Imperial College
London

Imperial College London

Department of Mathematics

Reservoir Computing in Alpha Forecasting of Foreign Exchange Market

Author: Yingzhuo He (CID: 02272118)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2022-2023

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Signature: [illegible]

Date: 2023/09/04

Acknowledgements

I would like to thank Edward White, Hendrik Muer, Amelia French, Qin Li, Oleg Groshev, Damon Lee, Roel Oomen, Hassaan Mohamed, Sophie Etheridge, Quentin Lam, Benedict Cater, and Andrea Bianchi from electronic FX Spot Trading team at Deutsche Bank for your professional guidance and generous help. It is my luck to have the chance working with you.

I also want to express my gratutide to my internal supervisor Lukas Gonon. Thank you for always providing solid technical supports, inspiring research ideas and answering my questions whenever needed.

Great thanks to my friends Xuanliang Wang and Chaolei Li, thank you for trying to help with my versatile stupid questions and tolerating my endless casual chats. Also a huge thank you to Wanci Li, Jinhan Sheng, Dekai Wu, and Rong Su for spending time with me on my crazy dining and city walk schedules.

Last, to my fluffy friends Fumo and Salem, thank you for letting me pat you, it is such a huge relief.

Abstract

Deep learning algorithms have long been implemented in financial industry for pricing, return prediction and so on. However, some complicated models may lead to expensive computational cost and time cost. In this thesis, we investigate two reservoir computing(RC) methods: echo state network(ESN) and randomized signature, which have similar structures as recurrent neural network but can be trained faster. Both models are implemented to forecast forward return of middle rate of USDMXN over the future 50 ticks. For the convenience of comparison, a simple linear regression serves as a baseline. Results show that even features are not informative enough, both RC algorithms can have some predictiveability. When informative features are added, significant improvement of performance exists in all the models and RC methods can outperform benchmark model. Among different RC methods, ESN using PCA and elastic net achieves highest average R^2 value while randomized signature might slightly underperform ESN but has more efficient initialization process. In all, we can conclude that RC methods have the capability in forward rate forecasting and is superior than linear regression under the same condition.

Contents

1 Literature review **7**

2 Methodology **9**

2.1 Order Book . . . 9

2.1.1 Ask & Bid Rates . . . 9

2.1.2 Middle Rate & Alpha . . . 9

2.1.3 Ask & Bid Quantities & Counts . . . 10

2.2 Simple Linear Regression . . . 10

2.2.1 Model Structure . . . 10

2.2.2 Coefficients Estimation . . . 11

2.2.3 Results Prediction . . . 12

2.2.4 Variance Inflation Factor . . . 12

2.3 Reservoir Computing . . . 12

2.3.1 Recurrent Neural Network . . . 13

2.3.2 Echo State Network . . . 15

2.3.3 Echo State Property . . . 16

2.3.4 Randomized Signature in RC . . . 17

2.3.5 Regression Methods . . . 18

2.3.6 Principal Component Analysis . . . 20

2.3.7 RC Method with PCA Embedding . . . 22

2.4 Evaluation Metric . . . 23

3 Experiment **25**

3.1 Data . . . 25

3.1.1 Raw data . . . 25

3.1.2 Data Preprocessing . . . 25

3.2 Modelling Details . . . 28

3.2.1 Model Setup . . . 28

3.2.2 Implementation Method . . . 29

3.3 Empirical Results . . . 29

3.3.1 Numerical Results of Single Variable Models . . . 29

3.3.2 Numerical Results of Multiple Variable Models . . . 34

3.3.3 Investigative Experiments & Optimal Multiple Variable RC Methods . . . 37

4 Possible Improvements **42**

Bibliography **46**

List of Figures

2.1 Limit order book structure 10
2.2 Mechanism of simple linear regression 11
2.3 Structure of RNN 13
2.4 Inner structure of RNN's hidden state 14
2.5 Overall structure of a RC method with feedback mechanism 15
2.6 Recurrent structure of a RC with feedback mechanism 16
2.7 Geometrical meaning of principal component 22
2.8 The overall structure of a RC method with feedback mechanism and PCA embedding 23

3.1 Process of data matching for single variable time series forecast 27
3.2 Moving block bootstrap 29
3.3 Density of R^2 from Different Models 33
3.4 Step wise R^2 value of OLS & ESN models initialized by standard uniform distribution 38

List of Tables

3.1 Order book and trade book variables at level $i = 0, 1, ..., 9$ 25
3.2 Features and target data at level $i = 0, 1, ..., 9$ 26
3.3 Different size of training and testing dataset 28
3.4 Mean, variance, skewness, and kurtosis of R^2 (in %) of simple linear regression and RC methods trained by different size training dataset 32
3.5 Average, variance, skewness, and kurtosis of R^2(in %) obtained from simple linear regression and RC methods with different embedding methods 32
3.6 VIF of features 35
3.8 Average R^2(in %) of different regression models 36
3.9 Average, variance, skewness, and kurtosis of R^2(in %) obtained from simple linear regression and RC methods with different embedding methods 38
3.10 Average R^2(in %) and percentage of positive R^2 values(in %) obtained from models trained with different dataset. RS stands for randomized Signatre method 40
3.7 Average(in %), variance, skewness, kurtosis, and percentage of positive values(in %) of R^2 obtained from models trained with different dataset. RS stands for randomized Signatre method 41

Introduction

Risen from initial needs of international business, foreign exchange(forex) trade gradually expand its aims from business needs to making financial gains. In exchange market, people use available arbitraging and hedging opportunities to make profits. Nowadays, foreign exchange market or forex market is known as the largest financial market [1] that trades 24 hours in each working day Over-the-Counter (OTC). Based on data collected, in April 2022, the average turnover of forex market is 7.2 trillion USD in daily basis [2]. Unlike other markets, forex market distinguishes itself as one of the most complex financial markets by its high volatility, nonlinearity, and irregularity [3]. Those characteristics also make it a good place where more profitable opportunities exist [4]. Besides, the forex market also benefits from several advantages such as free of insider trading, middlemen, and commissions, as well as having low transaction costs, high liquidity, low margins/high leverage, limited regulation, and online trading opportunities [5]. However, those advantages and characteristics not only bring chances of earning, but also make the best time of trading hard to be captured. In this scenario, being able to predict the future rate of foreign exchange rapidly and precisely can bring huge benefits.

