

B. Quantization of scale-factor values: For known values of $f(k^*+1)$, $\hat{f}(k^*+1)$ and $e(k^*)$, we obtain scale factor $F(k^*)$ from (9). Selection of the quantized bands of $F(k^*)$ is performed in two steps. In the first step, we sort the list of values of $F(k^*)$ obtained for different $e(k^*)$, in a fixed (ascending or descending) order. In the second step, we divide the range of $F(k^*)$ into m uniformly quantized intervals: $r_1(F), r_2(F), \ldots\ldots r_m(F)$ following similar conditions taken for the error bands. In the PLM (Fig. 1), $p(r_j / q_i)$ denotes the probability of selection of scale factor band r_j, given the probability of occurrence of error band q_i, satisfying the condition $\sum_{j=1}^{m} p(r_j / q_i) = 1$, for $i = 1$ to n.

C. PLM Dimension Selection: The choice of the number of error bands n and scale factor bands m depends on the length l of the time-series. In general, for a time-series of length l, we have l possible error sample values, one at each time point, and also l possible scale factor values, each one corresponding to one error satisfying (9).Thus, n and m should be less than l. Here, selection of n and m can be posed in the settings of an optimization problem with an aim to find n, m in (1, l), such that Root Mean Square Error (RMSE) [1] or any other metric of measuring time-series performance is minimum. The optimization problem can be solved by constructing PLMs of all possible

dimensions in (1, *l*)× (*1*, *l*) and then by exhaustively searching the optimal values of n and m, which yields the least RMSE.The principle of selection of optimal values of n and m is briefly presented below. A pseudo code of this is available in the Appendix.

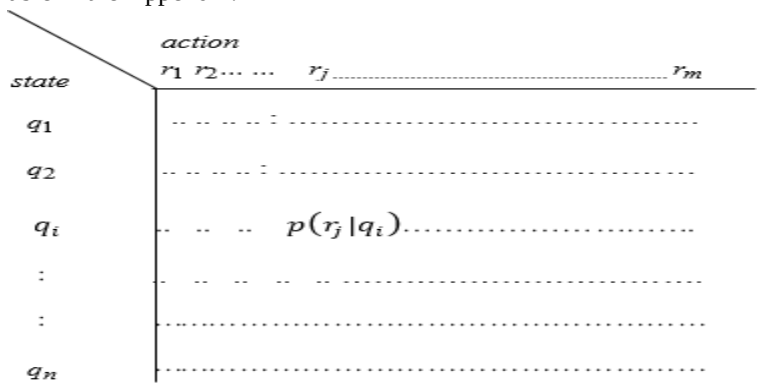


Fig. 1 Structure of the PLM as the state-action matrix

Principles of selection of optimal n and m

1. Choose m and n as all possible factors of *l*(=length of time-series)excluding one. If *l* has z factors, then m and *n* may be any arbitrary factors of *l* and need not be equal. So, there isz^2 number of possible PLMs one of which should be responsible for the least RMSE. Initialize the rewards ($p(r_j/q_i)$) for all *i, j* in the matrices in a manner, so that all elements in a row are equal and row-sum =1.
2. Now, for each time-series data point, predict the next point by Taylor series approach, obtain error by (8) and scale factor by (9). Repeat it for all the data points in the series. Save the errors in Array 1 and scale factors in Array 2 in ascending order of time. Find maximum error and minimum error in Array 1, and divide the difference of them into n quantized intervals/bands of equal width. Similarly, find maximum and minimum scale factors in Array 2, and divide their difference into m quantized intervals/bands of equal width for each pair of selected dimensions: m, n of individual matrices.
3. For each pair of error and its corresponding scale factor in arrays 1 and 2, find the error band q_i containing the error value, and scale factor band r_j containing the scale factor. Then increment a small positive reward δ at (*i, j*) cell in the PLM, and decrease the reward by $\delta/(m\text{-}1)$ from the rest of the columns in row *i* of all the tentative PLM matrices.
4. Repeat step 3 until all the data in Array 1 and Array 2 are consulted for adaptation of tentative PLM matrices. Alternatively, if the tentative PLM matrices converge (i.e., more than 70% of the matrix elements do not change in the last 50 learning epochs), stop. So z^2 number of tentative PLM constructions is over.
5. To test which of the z^2tentative PLMs yields the least RMSE, we need to predict the data points over an unknown segment of the same time-series (eg. TAIEX time-series) using each PLM one by one. Here, too for each time point value, we predict the next time-point value *Y* by Taylor series, and then compute error *E* at the predicted time point. Next, we compute the error band q_i containing the error Ein the present prediction. For the selected error band q_i, we identify the scale factor range r_j for which the adapted learning score $p(r_j/q_i)$ is the largest. Then we get the predicted value as *Y* + mid-point value of the selected scale factor range $r_j \times E$.
6. After the prediction at all points in the unknown segment of the series is over using all tentative PLMs, we compute RMSE using a formula to be introduced in the next section. The RMSE is evaluated for each tentative PLM, and the one with the smallest RMSE is selected as the winning PLM for prediction of the given time series. So, *n, m* and $p(r_j/q_i)$ for all *i,j* of the winning PLM are known.

D. Probability Updating (Learning)

For an error $e(k^*) \in q_i$ and $i \in [1, n]$,the selection of action $F\left(k^*\right)$ is found to be rewarding, if it satisfies $f(k^*+1) = \left\{ \hat{f}(k^*+1) + F(k^*).e(k^*) \right\}$, where $F\left(k^*\right) \in r_j$ and $j \in [1, m]$. We then adapt

$p(r_j / q_i) \leftarrow p(r_j / q_i) + \delta$

and, for the rest of the columns in row i, we adapt:

$$p(r_k / q_i) \leftarrow p(r_k / q_i) - \frac{\delta}{m-1}$$

where, δ is a small pre-defined positive number, and m is the number of actions, n is the number of states.
The above two learning equations are adapted in a loop until the matrix converges, i.e., the difference between the current value and the last s-th (s=50, say) iteration value of $p(r_j / q_i)$ lies below a pre-defined threshold ω (=10 × δ, say) for a significant number of (say, at least 70%) the matrix elements.

E. Prediction: For a given error $e(k^*) \in q_i$ where $i \in [1, n]$ in the PLM, we find the maximum value in the i^{th} row and select the mid value (say,u) of the corresponding action band from the PLM. We use this $F\left(k^*\right) = u$ for predicting $f(k^*+1)$ by using

$$f(k^*+1) = \hat{f}(k^*+1) + u.e(k^*) . \qquad (10)$$

3. Proposed Schemes for time-series prediction

In this section, we present three different schemes for time-series prediction. The first scheme is concerned with raw data for training and testing. Due to non-stationary characteristic of the selected time-series, the acquired time-series data has large amplitude randomness, and thus often is found to be rough and corrugated. Usually the randomness appears as short duration high frequency noise, which needs to be filtered with a low-pass Gaussian digital filter. The Gaussian filter smoothens the time-series, thereby eliminating the unwanted noise components present therein. The second scheme is developed to overcome the above problem. Here, we used filtered time-series data for both training and testing phases. However, it is apparent that learning the time-series from filtered data often misses important information in the series, resulting in poor performance in the prediction phase. This inspired us to learn the time-series from noisy instances, but testing is undertaken with filtered data in scheme 3. Experiments undertaken demonstrate that the performance of prediction usually is the best for the third scheme. Next, we outline the principles used for time-series filtering, which will be used later in the proposed schemes for time-series prediction.

3.1 Time-Series Filtering

One simple approach to filter a time-series is to pass the sample values of the series through a Gaussian filter. Usually, filtering in time-domain is performed by time-convolution of the input time-series with a Gaussian signal of well-defined mean m and variance σ^2. A Gaussian function in time-domain with parameters m and σ^2 is represented by (11).

$$g(t) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{\frac{-(t-m)^2}{2\sigma^2}} \qquad (11)$$

For a daily economic time-series, the mean m is the centre of the time-series data. Additionally, it is apparent from (11) that smaller the value of σ, the larger is the discrimination in $g(t)$. However, for daily time-series, the smallest

value of $\sigma = 1$. So, in our present application, we kept $\sigma = 1$.

In continuous time-domain, the convolution of an infinite length time-series $x(t)$ is defined by

$$(x * g)(t) = \int_{-\infty}^{+\infty} x(\tau).g(t-\tau)d\tau \ . \quad (12)$$

In discrete domain, we replace $t = nT$ and $\tau = kT$ for integers n and k, and thus obtain (13).

$$(x * g)[nT] = \sum_{k=-\infty}^{+\infty} x[kT].g[nT - kT] \quad (13)$$

Now, we consider that the time-series $x[nT]$ has a finite length of $n = 1$ to l. Replacing the bounds of the summation in (13) by $k = 1$ to l and presuming $T = 1$ for daily time-series, we obtain (14).

$$(x * g)[n] = \sum_{k=1}^{l} x[k].g[n-k]. \quad (14)$$

It is well-known that the Gaussian filter has poor filtering performance at the beginning and at the end side of the raw signal $x[kT]$. The poor performance in filtering in the end side hampers the prediction results. This inspired us to extend the end-side of the time series by early prediction of few points using Taylor series. Then we go for Gaussian filtering followed by Taylor series prediction of the filtered data and lastly error correction by machine learning.

We now present the pseudo codes for 3 possible schemes of time-series prediction. Each scheme contains a training phase for adaptation of the PLM matrix and a prediction/test phase, where the pre-converged PLM matrix is used for prediction at finer level. In scheme-1, the training and testing algorithms are run on raw time-series data without Gaussian filtering. In Scheme-2, the training and testing are both done on filtered data. In scheme 3, the training is done on non-filtered data, but testing is done on filtered data. We do not keep provisions for a possible scheme-4 with training using filtered data but testing without filtering. This is due to the fact that the PLM matrix constructed from filtered data (with high signal to noise ratio) is unable to predict over a non-stationary time-series with good accuracy due to its relatively smaller signal to noise ratio.

For training and testing, we split the time-series data into two components, where approximately 80% of the contiguous data is used for training and remaining 20% of the data for testing.

Pseudo Code for Scheme-1: a) Training with raw time-series data

---

Input: Raw time series data $f(k^*-2), f(k^*-1), f(k^*)$, for $k^* = 3$ to l ,where k^* is the current time-point and l is the length of the time-series data;
Output: Probabilistic Learning Matrix (PLM) till it converges.

---

begin

1. Predict $\hat{f}(k^*+1) = 2.5f(k^*) - 2f(k^*-1) + 0.5f(k^*-2)$ for $k^* = 3$ to $(l-1)$;
2. Compute prediction error $e(k^*) = f(k^*) - \hat{f}(k^*)$ for $k^* = 4$ to l ;
3. Compute Scale factor $F(k^*)$ from $f(k^*+1) = \hat{f}(k^*+1) + F(k^*).e(k^*)$ for $k^* = 4$ to $(l-1)$;
4. Quantize the range of error into uniform quantized intervals: $q_1(e), q_2(e), \ldots\ldots q_n(e)$;
5. Quantize the range of scale-factor into uniform quantized intervals: $r_1(F), r_2(F), \ldots\ldots r_m(F)$;
6. Initialize: $\left[p(r_j / q_i)\right] \leftarrow \left\lceil \frac{1}{m} \right\rceil$ for $i = 1$to n and $j = 1$ to m ;
7. for $k^* = 4$ to $(l-1)$ do begin

Repeat

if error $e(k^*) \in q_u$ for $u \in [1, n]$ then $i \leftarrow u$;

if $F\left(k^{*}\right) \in r_v$ for $v \in [1, m]$ then $j \leftarrow v$;

update $p(r_j / q_i) \leftarrow p(r_j / q_i) + \delta$

also for $b = 1$to m, and $b \neq j$, update $p(r_b / q_i) \leftarrow p(r_b / q_i) - \frac{\delta}{m-1}$;

Until the change in $p(r_j / q_i)$ over last s (=50, say) learning epochs < a pre-defined bound ω (=10 × δ) for majority

(more than 70%) of the conditional probabilities;

end for;

end.

Pseudo code for scheme 1 is self-explanatory. The noteworthy point is the convergence condition. Here, to test the convergence of the PLM, we need to check whether the change in conditional probabilities over the last 50 learning epochs is insignificant for majority of the elements in the PLM. By majority we mean at least 70% of the elements of the PLM. A threshold of 10×δ is used as the maximum allowable incremental changes in the conditional probabilities in the last 50 iterations. Such threshold is selected as we allow at most 10 adaptations in the matrix elements over last 50 learning epochs for majority of the elements in the PLM to qualify the PLM to achieve the convergence condition.

It may be added here that δ should be very small (=10^{-4}), so that none of the weight matrices go below zero by N number of PLM adaptation, i.e., if initial value of the matrix elements is $p_{i,j}$=0.1 then $p_{i,j} - N \times \delta >= 0$ holds for all i,j for N=1000 and $\delta = 10^{-4}$.

Pseudo Code for Scheme 1.b) Testing with raw time-series data

---

Input: Probabilistic Learning Matrix (PLM) and raw test series data, for $f(k^*-2), f(k^*-1), f(k^*)$, for $k^* = 3$ to l, where k^* is the current time point and l is the length of the time series data;

Output: Predicted data of the test series.

---

begin

1. Predict $\hat{f}(k^*+1) = 2.5 f(k^*) - 2 f(k^*-1) + 0.5 f(k^*-2)$ at $k^* = 3$;
2. for $k^* = 4$ to $l-1$ do begin

Compute prediction error $e(k^*) = f(k^*) - \hat{f}(k^*)$;

if error $e(k^*) \in q_u$ for $u \in [1, n]$ then $i \leftarrow u$;

if PLM $(i, j^*) \geq$ PLM (i, k) for all k, then $p \leftarrow j^*$;

$a \leftarrow \frac{1}{2}\left(\lfloor r_p \rfloor + \lceil r_p \rceil\right)$ where r_p denotes quantized interval of scale factor;

$\hat{f}(k^*+1) = 2.5 f(k^*) - 2 f(k^*-1) + 0.5 f(k^*-2) + \left(a \times e\left(k^*\right)\right)$;

end for;

end.

Pseudo Code for Scheme 2: a) Training with filtered time-series data

---

Input: Raw time series data $f(k^*-2), f(k^*-1), f(k^*)$, for $k^* = 3$ to l, where k^* is the current time point and l is the length of the time series data;

Output: Probabilistic Learning Matrix (PLM) till it converges.

---

begin

1. extend training data series by 3 samples:

$f(k^*+1) = 2.5 f(k^*) - 2 f(k^*-1) + 0.5 f(k^*-2)$ for $k^* = l$ to $(l+3)$;

2. perform Gaussian filtering operation on the training data of size $(l+3)$ and get $y(1), y(2), \ldots\ldots y(l+3)$;
3. predict $f(k^*+1) = 2.5 y(k^*) - 2 y(k^*-1) + 0.5 y(k^*-2)$ for $k^* = 3$ to l;
4. Step 2 to Step 7 of Algorithm 1.a.;

end.

Pseudo Code for Scheme 2: b) Testing with filtered time series data

Input: Probabilistic Learning Matrix (PLM) and raw test series data, for $f(k^* - 2)$, $f(k^* - 1)$, $f(k^*)$, for $k^* = 3$ to l, where k^* is

the current time point and l is the length of the time series data;
Output: Predicted data of the test series.

---

begin

1. extend test data series by 3 samples:

$f(k^*+1) = 2.5f(k^*) - 2f(k^*-1) + 0.5f(k^*-2)$ for k*= l to $l+3$;

2. Perform Gaussian filtering operation on the test data of size $l+3$; and get the series: $z(1), z(2), ..., z(l+3)$;

3. predict $f(k^*+1) = 2.5z(k^*) - 2z(k^*-1) + 0.5z(k^*-2)$ at $k^* = 5$;

4. for $k^* = 6$ to $(l-1)$ do begin

Compute prediction error $e(k^*) = f(k^*) - \hat{f}(k^*)$;

if error $e(k^*) \in q_u$ for $u \in [1, n]$ then $i \leftarrow u$;

if PLM $(i, j^*) \geq$ PLM (i, k) for all k , then $p \leftarrow j^*$;

$a \leftarrow \frac{1}{2}\left(\lfloor r_p \rfloor + \lceil r_p \rceil\right)$, where r_p denotes quantized interval of scale factor;

$\hat{f}(k^*+1) = 2.5z(k^*) - 2z(k^*-1) + 0.5z(k^*-2) + \left(a \times e\left(k^*\right)\right)$;

end for;

end.

Pseudo Code for Scheme 3: a) Training with raw time series data: same as Algorithm 1.a).
Pseudo Code for Scheme 3: b) Testing with filtered time series data: same as Algorithm 2.b).

---

4. Experiments and performance analysis

In this section, we would undertake 3 experiments. The first experiment is concerned with examining the performance of the proposed time-series prediction techniques for 3 different strategies introduced earlier. It may be remembered that the strategies proposed primarily examines the essence of Gaussian filtering in training and test cycles. The second experiment is designed to compare the relative performance of the proposed prediction algorithms with the state-of-the-art algorithms. The third experiment makes an attempt to give an interpretation of both the strength and weakness of the proposed algorithms, in order to identify the possible types of non-stationarity, where the proposed techniques are better/worse than the existing techniques.

To undertake the experiments indicated above, we used an experimental framework, comprising 7 time-series databases, 6 state-of-the-art algorithms for time-series prediction, and RMSE as the single metric to evaluate the performance of a prediction algorithm. Table-1provides the details of the time-series databases used, sample size and training and test data components used for the said experiments. Table-2 provides the list of algorithms/techniques used to prepare the comparative framework. In this Table, we present the algorithms used, the list of their parameters, their optimal value chosen in the present experiments and also the results of performance of the individual algorithm with respect to the RMSE metric. A description of the algorithms used along with its parameters is also given separately. Besides presenting the RMSE measures in Table 2, the visualization of the performance of the proposed and the existing techniques are graphically presented along with the raw time-series plot.

All the experiments presented here are performed using MATLAB 10 running under Windows 10 operating system on 3 GHz Intel Xeon dual-core processors enhanced with 4 MB advanced transfer L2 cache (2 MB per core).

4.1 Performance metric used

Let $\hat{y_t}$ be the predicted value, y_t be the observed value of that sample and n be the total number of forecasted samples.

RMSE: We use root-mean-square deviation or root-mean-square error (RMSE) to measure the differences between predicted value of the time series and the value actually observed. RMSE is computed by (15).

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y_t} - y_t)^2}{n}} . \quad (15)$$

4.2 Data sets used

The data sets used include FTSE-100 daily close price, Facebook daily closing stock price, SP 500 daily close price, N225 daily close price, Monthly gold price, Brent oil daily close price and TAIEX daily close price. In Table-2, column 2, 3, 4 and 5 respectively indicate the time-series database used, sample-size, training data size and test data size. For example, for FTSE-100 daily close price database, the sample-size is 7788 data, taken over the period January 1984 to October, 2013. Training data is taken from available FTSE-100 daily price data for the period 03/01/1984 to 29/10/2013, while test data set is taken from the same database for the period: 30/10/2013 to 30/10/2014. Rest of the attributes in the remaining 6 rows of Table-1 can be explained easily.

Table1: Data composition of different time series

Serial Number	Time-Series	Sample-size	Training Data Set	Test Data Set
1	FTSE-100 daily close price	Jan,1984 – Oct,2013 (7788 data)	03/01/1984 -onwards 3192 data	30/10/2013- 30/10/2014
2	Facebook daily closing stock price	21st May,2012 - 3rd Sept,2014(573 data)	472 data	Last 97 data
3	SP 500 daily close price	16445 data	08.02.1950 – 05.07.14 (16198 data)	03.07.14-30.06.16 (250 data)
4	N225 daily close price	7750 data	13.01.84-01.07.14 (7497 data)	07.07.14-30.06.15 (250 data)
5	Monthly gold price	1990-04 to 2008-07 (220 data)	First 200 data	Last 20 data
6	Brent oil daily close price	22nd June,1988- 2nd April,2018 (7589 data)	First 6069 data	Last 20% data
7	TAIEX daily close price	1990-1999 (2824 data)	1991-1999 January onwards (2322 data)	1991-1999 November and December (499 data)

4.3 State-of-the-Art Algorithms used for the present comparative study

The state-of-the-art algorithms we used to develop the comparative framework includes ARIMA [10]-[13], hybrid Genetic algorithm-based GA-WA model [26], Simple Averages (SA) [37], Bat Neural Network (BNN) [35], Sparse Bayesian Learning (SBL) [36], and, Taylor series and Artificial Neural Network (ANN) [38]. A brief description about the algorithms is presented below, with emphasis to their parameter sets.