One method to capture profitable opportunities is predicting the market so as to price and trade based on the prediction. Machine learning has long being used in financial market predictions. Specifically, in the past few years, researchers pay more and more attention to high frequency financial market predictions. In aspect of forex market, Zhelev and Avresky [6] use Long-short Term Memory (LSTM) to predict the future forex rate, which provides a basis of implementing deep learning method to forecast forex market. Ahmed & Hassan et al. [4] combined traditional LSTM model with forex loss function which greatly reduces the prediction loss. Abedin, Mohammad Zoynul, et al. [7] add Bagging Ridge (BR) algorithm to LSTM model to predict the exchange rate during COVID-19 pandemic. Galeshchuk, S. & Mukherjee, S. [8] use macroeconomic data and stacked LSTM to predict daily closing rates of dominant currency pairs.

All the previous researches show strong preference on LSTM model and prove its efficiency in forex market forecasting. However, in order to ensure the performance of model, many neural networks, including LSTM, contain numerous neurons. Also, as seen from the previous work, multipl layers of networks may be needed to improve model's performance. Complicated model structures can lead to large amounts of computation thus slow down the speed of training and predicting, which may miss the best time to trade.

To address the potential problem, reservoir computing(RC) algorithms are considered. A reservoir computing algorithm can be seen as a special version of RNN, it fixes the weight matrices between input layers and reservoir as well as the ones between neurons in reservoir. The only matrix needed to be trained is the weight matrix between reservoir and readout layer, which can be obtained via fast training method like linear regression. The fixed weight matrices and simple training model for the weight matrix between reservoir and readout layer greatly lower computational cost and training time of RC method. Moreover, the echo state property guarantees RC methods' ability of processing information. Thus, the RC models are viewed as promising choices of instantaneous prediction of high frequency time series data.

The motivation of this thesis is to test the predictability of RC methods in forex market. In this thesis, we test single and multiple variable RC algorithms and compare them with benchmark linear model. The two RC methods we use are echo state networks(ESNs) and randomized signature.

Results show that for single variable model, RC methods utilizing shrinkage regression and dimensional reduction can still achieve positive average R^2 even if the feature is of low information density. When more informative features are added, all the models achieve better results and all the multiple variable RC methods show significant ability in forex predicting.

This thesis is organized as follows. The first chapter reviews related work. The second chapter introduces methodologies used in experiments. Modelling details and empirical results are shown in the third chapter. Possible improvements are discussed in chapter four and conclusions are given in the last chapter.

Chapter 1

Literature review

Order book data has long being used to forecast future prices of different equities. The work of Cao, C. et al. [9] indicates that around 22% of future returns of stock market can be explained by historical public order book information. Chordia, T. et al. [10] discover that imbalance of order book and liquidity of asset will influence returns of market. The researches provide reliable foundation of utilizing order book data to predict returns of financial market.

Regarding using linear regression to forecast market return, Chantarakasemchit, O. et al. [11] use features processed via simple moving average (SMA) and linear regression to predict EURUSD. Babu, AS. and Reddy, SK. et al. [12] use ARIMA to predct exchange rate of multiple currencies and find that ARIMA outperforms neural network in India market.

Consider recurrent neural network models. The first known learning recurrent neural network(RNN) came to the world in 1972 when Shun-Ichi Amari made the Lenz-Ising recurrent architecture be able to convert its connection weight matrices so as to find the relationship between its input and output [13]. However, to obtain a more reliable RNN, the weights between states should get updated, which is solved by backpropagation methods like backpropagation-through-time (BPTT) [14].

In aspect of applying RNN in forex market, Ni, L. et al. [15] use C-RNN to predict the forex price, which achieves high accuracy. Zeng, K. et al. [16] combine attention mechanism with ARIMA and RNN and gain directional accuracy over 70%. Regrettably, even though RNN has great ability in time series predicting, its short-comes are also obvious. The complicated recurrent structures of RNN make its training difficult and time consuming, which may cause missing of opportunities considering the nature of high frequency market. Thus, a faster yet accurate algorithm is required.

Reservoir computing (RC), a class of methods that share similar ideas with RNN is then considered. RC was established by Jaeger, H. et al. [17] and Mass, W. et al. [18]. Their researches introduce the concept of echo state network from the engineering side and liquid state machines from the aspect of biology respectively, which are viewed as foundations of RC methods. In 2007, Verstraeten, D. et al. [19] suggest the class of methods to be named as reservoir computing and make it an independent researching field.

Nowadays, RC methods are widely used in a variety of industries. In financial field, the applications of RC mainly focus on its predictive power. Specifically, Maciel, L. et al. [20] use echo state network to predict the exchange rate. Lin, X. et al. [21] predict S&P 500 index by ESN. Kim, T. & King BR. [22] implement adaptive decomposition deep ESN model to both stationary and non-stationary time series financial data. These works all demonstrate the superior forecasting capabilities of ESN.

Except for ESN, signature transform and randomized signature transform are also of powerful abilities in time series predicting. Signature transform is a process that transforms a path to its signature [23], which originates from rough path theory [24] [25], while randomized signature can be viewed as a linear projection of signatures of given features [26]. In regards of applications, Compagnoni, E.M. et al. [27] apply randomized signature for function solving and signal generating, results show that randomized signature outperforms ESN, NCDE, LSTM, and NNARX.

Chapter 2

Methodology

In this thesis, we implement a benchmark model(OLS) and several comparative reservoir computing models on order book and trade book data. This chapter presents fundamental definitions of related data and underlying principals of used models.

2.1 Order Book

This section will give some brief introductions to financial terminologies used in later sections.