1. **ARIMA** [13]: ARIMA stands for a class of models that describes the current time-point value of a given time series as a linear combination of its own past values (i.e., its own lags), and the lagged forecast errors. Usually, the parameter *p* is used to describe the order (number of time-lags) of the autoregressive model and *d* is used to represent the degree of differencing (i.e., the number of times the data have had its past values subtracted) to eliminate the effect of trends, and *q* is the order of the moving-average model.

2. **GA-WA** [26]: GA-WA is a hybrid approach that utilizes Genetic Algorithm (GA) as an optimizer to determine the optimal weights for each of the prediction models: BPNN, SVM and SVR. The parameters

used in BPNN include number of nodes in first hidden layer, number of nodes in the second hidden layer, λ = learning rate and m = momentum constant. The parameters used in SVR/SVM are: c = regularization constant, γ =user-defined positive parameter and Q factor = quantization Factor.

3. **BNN** [35]: Bat neural network (BNN) is equipped with Bat algorithm to reinforce training phase of an Artificial Neural Network. Here the parameters used are: (i-h-o) = (input nodes-hidden nodes-output nodes) = (7-4-1); p_n =Number of population, g_n=number of generations; λ = wavelength, α = cooling factor, f_{min}=minimum pulse frequency and f_{max}= maximum pulse frequency.

4. **SBL** [36]: The parameters used in SBL are: s = the value of the scalar trade-off parameter for balancing sparsity, i_{max} = maximum number of iterations.

5. **Taylor Series and ANN** [38]: Mangai *et al.* proposed a forecasting technique based on Taylor series, comprising eight coefficients, error propagation using Artificial Neural Network (ANN) and used that technique for prediction of sunspot number. Here the parameters used are only that of ANN: (i-h-o) = (input nodes-hidden nodes-output nodes).

6. **SA** [37]: It is a simple moving average method and therefore does not have any specific parameter.

The optimal parameters of the algorithms used in our test runs are listed in Table-2 along with RMSE obtained for each algorithm.

Table 2: Parameter settings for the best performance of different time-series along with obtained RMSE

Data Set	Algorithm	Model Specifications	RMSE
Daily closing stock price of FTSE-100	ARIMA	(p, d, q): (1, 0, 0)	156.68
	GA-WA	BPNN:(20,5), $\lambda = 0.3, m = 2$ SVR: $c = 100, \gamma = 0.0001, QF = 100$ SVM: $c = 100, \gamma = 0.0003, QF = 100$	45.47
	SA		49.10
	BNN	ANN: $(7, 4, 1)$, p_n: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	20.89
	SBL	$s = 0.0004, i_{max} = 600$	39.64
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	19.86
	Proposed Algorithm: 2	Scale-factor Range: [-3, 3], Error Range: [-300,300], PLM Size: 12×12 , $l = 3192$	16.90
Daily closing stock price of Facebook	ARIMA	(p, d, q): (1, 0, 0)	2.17
	GA-WA	BPNN:(10,5), $\lambda = 0.3, m = 2$ SVR: $c = 100, \gamma = 0.0001, QF = 100$ SVM: $c = 100, \gamma = 0.0002, QF = 100$	1.58
	SA		1.29
	BNN	ANN: $(7, 4, 1)$, p_n: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	1.77
	SBL	$s = 0.0003, i_{max} = 400$	1.55
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	0.81
	Proposed Algorithm: 3	Scale-factor Range: [-5, 5], Error Range: [-11, 11], PLM Size: 8×4, $l = 472$	0.39

Daily closing stock price of S&P500 Index	ARIMA	(p, d, q): (1, 0, 0)	30.67
	GA-WA	BPNN:$(10,5), \lambda = 0.3, m = 2$ SVR: $c = 100, \gamma = 0.0001, QF = 100$ SVM: $c = 100, \gamma = 0.0003,\ QF = 100$	14.80
	SA		15.53
	BNN	ANN: $(7,4,1),\ p_n$: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	14.46
	SBL	$s = 0.0003, i_{max} = 400$	13.33
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	13.81
	Proposed Algorithm: 3	Scale-factor Range: [-10, 10], Error Range: [-31, 34], PLM Size: $13 \times 7, l = 16198$	6.31
Daily closing stock price of Nikkei 225 Index	ARIMA	(p, d, q): (1, 0, 0)	201.59
	GA-WA	BPNN:$(10,5), \lambda = 0.7, m = 10$ SVR: $c = 100, \gamma = 0.0003, QF = 100$ SVM: $c = 90, \gamma = 0.0002,\ QF = 10$	185.37
	SA		192.06
	BNN	ANN: $(7,4,1),\ p_n$: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	114.46
	SBL	$s = 0.0003, i_{max} = 400$	182.22
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	88.09
	Proposed Algorithm: 3	Scale-factor Range: [-3, 3], Error Range: [-300, 300], PLM Size: $7 \times 7, l = 7497$	71.22
Monthly observation of Gold Price	ARIMA	(p, d, q): (1, 1, 0)	166.89
	GA-WA	BPNN:$(10,5), \lambda = 0.7, m = 10$ SVR: $c = 100, \gamma = 0.0004, QF = 100$ SVM: $c = 100, \gamma = 0.0001,\ QF = 10$	87.23
	SA		162.06
	BNN	ANN: $(7,4,1),\ p_n$: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	21.26
	SBL	$s = 0.0003, i_{max} = 400$	74.37
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	25.68
	Proposed Algorithm: Taylor Series Approximation	-	18.47
Brent Oil	ARIMA	(p, d, q): (1, 0, 0)	1.21
	GA-WA	BPNN:$(10,5), \lambda = 0.7, m = 10$ SVR: $c = 100, \gamma = 0.0004, QF = 100$ SVM: $c = 100, \gamma = 0.0001,\ QF = 10$	0.93
	SA		2.06

Spot Price	BNN	ANN: $(7,4,1)$, p_n: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	0.89
	SBL	$s = 0.0004, i_{max} = 600$	1.19
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	0.85
	Proposed Algorithm: 3	Scale-factor Range: [-1, 1], Error Range: [-5, 5], PLM Size: $7 \times 7, l = 6069$	0.79
Daily Close Price of TAIEX	ARIMA	(p, d, q): (1, 0, 0)	123.67
	GA-WA	BPNN:$(10,5), \lambda = 0.7, m = 10$ SVR: $c = 100, \gamma = 0.0004, QF = 100$ SVM: $c = 100, \gamma = 0.0001,\ QF = 10$	87.22
	SA		102.06
	BNN	ANN: $(7,4,1)$, p_n: $15, g_n = 5, \lambda = 1.5, \alpha = 0.5$; $f_{min} = 0, f_{max} = 1$	79.99
	SBL	$s = 0.0004, i_{max} = 600$	81.70
	Taylor Series and ANN	Taylor series: up to 8 coefficients. ANN: (7,4,1)	77.75
	Proposed Algorithm: 3	Scale-factor Range: [-5, 5], Error Range: [-700, 700], PLM Size: $6 \times 6, l = 2322$	53.44

Experiment 1: Performance of the Proposed 3 Schemes

In this experiment, we examine the importance of Gaussian filter in Schemes 1, 2 and 3 introduced in the previous section. The experiments are undertaken for all the test data set presented in Table-1. However, for the sake of convenience and space limitation, we present the comparative study with respect to TAIEX time-series data, presented in Fig. 2 below. It is apparent from the plots in Fig. 2 that Scheme 1 (proposed Algorithm 1), which did not use Gaussian filter both in the training and test phase incurs large errors at many prediction steps. Particularly, errors are noticeable in large slope change points on the original data points. However, such errors do not appear when Scheme-2 (Algorithm 2), that utilizes Gaussian filter during training (PLM construction) and testing, is employed. It is also observed from Fig. 2 that the prediction error is reduced further, when scheme-3 that introduces no filtering in the training phase but a Gaussian filter during the testing phase is employed.

The justification in performance of the 3 techniques is apparent for the following reason. Without Gaussian filter in the training phase of Scheme-1, the Taylor series-based prediction yields large error during sharp changes in slope in the time-series. Additionally, the PLM is trained with noisy data. Naturally the trained PLM when used to predict an unknown noisy segment of the same time-series yields relatively large error almost everywhere (during large mean changes, large variance changes and trend changes) in the series because of the presence of random noise distribution around the time-series data points. It is important to mention here that the noise distribution of the trained and tested time-series may have difference, in case the segment of time-series predicted is not a part of the time-series used for training.

As scheme 2 employs Gaussian filter both in the training and test phases, the effect of noise distribution is not prominent in prediction. The PLM seems to have correctly encoded the error band to scale factor band mapping. Further, scheme 3 yields the best results in prediction performance as training is done on noisy data, but prediction is done on filtered time-series data. Because of low noise amplitude in the time-series during the prediction phase, Taylor series produces less noise, and the PLM eliminates the small noise amplitude, whatever is produced by Taylor series-based prediction. Table 3 provides the results of RMSE for the 3 schemes when applied to a small segment of the TAIEX time series. The optimal PLM-size was found to be (6×6) with a training set of l= 2304 data points. The RMSE is the least for scheme 3, and is relatively larger in scheme 2 and the worst in scheme 1.

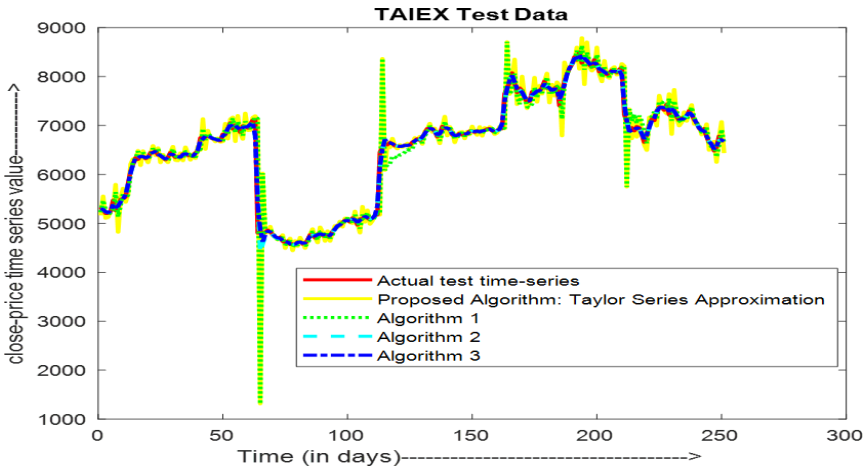


Fig. 2 TAIEX Time-series prediction (plotted from 200th data point to 450th data point in the selected year) representing the actual time series, and predicted time series obtained by Taylor series approximation, Pseudo codes 1, 2 and 3

Table 3: Results of Experiment 1

Specifications					**RMSE**
Scheme used	**Error range**	**SF Range**	**band-width, d-width**	**Size, *l***	***by prediction with Taylor series only (by Taylor series and PLM)***
1: Training and Testing without Gaussian Filter	[-702,702]	[-6,6]	234, 2	6 × 6, 2304	338.04 (318.49)
2: Training and Testing with Gaussian Filter	[-396,396]	[-6,6]	132, 1	6 × 6, 2304	79.40 (69.19)
3: Training without and test with Gaussian Filter	[-702,702]	[-6,6]	234, 2	6 × 6, 2304	79.40 (53.44)

Experiment 2: Comparative Performance of the Proposed Schemes with the State-of-the-art Algorithms

This section provides a through comparative study of the proposed schemes with the existing techniques presented in Table 2. Fig. 3-16 provide the prediction results for visual examination in competitive performance of the proposed techniques, and also the relative performance of the proposed techniques with the state-of-the art techniques listed in Table 2. In general, a noticeable performance of the proposed techniques is observed in Fig. 4, 6, 8, 10, 12, 14 and 16, where irrespective of variation in mean, variance and trends in the time-series, the predictions performed by the proposed techniques outperform the existing techniques significantly. The relatively better performance of the proposed techniques also is evident from the RMSE measures given in Table 2.

In Table 2, we consider the Taylor series approximation technique, Scheme 2 and scheme 3 because they have shown remarkable performance in the prediction of selected time-series. For example, in *Gold price test data* prediction, we used the basic Taylor series prediction model. This is due to the reason that we had only 200 data points, and so with fewer data points the PLM would hardly converge. However, even with only Taylor series prediction model, the prediction accuracy is good enough, yielding an RMSE of 18.47. The plots given in Fig. 5 and 6 also indicate that the Taylor series model outperforms the existing techniques.

It is noted from Table 2 that for *Daily closing stock price of FTSE*-100 database, scheme 2 yields good performance with an RMSE = 16.90. The result is also evident from the prediction results presented graphically in Fig. 15 and 16. It is important to mention here that the FTSE-100 time-series data has large changes in variance with more or less uniform mean. Our proposed scheme 2 could easily track the large changes in variance, and thus may be selected for prediction of similar time-series.

In Fig. 3 we examine the relative performance of the proposed 3 prediction algorithms for TAIEX time-series, and in Fig. 4 we present the relative performance of our proposed algorithm 3 with the state-of-the art algorithms for prediction of the same series. It is apparent from Fig. 4 that the proposed algorithm 3 outperforms all the considered algorithms in RMSE measure. In addition, it is noteworthy that there is hardly any error spike at the sharp 90^{O} changes in the time-series. In other words, the errors originated at the high slope changing points by Taylor series prediction could be eliminated in the second phase of prediction by scheme 3.

A thorough inspection of Fig. 3 to 16 and Table 2 reveals that the proposed techniques can outperform the present state-of-the-art techniques by a considerably large margin of RMSE measures, irrespective of the stochasticity of the series. However, identifying the right technique for a given time-series needs trial predictions with prior training of the time-series. In general, for a time-series containing all the four types of non-stationarity, scheme 3 is the best. In case the variance-change is large throughout the time-series, scheme 2 is preferred. Additionally, if there are fewer data points, Taylor series is the right option, as training with fewer data points is not feasible.

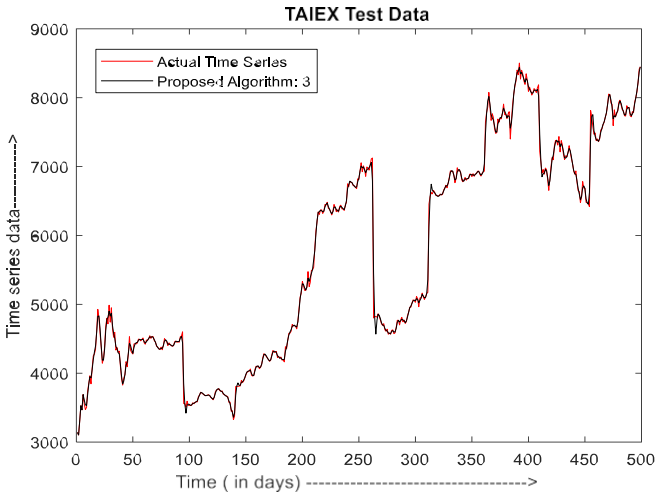


Fig. 3 The actual TAIEX test data and the predicted time-series using the proposed scheme-3

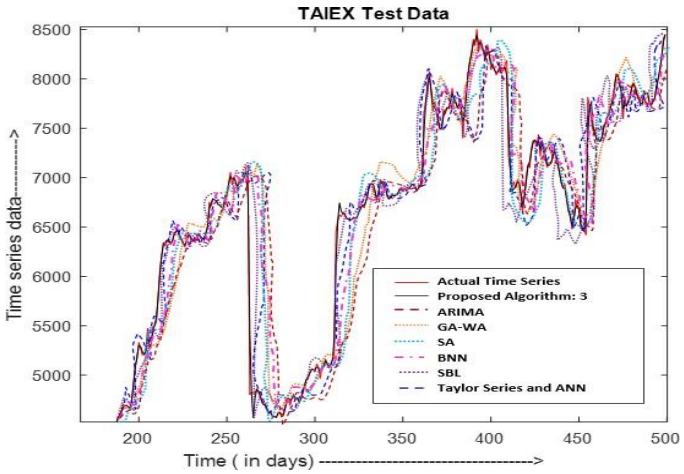


Fig. 4 Comparative analysis of the proposed algorithm with the existing algorithms for time-series prediction on TAIEX test data

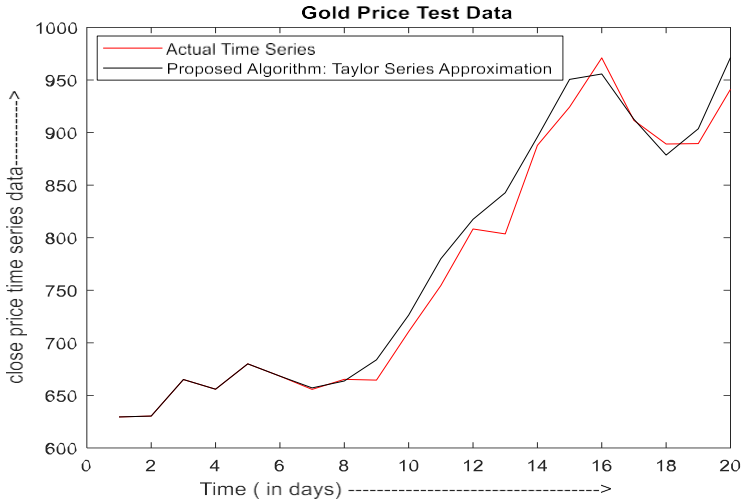


Fig. 5 The actual Gold price test data and the predicted time-series using the proposed Taylor-series approximation

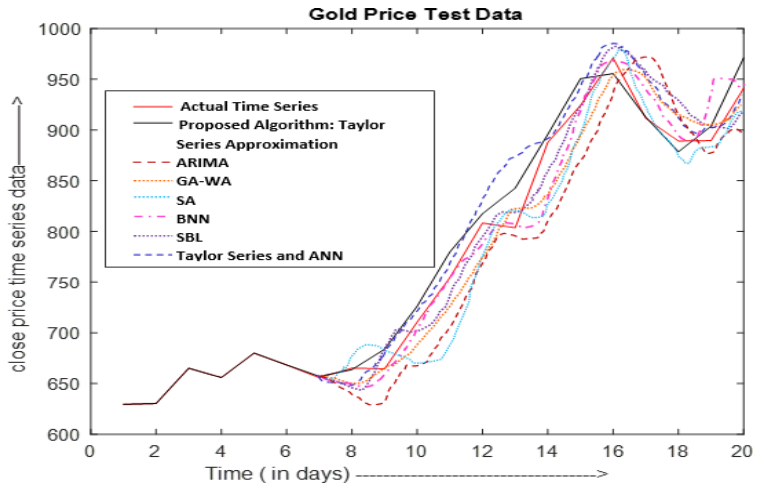


Fig. 6 Comparative analysis of the proposed Taylor-series approximation with the existing ones for time-series prediction on Gold price test data

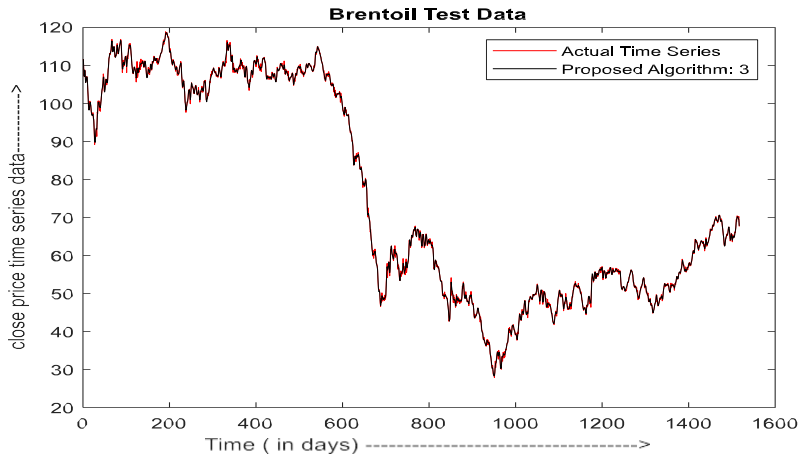


Fig. 7 The actual Brent oil test data and the predicted time-series using the proposed algorithm 3

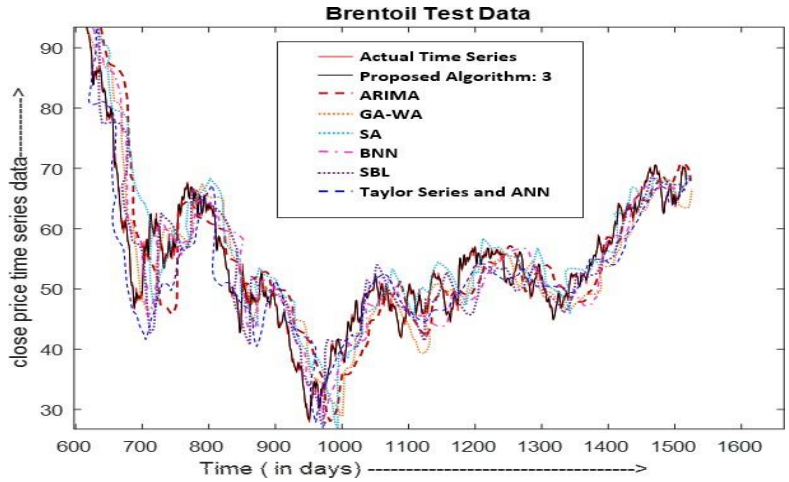


Fig. 8 Comparative analysis of the proposed scheme 3 with the existing algorithms for time-series prediction on Brent oil test data

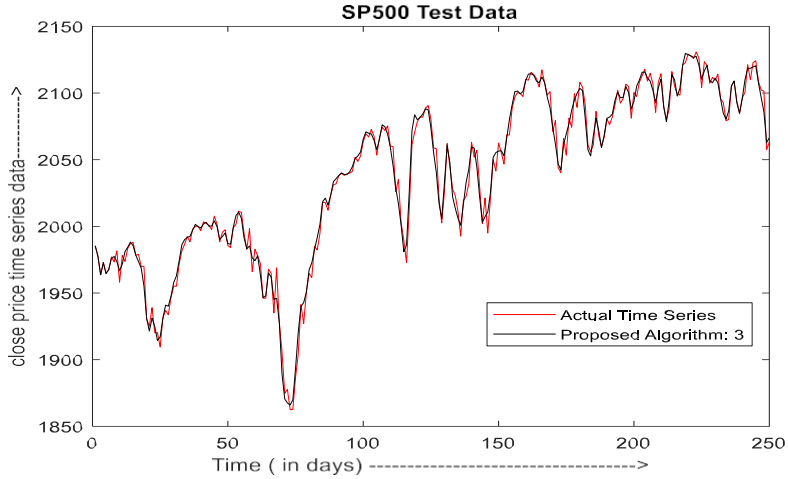


Fig. 9 The actual SP500 test data and the predicted time-series using the proposed **scheme-3**

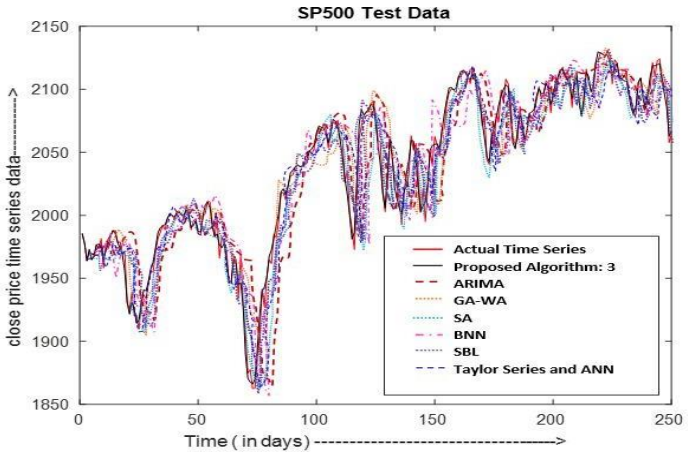


Fig. 10 Comparative analysis of the proposed scheme-3 with the existing algorithms for time-series prediction on SP500 test data

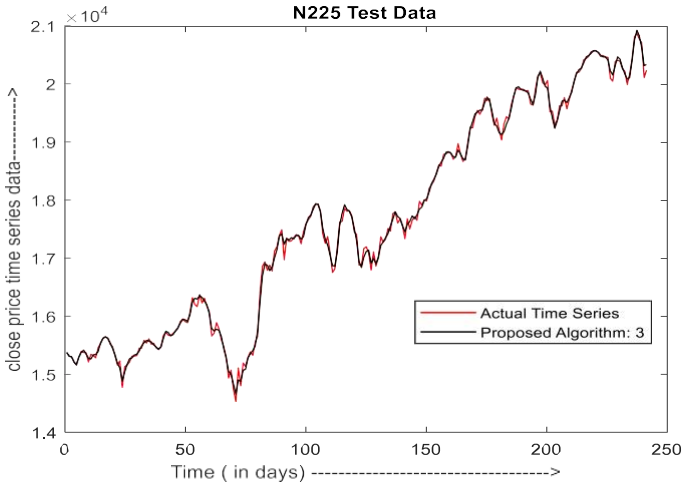


Fig. 11 The actual N225 test data and the predicted time-series using the proposed scheme-3

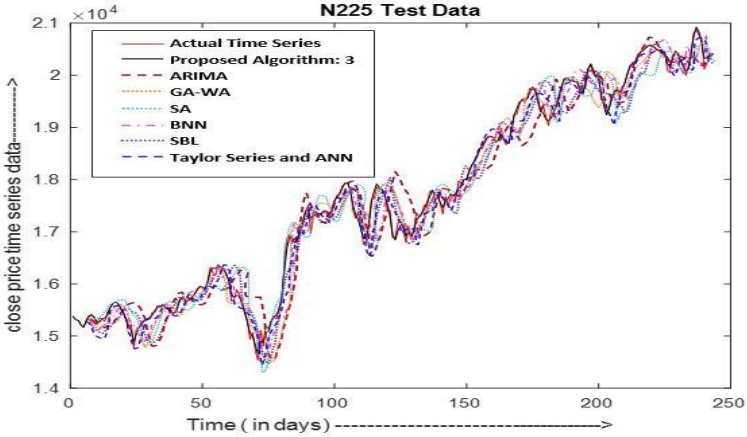


Fig. 12 Comparative analysis of the proposed scheme-3 with the existing algorithms for time-series prediction on N225 test data

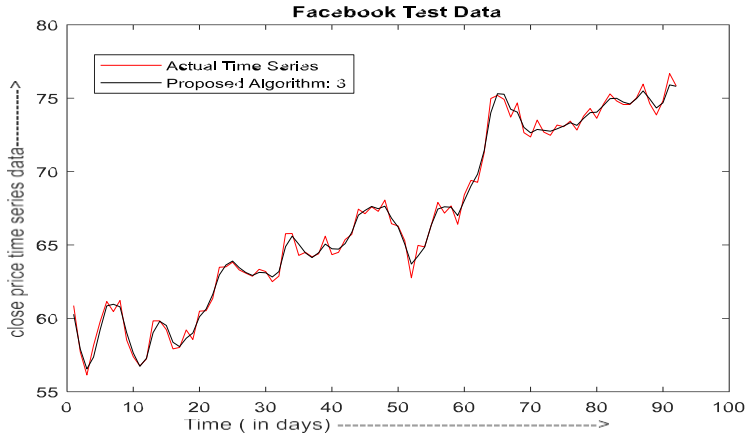


Fig. 13 The actual Facebook test data and the predicted time-series using the proposed algorithm 3

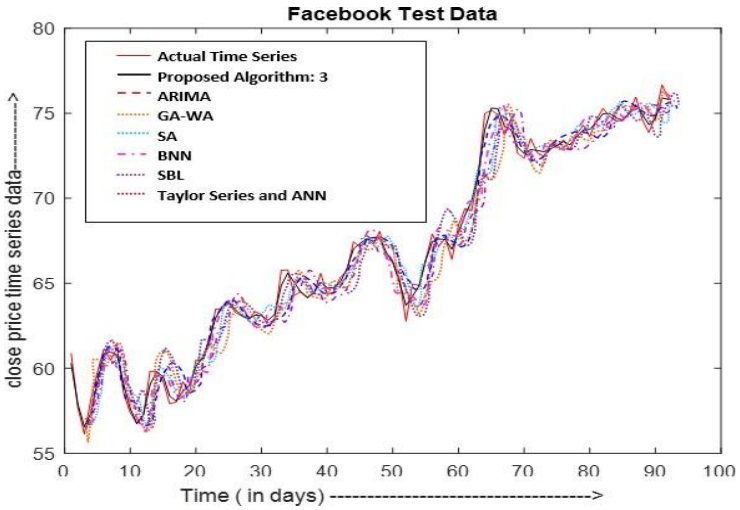


Fig. 14 Comparative analysis of the proposed scheme-3 with the existing algorithms for time-series prediction on Facebook test data

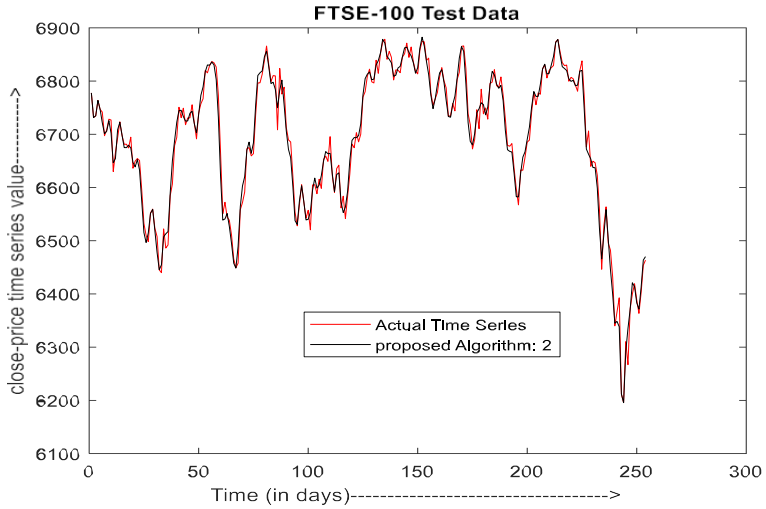


Fig. 15 The actual FTSE-100 test data and the predicted time-series using the proposed scheme-2

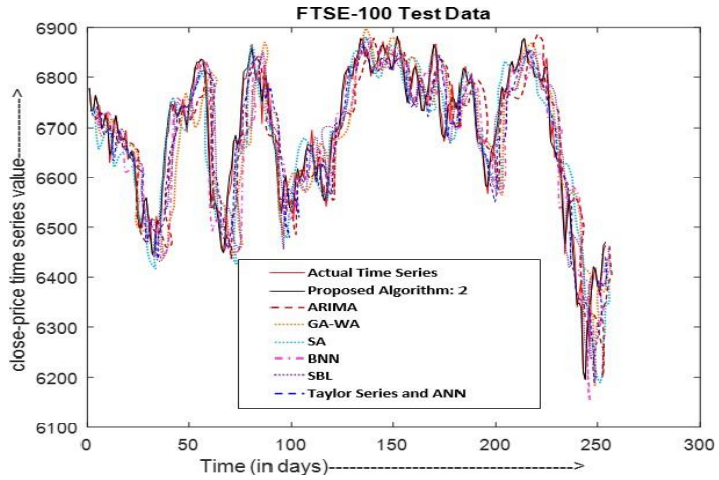


Fig. 16 Comparative analysis of the proposed scheme-2with the existing algorithms for time-series prediction on FTSE-100 test data

Experiment 3: Attempt to identify possible failures of the proposed schemes

This experiment is concerned with identifying the possible failures in prediction in presence of different types of non-stationarity in the given time-series. One approach to undertake the experiment is to examine the prediction performance of the proposed techniques on a set of time-series containing one or more types of non-stationarity, including large changes in mean, variance, both mean and variance simultaneously and trend-changes. Fig. 17 includes a TAIEX time-series predicted by Taylor series approximation, and schemes 1, 2 and 3 presented in Section 3. It is noteworthy that the given time-series includes mean-change on 17th, 62nd and 83rd days. So, it has constant mean during the interval of (1, 16), (63, 82) and (84, 100) days. The time-series also includes variance change during (62, 83) days interval, and trend change on the days: 17 and 62. So, it includes most of the non-stationarities. The following observations are obtained from the experimental plots of Fig. 17.

1. Taylor series-based prediction yields large error during the prediction of 90^{o} slope changing points. These 90^{O} slope changing points fall between the interface of 2 trends/structures [2], during the days 17 and 62. Once the algorithm crosses this large slope changing point, it remains in the new structure, and prediction error becomes low. So, the error correction is high even after producing large error due to large change in slope.
2. Fortunately, for large variance and mean changes in the said intervals can easily be predicted by Taylor series prediction.
3. As scheme 1 is examined on noisy time-series data (without using any filter), the scheme cannot rectify the Taylor series produced errors in prediction.
4. However, scheme 2 can easily rectify the errors obtained from Taylor series based prediction because of learning of the PLM from noise-free (filtered) data samples.
5. Further scheme 3 yields prediction with high accuracy because of its additional characteristic of learning the error-scale factor mapping in the PLM from noisy data samples, and then predicting a filtered time-series.

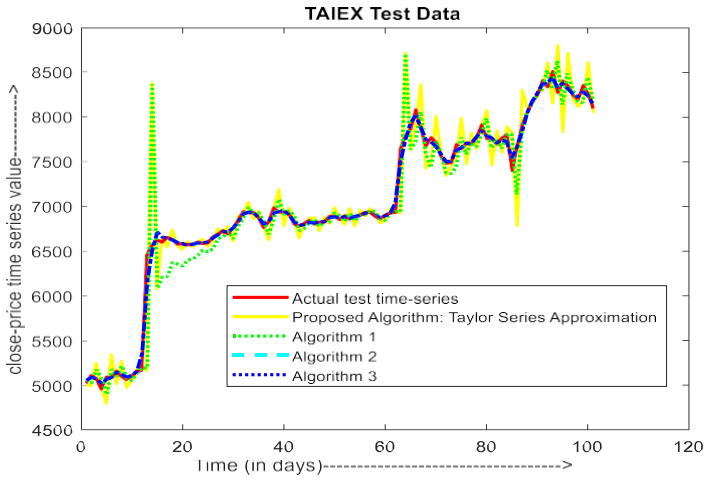


Fig. 17 Identification of possible failures in prediction of TAIEX time series from 300th to 400th test data points in presence of different types of non-stationarity

So, the proposed schemes are capable of predicting time-series with the above-mentioned types of non-stationarity. However, we need to test which of the 4 methods: only Taylor series, scheme 1, 2 and 3 is the best choice for a given time-series.

5. Statistical validation of the Proposed Algorithms

Statistical tests have been undertaken to test whether the relative difference in performance (RMSE) of 2 or more algorithms is statistically significant. Two non-parametric statistical tests, namely Combined 5 by 2 paired t-test [35] for single database and Friedman test [36] for multiple databases are considered to extensively carry out the experiments.

5.1 Combined 5 by 2 paired t-test

Let A be the reference algorithm (here, proposed algorithm 3) and B be any one of the 8 algorithms listed in Table-7. To validate the null hypothesis H_0, which states that the relative difference in performance of A and B is not significance, we evaluate *t*-measure, given by

$$\tau = \frac{m_A - m_B}{\sigma_A^2 + \sigma_B^2} \qquad (16)$$

where m_A and m_B are the mean values of the distributions of RMSE values obtained by algorithm A and B respectively, and σ_A and σ_B are the standard deviations of the distributions of RMSE values obtained by algorithm A and B respectively.

The results of the combined 5 by 2 paired t-test for each of the 7 databases are provided in Table-7, considering the proposed method 3 as the reference algorithm. The plus (minus)sign represents that the difference in the RMSE metric of algorithm A and B is significant (not significant). Here, 95% confidence level is achieved with the degree of significance, at the level of 0.05.

Further, it is apparent from Table-7 that for most of the databases, the combined 5 by 2 paired t-test cannot distinguish the relative performances of the 4 regression algorithms: Proposed algorithm 1 and 2, BNN and Taylor series & ANN that support the null hypothesis. Therefore, another non-parametric statistical test, Friedman test is introduced to evaluate the rank of each algorithm.

Table 4. Statistical Validation using Combined 5 by 2 paired t-test with 7 individual datasets

Algorithms for comparison	Reference method: Proposed algorithm 3						
	FTSE-100 Database	Facebook Database	S&P500 Database	Nikkei 225 Database	Gold Price Database	Brent Oil Spot Price Database	TAIEX Database
Proposed algorithm 1	-	-	-	-	-	-	-
Proposed algorithm 2	-	-	-	-	-	-	-
Taylor series and ANN	-	+	-	-	-	-	-
BNN	-	-	+	-	-	-	-
SBL	+	+	-	+	+	+	+
SA	+	-	+	+	+	+	+
GA-WA	+	+	+	+	+	+	+
ARIMA	+	+	+	+	+	+	+

5.2 Friedman Test

A non-parametric Friedman 2-way statistical test is employed here to validate the relative performance of the proposed 3 schemes/algorithms over the state-of-the-art algorithms. According to the Friedman 2-way statistical test, each of the 8 algorithms is ranked based on the median of the RMSE values for the 7 datasets, as reported in Table-8. The null hypothesis, H_0 is defined here as: there is no significant difference between the medians of the RMSE values obtained by each of the 8 algorithms for all the 7datasets. Based on the null hypothesis, the Friedman's statistic score χ_F^2 with degree of freedom (k-1) is defined as follows

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{i=1}^{k}\left(\frac{1}{7}\sum_{j=1}^{7} r_i^j\right)^2 - \frac{k(k+1)^2}{4}\right] \quad (17)$$

Where k is the number of algorithms to be compared (here, 8), N is the number of datasets (here, 7). Here r_i^j denotes the rank of i-th algorithm for the j-th dataset. The average ranks of all the 8 algorithms with respect to the respective RMSE values, computed by Friedman test, are depicted in Table-8. The smaller average rank indicates better performance of the algorithm. Using the number of experiments, N = 7 and number of time-series prediction algorithms to be compared, k = 8, we obtain χ_F^2 with respect to the RMSE values, which is greater than $\chi^2_{\{7,0.05\}} = 12.592$. Thus, the null hypothesis, which states that the 8 algorithms are equivalent, is rejected at 7 degrees of freedom at probability of 0.05.

Table 5. Statistical Validation using Friedman Test

Performance Metric		Prediction Algorithms							
		Proposed algorithm 2	**Proposed algorithm 3**	ARIMA	GA-WA	SA	BNN	SBL	Taylor Series and ANN
	Average Rank over 7 datasets	2.06	**1**	7.92	5.56	7.21	4.28	6.77	3.14

RMSE	Friedman Statistics χ_F^2	**83.14 (Null Hypothesis rejected)**

6. Discussion

This section explains the possible justification of better relative performance of the proposed technique with respect to the state-of-the-art techniques in time-series prediction. It may be recounted that the proposed prediction technique has two main components: i) Taylor series based coarse forecasting of the time-series, and ii) PLM based error correction/reduction for finer prediction. It is apparent that Taylor series-based prediction is advantageous to traditional prediction algorithms for its inherent merits of using slopes in the prediction equation. Slope-based prediction algorithms are advantageous to their non-slope-based competitors, as the former can recognize small changes in slope, thus leaving no/small chances to miss linear (rising/falling) trends in the series.

In fact, for small changes in slope (of the predicted point with respect to the current time point), the proposed Taylor series-based prediction technique works exceptionally well. However, (the single-point) prediction performance degrades, if the change in slope of the predicted point with respect to the current point is large (close to 90^{o}). So, the algorithm fails to predict the next point once when there is a change in structure (due to large slope change) [2], but regains its performance in the prediction of the next time-point value (as the prediction now takes place on the new structure). So, the chances of 2 or more successive failures by Taylor series are rare. Existing algorithms such ARIMA and ANN based prediction techniques, occasionally commit errors in successive steps of predictions.

Additionally, in non-stationary time-series, there are high chances of structural changes [2] at several time-points on the series. Consequently, taking several past values (as done in ARIMA and time-delayed NN) may not always have any tangible contribution in the prediction. Rather, due to possible structural change just marginally before the time-point of prediction, the past sample values taken may introduce errors in the prediction of the next time- point value. So, in the proposed Taylor series-based formulation, only the current and the last two sample values are considered for the prediction of the next time point value. It may be recollected that in [38], the authors considered first 8 terms of Taylor series expansion, costing large run-time complexity with no real gain.

The PLM based prediction learns the selection of scale factors for bands of quantized errors. The convergence in the PLM matrix ensures that the mapping from quantized error bands to quantized scale factor bands is complete, and the mappings can now efficiently be utilized for prediction. The optimal choice of grid-size is an important issue as the optimal choice effectively yields the smallest RMSE. Naturally, the optimal PLM matrix carries valuable knowledge to utilize the optimal error to scale factor mapping in the prediction phase. It is noted that the errors committed by Taylor series-based prediction particularly at the time-points of structural changes (large slope change), often is corrected by error band to scale factor mapping encoded by the PLM matrix, thereby improving prediction accuracy due to inclusion of the error corrective step.