2.1.1 Ask & Bid Rates

Similar to any other financial markets, in foreign exchange market, buyers, sellers and brokers will offer, ask and match prices for currency pairs. Bid rate is the exchange rate that a buyer can buy a currency pair. While ask rate is the exchange rate that a seller can sell a currency pair. For one specific currency pair, there are several different levels of bid and ask rates at the same time, here we focus on the top 10 levels. The best(respectively, worst) bid rate is labelled as level 0(respectively, 9), indicating the highest(respectively, lowest) prices that buyers offer to buy the currency pair. The best(respectively, worst) ask rate is also labelled as level 0(respectively, 9), which stands for the lowest(respectively, highest) exchange rate that the currency pair can be sold. Normally, at each tick time, a currency pair's ask rates are higher than its bid rates. The difference between the best ask and bid rate is called the bid-ask spread, writes as $spread_i = ask_i - bid_i,\ i = 0, 1, ..., 9$. A graph version of bid and ask prices is shown in Figure 2.1

2.1.2 Middle Rate & Alpha

Since there are bid and ask rates, it is natural to think that there is a 'fair' exchange rate of currency pairs. This 'ground truth' rate is called middle rate, which is computed as the average of the best bid and ask rates, writes as:

$$Middle\ Rate = \frac{Bid\ Rate_0 + Ask\ Rate_0}{2}$$

Simple forward return of middle price is defined as alpha, writes as:

$$Alpha_t = Middle\ Rate_{t+k} - Middle\ Rate_t$$

Where t is the current tick time and $k \in \mathbf{N}^+$ is chosen manually.

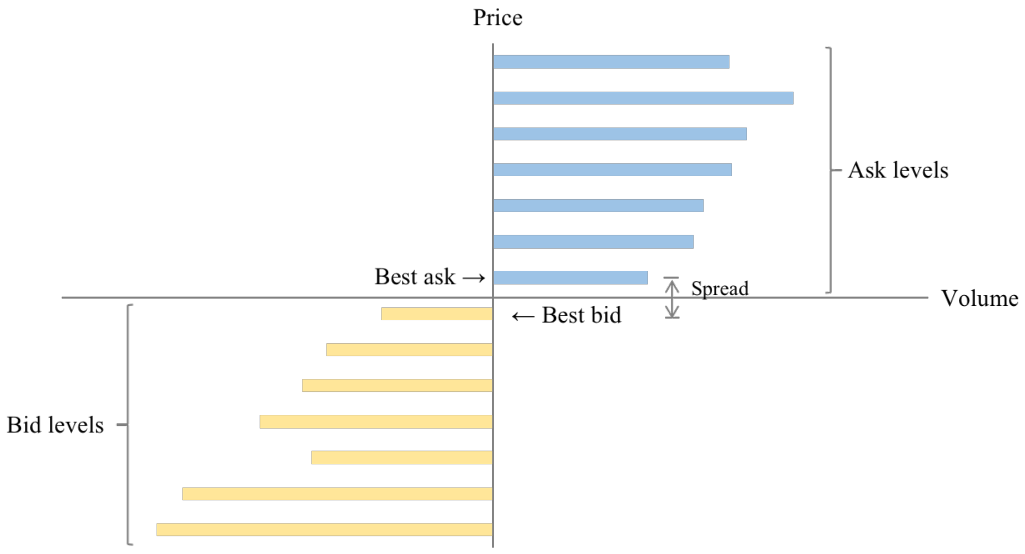


Figure 2.1: Limit order book structure

2.1.3 Ask & Bid Quantities & Counts

In an order book, recorded data are orders that are only offered but not surely executed, order book will show the amount of all the intended bid and ask orders, which refers to quantities of bid and ask offers. Also, order book displays how many clients made orders on the given currency pair, which can be recorded as count of ask or bid orders.

2.2 Simple Linear Regression

Linear regression is one of the most commonly used machine learning algorithms. It raises from an assumption that suggests linear dependent relation exists between dependent variables and independent variables. The regression is then used to capture the relation. In this section, we will introduce the structure and mechanism of linear regression in details.

2.2.1 Model Structure

Assume an input data series $X = (x_1, x_2, ..., x_n)^T$, $x_i \in \mathbf{R}^d$, $i = 1, 2, ..., n$, and an output data series $Y = (y_1, y_2, ..., y_n)^T$, $y_i \in \mathbf{R}$, $i = 1, 2, ..., n$, which are defined as independent variables and dependent variables respectively. To investigate the linear relation between independent and dependent variables, we setup a linear regression model as (2.2.1)

$$Y = XW + \epsilon \tag{2.2.1}$$

Where $W \in \mathbf{R}^d$ is a coefficient matrix(can include constant term) and $\epsilon \in \mathbf{R}^n$ is the error term which is usually a sequence of identical independent distributed standard normal random variables. Also, to make sure there is no overlap information between X and ϵ, the two terms are usually considered as independent, if they are dependent, then the model will be invalid. Under all these assumptions, the goal of regression is to find an optimal $\hat{W}$ that can represent the relations between X and Y as much as possible. There are multiple ways to achieve the goal, here we introduce a basic but commonly used one, ordinary least square regression.

2.2.2 Coefficients Estimation

As mentioned before, the purpose of linear regression is finding the optimal W. A straightforward method to measure the quality of fitting is to examine the distance between the estimated results and ground truth data, show as Figure 2.2.3. The closer the distance, the better the estimations. Thus, the problem of obtaining the optimal coefficients equals to finding a matrix that can lead to the shortest overall distance between ground truth and estimated data. The sum of distances can be expressed as square error, written as (2.2.2):

$$SE = (Y - \hat{Y})^T (Y - \hat{Y}) \tag{2.2.2}$$

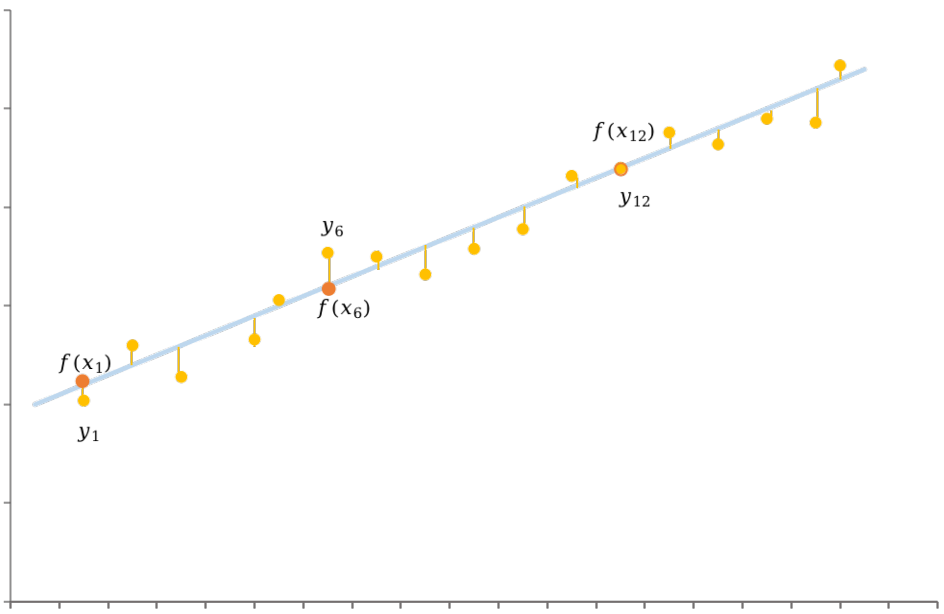


Figure 2.2: Mechanism of simple linear regression

To obtain the optimal $\hat{W} = argmin(SE)$, it is nature to differentiate (2.2.2) on W, which writes (2.2.3)