It is noteworthy that out of 3 schemes proposed, scheme 3 includes a training session without smoothing, but the prediction session requires a Gaussian smoothing of the time-series to yield the best results in prediction accuracy of the time-series. Without Gaussian smoothing of the time-series in the prediction phase, the proposed PLM-based finer prediction, does not yield good performance, as the noise characteristics in the test data differs significantly from the training data in a non-stationary time-series. Naturally, the PLM encoding undertaken during the training phase cannot help in error correction in the prediction phase due to the variation in the noise characteristics in the unknown segment of the non-stationary time-series.

7. Conclusion

The paper introduced a novel 2-step approach for time-series prediction. In the first step, an approximate prediction of the series is undertaken by a Taylor series formulation, while in the second step, a reinforcement learning inspired probabilistic learning technique is employed to rectify the prediction errors. The Taylor series-based prediction yields an exceptional performance even in presence of dynamic variation in mean and variance in the series, but yields error spikes at 90^O slope changing points on the time-series. Such large slope change generally takes place during structure/trend changes in the time-series. The probabilistic learning model proposed, significantly reduces the prediction errors, and thus works in anticipatory modality with the Taylor-series based prediction.
Noise plays an important impediment to time-series prediction. To eliminate the effect of noise in the prediction problem, the paper introduces a Gaussian filter before to invoke the learning and prediction phases. It is noticed that in presence of the filter, the prediction performance improves significantly. Three different schemes of prediction algorithms have been proposed, by considering the employment of Gaussian filter in the training and prediction phases. Of these 3 schemes, the one having training without Gaussian filter and prediction with Gaussian filter yields the best performance in prediction. Experiments undertaken on different time-series databases confirm that the proposed techniques outperform all its competitors by a statistically significant difference in RMSE measure.

8. Appendix

Selection of the Optimal PLM
Given a time-series of l data points, and let the PLM be of ($n \times m$), where n, m <l. Here, we present two pseudo codes, the first one to construct a feasible set of tentative PLM matrices, while the second one is to select the optimal PLM matrix among the feasible choice of PLMs in order to minimize the RMSE metric. After selection of the optimal PLM for a given time-series, the same matrix can be used for prediction on the time-series.

Pseudo Code to Build PLMs

---

1. **Compute Errors at l time-points using Taylor series:** Run the Taylor series program on a time-series of l data points, and compute error at each time point. Save errors in ascending order of time in a one-dimensional array A= [a_i] of length l. Also obtain the minimum and the maximum errors, called error-min and error-max respectively by 2 linear searches in the array A.

2. **Quantize errors:** Divide the difference (error-max minus error-min) into n intervals of equal size.

3. Computation and saving of scale factors in an array: For each a_i, i= 1 to l, obtain scale factor F(k*), satisfying equation (9). Save scale factors in ascending order of time in a one-dimensional array B=[b_i] of length l. Obtain minimum and maximum scale factors by again 2 linear searches in the array B. Get the range of scale factor by taking difference of maximum and minimum scale factors, and divide the range into equal m intervals.

//Note that Array A and B have correspondence, as the i-th element of A returns the i-th element of B by Taylor series prediction. //

4. **Window-size in the PLM:** Let the tentative PLMs be of size ($n \times m$), such that for all feasible n, m<l, and modulo (l/m)=0 and Modulo (l/n)=0, indicating that n and m are factors of l. The widow size is automatically fixed by equal-width partitioning of the range of errors and the range of scale factors. Compute window-height=l/n, and window-width= l/m.

5. **Initialization of PLM:** Suppose modulo (l/m)=0 and Modulo (l/n)=0 occur due to p possible choice of m and q possible choice of n. Then total number of tentative PLMs= D= $p \times q$. Initialize D PLMs of variable sizes ($n \times m$) in a manner, such that all elements in each row are equal, and row-sum=1.

6. **Widow filling:** For each i = 1 to l, find q_k and r_j, such that a_i and b_i fall in window (k, j), where k=

Modulo(n/widow-height) and j=Modulo(m/widow-width) in all the D PLMs. Update $p(q_k / r_j)$ and $p(q_k / r_s)$ for all s, except s=j following probability update equations in the PLM matrix.

7. End.

Pseudo code for optimal choice of PLM

---

1. Predict the data points, one by one in a user-provided segment of the selected time-series using Taylor series followed by PLM matrix-based prediction. Repeat this for D PLM matrices one by one.

2. Compute RMSE obtained for each used PLM. Find the PLMfor which the RMSE is the smallest. Return the selected PLM.

References

[1] A. Konar, and D.Bhattacharya, *Time-series prediction and applications*, Springer, 2017.
[2] D. Bhattacharya, J. Mukhoti, *and* A. Konar*:* "Learning regularity in an economic time-series for structure prediction," *Appl. Soft Comput.*vol. 76, *pp. 31-44, 2019*
[3] L. Ghosh, A. Konar and P, Rakshit, Cognitive Modelling of Human Memory and Learning: A Brain-Computer Interface Approach, Wiley-IEEE Press, October, 2020 (to appear).
[4] G. E.Box, G. M.Jenkins, G. C.Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*, *John Wiley & Sons*, 2015.
[5] D. Bhattacharyand A. Konar, *and P. Das, "Secondary factor induced stock index time-series prediction using Self-Adaptive Interval Type-2 Fuzzy Sets," Neurocomputing*, vol. 171, *pp. 551-568, 2016.*
[6] Scarborough, J. B., *Numerical Mathematical Analysis*, Oxford and IBH Publishing, 1955.
[7] E. Kreyszig, Advanced Engineering Mathematics, John-Wiley, 1972.
[8] Hamilton, J.D., *Time Series Analysis*, vol. 2, Princeton University Press, Princeton, 1994.
[9] K.W. Hipel, A. I. McLeod, *Time Series Modelling of Water Resources and Environmental Systems*, Elsevier 1994.
[10] A. A. Ariyo, A. O.Adewumi, and C. K. Ayo, "Stock price prediction using the ARIMA model", In *Computer Modelling and Simulation (UKSim), 16th IEEE International Conference on UKSim-AMSS*, pp. 106-112, 2014.
[11] C. Liu, S. C. Hoi, P. Zhao and J. Sun, "Online ARIMA algorithms for time series prediction", *Proceedings of 30th International AAAI Conference of Artificial Intelligence*, 2016.
[12] D. Ö. Faruk, "A hybrid neural network and ARIMA model for water quality time series prediction," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp: 586-594, 2010.
[13] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp: 159-175, 2003.
[14] F. R. Engle, *"GARCH 101: The Use of ARCH/GARCH Models in Applied Econometrics," Journal of Economic Perspectives, vol.* ***15****, no. 4, pp. 157–168, 2001.*
[15] Z. Tang and P. A. Fishwick, "Feed-forward neural nets as models for time series forecasting," *ORSA journal on computing*, vol. 5, no. 4, pp. 374-385, 1993.
[16] M. Rosen-Zvi, I. Kanter and W. Kinzel, "Time series prediction by feed-forward neural networks: is it difficult?" *Journal of Physics A: Mathematical and General*, vol. 36, no. 16, p. 4543, 2003.
[17] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of clinical epidemiology*, vol. 49, no. 11, pp.1225-123, 1996.
[18] S. Jabin, "Stock market prediction using feed-forward artificial neural network-growth," *International Journal of Computer Applications*, vol. 99, no. 9, 2014.
[19] V. Vapnik, *Statistical Learning Theory*, John-Wiley, 1998
[20] C. Cortes, and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273-297, 1995.
[21] T. Van Gestel, J.A.K. Suykens, D. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle, "Financial time series prediction using least squares support vector machines within the evidence framework", *IEEE Trans. Neural Networks*, vol. 12, no. 4, pp. 809 -821, July 2001.
[22] U. Thissen, R. V.Brakel, A. P. De Weijer, W. J. Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometrics and intelligent laboratory systems*, vol. 69, no. 1-2, pp: 35-49, 2003.
[23] S. Rüping, "SVM kernels for time series analysis," *Technical Report*, SFB, vol. 475, no. 43, 2001.
[24] J. Suykens, "Support vector machines and kernel-based learning for dynamical systems modelling," *In Proceedings of the 15th IFAC symposium on system identification*, Saint-Malo, France, 2009.
[25] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, "Support vector regression with chaos-based firefly algorithm for stock market price forecasting,"*Applied soft computing*, vol. 13, no. 2,pp: 947-958, 2013.
[26] B.Al-hnaity, and M. Abbod, "Predicting financial time series data using hybrid model,"*In Intelligent Systems and Applications* Springer, pp. 19-41, 2016.
[27] E. Egrioglu, C. H. Aladag, and U. Yolcu, "Fuzzy time series forecasting with a novel hybrid approach combining fuzzy c-means and neural networks,"*Expert Systems with Applications*, vol. 40, no. 3,pp. 854-857, 2013.
[28] C. H. Aladag, U. Yolcu, E. Egrioglu, and A. Z. Dalar, "A new time invariant fuzzy time series forecasting method based on particle swarm optimization,"*Applied Soft Computing*, vol. 12, no. 10, pp: 3291-3299, 2012.
[29] S. A. Abdulkarimand A. P. Engelbrecht. "Time Series Forecasting Using Neural Networks: Are Recurrent Connections Necessary?" *Neural*

Processing Letters, vol. 50, no. 3, pp. 2763-2795, 2019.
[30] H. Yang, K. Huang, I. King, and M. R. Lyu, "Localized support vector regression for time series prediction." *Neurocomputing* , vol.72, no. 10-12, pp.2659-2669, 2009.
[31] R. Samsudin, A. Shabri, and P. Saad, "A comparison of time series forecasting using support vector machine and artificial neural network model." *Journal of applied sciences, vol.* 10, no. 11, pp.950-958, 2010.
[32] U. Thissen, R. Van Brakel, A. P. De Weijer, W. J.Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction." *Chemometrics and intelligent laboratory Systems, vol.* 69, no. 1-2,35-49, 2003.
[33] O. C. Yolcu, E. Bas, E. Egrioglu, and U. Yolcu."Single multiplicative neuron model artificial neural network with autoregressive coefficient for time series modelling," *Neural Processing Letters*, vol.47, no. 3, pp.1133-1147, 2017.
[34]L. M.Liu, and M. H. Dominique, "A Bayesian approach to time-varying cross-sectional regression models,"*Journal of Econometrics, vol*, 15, no. 3, pp. 341-356, 1981.
[35] R. Hafezi, and A. Akhavan, "Forecasting gold price changes: Application of an equipped artificial neural network, "*AUT Journal of Modeling and Simulation*, vol. 50, no. 1, pp.71-82, 2018.
[36] Z. H. Li, Y.Jia, J. Wu, and Y. Zhou. "Forecasting crude oil prices using ensemble empirical mode decomposition and sparse Bayesian learning," *Energies*, vol. 11, no. 7, pp. 1882, 2018.
[37] R. Adhikari,G. Verma, and I. Khandelwal, "A model ranking based selective ensemble approach for time series forecasting,"*Procedia Computer Science*, vol. 48, pp: 14-21, 2015.
[38] S. A. Mangai, B. R. Sankar, and K. Alagarsamy, "Taylor Series Prediction of Time Series Data with Error Propagated by Artificial Neural Network," *Int. Journal of Computer Applications*, vol. 89, no. 1, 2014.
[39] TAIEX [Online]. Available http://www.twse.com.tw/en/products/indices/tsec/taiex.php, May 23, 2017.
[40] NASDAQ, DOW-JONES, FACEBOOK [Online]. Available https://in.finance.yahoo.com/, April 22, 2018.
[41] FTSE-100 [Online]. Available http://markets.businessinsider.com/index/historical-prices/ftse_100/, May 17, 2018.
[42] Sunspot data [online]. Available http://www.sidc.be/silso/datafiles, April 12, 2017.
[43] India's monthly minimum temperature data [online]. Available https://data.gov.in/catalog/all-india-seasonal-and-annual-minmax-temperature-series, May 11, 2028.

CHAPTER - 2

Learning Regularity in an Economic Time-Series for Structure Prediction

***Abstract*— Although an economic time-series has an apparently random fluctuation over time, there exists certain regularity in the functional behavior of the series. This paper attempts to identify the regularly occurring structures in an economic time-series with an aim to represent the series as a specific sequence of such structures for forecasting applications. The applications include prediction of the most probable structure with its expected duration, along with predicted values lying thereon. Representation of a time-series by a set of regularly recurring structures is undertaken by invoking three main steps: i) non-uniform length segmentation of the series, ii) identification of the recurrent patterns by clustering of the generated segments, and iii) representing the sequence of regular structures using a specially designed automaton. The automaton is used here to both encode the sequence of structures representing the time-series and also to act as an inference engine for stochastic forecasting about the time-series. Experiments undertaken on large (28 years') daily economic time-series datasets confirm the success in automated structure prediction with an average prediction accuracy of 88.05%, average precision of 91.24% and average recall of 93.42%.**

***Keywords*—Knowledge acquisition, time-series segmentation, multilevel clustering, stochastic automaton, business forecasting**

1. Introduction

A time-series represents a sequence of the time-functional values available at regular interval of time, such as yearly population-growth [4], daily recordings of temperature [11], humidity and rainfall [56], daily variation in economic growth [31] of a country, and many others. Prediction of a time-series refers to determining the value of the time-series at future time points from its current and preceding values. Among the well-known principles of time-series prediction, deterministic [53], stochastic [2], neural [16], [39], [47], fuzzy [8], [10], [12]-[14], [60], chaos theoretic [29], regression [40] and other approaches indicated in [28] need special mention. Besides prediction of time-series, there exist a few research works dealing with matching [22], segmentation [1], [3], [9], [15], [26], [32], [33], [51] dimensionality reduction [63], clustering [28], and classification [27], [55] of time-series. Knowledge acquisition [17], [34] from a time-series by time-series sub-sequence clustering is another interesting research arena. One important but scarcely studied aspect of time-series research is forecasting of (up/down/side-ways) moves/structures emerging from selected time- points. A structure comprises a block of contiguous data points, having a meaningful shape/geometry. Prediction of structures is important in economic time-series, as the forecasted structures help investors and traders in their business planning.

The motivation of this paper is two-fold. First, it would be able to automatically recognize the repetitive structures present in an economic time-series in absence of any knowledge about structure-shape, duration and location of their endpoints. Second, it offers a method to predict the (most) probable structures expected to emerge from the selected time-points. It would also infer the sequence of the most probable structures between any pair of partitions of a pre-partitioned time-series. The first motivation helps in determining the approximate duration, shape and probability of occurrence of the primitive structures present in the series. The second motivation helps in predicting structure-shape, approximate duration and probability of occurrence, which provide useful insights to business traders and investors.

Identification of the primitive structures present in a time-series is performed here in two steps. In the first step, the time series is segmented heuristically based on the similarity in local slopes of the contiguous data points in the series. Second, the segments extracted are clustered (grouped) based on similarity of the geometric shapes of the segments. Given a daily (economic) time-series of significant length (≈10 years), it is noted from experiments that there exist approximately 7 to 10 clusters of the segments. In other words, there exist 7 to 10 possible primitive structures embedded in the time-series. So, a time-series of finite length can

be represented by a sequence of known structures. While predicting the most probable structure, emanating from a given time-point, any one of these 7-10 possible structures is expected as the solution. One natural question is: how to predict the most probable structure. This is undertaken here with the help of a Dynamic Stochastic Automaton (DSA), which is briefly outlined below.

The DSA employed here includes nodes (states) representing partitions of the series and directed arcs representing state-transitions from one partition to the other due to the presence of a structure between the partitions. The structure p_k originating from a partition Z_i and terminating to a partition Z_j is represented in DSA as two states s_i and s_j and one input symbol occurring at s_i causing transition to state s_j. The probability of transition from a state to other (feasible) states due to occurrence of an input symbol to the former state is attached to the respective arcs along with the average duration of the transition in the series. Suppose, the structure emanating at time t needs to be predicted. As the time-point 't' is given, the partition containing the time-series value at time t and hence the state describing the partition are known. Then the input symbol, representative of a structure with the highest probability of occurrence, is searched in the selected state of the DSA. The said structure is inferred as the next possible structure of the time-series emanating at time t.

The originality of the paper thus is 2-fold. First, a new technique for online structure prediction in a time-series, involving 3 basic steps: i) segmentation, ii) clustering and iii) prediction using DSA, is proposed. Second, the algorithms developed for the individual steps are novel and have shown promising performance with their existing counterparts. An outline to the original contributions of the proposed segmentation, clustering and DSA-based prediction and their relative merits with existing algorithms are summarized below.

Most of the traditional segmentation algorithms utilizes *piecewise linear approximation* (PLA) criterion [64] to distinguish segments based on their inter-segment homogeneity (here uniform slope). In other words, a segment boundary is selected at a time point, where the slopes of the current and the next segments have significant difference. Two common varieties of segmentation algorithms, developed satisfying the PLA criterion, include the top-down [51], the bottom-up [33] algorithms. While the top-down algorithm recursively splits a fragment of a given time-series into two components based on difference in slopes, the bottom-up algorithms recursively merge segments greedily as long as PLA criterion within the segment is satisfied. The top-down and the bottom-up algorithms work in offline mode. Among the online segmentation algorithms, the Sliding Window (SW) algorithm is popular. The SW algorithm aims at identifying the position of the next segment boundary by iteratively widening the present segment, until the measure of the "approximation error" [37] (like Maximum Vertical Distance (MVD) [41] or Root Mean Square Error [10]) of the segment exceeds a predefined threshold. Several extensions [48], [49] to the basic SW algorithm have been undertaken to reduce its time-complexity. Keogh, for example, combined sliding window and bottom up principles to design a new algorithm called SWAB [49]. Among other interesting techniques of segmentation, Dynamic Programming [5], [9], Guaranteed Error-Bound based segmentation [41], [57], [62], clustering [1], least square approximation [26] and evolutionary algorithm [15] based techniques need special mention.

The proposed segmentation algorithm differs from the existing ones by the following counts. First, it labels the transition between pairs of successive time-points in the series as 'rise'(R), 'fall'(F) and 'equal'(E) depending on increase, decrease and no change in amplitudes of the successive points. Second, it labels a sequence of a fixed number of time-points into v = R/F/E, if the frequency count of v is maximum within the predefined fixed number of data points. Third, it groups two or more consecutive sequence of time-series points having common label x into a segment. Thus the proposed algorithm is similar with natural segmentation of ridges on a mountain by a traveler based on the local gradients.

The structures extracted from a long time-series of several years are often found to have shape- similarity, which can be easily recognized manually by visual inspection. However, because of non- uniformity in the duration of the extracted structures and their peak-to-peak amplitude, they need to be pre- processed before automatic clustering. The z-score normalization is employed to normalize their peak-to-peak amplitude and fixed number of sampling is used to normalize their duration, irrespective of structure-length. Once normalized, a clustering algorithm [45], [59] could be employed to group similar structures of fixed duration and amplitude using certain distance metrics [49], [52], [54] to measure similarity [18], [50], [61]. If p number of clusters is discovered, then the cluster centers of p clusters are recorded as the primitive structure-shapes in the normalized form. Although any traditional clustering algorithms, such as k-means, k- medoids, fuzzy c-means clustering algorithm and the like could serve the purpose, they suffer from 2 fundamental limitations: i) performance of clustering on the initialization of cluster centers [6] and the requirement to specify the number of clusters [59], which is not known in the present situation. The above problems can be alleviated by data-density based clustering algorithms. *Density Based Spatial Clustering of*

Applications with Noise (DBSCAN) [23] is one such well-known data-density dependent clustering algorithm. The DBSCAN algorithm clusters points located in the highest data density region and also labels points with lower surrounding density as noise. DBSCAN-DLP [58], which is also an extension of classical DBSCAN, attempts to cluster data points based on their variety of data densities. In this paper, we attempt to extend the classical DBSCAN algorithm to *hierarchically cluster* [19], [46] data points at different density levels. The DBSCAN-DLP pre-processes the input data and then organizes them into layers based on their density. Next for each layer of data, the classical DBSCAN is utilized to cluster the data points. Here, an analogous technique with one fundamental difference is proposed. In the present settings, a greedy recursive technique, which in lieu of preprocessing the data set, layers the data set by isolating the cluster with the highest density and dropping the points with relatively lower density as outliers, is proposed. A recursive realization of the data layering autonomously places the cluster centers in the relative order of data density. Besides, data points of uniform density, irrespective of their spatial locations, are clustered at the same level.

After clustering of the structures by the proposed extension of DBSCAN clustering, the cluster centers representing the ideal members of each cluster are preserved for future usage. In fact, the stochastic automation is developed to encode the knowledge for state-transition, considering the structures representative of the cluster centers, as the input symbol of the automaton. The idea used for knowledge encoding [17] in the proposed Dynamic Stochastic Automaton is novel and unique in comparison to the related works reported in [36], [40], [42], [43].

The paper is divided into eight sections. Section 2 provides an overview of the proposed structure prediction technique. Section 3 provides a novel algorithm for segmentation of a time-series. Section 4 extends the DBSCAN algorithm for hierarchical multi-resolution clustering. Section 5 provides the principles of time-series representation by a stochastic automaton. Section 6 deals with prediction experiments on 3 well-known economic time-series: Taiwan Stock Exchange Index (TAIEX) [65], National Association of Securities Dealers Automated Quotations (NASDAQ) [66] and Dow Jones Industrial Average (DJIA) [67] for each year in the period 1990-2017, resulting in an average structure prediction accuracy of 88.05%. A study of the performance analysis of the proposed algorithm is undertaken in Section 7.The concluding points are summarized in section 8.

2. An Overview of the Proposed Structure Prediction Technique

Structures are meaningful contiguous blocks of time-series data. Structure prediction, here, is formulated as a 2-phase problem. In the first phase, the knowledge acquired from the time-series is encoded in a DSA. In the second phase, the DSA is used to forecast the possible shape of the meaningful moves in the time-series at a desired time-point.

Fig.1 provides a schematic overview of the DSA construction from a time-series. First, the time-series is segmented into (homogeneous) time-blocks of uniform slopes, called segments [30]. One or more consecutive segments having slope-similarity are grouped into a structure. It is interesting to note that a time- series usually contains a few repeatedly occurring primitive structures. In order to identify the primitive structures, present in the time-series, the structures acquired from the series are first normalized to have a uniform amplitude and duration, and the normalized structures are then clustered based on shape-similarity. Clustering returns a cluster-center-structure for a group of similar shaped structures. A cluster centre is usually regarded as an ideal member of the cluster. The cluster centers here have specific geometric shapes, carrying meaningful information. Suppose the structures present in a (pre-segmented) time-series are labeled by their nearest matched cluster centers, where the matching is undertaken by shape-similarity with respect to Euclidean distance metric. A pre-partitioned and structure-labeled time-series can be easily encoded in the parametric space of a DSA. The principles employed for encoding in a DSA are explained next.

The DSA includes nodes (called states) to represent the partitions of the time-series and directed arcs/edges to represent transitions from one partition to the other. Each transition is represented by one of the cluster-center-structures acquired from the time-series by means of clustering. A transition of the time-series from partition Z_i to partition Z_j involving a cluster center structure p_k, is encoded in the DSA by 2 states s_i and s_j and a directed arc from s_i to s_j with an attached label p_k. In addition to p_k, the probability of occurrence of the structure p_k and its duration are also attached as labels to the transition (directed arc from s_i to s_j). Fig. 2 illustrates the principles of DSA construction from a time-series. Fig. 2(a) provides a time-series with five partitions: Z_1, Z_2, Z_3, Z_4, Z_5 and 10 segments/structures segregated by vertical dotted lines. The structures are normalized in both amplitude and duration, and then clustered (grouped) based on shape-similarity. This resulted in 4 cluster center (primitive) structures: p_1, p_2, p_3, and p_4. The segments in Fig. 2(b) are replaced by

cluster-center-structures. Next the partition-to-partition transitions are encoded by using the automaton (Fig. 2(c)). For example, the first structure originating at partition Z_3 and terminating at partition Z_5 (Fig.2 (b)) is realized by cluster-center- structure p_1. This is encoded in the automaton of Fig. 2(c) as 2 states: s_3 and s_5 and one directed arc labeled with p_1 describing the transition of the time-series from partition Z_3 to partition Z_5. In the terminology of automata theory, it is said that an input symbol p_1 appearing at state s_3 causes a transition of the time-series to state s_5. The rest of the automaton in Fig. 2(c) is self-explanatory.

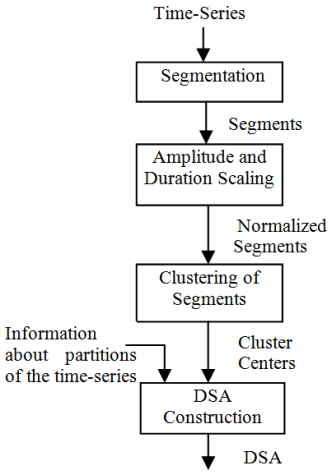


Fig. 1. Basic steps of knowledge encoding in a DSA from a given time-series during the training phase

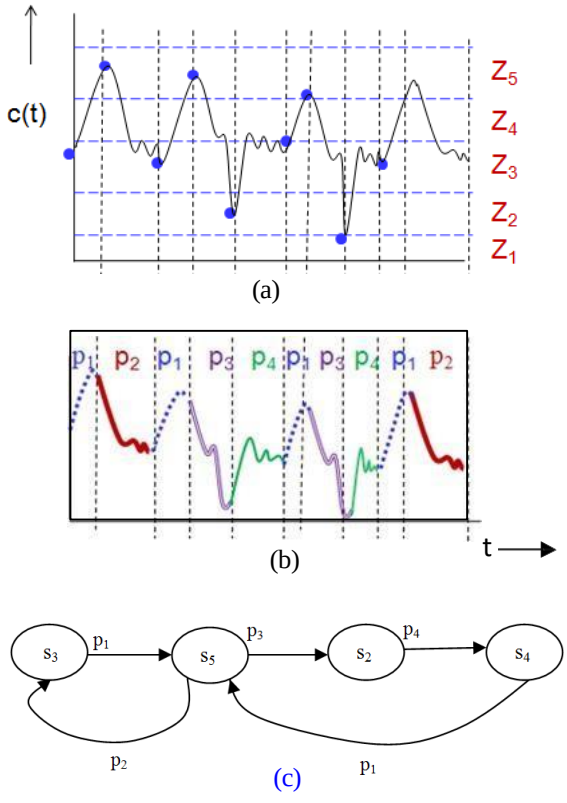


Fig. 2. Construction of an automaton (c) for a segmented and partitioned time-series (a) with discovered primitive structures (cluster centers) p_1, p_2, p_3 and p_4, shown in (b)

In a Dynamic Stochastic Automaton (DSA), the edges/transitions are labeled with 3 parameters: input symbol p_i/<probability of p_i>, <duration of p_i>. For example, in Fig. 3, the state-transitions from state s_3 to s_5

due to occurrence of an input symbol (cluster-centre-structure) p_1 is represented by an edge-label: p_1/ 0.7, 5, where p_1 is the input symbol (cluster-centre-structure) causing state transition from s_3 to s_5, 0.7 denotes the probability of state-transition due to input symbol p_1 and 5 denotes the time required for the said state-transition.

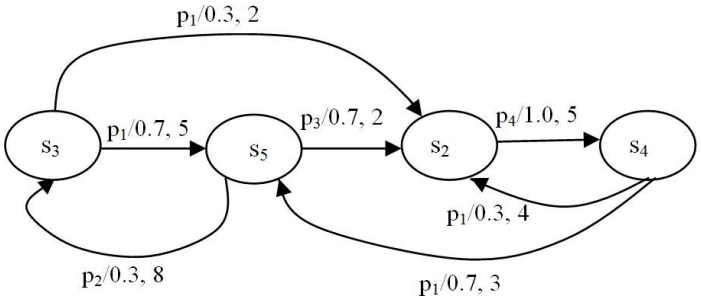


Fig. 3 Illustrating a Dynamic Stochastic Automaton

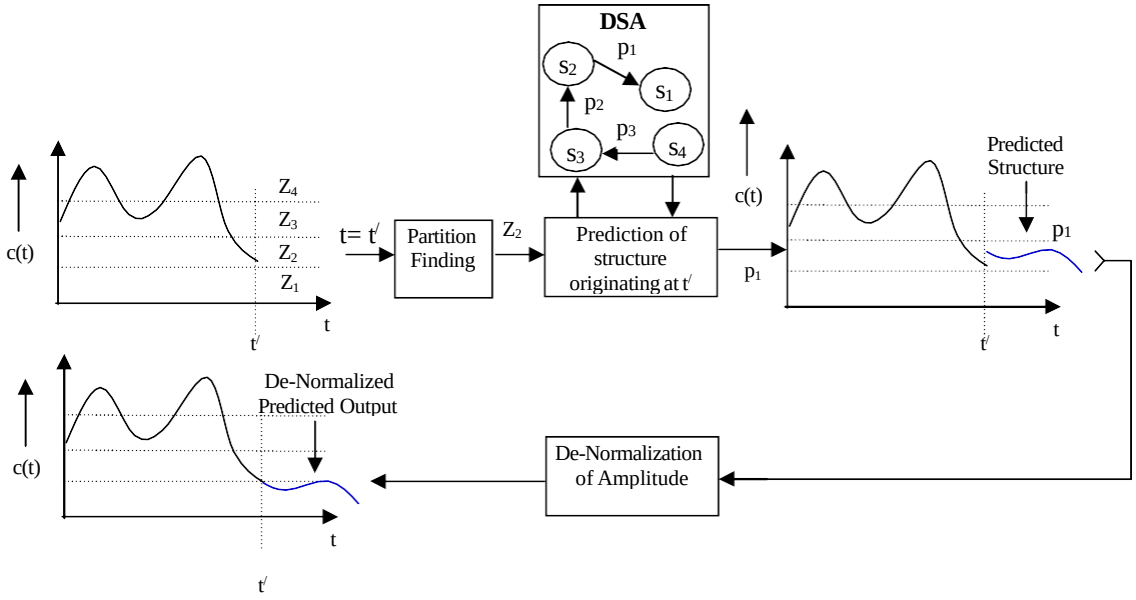


Fig. 4. Basic steps of structure prediction in a time-series using a DSA during the application phase

The automaton in Fig. 2(c) is called deterministic, as an input symbol appearing at a state, results in a unique next state. For example, occurrence of p_1 at s_3 would have a unique transition to next state s_5. In case there exists more than one next state due to occurrence of p_1 at s_3, the automaton becomes non-deterministic. The non-determinism is handled here stochastically by attaching probability to the transitions. Fig. 3 represents a typical DSA with arcs labeled with 3 parameters, the input symbol, probability of occurrence of the state transition due to the input symbol and approximate duration of the transition. One restriction in DSA is that the sum of the probabilities for the transitions from a given state (say, s_3) to multiple states (say, s_5 and s_2) due to occurrence of all possible input symbols (here, p_1) (Fig. 3) is one. The rest of Fig. 3 is self- explanatory. One interesting application of structure prediction is to determine the most probable structure/sequence of structures between two partitions. The most probable sequence of structures is the highest probability pathway between selected states (indirectly, partitions). For example, the most probable sequence of structures between state s_3 and s_2 in Fig. 3 is p_1 p_3 with probability of occurrence = 0.7×0.7 =0.49.

After the automaton is built up, the details of transitions from one partition to others by the cluster center structures are graphically available to the users. This makes structure prediction easy. For example, consider the close price time-series c(t) presented in the top left corner of Fig. 4. Here, the problem is to predict the shape of the probable structure that would emanate from the time point t= t/. As the amplitude of the time-

series at t= t′ is known, the partition (here, Z_2) containing the time-series value at t= t′ is determined. Next, the automaton is consulted to identify the transition originating at sate s_2. It is found that the transition s_2 to s_1 takes place due to occurrence of the input symbol p_1. So, the structure p_1, previously obtained as a cluster center, is identified and presented as the predicted structure. However, the structure was saved in amplitude- normalized form during the training phase, and so needs a de-normalization. The final structure obtained after de-normalization is given in the left bottom time-series, emanating at time-point t= t′. Details of probabilistic structure prediction will be given in Section 5.

3. **Slope-Sensitive Natural Segmentation**

The Slope-Sensitive Natural Segmentation (SSNS) algorithm presented here, segments a time-series in a manner similar to natural segmentation of ridges on a mountain based on slope changes. The algorithm works in two phases. The first phase is required to label the lines joining consecutive pairs of data points as rise (R), fall (F) and zero-slope (E). In the second phase, it classifies the windows of fixed length (intuitively chosen as five) containing the maximum number of labels of R/F/E into the respective type: rise, fall and zero-slope. A set of one or more consecutive windows of same type (R/F/E) forms a segment. The algorithm is found to generate segments of a fixed set of geometry over a given time-series of varying length, and different sets of geometric patterns for different time-series. In addition, the algorithm is less sensitive to small fluctuations in the time-series.

Consider a time-series: $\vec{X} = [x_i]$ for i= *1 to n*. Let x_{max} and x_{min} be the maximum and minimum of the possible sample values in the series. The range Z=[x_{min} , x_{max}] of the series is partitioned into k non- overlapped equal-length intervals: $Z_1, Z_2, ..., Z_k$, called partitions, such that union of the partitions yields Z. The partitions are ordered sequentially, such that for any two successive partitions: $Z_p = [Z_p^-, Z_p^+]$ and $Z_{p+1} = [Z_{p+1}^-, Z_{p+1}^+]$ for p= *1 to (n - 1),* $Z_{p+1}^- > Z_p^+$ holds.

Now, for 2 successive data points (x_i , x_{i+1}) in the time-series $\vec{X}$, let $x_i \in Z_p$ and $x_{i+1} \in Z_q$. Then the **transition** of x_i to x_{i+1}, denoted by S_i is associated with one of the following three labels: Rise (R), Fall (F) and Zero-slope (E) following (1).

$$\begin{aligned} S_i &= \text{R, if } q > p \\ &= \text{F, if } q < p \\ &= \text{E, if } q = p. \end{aligned} \qquad (1)$$

Now, a time-series $\vec{X} = [x_1, x_2, ..., x_n]$, comprising n points has (n–1) transitions. Let $\vec{S} = [S_1, S_2, ..., S_{n-1}]$ be a **Ternary String of Transitions** (T-SOT) obtained from the given time-series $\vec{X} = [x_1, x_2, ..., x_n]$ where S_i denotes one of three labels: R/F/E by (1). A **sliding window** W_i **of length L** for i=*1* to n–L is obtained by selecting L consecutive elements from $\vec{S}$, given by $\vec{W}_1 = [S_1, S_2, ..., S_L]$, $\vec{W}_2 = [S_2, S_3, ..., S_{L+1}]$, …, $\vec{W}_{n-L} = [S_{n-L}, S_{n-L+1}, ..., S_{n-1}]$. Each window is assigned one of three labels: R, F, E based on the frequency of occurrence of the symbols R, F and E in the window. Let the label of a window W_i, called $L(\vec{W}_i) = v$, if for v∈{R, F, E} the frequency count of v in $\vec{W}_i$ is maximum. In case v is not distinct but v is also the label of $\vec{W}_{i-1}$ then also assign v to $L(\vec{W}_i)$ i.e., $L(\vec{W}_i) = v$. If both the conditions fail, then any one

of the labels with the highest frequency count is assigned to W_i .

A **structure** is defined as a sequence of one or more contiguous windows with the same label. For

example, if $L(W_3)=L(W_4)=L(W_5)=R$ then $W_3W_4W_5$ is called a structure with the same label R, as carried by the participating windows. Like windows, labels are assigned to structures as well. The label associated with the i^{th} structure is represented by $L(struct_i)$. For two consecutive structures: $struct_i$ and $struct_{i+1}$, if $L(struct_i) \neq L(struct_{i+1})$, we identify a segment boundary located at the center point S_j of the last window $\vec{W_j}$ (say) in $struct_i$. As S_j represents the transition between the data points x_j and x_{j+1} in the series, a segment boundary exists between the above two data points.

Example 1: Let $\vec{x}$ = [2, 4, 3, 4, 6, 5, 8, 9, 8,10, 8, 9, 7, 6, 5] be a time-series. Let the range Z= [2, 10] of the series be partitioned horizontally into 6 partitions with $Z_1 = [2,\ 3.5)$, $Z_2 = [3.5,\ 5)$, $Z_3 = [5,\ 6.5)$, $Z_4 = [6.5,\ 8)$, $Z_5 = [8,\ 9.5)$, $Z_6 = [9.5,\ 10]$. For the first two points $x_1 = 2$ and $x_2 = 4$, in the series, i.e., $S_1 = R$ as $x_1 \in Z_1$ $(= Z_p)$, and $x_2 \in Z_2 (= Z_q)$ *and* q >p. In the same manner, S_i for i=1 to 14 is evaluated in the given time-series comprising 15 data points. The T-SOT has been computed as [*R*, *F*, *R*, *R*, *E*, *R*, *E*, *E*, *R*, *F*, *E*, *F*, *F*, *E*]. Considering window-length= 5 characters, as many as 10 windows are obtained, with W_1=[R, F, R, R, E], W_2= [F, R, R, E, R], …,W_{10}=[F, E, F, F, E]. Here, $L(W_1)$= R, as frequency count of R in W_1 is the highest. Similarly, $L(W_2)$=R, $L(W_3)$=R, but $L(W_4)$=E. Since $L(W_1)= L(W_2) = L(W_3)$, we obtain a structure, called structure$_1$ = $W_1W_2W_3$. Since $L(W_4) \neq$ R, it will fall in structure$_2$, and a segmentation boundary is fixed at the middle of the last window (i.e.,W_3) of structure$_1$. Details of computation for segmentation of the given time-series are given in Fig.5.

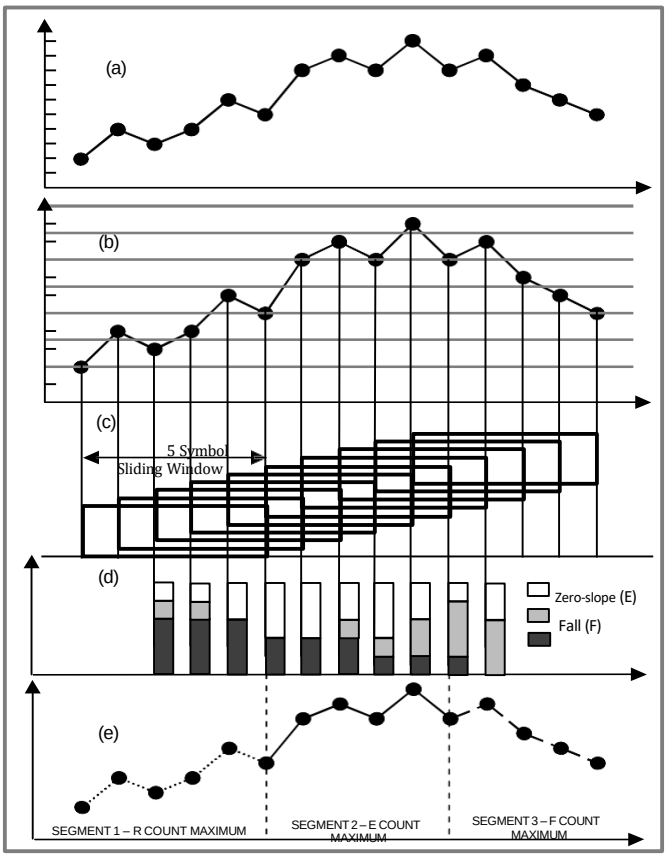


Fig. 5. Illustrating the SSNS segmentation: (a) time-series, (b) partitioning and T-SOT construction, (c) windowing, (d) freq. of window labels, (e) segmentation

The SSNS algorithm, given in Table 1, includes four steps. In step 1, a given time-series $\vec{X}$ =[x_i] of a finite range $[min(X), max(X)]$ is divided into k partitions of uniform width w, where w denotes the average of the successive differences of the time-series data points, and $min(\vec{X})$ and $max(\vec{X})$ denote the minimum and maximum values in the array $\vec{X}$. Here, $w = (\sum_{i=1}^{n-1} (x_{i+1} - x_i)) / (n-1)$ for the time-series [x_i] for i=1 to n and

$k = (max(\vec{X}) - min(\vec{X}))/w.$ Step 2 of the SSNS algorithm is concerned with labeling of the transition S_i between successive pair of data points: x_i and x_{i+1} into 3 distinct labels: R, F and E following (1), and saving the transition labels in an array T-SOT: $\vec{S} = [S_1, S_2, ..., S_{n-1}]$. In step 3, the label $\vec{L(W_i)}$ of the window $\vec{W_i}$, comprising 5 successive symbols in T-SOT, is determined for $i = 1$ to $(n - L)$. Step 4 of the SSNS algorithm declares the boundary of a segment at the center of $\vec{W}_{i-1}$ (at symbol S_{i-1}) if $L(\vec{W}_{i-1}) \neq L(\vec{W}_i)$, and also saves the time-point corresponding to the centre of x_{i-1} and x_i as an element i_i in an output array $I = [i_i]$.