$$\frac{\partial SE}{\partial W} = 2X^T (Y - X\hat{W}) \tag{2.2.3}$$

Let (2.2.3) equals 0, the optimal $\hat{\beta}$ is then calculated as:

$$\hat{W} = (X^T X)^{-1} X^T Y \tag{2.2.4}$$

Sometimes $X^T X$ might not be invertable, which leads to invalid of (2.2.4). In this situation, adding some penalties will work, the alternative models will be discussed in details in later sections.

2.2.3 Results Prediction

After obtaining optimal coefficients, it is nature to implement trained model for prediction. The way of predicting can be expressed as:

$$\hat{Y}_{test} = X_{test}\hat{W} + \hat{\epsilon} \tag{2.2.5}$$

Where X_{test} is the matrix of testing independent variables and $\hat{Y}_{test}$ is the vector of estimated dependent variables. After predicting, we can use $\hat{Y}_{test}$(if available) to evaluate the performance of model, specific methods will be introduced in Section 2.4.1.

2.2.4 Variance Inflation Factor

When implementing multiple variable simple linear regression model. One vital step is to make sure there is no multicollinearity between features. If multicollinearity exists, the regression will have high training R^2 but no significant coefficients exists.

Here we use variance inflation factor(VIF) to test if multicollinearity exists. First, assume a multiple variable simple linear regression:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_k x_k + \epsilon$$

Where $x_i,\ i = 1, 2, ..., k$ is the i^{th} feature. The aim is to calculate VIF to test the multicollinearity between features. The steps of calculating are as follow:

- Select an $x_i,\ i \leq k$, conduct simple linear regression on it over the rest features:

$$x_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{i-1} x_{i-1} + \beta_{i+1} x_{i+1} + ... + \beta_k x_k + \epsilon$$

- Get R_i^2 of this regression, compute VIF_i via:

$$VIF_i = \frac{1}{1 - R_i^2}$$

Where $R_i^2\ i = 1, 2, ..., k$ is the multi-correlation coefficient between x_i and the other features.

After VIF is calculated, it is time to analysis the results. Two important values of VIF analysis are 1 and 10, if VIF of a feature is close to 1, the feature faces less possibility of having multicollinearity with others. If a feature's VIF is smaller than 10 but larger than 1, the feature is then thought to have weak multicollinearity with the rest features. If VIF of a feature is larger than 10, then it has strong multicollinearity with other features.

2.3 Reservoir Computing

In this section we will show the principals of reservoir computing(RC) methods and some algorithms related to RC.

2.3.1 Recurrent Neural Network

RC method can be viewed as a special form of recurrent neural network(RNN). Thus, here we first introduce the structure of RNN to better explain RC methods.

General Structure

RNN is a neural network with recurrent states. There are multiple forms of it, including using single variable input to obtain multiple variable output(one-to-many), using multiple variable input to obtain single variable output (many-to-one), and using single variable input to get single variable output(one-to-one). Different models have different application scenarios. Here we focus on how an one-to-one model is used for time series prediction.

Assume a sequence of input data $X \in \mathbf{R}^{T\times m}$, where T is the number of time steps, m is the amount of features. The sequence can be expressed as $X = \{x_1, x_2, ..., x_T\}^T,\ x_i \in \mathbf{R}^m,\ i = 1, 2, ..., T$. In an one-to-one model, each time an x_i is inputted, a $\hat{y}_i$ will be returned. A more understandable version of the process is presented as Figure 2.3. Where x_t is the input at time t, $W_{in,t} \in \mathbf{R}^{k\times m}$ is the weight matrix between input layer and hidden layer at time t, $h_t \in \mathbf{R}^k$ is the state calculated at time t, $W_{h,t} \in \mathbf{R}^{k\times k}$ is the weight matrix connect the $t-1^{th}$ hidden state and the t^{th} hidden state, y_t is the output at time t and $W_{out,t} \in \mathbf{R}^{1\times k}$ is the weight matrix between hidden layer and output layer. Figure 2.3 shows that at each time step, the network will pass the activation obtained at current time to the next state recurrently, which explains its name.

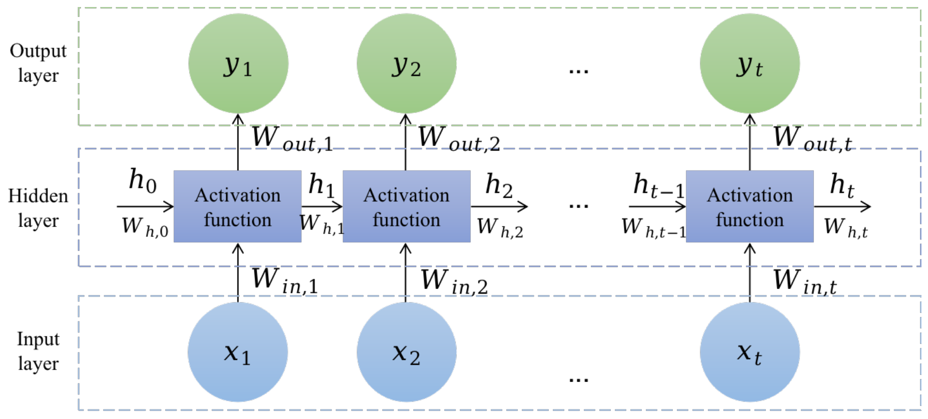


Figure 2.3: Structure of RNN

Inner structure of each hidden state within RNN is shown as 2.4, where g is an activation function, commonly chosen as $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and f is an output activation function, which is normally chosen as a softmax function, defined as $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$. The detailed math expression of each states in RNN can be expressed as follow:

$$\begin{aligned} u_t &:= W_{in,t}x_t + W_{h,t}h_{t-1} + b_h \\ h_t &= g(u_t) \\ v_t &:= W_{out,t}h_t + b_o \\ y_t &= f(v_t) \end{aligned}$$

Where b_o, b_h are error terms randomly generated via standard normal distribution.