Table 1: Pseudo code of the SSNS Algorithm

Input: A given time-series $X = [x_1, x_2, ..., x_n]$, containing n data points;

Output: A vector $\vec{I} = [i_1, i_2, ..., i_m]$, comprising the segment boundaries

Begin

1. Partitioning of the Series: Partition the range: $[min(\vec{X}), max(\vec{X})]$ of X into k partitions, where $k = (max(\vec{X}) - min(\vec{X}))/w$; and $min(\vec{X})$ and $max(\vec{X})$ represent the smallest and the largest elements respectively in the time-series $\vec{X}$, where partition-width $w = (\sum_{i=1}^{n-1}(x_{i+1} - x_i))/(n-1)$. Merge partitions containing no data points with their immediate lower partition.

2. Labelling of transitions: For each pair of successive data points x_i and x_{i+1}, $1 \le i \le (n-1)$, determine the labels of the transition between x_i and x_{i+1} following (1). Store the obtained transition-labels in a T-SOT: $\vec{S} = [S_1, S_2, ..., S_{n-1}]$ where S_i *denotes the transition from* x_i *to* x_{i+1} for *i=1 to (n-1).*

3. Labelling of windows: For a sequence of five successive symbols in the T-SOT $\vec{S}$, determine the windows $\vec{W}_i = [S_i, S_{i+1}, ..., S_{i+L-1}]$. for $i = 1$ to $(n - L)$. Assign the label $L(\vec{W}_i)$ to the window W_i, using the principles introduced earlier.

4. Segmentation boundary detection: For two successive windows, $\vec{W}_{i-1}$ and $\vec{W}_i$, if $L(\vec{W}_{i-1}) \neq L(\vec{W}_i)$, declare the boundary of a segment at the center of $\vec{W}_{i-1}$ (i.e., at symbol S_{i-1}) which in turn refers to the center of x_{i-1} and x_i. Save the time-point corresponding to the center of x_{i-1} and x_i as i_i, where i_i = i-th element of the output $\vec{I}$.

End.

Results of complexity analysis: The SSNS algorithm has four main components. The partition-width

evaluation requires Max and Min computation over an array of length n, where n is the number of data points in the time-series. Hence, the time complexity for partitioning is $O(n)$ and the space complexity is $O(n + z)$, where z denotes the number of partitions.

Next, the T-SOT computation requires iterating over each pair of consecutive data points in the time-series to identify the partition containing each data point. The array of partitions being ordered, a binary

search is used, requiring a time-complexity of $O(\log z)$. The above computation is carried out for n data points and so, the time-complexity taking into account n points is $O(n \log z)$. Third, the windowing, window labeling and segmentation steps together, iterate over the elements in the T-SOT to place a marker at each segment boundary. The window having a fixed length, the decision to place a segment boundary takes a constant time. As the T-SOT contains $n-1$ elements, this step has a time complexity of $O(n)$. Thus, the overall time-complexity is $O(n) + O(n \log z) + O(n) \approx O(n \log z)$ and the space complexity is $O(n + z)$.

4. Clustering of Segmented Structures

A typical time-series of 5000 to 10,000 data points contains several hundred segmented structures of similar shapes. To determine the primitive structures in a time-series, we need to perform two steps: i) pre-processing and ii) clustering of the pre-processed structures. The pre-processing is again done in two phases. The first phase includes representing the segmented time-blocks/structures of different lengths into vectors of fixed dimension b. The value of b is selected heuristically as 12. The second phase is concerned with normalization of the range of the time-blocks. The phase-1 of pre-processing includes 3 steps.

1. Construct straight lines by joining pairs of successive points within a segment to develop a PLA of the series within the segment.
2. Represent the segment as a sequence of b number of equally spaced points, covering the entire segment.
3. Obtain the ordinates of the above b points in the constructed PLA of the series within the segment.

The phase 2 involves normalization of the range of the segments. Suppose, we have altogether l segments in a time-series, each being represented as a vector of b (=12) dimensions, as done in phase 1. Thus we have a matrix M of $(12 \times l)$ elements, where element (i, j) denotes i-th dimension of segment j. Let $mean(M, j)$ and $std(M, j)$ be the respective mean and the standard deviation of the information under the j^{th} column of matrix M. A Z-score standardization, given by (2) is undertaken.

$$M_{i,j} = \frac{M_{i,j} - mean(M, j)}{std(M, j)} \tag{2}$$

The merit of Z-score standardization lies in scaling the segments with an aim to have zero mean and unit variance.

After the two phases of pre-processing is over, the segmented structures are clustered to obtain the cluster centers, representing primitive structures. Although quite a vast literature on clustering algorithms [29] is available, a density-based clustering algorithm, capable of generating a cluster center for the densest region of the data points is selected. The DBSCAN algorithm offers density-dependent spatial clustering of data points. It clusters points located in the highest data density region and also labels points with lower surrounding density as noise. This inspired us to employ DBSCAN for the present application. Here, the DBSCAN algorithm is extended for its hierarchical realization, with a motivation to cluster the data points of a particular density at one level, and to transfer the data points located in the less dense regions for possible clustering in the next level in a recursive manner. The recursion stops when the data density at a level falls below a pre-defined threshold. An overview of traditional DBSCAN algorithm is presented first, before undertaking the proposed hierarchical extension.

4.1 Outline of the DBSCAN Clustering

The DBSCAN algorithm has two input parameters: i) the radius ε and ii) a threshold m. The neighborhood of a point is represented by ε. On the other hand, m stands for the minimum number of points to qualify as the neighborhood of a selected point. The following terminologies are presented to explain DBSCAN.

Definition 1: A point P is referred to as a **core** point in a domain of points D, if the following two conditions jointly hold.

1. For a given set of points $P' = \{P_1, P_2, P_3, ..., P_k\}$,

$$\|P - P_i\| \le \varepsilon, \ \forall P_i \in P', \tag{3}$$

i.e., P_i lies in the ε **-neighborhood** of P. This is referred to as ε **-neighborhood criterion.**

2. $|P'| = k \geq m$ (4)

i.e., the number of points lying in $P' = k \geq m$, a predefined empirically selected threshold. This is referred to as **threshold criterion.**

Definition 2: A point P_i that falls in the ε **-neighborhood** of P is **directly density-reachable** from P. We express **data-density of point P as** e/ε where e denotes the number of points lying in the ε -neighborhood of P, and ε is the radial distance between P_i and P.

Definition 3: Let there be a sequence of points $P_1, P_2, ..., P_n$ so that P_{i+1} is directly density-reachable from P, for all i in [1, n -1]. Again, P and Q are directly density-reachable from P and P respectively. Then we say that P is **density-reachable** from Q.

Definition 4: Two points P_i and P_j are **density-connected,** if there is a common point P so that both the points P_i and P_j are density-reachable from P.

Definition 5: A point P_i is referred to as **processed,** when it is selected for examination of all the points in its ε – neighborhood. A point, which has not yet been processed yet, is referred to as **unprocessed**.

The DBSCAN algorithm has three main steps.

1. Given a set of data points D, this step picks up an unprocessed point P_i from D and examines the possibility of P_i to be a core point. If P_i is detected as the core point, all the ε – neighborhood points of P_i are recorded in set N. The set N is used to keep track of all unprocessed points. If P_i is not a core, it is simply ignored.

2. For each unprocessed point P_j belonging to set N, determine points lying in the ε – neighborhood of P_j and save those points in M. Set $N \leftarrow N \cup M$. The inclusion of the new unprocessed points in N, with gradually increasing distance from the initial core point leads to the formation of the cluster boundary. If all points in N are processed, a cluster is formed and a new core point is searched.
3. The algorithm comes to an end when there is no unprocessed core point in N,
The following two observations follow as a natural consequence of DBSCAN clustering.

1. Two points P_i and P_j, that lie in the same cluster are mutually density-connected.
2. If point P_i is density-reachable from point P_j where $P_j \in cluster\ C$, then P_i also belongs to cluster

4.2 Extending DBSCAN for Multi-level Clustering

The proposed extension includes the following fundamental changes over the classical DBSCAN algorithm.
1. The multilevel DBSCAN employs a recursive realization of the classical DBSCAN, where at each level of recursion two tasks are undertaken as in classical DBSCAN. The first task refers to clustering of the data points having the highest local data density. The second task is to pass the lower density data points (outliers)

to the next level for subsequent clustering. Fig. 6 illustrates the scheme.

2. At each level of recursion, compute two items; i) the ε parameter and ii) a point-information table (PIT).

a) *Computation of the ε parameter for each level:* The Z-score normalized l data points of 12 dimensions,

representing the structures to be clustered are readily available. Each structure thus is a point in 12-dimensional space. To evaluate ε for the first recursion level, i)first construct a distance matrix $D= [d_{ij}]$ of $(l \times l)$ dimension, where d_{ij} denotes the Euclidean distance between the pair of structures i and j, ii) next sort the distances in descending order and iii) finally take the mean of the top h distances as ε. Here, h is empirically selected as the top s=10% members of the list of distances. The computation of ε parameter for the other levels is done in a similar manner, where the number of distances is less than l^2.

b) *PIT construction:* The essence of the PIT lies in easy organization of the data points that lie within an ε -radius of a core point. Although several alternative data structures could be used to realize the PIT, we here organize it using an array of linked lists. Let there be l points of 12 dimensions, representing l structures. Suppose, out of these l points, $l - m$ number of points satisfy both the ε -neighborhood criterion and the threshold criterion, So, preserve an array of pointers of length $l - m$, whose i-th element points to a linear linked list (Fig. 7) to the store the number of ε -neighborhood points (N_i) of point P_i along with the indices of the neighborhood points: $i_1, i_2, ..., i_{Ni}$. The m points that disagree with the ε -neighborhood and/or threshold criteria are declared as outliers, and are passed on to the next recursion level for clustering.

3. Outlier Detection: The following principle is used to detect the outliers. Let N_i be the number of points in the ε -neighborhood of a given point p. If there exist a total number of l points in a recursion level, we consider the local ε -neighborhoods of these l *points.* We check whether N_i is less than the average of the local neighborhood points of the given l *points.* If the condition holds, we declare p_i as an outlier. This is formally stated in condition (5).

$$outlier(p_i) = true,\ if\ N_i < \sum_{j=1}^{l}(N_j/l) \\ = false, \text{otherwise.} \Big\} \tag{5}$$

In addition, the PIT is scanned to identify the seed point, i.e., a point with the highest neighborhood density to initiate a new cluster. This justifies the why the cluster canters are hierarchically organized in the decreasing order of data- density.

4. The outliers obtained at one level are passed on to the next level recursively for clustering until the data-density goes below a predefined threshold.

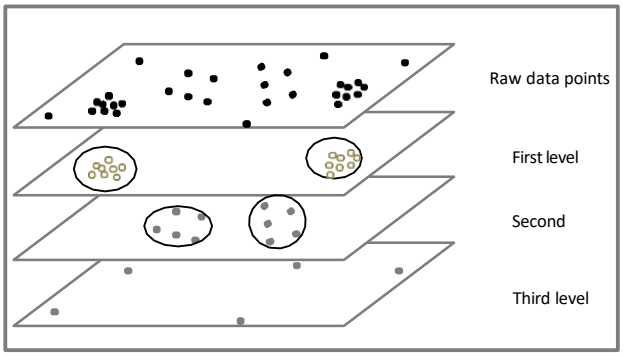


Fig. 6. The mechanism of Multi-level clustering: Original data set is clustered at different levels based on local neighborhood data density. Points of higher density are clustered at higher level, allowing the low density points to get clustered at subsequent lower levels.

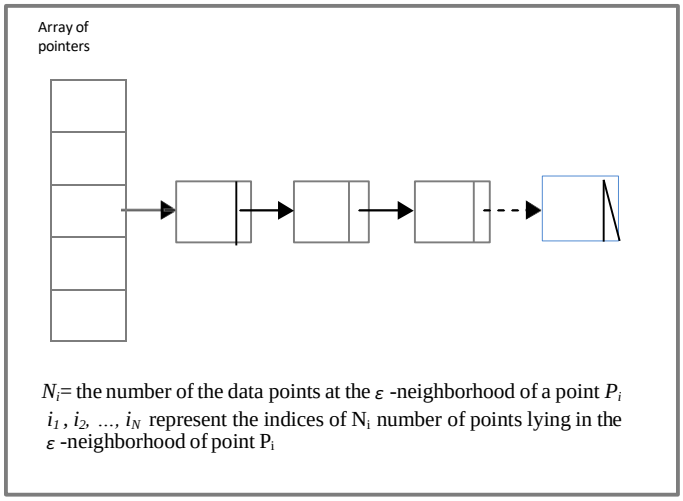


Fig. 7. Linked list representation of the PIT

In brief, the Multi-level DBSCAN algorithm has three steps in each level of recursion.

1. ε -parameter evaluation for the current recursion level.
2. Construction of the PIT to select the core point (a point with highest neighborhood density at one level) and outliers at that level.
3. The outliers obtained from one level are transferred as the input set of data for the immediately next level for

clustering.

The above three steps are recursively executed until the data density falls below a pre-defined threshold.

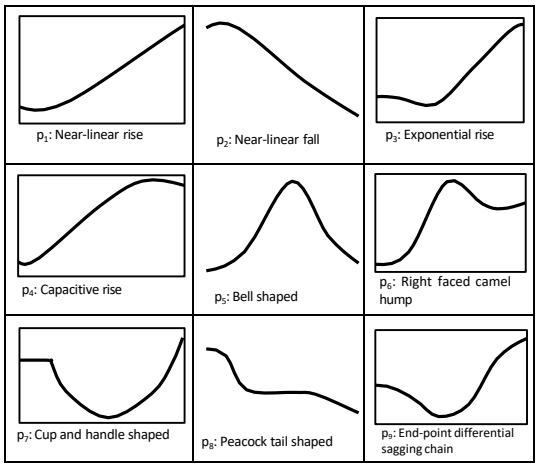


Fig. 8. The resulting cluster centers obtained by segmentation followed by multi-level DBSCAN clustering of the TAIEX 1990-2017 time-series

In Fig.8, we present 9 cluster centers (primitive patterns) obtained by multilevel DBSCAN clustering of TAIEX time-series for the period 1990-2017.

Results of Complexity Analysis: In the first step, for l temporal segments, the time required for the distance matrix computation is $O(l^2)$. The second step constructs the PIT of l entries where the i^{th} entry stores the indices of points lying in the ε-neighborhood of the i^{th} point. This also takes $O(l^2)$ time. The third step involves removal of outliers from a density-based stratum. It iterates over each entry in the PIT, taking constant time to decide, if a single entry (point) is an outlier. Hence, this step can be done in $O(l)$ time. The fourth step, i.e., traditional DBSCAN clustering of the remaining points normally has a time complexity of $O(l^2)$. With the present extension of DBSCAN, a search for a core point takes a computational overhead of $O(l)$. However, the PIT can be used to identify the ε – neighborhood points of the core point in constant time. Thus, the total time-complexity of modified-DBSCAN is also $O(l^2)$. It is important to note that the number of density based strata is significantly less compared to the number of points l, to be clustered and hence, can be considered as a constant. Thus the overall time-complexity for the proposed clustering algorithm is $O(l^2)+O(l^2)+O(l)+O(l^2)\approx O(l^2)$. Since, the distance matrix takes $O(l^2)$ space, the space complexity of the algorithm is $O(l^2)$.

5. Knowledge Encoding and Prediction by Dynamic Stochastic Automaton

(6)

A **dynamic stochastic automaton (DSA)** [38] is a 7-tuple, given by

$$F = (Q, I, Q_G, V, \delta, \delta_t, \delta_G)$$

where,

Q = A finite set of states, representing partitions,

I = A finite set of input symbols, representing primitive patterns (cluster centers) obtained by clustering of extracted segments,

Q_G = Set of next states (called a group) for a given state and a given input symbol: $Q_G : Q \times I \rightarrow P(Q)$, where $P(Q)$ denotes the power set of Q. It should be noted that Q_G has two representations. It can both be represented as a set of states or as a mapping from the Cartesian product of Q and I to the power set of Q, i.e., $P(Q)$,

V =[0, 1] represents the real valuation space in the said interval,

δ = Probabilistic transition function, representing the probability of transitions from a given state to a next state. Formally, δ describes a mapping given by $\delta : Q \times I \times Q \rightarrow V$,

δ_t = Temporal transition function $\delta : Q \times I \times Q \rightarrow \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers. δ_t represents the time required for the state to next state transition.

δ_G = Probabilistic transition function from a state to a group of states for a common occurrence of input symbol. Formally, $\delta_G : Q \times I \times Q_G \rightarrow V$

For a given pair of states $p, q \in Q$, the transition from p to q following the input symbol $a \in I$ is associated with a label of the form: $a / \delta(p, a, q)$, $\delta_t(p, a, q)$.

The model of DSA introduced here has difference from its traditional version [30] by the following three aspects.

1. Conventional automata have fixed beginning- and terminal-states, whereas in a DSA they are fixed dynamically as per users' requirement.
2. Traditional automata do not have any temporal transition function δ_t, which is required here to predict approximate time-duration of predicted structures.
3. Traditional automata do not model group transitions from a given state for a fixed input symbol, which is required here for transition-probability evaluation.
4. The computation of transition probability from state s_i to s_j in a DSA is obtained here by the following formula, which is different from traditional Stochastic Automata [20]:

$$\delta(s_i, a, s_j) = \frac{f(s_i, a, s_j)}{\sum_{s \in Q} \sum_{x \in I} f(s_i, x, s)}. \quad (7)$$

The justification of the above formula is given in the Appendix.

5. In addition, traditional stochastic automata have no temporal transition function. Here, computation of $\delta_t(s_i, a, s_j)$ is performed here by exhaustively determining the duration of all feasible transitions in the given time-series with the beginning and the terminal-states s_i and s_j respectively, due to the occurrence of a structure (cluster center) a. Suppose, the list of structures lying in the same cluster as that of structure "a" that causes state-transitions from partition s_i to partition s_j be $a_1, a_2, ..., a_n$. Then, $\delta_t(s_i, a, s_j)$ is obtained as

$$\delta_t(s_i, a, s_j) = \sum_{k=1}^{n} \tau(s_i, a_k, s_j) / n. \quad (8)$$

Here, $\tau(s_i, a_k, s_j)$ denotes the time duration for the state-transition partition s_i to partition s_j due t occurrence of the input structure a_k.

Example 2: Consider an illustrative DSA shown in Fig. 9. The computation of transition-probabilities from state s_1 to all the other states in the automaton is illustrated here. The frequency counts (i.e., number of occurrences) of transitions from state s_1 to the other states are given in Table 2. The element in Table 2 corresponding to the transition from s_1 to some state s_j is represented as a set of ordered pairs of the form (r, u), where r represents the input symbol and u represents the corresponding frequency count $f(1, r, j)$ of transitions from s_1 to s_j following the input symbol r.

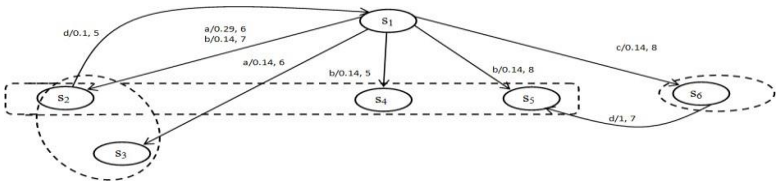


Fig. 9. A DSA to illustrate its parameters, the labels in each arc are of the form *"a/r,u"* where *a* is the input symbol, *r* is the probability of occurrence of the transition and *u* is the expected duration of the transition. The groups G_1, G_2 and G_3 corresponding to state s_1 are shown using closed dotted curves.

Table 2: Frequency counts of transitions from state s_1 to other states in the DSA of Fig. 9

From state \ To states	S_1	S_2	S_3	S_4	S_5	S_6
S_1	–	(a,2) (b,1)	(a,1)	(b,1)	(b,1)	(c,1)

Here, $Q=\{s_1, s_2, s_3, s_4, s_5, s_6\}$, $I=\{a,b,c,d\}$,
$Q_G(s_1,a)=\{s_2, s_3\}=G_1$, $Q_G(s_1,b)=\{s_2, s_4, s_5\}=G_2$,
$Q_G(s_1,c)=\{s_6\}=G_3$.

Having determined the groups of next states G_1, G_2 and G_3, corresponding to state S1, we can compute the transition probability $\delta(S1,a,S2)$ following the simplified formula presented in equation (7) as follows:

$$\delta_G(s_1,a,s_2)=\frac{f(1,a,2)}{f(1,a,2)+f(1,a,3)+f(1,b,2)+f(1,b,4)+f(1,b,5)+f(1,c,6)}$$

$$=\frac{2}{2+1+1+1+1}=\frac{2}{7}\approx 0.29.$$

Similarly, the other transition probabilities from state s_1 are computed as $\delta(s_1,a,s_3)\approx 0.14$, $\delta(s_1,b,s_2)\approx 0.14$, $\delta(s_1,b,s_4)\approx 0.14$, $\delta(s_1,b,s_5)\approx 0.14$ and $\delta(s_1,c,s_6)\approx 0.14$. We also provide the values of the temporal transition function δ_t obtained from Fig. 5 as

$\delta_t(s_1,a,s_2)=6$, $\delta_t(s_1,a,s_3)=6$, $\delta_t(s_1,b,s_2)=7$, $\delta_t(s_1,b,s_4)=5$, $\delta_t(s_1,b,s_5)=8$ and $\delta_t(s_1,c,s_6)=8$. ϒ

The algorithm for construction of a DSA is presented in Table-3.