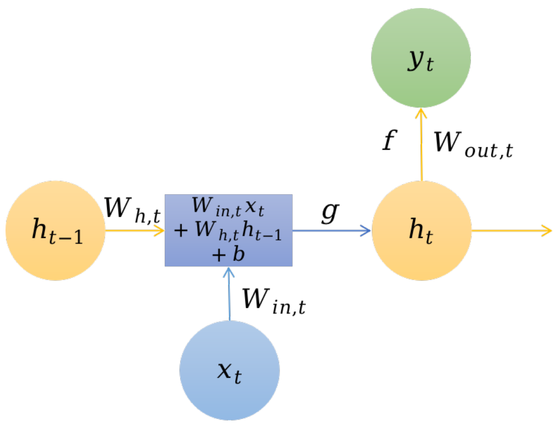


Figure 2.4: Inner structure of RNN's hidden state

Back Propagation

In RNN, the update of weight matrices is based on back propagation method. Here we will explain the process with loss function set as $L = \frac{1}{2}\sum_{t=1}^{T}||y_t - d_t||^2$, where d_t is the ground truth value at time t and y_t is the prediction of target data forecast by RNN. The ultimate goal of training an RNN model is to minimize the loss function, which can be achieved by updating the weight matrices via gradient descent. The process writes as $W_{new} = W - \lambda\frac{\partial L}{\partial W}$ where λ is the learning rate and $\frac{\partial L}{\partial W}$ is the gradient.

In order to calculate the gradient, two error terms are defined: $\delta_t^y(j) = -\frac{\partial L}{\partial v_t(j)}$ and $\delta_t^h(j) = -\frac{\partial L}{\partial u_t(j)}$. Applying chain rule to the error terms, they can then be expanded as:

$$\begin{aligned}
\delta_t^h(j) &= -(\sum_i \frac{\partial L}{\partial y_t(i)}\frac{\partial y_t(i)}{\partial v_t(i)}\frac{\partial v_t(i)}{\partial h_t(j)} + \sum_i \frac{\partial L}{\partial h_{t+1}(i)}\frac{\partial h_{t+1}(i)}{\partial u_{t+1}}\frac{\partial u_{t+1}(i)}{\partial h_t(j)})\frac{h_t(j)}{\partial u_t(j)} \\
&= (\sum_i \delta_t^y(i)W_{out,t}(j,i) + \sum_i \delta_{t+1}^h(i)W_{h,t}(j,i))g^{'}(u_t(j)) \\
\delta_t^y(j) &= -\frac{\partial L}{\partial y_t(j)}\frac{\partial y_t(j)}{\partial v_t(j)} \\
&= (d_t(j) - y_t(j))f^{'}(v_t(j))
\end{aligned}$$

Where $j = 1, 2, ..., m$, $t = 1, 2, ..., t-1$, and $W_{out,t}(j,i)$ represents the weight matrix between j^{th} hidden state and i^{th} output cell. The error terms can be rewrote into vector form as:

$$\delta_t^h = (W_{out}^T\delta_t^y + W_h^T\delta_{t+1}^h)^T \cdot g^{'}(u_t)$$
$$\delta_t^y = (d_t - y_t) \cdot f^{'}(v_t)$$

Here $\cdot$ indicates element wise multiplication. Based on the error terms, updated weight matrices can then be expressed as:

$$W_{out}^{new} = W_{out} + \lambda\sum_{t=1}^{T}\delta_t^y h_t^T$$

$$W_{in}^{new} = W_{in} + \lambda\sum_{t=1}^{T}\delta_t^h x_t^T$$

$$W_h^{new} = W_h + \lambda\sum_{t=1}^{T}\delta_t^h h_{t-1}^T$$

By repeating the update process till loss is within a determined expected range, the training process can be completed.

2.3.2 Echo State Network

Echo state network(ESN) can be viewed as a simplified version of one-to-one RNN, it commonly contains an input layer, a reservoir consisted of some sparsely connected neurons [28], and an output layer. Its whole structure is displayed as Figure 2.5, where the weight matrix between input layer and hidden layer W_{in}, the one between neurons W_{res}, and the matrix directing from output layer to reservoir W_{back} are fixed manually once it is initialized. The only exempt is the output weight matrix W_{out} between reservoir and output layer, which is trained by fitting a linear regression.

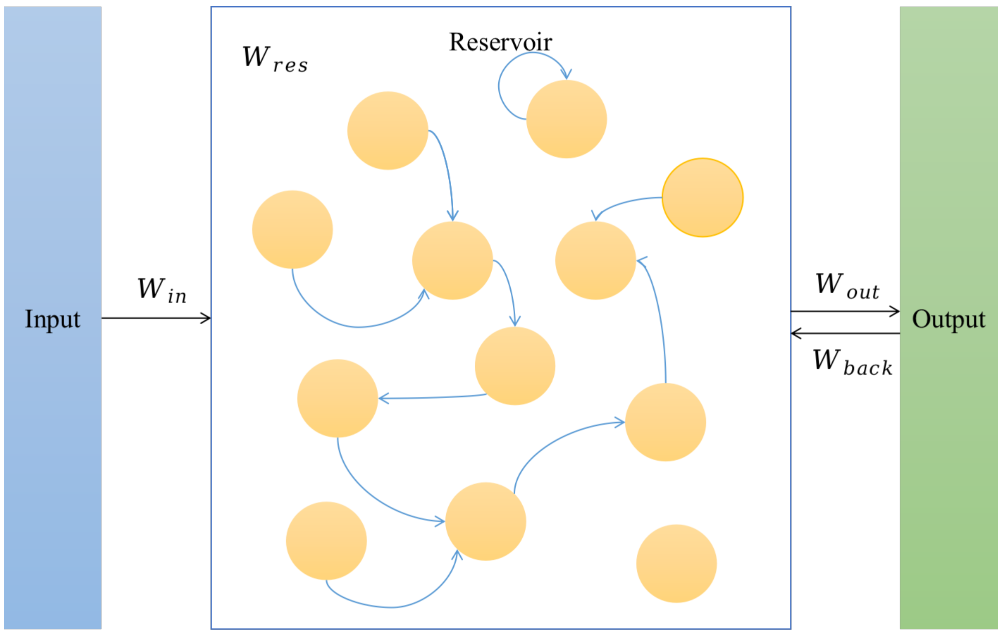


Figure 2.5: Overall structure of a RC method with feedback mechanism

Assume the input of an ESN as $x \in \mathbf{R}^{D\times T}$, where D indicates the dimension of features, T denotes the total time steps of input series, let state matrix be written as $s \in \mathbf{R}^{N\times T}$, N is the number of neurons in the reservoir, and define $y \in \mathbf{R}^{M\times T}$ as an M dimensional output. At each time step, the update function can be defined as (2.3.1)

$$\begin{aligned} s_t &= (1-\alpha)\times f(W_{in}x_t + W_{res}s_{t-1} + W_{back}y_{t-1} + \epsilon) + \alpha \times s_{t-1} \\ y_t &= W_{out}[s_t; x_t] \end{aligned} \tag{2.3.1}$$

Here α is called leaking decay rate, which decides how much information the current state will take over from the previous state, $[s_t; x_t] \in \mathbf{R}^{(N+D)\times 1}$ is a vector concatenated by hidden state and input data at time t, f is an activation function selected manually(commonly chosen as tanh), ϵ_t is an extra error term added to model at time t, and the weight matrices are: $W_{in} \in \mathbf{R}^{N\times D}$, $W_{res} \in \mathbf{R}^{N\times N}$, $W_{out} \in \mathbf{R}^{M\times(N+D)}$, and $W_{back} \in \mathbf{R}^{N\times M}$. At time $t = 1, 2, ..., T$, we can have $x_t \in \mathbf{R}^{D\times 1}$, $s_t \in \mathbf{R}^{N\times 1}$, and $y_t \in \mathbf{R}^{M\times 1}$.

Among all the parameters, only the weight matrix directs from reservoir to output layer requires training. Common training methods include simple linear regression, lasso regression and other fast training regression methods. The detailed algorithms will be discussed in Section 2.3.5. The rest parameters are initialized in specified ways, here are some rules of generating them.

- Elements of W_{in} are commonly generated by a uniform distribution ranging from -a to a, $a \in [0, 1]$
- Spectral radius: $max(\lambda(W))$, the largest eigenvalue of W and adjust parameter Rho: $\rho(W_{res})$ are parameters used to help generate W_{res}. In accordance with echo state property, value of spectral radius should fall within $[0, 1]$ to ensure the robustness of ESN
- To generate W_{res}, the first step is generating a sparse, uniformly distributed random matrix W. The adjust W by alpha and spectral radius to produce W_{res}. The process writes:

$$W_{res} = \rho(W_{res})\frac{W}{max(\lambda(W))}$$

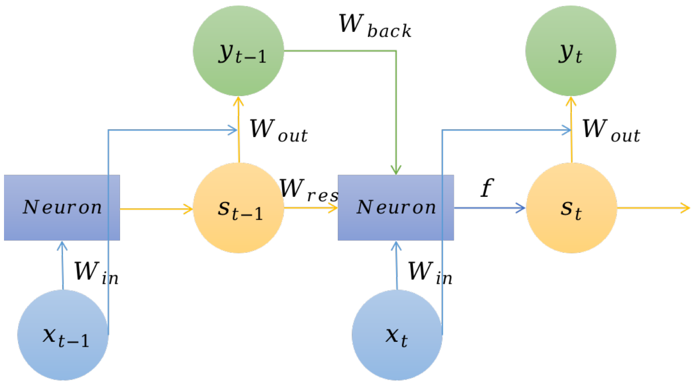


Figure 2.6: Recurrent structure of a RC with feedback mechanism

After initialization, training process starts and W_{out} will be calculated. Testing or forecasting results can then be achieved by fitting testing input dataset to the trained model, the results can be expressed as (2.3.2).

$$\begin{gathered}\hat{s}_t = (1-\alpha) \times f(W_{in}x_t + W_{res}s_{t-1} + W_{back}\hat{y}_{t-1} + \epsilon) + \alpha \times s_{t-1} \\ \hat{y}_t = W_{out}[\hat{s}_t; x_t]\end{gathered} \tag{2.3.2}$$

Where x_t is testing input, $\hat{s}_t$ is estimated state at time t, and $\hat{y}_t$ is the predicted target data at time t.

2.3.3 Echo State Property

To obtain a well-functioned and stable ESN model, applying restrictions on initialization matrices W_{res} and W_{in} is essential. The matrices are obtained via satisfying some restrictions, which can be called as echo state property. The property can be described as: no matter what is the initial state of the reservoir in ESN, the final output of model should always be similar, which can be expressed by mathematics formula writes as:

$$||\hat{F}(s_k, X_n) - \hat{F}(\tilde{s}_k, X_n)|| \to 0 \; when \; k \to \infty$$

Where $\hat{F}$ can be seen as a nonlinear map of input(reservoir in ESN here), $X_n = (x_1, x_2, ..., x_n)$ is the input and s_k, $\widetilde{s}_k$ are different states. The formula indicates that influence of states will fade away in an efficient ESN model.

To ensure that the reservoir of an integrator leaky ESN satisfies ESP, the sufficient condition of writes as: $\rho(W) < 1$ algebraically, here ρ is the spectral radius of W, W is constructed as: $W = (1-\alpha) \times W_{res} + \alpha \mathbf{I}$, $\mathbf{I}$ is an identity matrix [17]. The proof of above conditions is analogous to proofing $\rho(W_{res}) < 1$ is the sufficient condition of a standard ESN. Thus, proof of basic ESN's sufficient condition will be presented and the proof of integrator leaky ESN can be obtained similarly.

$$\begin{aligned}
||\hat{F}(s_t, X_n) - \hat{F}(\widetilde{s}_t, X_n)|| &= d(s_t, \widetilde{s}_t) \\
&= d(f(W_{in}x_t + W_{res}s_{t-1}), f(W_{in}x_t + W_{res}\widetilde{s}_{t-1})) \\
&\leq d(W_{in}x_t + W_{res}s_{t-1}, W_{in}x_t + W_{res}\widetilde{s}_{t-1}) \\
&= d(W_{res}s_{t-1}, W_{res}\widetilde{s}_{t-1}) \\
&= ||W_{res}(s_{t-1} - \widetilde{s}_{t-1})|| \\
&= ||W_{res}||d(s_{t-1}, \widetilde{s}_{t-1}) \\
&\leq \rho(W_{res})d(s_{t-1}, \widetilde{s}_{t-1}) \\
&= \rho(W_{res})||\hat{F}(s_{t-1}, X_n) - \hat{F}(\widetilde{s}_{t-1}, X_n)||
\end{aligned}$$

Where $d(\cdot, \cdot)$ means the distance between two vectors. Thus, when $\rho(W_{res}) < 1$ is satisfied, convergence of different states is obviously achieved.