Table 3: Pseudo code of the DSA Construction Algorithm

Input: A set $\{a_1, a_2, ..., a_l\}$ of cluster centers (structures); *k*, the total number of partitions (states in the DSA), $\delta(s_i,a,s_j)$, $\delta_t(s_i,a,s_j)$ for all i, j;
Output: A DSA representative of the given time-series

Begin
For $i=1$ to k construct a state s_i corresponding to partition z_i ;
For each temporal segment a_k, $k=1$ to l Do
Begin
If $\delta(s_i,a_k,s_j)>0$ Then construct a directed arc from state s_i to state s_j in the DSA and label it with a_k, $\delta(s_i,a_k,s_j)$ and $\delta_t(s_i,a_k,s_j)$ in order following (7) and (8).

```
        End For;
    End For;
End.
```

Forecasting Using the Dynamic Stochastic Automaton

The DSA constructed for a time-series can be used to predict the *most probable structure* (MPS)/*most probable sequence of the structures* (MPSS) between dynamically selected beginning and terminal states. The following steps are carried out for forecasting using DSA.

1. For a given beginning state s_1 and a given terminal state s_2, identify all possible structures/sequence of structures p where $\delta(s_1, p, s_2)$ is maximum. If p is a single structure, it needs no computation. However, if p is a sequence of structure of the form: $p_i\ p_j\ p_k$ then $\delta(s_1, p_i\ p_j\ p_k, s_2)$ is computed by taking the product of $\delta(s_1, p_i, s_3)$, $\delta(s_3, p_j, s_4)$ and $\delta(s_4, p_k, s_2)$, where s_3 and s_4 are intermediate states between states s_1 and s_2 such that transitions from s_1 to s_3 using input symbol p_i and transition of s_3 to s_4 using input symbol p_j and transition of s_4 to s_2 using input symbol p_k are valid.

2. If multiple structures/sequence of structures between states s_1 and s_2 are detected with the maximum probability of occurrence, report the MPS/MPSS whose duration is the smallest. The duration of the MPSS is the sum of the duration of the comprising structures.

Example 3: Let there be a DSA as shown in Fig. 10.

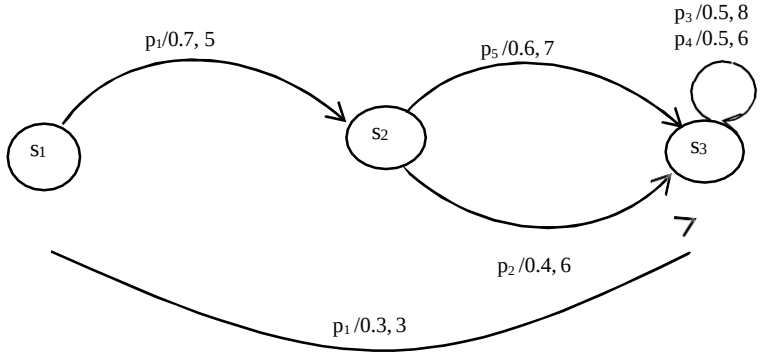


Fig. 10. Example of a time-labeled dynamic stochastic automaton

Table 4: MPS/MPSS with their probability of occurrence and expected duration for each pair of states in the DSA of Fig. 10

Terminal state / Beginning state	**S1**			**S2**			**S3**		
	MPS	δ	δ_t	**MPS**	δ	δ_t	**MPS**	δ	δ_t
S1	-	-	-	P1	0.7	5	P1,P5	0.42	5+7
S2	-	-	-	-	-	-	P5	0.6	7
S3	-	-	-	-	-	-	P4	0.5	6

In Table 4, the computation of the MPS/MPSS along with its probability of occurrence δ and expected duration δ_t, for all possible pairs of states in the DSA shown in Fig. 10. It may be noted that most probable sequence of transitions between S1 and S3 is P1, P5 with probability of occurrence= 0.7×0.6= 0.42 and duration= 5+7=12 days. In all other valid state transitions, a MPS rather than most probable sequence of structures (MPSS) is

obtained with time labels associated with the transition as its duration.

Table 5: Computational and Runtime Complexities of Segmentation, Clustering and DSA Construction Algorithms

Algorithm	Time-complexity	Space-complexity	CPU Runtime (milliseconds)
SSNS Segmentation	$O(n\log z)$	$O(n+z)$	1878.6
Multi-layered DBSCAN Clustering	$O(l^2)$	$O(l^2)$	3869.4
DSA Construction	$O(l)$	$O(k^2m)$	849.2

Results of Complexity for DSA Usage: The search time to identify the most probable structure is proportional to the number of structures/segments l. So, the time complexity is $O(l)$. The space complexity of storing the DSA in system memory in the worst case is proportional to the number of state-transitions required in computing the most probable sequence of transitions, which in turn is proportional to k^2m, where k and m denote the number of states and the number of structures respectively. So, the space-complexity required for DSA-based structure prediction is $O(k^2m)$ respectively.

Table 5 provides a summary of the results for computational and run-time complexities for SSNS segmentation, multi-level DBSCAN clustering and DSA construction. We executed the MATLAB codes on an Intel Core i5 processor with a CPU speed of 2.30 GHz running under WINDOWS-7 operating system.

6. Prediction Experiments

Prediction of a time-series involves two phases, where the former phase, referred to as training phase, is concerned with knowledge acquisition, while the latter phase, called test phase, deals with forecasting of the time-series at future time-points. Here, segmentation, clustering and DSA construction, introduced earlier, together constitutes the training phase, and forecasting using the DSA is undertaken in the test phase.

Let x_t be a point on a time-series at time t. The test phase here is employed to determine the MPS/MPSS that might emanate from the partition containing the time point value x_t. Along with MPS/MPSS, the other parameters forecasted in the test phase include the probability of occurrence and expected duration of the MPS.

In addition, the test phase prediction also includes determining the sequence of time-points in the predicted structure. This can be done in two steps: i) z-score de-normalization of the structure corresponding to the MPS such that the range of the de-normalized structure lies within the beginning and terminal partitions, ii) sampling of the de-normalized structure at a daily interval to obtain the complete sequence of data points between the beginning and the terminal states. This extrapolation technique enables us to predict the daily time-series data points lying on a structure. It also helps us in building a comparative framework with other existing prediction models using the well-known Root Mean Square Error (RMSE) [12] metric.

The experiments are performed on three daily stock time-series: TAIEX [65], NASDAQ [66] and DJIA [67] for the period 1990-2017. Here, the training and the test phases include January 1 to October 31 and November 1 to December 31 respectively. In Fig. 11, we present one illustrative case of prediction experiment, undertaken on the TAIEX time-series in 2017. The query input for the present problem is to predict the MPSS along with its probability of occurrence and approximate duration, when the beginning and the terminal states are S5 and S3, corresponding to partitions P_5 and P_3 respectively in TAIEX 1995 time- series.

To examine the performance of different competitive algorithms in solving the above problem, we employed 15 different combinations of segmentation and clustering algorithms (listed in Table 6), of which only two well-known combinations: SWAB segmentation followed by k-means clustering, and FSW segmentation followed by fuzzy C-means clustering are shown in Fig. 11 for illustration. The other combinations are not shown here to avoid clumsiness of the figure. In Fig. 7, we used vertical arrows to indicate the structure boundaries. It is noteworthy that the structure predicted by the proposed algorithm in Fig. 11 includes 3 vertical arrows, thus containing three structures: p_7, p_2 and p_9, the shapes of which are

available in Fig. 8. The DSA constructed from TAIEX data for 2017 January to October, given in Fig. 12, supports the transitions from state S5 to S6 using structure p_7, from state S6 to state S3 using structure p_2, and S3 to S3 using structure p_9 respectively.

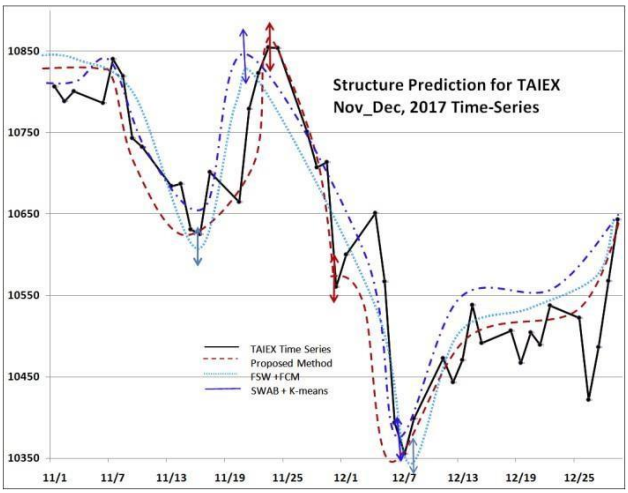


Fig. 11. Results of structure prediction for TAIEX 2017, November 1 to December 31 time-series by three methods: Proposed, FSW + Fuzzy C-means (FCM) and SWAB + k-means clustering

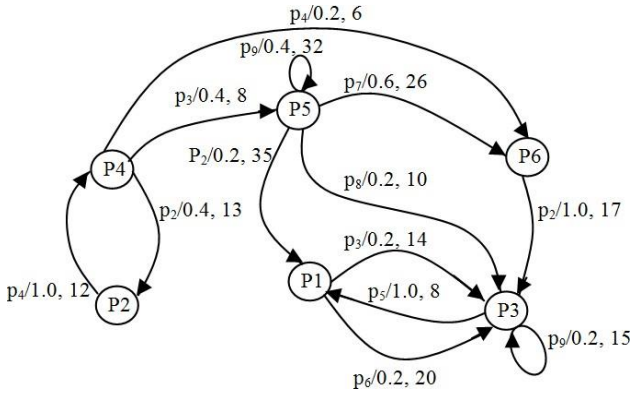


Fig. 12. The DSA obtained from TAIEX time-series for the periods: January to October in 2017 for Fig. 11

Similarly, the structure predicted by FSW plus FCM in Fig. 11 includes 2 vertical arrows, and thus there exists 3 structures: p_2, p_5 and p_9 , the shapes of which are slightly different from those listed in Fig. 5. The difference in shape is apparent as the structures generated by FSW and SWAB segmentation are different from those obtained by SSNS segmentation. The difference in structure shapes obtained by different segmentation and clustering algorithms is also evident from Fig. 11.

It is to be noted that the structures p_7-p_2-p_9 selected between states S5 and S3 in Fig. 12 has the highest probability of $0.6 \times 1.0 \times 0.2 = 0.12$, among all possible sequence of transitions from S5 to S3. The time required for the sequences p_7-p_2-p_9 is obtained as 26+ 17+ 15==58 days.

7. Performance Analysis

The analysis includes both the overall and the individual performances of the proposed segmentation and clustering algorithms.

7.1 Performance Analysis by RMSE and Prediction Accuracy

Here, we examine the overall performance in structure prediction by composite selection of SSNS and extended DBSCAN algorithms among their competitors. In Table 6, we use two metrics: percentage accuracy in structure prediction averaged over 3 time series (TAIEX, NASDAQ and DJIA), and RMSE averaged over

10 years across 3 time-series. It is apparent from the Table that RMSE averaged over 2 months' prediction in 10 years (1990-1999) and across 3 time-series indicated earlier is least for the proposed method, justifying its significance over others. In addition, the percentage accuracy in structure prediction averaged over 3 time-series is the largest by the proposed approach in comparison to its competitors.

*7.2 **Precision, Recall and Accuracy***

To evaluate the performance of the proposed model for structure prediction, we use three well-known metrics: Precision (*P*), Recall (*R*) and Accuracy (*A*). For calculating these metrics we first construct a confusion matrix **C**, where the element $\mathbf{C}_{i,j}$ corresponding to the i^{th} row and j^{th} column is equal to the number of times that structure *j*, is predicted as structure *i*. Given a confusion matrix **C**, the precision *P* and recall *R* for structure *i* are computed as follows:

$$P = \frac{\mathbf{C}_{i,i}}{\sum_{\forall j} \mathbf{C}_{i,j}} \text{ and } R = \frac{\mathbf{C}_{i,i}}{\sum_{\forall j} \mathbf{C}_{j,i}}. \quad (9)$$

The overall precision and recall are computed as the mean of the precision and recall measures of individual structures. The accuracy *A* from a given confusion matrix is computed as follows:

$$A = \frac{\sum_{\forall i} \mathbf{C}_{i,i}}{\sum_{\forall i} \sum_{\forall j} \mathbf{C}_{i,j}}. \quad (10)$$

In Table 7, the overall evaluation measures obtained from the three time-series is presented. It is evident from the Table, that the precision, recall and accuracy of structure prediction, averaged over the three time- series TAIEX, NASDAQ and DJIA are very encouraging with values of 91.24% precision, 93.42% recall and 88.05% overall accuracy.

Table 6: Percentage accuracy and RMSE averaged over 3 time-series

Method used	Segmentation Algorithm used	Clustering Algorithm used	% accuracy in structure prediction averaged over 3 time-series	RMSE averaged over 10 years across 3 time-series
Proposed Method	**SSNS**	**Extended DBSCAN**	**88.05%**	**23.4**
Method 2	SSNS	FCM	70.2%	31.2
Method 3	SSNS	k-means	69.2%	33.8
Method 4	SSSW	Extended DBSCAN	73.1%	29.8
Method 5	SSSW	FCM	72.4%	30.2
Method 6	SSSW	k-means	70.2%	31.48
Method 7	SW	Extended DBSCAN	71.1%	28.2
Method 8	SW	FCM	69.2%	29.2
Method 9	SW	k-means	69.7%	30.8
Method 10	FSW	Extended DBSCAN	78.2%	29.2
Method 11	FSW	FCM	77.8%	31.4
Method 12	FSW	K-means	80.4%	32.12
Method 13	SWAB	Extended DBSCAN	81.8%	28.6
Method 14	SWAB	FCM	76.2%	29.4
Method 15	SWAB	k-means	72.14%	30.5

Table 7: Evaluation metrics for assessing the performance of the proposed model on TAIEX, NASDAQ and DJIA time-series

Evaluation measure / Time-Series	Precision	Recall	Accuracy
TAIEX	91.24%	93.42%	88.05%
NASDAQ	92.6%	90.8%	86.60%
DJIA	88.4%	89.2%	82.5%

Table 8: Optimal parameter setting by sensitivity analysis

Window length L in SSNS Algorithm (number of transitions in a Window)	DBSCAN parameters			% Accuracy of Structure Prediction Averaged over 3 Time-Series	RMSE Averaged over 3 Time-Series
	s in %	Stopping threshold number of points	Length of the structure b		
3	8	6		78.02	31.40
5	10	4	10	80.2	34.82
5	**10**	**4**	**12**	**88.05**	**30.24**
5	10	4	14	85.20	32.14
7	8	6	14	80.24	40.8
7	10	4	14	65.23	59.23

*7.3 **Sensitivity Analysis***

For sensitivity analysis, we consider three parameters: i) window length L in SSNS algorithm, ii) parameter s, iii) stopping threshold number of points and iv) parameter b used in extended hierarchical DBSCAN algorithm. In Table 8, the average RMSE across all experiments over 3 time-series and % accuracy of structure prediction, averaged over 3 time-series are examined. It is observed from the Table that average RMSE has the smallest value, while average accuracy has the highest value for L=5, s=10%, stopping threshold number of points =4 and the length of the structure b=12 points.

*7.4 **Statistical t-test***

Let $\rightarrow = [x]$, i=1 to 48 be a given time-series of 48 data points, corresponding to 48 trading days in two months during November-December. Now, consider [y_1, y_2, y_3, ..., y_{48}] and [u_1, u_2, u_3, .., u_{48}] be two sets of predicted data points obta→ined by the reference (proposed) al,gorithm R and a test algorithm A taken over the same time interval as in $X = [x]$. We compute errors e and e with respect to the actual series by

$$e_i = x_i - y_i, i = 1 \text{ to } 43$$

and $e_i' = x_i - u_i, i = 1 \text{ to } 43.$

Now, we compute mean M_1 of the error sample e_i and mean M_2 of the error sample e_i' for i=1 to 43. Next we compute standard deviation of the two samples e_i and e_i', for i=1 to 48 and call them σ_1 and σ_2. We also compute standard error, defined by

$$S = \sqrt{\frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}}$$

where, n_1 and n_2 denote sample size of e_i and e_i'. Here, $n_1=n_2= 43$.

Table 9: Results of 2-Sample t^2 test, considering the proposed technique as the Reference Algorithm

Algorithm A	$M_1 - M_2$ obtained for 43 data points over the months: November-December of 2017 in TAIEX time-series	Computed value of S for the error samples of the same year	t^2-test criterion satisfied/ failed
SSNS-FCM-DSA	51.8	0.41	Failed
SSNS-k-MEANS-DSA	52.6	0.42	Failed
SSSW-Ext-DBSCAN-DSA	55.1	0.40	Failed
SSSW-FCM-DSA	53.2	0.41	Failed
SSSW-k-MEANS-DSA	55.9	0.42	Failed
SW-Ext-DBSCAN-DSA	53.8	0.42	Failed
SW-FCM-DSA	54.2	0.44	Failed
SW-k-MEANS-DSA	56.4	0.45	Failed
FSW-Ext-DBSCAN-DSA	56.1	0.47	Failed
FSW-FCM-DSA	54.1	0.46	Failed
FSW-k-MEANS-DSA	56.1	0.47	Failed
SWAB-Ext-DBSCAN-DSA	55.8	0.47	Failed
SWAB-FCM-DSA	55.2	0.49	Failed
SWAB-k-MEANS-DSA	57.1	0.49	Failed

Lastly, we test whether the mean difference of the two populations: $\mu_1 - \mu_2$ satisfy the inequality:

$$(M_1 - M_2) - S \times \mathrm{CL}_{95\%} < \mu_1 - \mu_2 < (M_1 - M_2) + S \times \mathrm{CL}_{95\%}$$

where $\mathrm{CL}_{95\%}$ = t-statistic determined by 95% confidence level. For, 95% confidence level, and degrees of freedom =43-1-1=41, $\mathrm{CL}_{95\%}$ in t-table is found to be 2. Here, $\mu_1 - \mu_2 = 50.1$ is calculated from error population of e_i and $e_i^{/}$ over 245 TAIEX trading days in 2017.

Now, consider the hypothesis *H_o: Both the sample error distributions are similar*.

If the given inequality is satisfied, we understand that μ_1 is statistically closer to μ_2 else μ_1 and μ_2 are

different. The 2-sample t^2–test, introduced above, is undertaken to obtain the results by the proposed algorithm and its competitive realizations.

It is apparent from Table 9 that there is enough statistical evidence to reject the null hypothesis H_o. Similar results are obtained for all the years between [1990, 2017] on TAIEX time-series prediction. Thus it is confirmed that the proposed algorithm has statistically significant performance with its competitors.

7.5 ***Performance of SSNS Algorithm***

Performance of a segmentation algorithm usually is measured by comparing the segment boundaries obtained by an algorithm with manual segmentation of the same time-series. Now to measure the degree of matching of the segmentation boundaries produced by an algorithm with a reference algorithm, certain metrics need to be measured. The following three metrics: i) match-score (m), ii) number of extracted segment boundaries (S) and iii) percentage of segment boundaries matched ((m/s) × 100%) have been employed to compare the performance of the proposed SSNS algorithm with a few standard algorithms, such as Top-down [51], Bottom-up [33], Sliding Window (SW) [56], FSW [41], and SWAB [49] algorithms. The parameter match-score determines how many segment boundaries extracted by an algorithm match with the segment boundaries obtained by hand-crafted segmentation.

Experiments are performed on 150 consecutive points in the TAIEX close-price time-series. Table 10 offers

a summary of results [38], indicating that the SSNS algorithm outperforms its competitors with respect to all the three metrics. While percentage segmentation boundaries matched by SSNS is 76.5%, the same for SWAB, one of the most promising algorithms, is 75%. The other four algorithms have relatively worse performance in this metric. Same situation is noticed in Table 10 with regard to m and S. It is noted that SSNS identifies fewer segment boundaries, which are true boundaries of the segments in the time-series. However, number of segment boundaries obtained by most of the algorithms is unexpectedly large.