2.3.4 Randomized Signature in RC

Predicting high frequency time series data can be analogized as solving a stochastic differential equation that maps an known input series called control to an known output value called solution trajectory by a unknown smooth function. The equation can be solved(the smooth function can be found) using RC methods, one applicable RC method is signature transform, also called signature. In this section, we will showcase the principals of signature and introduce a modified version of it: randomized signature.

Path

To understand signature, comprehending the definition of path is a must. Here we discuss paths in Educlidean space.

Assume a path X as a continuous mapping from an interval $[a,b]$ to $\mathbf{R}^d$ [29], denotes as $X : [a,b] \mapsto \mathbf{R}^d,\ X_t = \{X_t^1, X_t^2, ..., X_t^d\}$, where $X^i,\ i = 1, 2, ..., d$ is a real-valued path [30].

Signature

Over a time period $[a,b]$, signature of X is defined as:

$$S_{a,b}(X) = (1, S(X)^1_{a,b}, S(X)^2_{a,b}, ..., S(X)^d_{a,b}, S(X)^{1,1}_{a,b}, S(X)^{1,2}_{a,b}, ...)$$

The signature is an infinite sequence with each element calculated by path integral:

$$S^{\mathbf{I}}_{a,b}(X) = \int_{a<t_1<t_2<...<t_k<b} dX^{i_1}_{t_1} dX^{i_2}_{t_2} ... dX^{i_k}_{t_k}$$

Where $\mathbf{I} = (i_1, i_2, ..., i_k)$, with $i_p \in \{1, 2, ..., d\},\ p = 1, 2, ..., k$.

Truncated Signature

Truncated signature is without doubt, a truncated version of signature, making it a finite sequence. A p order truncated signature of X writes as (2.3.3):

$$S^{p}_{a,b}(X) = (1, S(X)^1_{a,b}, S(X)^2_{a,b}, ..., S(X)^p_{a,b}) \tag{2.3.3}$$

However, (2.3.3) shows that the cost of computing truncated signature as a reservoir might be very expensive due to high complexity. One possible solution to the problem is randomized signature, which will be introduced in the next section.

Randomized Signature

Randomized signature can be viewed as a random projection of truncated signature mapped by a linear operator [26]. A mathematical definition of randomized signature used as reservoir can be written as (2.3.4) [27]:

$$\begin{cases} s_0 = 0 \\ s_t = \alpha \times s_{t-1} + f(A_1 s_{t-1} + \epsilon_1) + \sum_{i=1}^{d} f(A^i_2 s_{t-1} + \epsilon^i_2) x^i_t \end{cases} \tag{2.3.4}$$

In (2.3.4), $t = 1, 2, ..., T$. $A_1 \in \mathbf{R}^{k\times k}$ and $A^i_2 \in \mathbf{R}^{k\times k},\ i = 1, 2, ..., d$ are independent sparse matrices generated from standard normal distribution. $\epsilon_1 \in \mathbf{R}^k$ and $\epsilon^i_2 \in \mathbf{R}^k,\ i = 1, 2, ..., d$, where d is dimension of features, are independent random normal distributed error terms. x^i_t is the i^{th} element of x_t, s_t is the randomized signature of x_t, and f is an activation function set as tanh. Randomized signature s_t is viewed as the state at time t while implementing this recursion as a reservoir.

2.3.5 Regression Methods

In this section, we will showcase some regression methods that can be applied to the training process of RC. Here X is defined as $(x_1, x_2, ..., x_n)^T,\ x_i \in \mathbf{R}^d,\ i = 1, 2, ..., n$, and $Y = (y_1, y_2, ..., y_n)^T,\ y_i \in \mathbf{R},\ i = 1, 2, ..., n$.

Ridge Regression

The most commonly used regression method in ESN is simple linear regression. However, as shown in (2.2.4), the optimal $\hat{W}$ of (2.2.1) only holds when X^TX is a full trace matrix. Unfortunately, the presumption is not always satisfied since sometimes independent variables are highly correlated, which can invalid the simple linear regression. In order to ensure the efficiency of regression, some restrictions should be applied to simple linear regression to transfer X^TX to $X^TX + \lambda\mathbf{I}$. One algorithm developed from the idea is ridge regression. Ridge regression works by adding an

$l2-norm$ term to a simple linear regression, which writes as:

$$Y = W^T X + \epsilon$$
$$W^T W \leq r$$

Where r is a threshold of the coefficients. The corresponding loss function of ridge regression is:

$$J(W) = \frac{1}{2}(Y - XW)^T(Y - XW) + \lambda W^T W$$

Here $\lambda \in \mathbf{R}^+$ is the penalty coefficient. By differentiating the loss function, the optimal coefficient matrix $\hat{W} = (X^T X + \lambda \mathbf{I})^{-1} X^T Y$ can be obtained. $\hat{W}$ shows that the larger the λ, the more severer the penalty, the more compressed the estimated coefficients. When $\lambda = 0$, the regression is equal to OLS regression.

Lasso Regression

Another method used to handle multicollinearity is lasso regression. Similar to ridge regression, lasso regression applies an $l1-norm$ regularization to OLS regression. The definition and loss function of lasso regression write as:

$$\begin{cases} Y = W^T X + \epsilon \\ ||W||_1 \leq r \end{cases}$$

$$J(W) = \frac{1}{2}(Y - XW)^T(Y - XW) + \lambda ||W||_1$$

Where $||W||_1 = \sum_i w_i$, represents the sum of values calculated by row, λ is the penalty coefficient and r is the threshold of restriction. Optimal $\hat{W}$ is calculated via coordinate descent. Empirical results show that when the penalization is harsh enough, lasso regression can discard(compress the coefficients of corresponding features to 0) some features to eliminate the influence of multicollinearity.