Table 10: Comparison of time-series segmentation algorithms b matching of generated segments with hand-crafted ground-truth segments

Algorithm	Number of matched segment boundaries (m)	Number of segment boundaries obtained (S)	Percentage of segment boundaries matched $(m / S) \times 100$
Top-down [51]	7	21	33.3
Bottom-up [33]	7	19	36.8
SW [56]	11	26	42.3
FSW [41]	9	18	50.0
SSNS	**13**	**17**	**76.5**
SWAB	12	16	75.0

*7.6. **Performance of the Extended DBSCAN Clustering Algorithm***

A clustering algorithm is said to have good performance, if the intra-cluster separating distance of all points in the clusters with their cluster centre is small, and the inter-cluster separating distance, indicating the distances between each pair of cluster centres is large. Two metrics, called *sum of squared intra-cluster distance* (SSID) and the *sum of inter-cluster distances* (SICD) [7] have been used to measure the relative performance of the proposed hierarchical DBSCAN algorithm with other well-known clustering algorithms, including k-means clustering [24], fuzzy c means (FCM) clustering [6] and traditional DBSCAN [19] algorithms. The SISD and SICD metrics are now formally defined below.

Let there be m clusters with cluster centers $(Z_1, Z_2, ..., Z_m)$. Then SSID index is defined by

$$SSID = \sum_{i=1}^{m} \sum_{X \in S_i} \left\| X - Z_i \right\|, \tag{11}$$

where, S_i denotes the set of points in the i^{th} cluster, and || . || is a Euclidean norm. The SICD index measures the distance between each pair of cluster centers, and is obtained by

$$SICD = \sum_{i \neq j} \left\| Z_i - Z_j \right\| \tag{12}$$

for all i, j ∈ [1, m]. A large SICD and a small SSID jointly indicates a good clustering.

Table 11 provides the relative performance of four clustering algorithms using SICD and SSID metrics, tested on the TAIEX close-price time-series for the period 1990 to 2017. It is apparent from the Table that Multi-level DBSCAN algorithm outperforms its three competitors with respect to both the metrics.

Table 11: Comparison of existing clustering algorithms with the proposed technique based on the SSID and SICD metrics

Algorithm / Performance index	k-means [24]	Fuzzy c-means [6]	DBSCAN [23]	**Multilevel DBSCAN**
SSID	15.8917	11.2476	12.5371	**10.6215**
SICD	1186.9	1147.4	1222.2	**1293.2**

8. Conclusions

The paper introduced a novel scheme to encode a time-series by an automaton using three main steps: segmentation, clustering and DSA construction. These steps are usually performed in the training phase.in the test phase, the automaton is consulted to predict the most probable structure/sequence of structures between any two given states (partitions) along with the probability of the MPS/MPSS and their expected duration. The online prediction scheme proposed here is useful, as it does not require any search and thus is very time-efficient for forecasting applications.

The accuracy of the structures predicted in the test phase here greatly depends on the accurate encoding of the time-series by the DSA. The DSA here indirectly encodes a time-series by a sequence of repeatedly occurring structures. An accurate encoding of the time-series depends on the structures selected to describe the series, which in turn depends on the performance of the segmentation and clustering algorithms.

In the performance analysis section, it is clearly demonstrated that SSNS segmentation and the proposed hierarchical DBSCAN clustering algorithms outperform their competitors significantly. Extensive analysis of the proposed algorithms further reveals that the selected combination: SSNS segmentation and extended hierarchical DBSCAN algorithm outperforms all other 15 competitors with respect to average percentage accuracy in structure prediction and RMSE, thereby justifying its selection. Consequently, application of the principles of structure prediction in economic time series would provide an assistive support to the users for trading and investment planning.

Although tested for economic time-series, the principles narrated for structure prediction are equally applicable in many other time-series as well. For example, the technique may help a weather-forecast scientist to forecast the possible recurrence of a catastrophic change in the coming future. The proposed technique would also serve ecologists to determine possible recurrent extinction of an endangered species due to lack of environmental resources.

9. Appendix

Computation of transition probabilities in a DSA

The proposed DSA has the following properties.

Property 1: *The sum of the transition-probabilities from a given state q to all possible groups* $G \in Q_G(q,x)$ *due to occurrence of any input symbol* $x \in I$ *is unity, i.e.,*

$$\sum_{x \in I} \sum_{G \in Q_G(q,x)} \delta_G(q,x,G) = 1. \qquad \text{(A.1)}$$

Property 2: *The transition-probability* $\delta(q,x,q')$ *from a given state q to the next state* q' *due to the occurrence of an input symbol x is obtained by the product of two probabilities, i)the transition-probability* $\delta_G(q,x,Q_G(q,x))$ *from q to the group* $Q_G(q,x)$ *due to the input symbol x, and ii) the conditional group transition-probability* $\delta(q,x,q' \mid Q_G(q,x))$ *from q to* q' *due to x, presuming that the transition to the group* $Q_G(q,x)$ *has already taken place, i.e.,*

$$\delta(q,x,q') = \delta_G(q,x,Q_G(q,x)) \times \delta(q,x,q' \mid Q_G(q,x)). \qquad \text{(A.2)}$$

After we construct a DSA, we need to assign transition probabilities. The following steps are required to evaluate the transition probabilities.

1. Computation of $\delta_G(s_i,a,Q_G(s_i,a))$: The probability of transition $\delta_G(s_i,a,Q_G(s_i,a))$ from state $s_i \in Q$ to the group of states $Q_G(s_i,a)$ following input symbol $a \in I$ is obtained as

$$\delta_G(s_i,a,Q_G(s_i,a)) = \frac{\sum_{s \in Q_G(s_i,a)} f(s_i,a,s)}{\sum_{s \in Q} \sum_{x \in I} f(s_i,x,s)}. \qquad \text{(A.3)}$$

where $f(s_i,a,s)$ denotes the frequency count of the state-transitions from a given state s_i the occurrence of an input symbol a. to a state s due to

2. Computation of $\delta(s_i, a, s_j \mid Q_G(s_i, a))$: Similarly, we obtain the conditional probability $\delta(s_i, a, s_j \mid Q_G(s_i, a))$ that a transition takes place from the state $s_i \in Q$ to the state $s_j \in Q_G(s_i, a)$ following the input symbol a, presuming that the group-transition $Q_G(s_i, a)$ has taken pace, is obtained by

$$\delta(s_i, a, s_j \mid Q_G(s_i, a)) = \frac{f(s_i, a, s_j)}{\sum_{s \in Q_G(s_i, a)} f(s_i, a, s)}. \quad \text{(A.4)}$$

3. Computation of $\delta(s_i, a, s_j)$: Finally, the probability of a transition $\delta(s_i, a, s_j)$ from state s_i to state s_j due to the input symbol a, where $s_i, s_j \in Q$ and $a \in I$, is computed below following Property 2.

$$\begin{aligned}\delta(s_i, a, s_j) &= \delta_G(s_i, a, Q_G(s_i, a)) \times \delta(s_i, a, s_j \mid Q_G(s_i, a)) \\ &= \frac{\sum_{s \in Q_G(s_i, a)} f(s_i, a, s)}{\sum_{s \in Q} \sum_{x \in I} f(s_i, x, s)} \times \frac{f(s_i, a, s_j)}{\sum_{s \in Q_G(s_i, a)} f(s_i, a, s)} \\ &= \frac{f(s_i, a, s_j)}{\sum_{s \in Q} \sum_{x \in I} f(s_i, x, s)}. \end{aligned} \quad \text{(A.5)}$$

REFERENCES

[1] J. Abonyi, B. Feil, S. Nemeth, and P. Arva, "Modified Gath-Geva clustering for fuzzy segmentation of multivariate time-series," Elsevier, Fuzzy Sets and Systems, vol. 149, no. 1, pp. 39-56, Jan. 2005.

[2] B. Abraham and J. Ledolter, Statistical Methods for Forecasting, John Wiley and Sons, Inc., Hoboken, NJ, USA, 2008.

[3] U. Appel and A.V. Brandt, "Adaptive Sequential Segmentation of Piecewise Stationary Time Series," Information Science, vol. 29, no. 1, pp. 27-56, 1983.

[4] O. Barnea, A. R. Solow, and L. Stone, "On Fitting a Model to a Population Time Series with Missing Values,"Israel Journal of Ecology and Evolution, vol. 52, pp. 1-10, 2006.

[5] R. Bellman, "On the approximation of curves by line segments using dynamic programming," Communications of the ACM, volume 4, Issue 6, Page 284, June 1961, NY, USA.

[6] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York and London, 1987.

[7] J.C. Bezdek and N.R. Pal, "Some New Indexes of Cluster Validity," IEEE Trans. Syst. Man Cybern., Part-B, vol. 28, no. 3, pp. 301–315, 1998.

[8] Diptendu Bhattacharya, Amit Konar and Pratyusha Das, "Secondary factor induced stock index time-series prediction using self-adaptive interval type-2 fuzzy sets," Neurocomputing, vol.171, pp. 551-568, 2016.

[9] G. F. Bryant and S.R. Duncan, "A Solution to the Segmentation Problem Based on Dynamic Programming," Proc. Third IEEE Conf. Control Applications (CCA '94), pp. 1391-1396, 1994.

[10] Q. Cai, D. Zhang, W. Zheng and S.C.H Leung, "A New Fuzzy Time Series Forecasting Model Combined with Ant Colony Optimization and Auto-regression," Elservier, Knowledge-Based Systems, vol. 74, pp. 61-68, Jan. 2015.

[11] S.M. Chen, and J.R. Hwang, "Temperature Prediction Using Fuzzy Time Series," IEEE Trans. Syst., Man, Cybern., Part-B, vol. 30, no. 2, pp. 263-275, Apr. 2000.

[12] S.M. Chen, C.D. Chen, "TAIEX Forecasting Based on Fuzzy Time Series and Fuzzy Variation Groups,"IEEE Trans. Fuzzy Syst.,vol. 19, no. 1, pp. 1-12, Jan. 2011.

[13] S. M. Chen, H.P. Chu, and T.W. Sheu, "TAIEX Forecasting Using Fuzzy Time Series and Automatically Generated Weights of Multiple Factors," IEEE Trans. Syst. Man Cybernet. Part A Syst. Hum.,vol. 42, no. 6, pp. 1485–1495, Apr. 2012.

[14] S.M. Chen, and P.Y. Kao, "TAIEX Forecasting Based on Fuzzy Time Series, Particle Swarm Optimization Techniques and Support Vector Machines," Elsevier, Inform. Sci. vol. 247, pp. 62-71, Oct. 2013.

[15] F.L. Chung, T.C. Fu, V. Ng, and R.W.P. Luk, "An Evolutionary Approach to Pattern-Based Time Series Segmentation," IEEE Trans. Evolutionary Computation, vol. 8, no. 5, pp. 471-489, Oct. 2004.

[16] J.T. Connor, R.D. Martin, and L.E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction," IEEE Trans. Neural Networks, vol. 5, no. 2, pp. 240-254, Mar. 1994.

[17] G. Das, K.I. Lin, H. Mannila, G. Renganathan and P. Smyth, "Rule Discovery From Time Series," in Proc. 4th International Conference on Knowledge Discovery and Data Mining, pp. 16-22, 1998.

[18] H. A. Dau, N. Begum, E. Keogh, "Semi-Supervision Dramatically Improves Time Series Clustering under Dynamic Time Warping," Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16), pp.999-1008, Indiana, USA October 24 -28, 2016.

[19] C. Ding and X. He, "Cluster aggregate inequality and multi-level hierarchical clustering,"PKDD'05 Proc. of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases, pp.71-83, Porto, Portugal, October 03 - 07, 2005.

[20] E.R. Dougherty, and C.R. Giardina, Mathematical Methods for Artificial Intelligence and Autonomous Systems, Prentice-Hall international editions publisher, the University of Michigan, USA, 1988.

[21] R. D. Edwards, J. Magee, and W.H.C. Bassetti, "The Dow Theory," in Technical Analysis of Stock Trends, 8th ed., 2000 N.W. Corporate Blvd., Boca Raton, Florida: CRC Press LLC, 2001, pp. 21-22.

[22] P. Esling and C. Agon, "Multiobjective Time Series Matching for Audio Classification and Retrieval," IEEE Trans. on Audio Speech and Language Processing, vol. 21, no. 10, pp. 2057-2072, May 2013.

[23]M. Ester, H. P. Krigel, J. Sander and X. Xu, " A density based algorithm for discovering clusters in large spatial databases with noise," Proc. of the Second Int. Conf. on Knowledge Dicovery and Data Mining (KDD-96), AAAI Press, pp. 226-231, 1996.

[24]E.W. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," Biometrics, vol. 21, no. 3, pp. 768–769,1965.

[25]T. C. Fu, "A review of time series data mining," Engg. Applications of Artifi. Intelli., vol. 24, pp. 164-181, 2011.

[26]E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations," IEEE Trans. Pattern Analysis, Machine Intelligence, vol. 32, no. 12, pp. 2232-2245, Dec. 2010.

[27] B. D. Fulcher and N. S. Jones, "Highly Comparative Feature-Based Time-Series Classification," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 12, pp. 3026-3037, December, 2014.

[28] Jan G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," Elsevier, International Journal of Forecasting, vol. 22, pp. 443-473, 2006.
[29] M. Han, J. Xi, S. Xu and F.L. Yin, "Prediction of Chaotic Time Series Based on the Recurrent Predictor Neural Network," IEEE Trans. on Signal Processing, vol. 52, no. 12, pp. 3409-3416, Dec. 2004.
[30] H. Hiisila, Segmentation of time series and sequences using basis representations, Ph.D. Thesis, Helsinki University of Technology, 2007.
[31] A. Jalil, and M. Idrees, "Modeling The Impact Of Education On The Economic Growth: Evidence From Aggregated And Disaggregated Time Series Data Of Pakistan," Elsevier, Econ. Model, vol. 31, pp. 383-388, 2013.
[32] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: A Survey and Novel Approach," Data Mining in Time Series Databases, second ed. World Scientific, 2004.
[33] E. Keogh and P. Smyth, "A Probabilistic Approach to Fast Pattern Matching in Time Series Databases," Proc. ACM SIGKDD'97, pp. 20-24, 1997.
[34] E. Keogh and J. Lin, "Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research," Knowledge and Information Systems, vol. 8, no. 2, pp. 154-177, Aug. 2005.
[35] E. J. Keogh and M. J. Pazzani, "Relevance Feedback Retrieval of Time Series Data," Proc. SIGIR '99 Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 183-190, 1999.
[36] E. Keogh, J. Lin and A. Fu, "HOT SAX: Finding the Most Unusual Time Series Subsequence: Algorithms and Applications," Proc. of the Sixth International Conference on Data Mining (ICDM), January 2005.
[37] E. Keogh, S.Chu, D. Hart and M. Pazzani, "An online algorithm for segmenting time series," Proc. IEEE International Conference on Data Mining (ICDM, 2001), pp. 289-298, 2001.
[38] A. Konar and D. Bhattacharya, Time-Series Prediction and Applications, Springer, 2017.
[39] X. Lin, Z. Yang and Y. Song, "Short-term Stock Price Prediction Based on Echo State Networks," Elsevier, Expert Systems with Applications, vol. 36, no. 3, pp. 7313-7317, Apr. 2009.
[40] Q. Lin, C. Hammerschmidt, G. Peellegrino and S. Verwer, "Short-term Time Series Forecasting with Regression Automata," Proc. of ACM SIGKDD, Aug. 13-17, 2016.
[41] X. Liu, Z. Lin, and H. Wang, "Novel Online Methods for Time Series Segmentation," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 12, pp. 1616-1626, Dec. 2008.
[42] L. M. López, I. F. Ruiz, R. M. Bueno, F. T. Ruiz, "Dynamic Discretization of Continuous Values from Time Series," European Conference on Machine Learning, ECML 2000, pp. 280-291, 2000.
[43] L. M. Lopez and M. S. D. Cardonab, "Using probabilistic finite automata to simulate hourly series of global radiation," Solar Energy, vol. 74 pp. 235–244, 2003.
[44] J. Lundberg, "Lifting The Crown-Citation Z-score", Elsevier, Journal of Informetrics, vol. 1, pp. 145-154, 2007.
[45] Tom M. Mitchell, Machine Learning, McGraw-Hill, New York, USA, 1997.
[46] R. Mall, R. Langone and J. A.K. Suykens, "Multilevel Hierarchical Kernel Spectral Clustering for Real-Life Large Scale Complex Networks," PLOS ONE, Vol 9, Issue 6, June 2014.
[47] D.T. Mirikitani and N. Nikolaev, "Recursive Bayesian Recurrent Neural Networks for Time-Series Modeling," IEEE Trans. Neural Networks, vol. 2, no. 2, pp. 262-274, Feb. 2010.
[48] Palpanas T, Vlachos M, Keogh E J, Gunopulos D, "Streaming time series summarization using user-defined amnesic functions," IEEE Trans Knowl Data Engg., vol. 20, no. 7, pp. 992–1006, 2008.
[49] T. Rakhanmanon, B. Campa, A. Mueen, G. Batiska, B. Westover, Q. Zhu, J. Zakaria and E. Keogh, "Searching and mining trillions of time series subsequences under Dynamic Time Warping," Proc. of ACM SIGKDDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 262-270, 2012.
[50] C. Ratanamahatana, E. J. Keogh, A. J. Bagnall and S.Lonardi," A novel bit level time series representation with implication of similarity search and clustering," Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining (Proceeding PAKDD'05), pp. 771-777, Hanoi, Vietnam, May 18 - 20,2005.
[51] H. Shatkay and S.B. Zdonik, "Approximate Queries and Representations for Large Data Sequences," Technical Report CS-95-03, Dept. of Computer Science, Brown Univ., 1995.
[52] J. Shieh and E. Keogh, "iSAX: indexing and mining terabyte sized time series," KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp.623-631, Las Vegas, Nevada, USA , August 24 - 27, 2008.
[53] M. Small and C.K. Tse, "Detecting Determinism in Time Series: The Method of Surrogate Data," IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, vol. 50, no. 5, pp. 663-672, May 2003.
[54] T. Warren Liao, "Clustering of time series data – a survey," Elsevier, Pattern Recognition, vol. 38, no. 11, pp. 1857-1874, Nov. 2005.
[55] L. Wei and E. Keogh, "Semi-supervised time series classification," KDD '06 Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 748-753, Philadelphia, PA, USA, August 20 - 23, 2006.
[56] C.L. Wu, and K.W. Chau, "Prediction of Rainfall Time Series Using Modular Soft Computing Methods," Elsevier, Engineering Applications of Artificial Intelligence, vol. 26, pp. 997-1007, 2013.
[57] Q. Xie, C. Pang, X. Zhou, X. Zhang, K. Deng, "Maximum error-bounded Piecewise Linear Representation for online

stream approximation," The VLDB Journal, Springer, 2014.
[58] Z. Xiong, R. Chen, Y. Zhang, and X. Zhang, “Multi-density DBSCAN Algorithm Based on Density Levels Partitioning,” Journal of Information & Computational Science, vol. 9, no. 10, pp. 2739-2749, 2012.
[59] R. Xu and D. Wunsch II, “Survey of Clustering Algorithms,” IEEE Trans. Neural Networks, vol. 16, no. 3, May 2005.
[60] H.K. Yu, “Weighted Fuzzy Time Series Models for TAIEX Forecasting,” Elsevier, Physica A, vol. 349, no. 3-4, pp. 609-624, Apr. 2005.
[61] J. Zakaria, A. Mueen and E. Keogh, "Clustering Time Series Using Unsupervised-Shapelets," ICDM '12 Proceedings of the 2012 IEEE 12th International Conference on Data Mining, pp. 785-794, December 10 - 13, 2012.
[62] J. Qi, R. Zhang, K. Ramamohanarao, H. Wang, Z. Wen, D. Wu, "Indexable online time series segmentation with error bound guarantee," World Wide Web, Springer, volume 18, issue 2, pp. 359–401, 2015.
[63] Y. Zhao and S. Zhang, “Generalized Dimension Reduction Framework for Recent-Biased Time Series Analysis,” IEEE Trans. Knowledge and Data Engineering, vol. 18, no. 2, pp. 231-244, Feb. 2006.
[64] H. Y. Zhao, G. X. Li, H. Zhang, "An improved algorithm for segmenting online time series with error bound guarantee," International Journal of Machine Learning and Cybernetics, vol. 7, Issue 3, pp 365–374, June 2016.
[65] TAIEX [Online]. Available: http://www.twse.com.tw/en/products/indices/tsec/taiex.php
[66] NASDAQ [Online]. Available: http://www.nasdaq.com
[67] DJIA [Online]. Available: https://www.google.com/finance?cid=983582