However, in some scenarios, lasso regression has several disadvantages:

- When $n \approx d$, lasso regression can not choose features effectively
- When input features have grouping effect, lasso regression can only pick one feature among other similar ones, causing the waste of information [31]
- If $n > d$ and multicollinearity exists, lasso regression might be dominated by ridge regression [32]

To address these problems, another sparse regression method is considered.

Elastic Net Regression

Aiming at overcoming the weaknesses raised from lasso regression, a regression called elastic net regression which combines the sparsity of lasso regression and the coefficient compress ability of ridge regression is defined. Elastic net comes form naive elastic net, whose loss function writes as:

$$J(W) = \frac{1}{2}(Y - XW)^T(Y - XW) + \lambda_1 ||W||_1 + \lambda_2 ||W||_2^2$$

Here λ_1, λ_2 stand for the penalty factors of $l1-norm$ and $l2-norm$ respectively, $||W||_1$ is the $l1-norm$ defined before, while $||W||_2^2 = W^TW$ is an $l2-norm$ term.

Far from expectation, empirical experiments show that naive elastic net does not outperform ridge regression nor lasso regression [31]. Thus, elastic net regression, a calibrated shrinkage regression method is developed. The loss function of elastic net regression writes as:

$$J(W) = W^T \frac{X^TX + \lambda_2 \mathbf{I}}{1+\lambda_2} W - 2Y^TXW + \lambda_1 ||W||_1$$

The loss function can be viewed as a more robust form of lasso regression's loss function, thus leading to more robust results. In experiments, the values of λ_1 and λ_2 are usually chosen via cross validation.

2.3.6 Principal Component Analysis

In this section, we will look into a dimensional reduction method, PCA, and some linear algebra knowledge related to it.

Singular Value Decomposition

To better understand PCA, we first introduce singular value decomposition(SVD). In linear algebra, a normal matrix can be decomposed in several methods. However, the matrix we meet is not always formal. Thus, a more universally applicable matrix decomposition method is needed. Assume a matrix $A \in \mathbf{R}^{m\times n}$, there always exists two orthogonal matrices $U \in \mathbf{R}^{m\times m}$, $V \in \mathbf{R}^{n\times n}$ and a diagonal matrix $\Lambda \in \mathbf{R}^{m\times n}$ that satisfy:

$$A = U^T \Lambda V$$

$$\Lambda = \begin{pmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}$$

Where U is a matrix consisted of standard orthogonal eigenvectors of AA^T, V is a matrix made of standard orthogonal eigenvectors of A^TA, and $\sigma_i = \sqrt{\lambda_i},\ i = 1, 2, ..., r,\ r \leq m$ is the square root of corresponding eigenvalue of A, defined as singular value. The singular values $\sigma_1, \sigma_2, ..., \sigma_r$ are ranked in an decreasing order.

PCA

Principal component analysis (PCA) is a technique commonly implemented to reduce the dimension of features. Given $X = (x_1, x_2, ..., x_m)^T,\ X \in \mathbf{R}^{m\times n}$, where m is the dimension of data and n is the number of data, the covariance matrix of X is defined as Σ. Assume that we want to reduce the dimension of X to k ($k \leq m$) but at the same time retain as much information from X as possible. It is natural to consider its linear transform written as (2.3.5), then obtain k elements from the transform results.

$$\begin{cases} Z_1 = \alpha_1^T X = \alpha_{11}x_1 + \alpha_{21}x_2 + ... + \alpha_{m1}x_m \\ Z_2 = \alpha_2^T X = \alpha_{12}x_1 + \alpha_{22}x_2 + ... + \alpha_{m2}x_m \\ ... \\ Z_n = \alpha_n^T X = \alpha_{1n}x_1 + \alpha_{2n}x_2 + ... + \alpha_{mn}x_m \end{cases} \tag{2.3.5}$$

Where $\alpha_i \in \mathbf{R}^m,\ i = 1, 2, ..., n$ are the coefficients of linear transformation. $Z_i,\ i = 1, 2, ..., n$ is called the i^{th} principal component if :

- $\alpha_i^T \alpha_i = 1,\ i = 1, 2, ..., n$
- When $i > 1$, $\alpha_i^T \Sigma \alpha_j = 0,\ j = 1, 2, ..., i-1$
- $Var(Z_i) = \underset{\alpha^T\alpha=1, \alpha_i^T\Sigma\alpha_j=0, j=1,2,...,i-1}{max} Var(\alpha^T X)$

These restrictions can be explained by maximize variance theory. The theory indicates that data with larger variance tends to contain more useful information while data with smaller variance has a higher chance to be noisy. Thus, a larger variance is preferred. The first condition $\alpha_i^T \alpha_i = 1$ is a restriction when $Var(Z_i) \to \infty$. Also, to make principal components as efficient as possible, different principal components should has no overlap information. The idea can be expressed as: $Cov(Z_i, Z_j) = \alpha_i^T \Sigma \alpha_j = 0$, which is equal to the second condition.

Geometrical illustration of principal component is that the process of linear transform and restrictions are equivalent to rotating the coordinate system where the datapoints are located to find a new coordinate system that can maximize the variance of data. For example, as shown in Figure 2.7. In a two dimensional space, sample points are scattered in an oval. Its projection on the original coordinate system(black one) indicates a smaller variance(less informative) comparing with the one in the new coordinate system(blue one). Thus, rotating the axis is essential for clustered data.

After interpreting the underlying principal of PCA, we start to discuss how to implement it. The equations (2.3.5) with restrictions can be solved via Lagrange multiplier method. Let $\phi(\alpha_i) = Var(\alpha_i^T X) - \lambda(\alpha_i^T \alpha_i - 1)$, consider:

$$\begin{cases} \frac{\partial \phi}{\partial \alpha_i} = 2(\Sigma - \lambda \mathbf{I})\alpha_i = 0 \\ \frac{\partial \phi}{\partial \lambda} = \alpha_i^T \alpha_i - 1 = 0 \end{cases} \tag{2.3.6}$$

Since $\alpha_i \neq 0$, solving (2.3.6) equals computing the eigenvalues and eigenvectors of Σ, writes as $|\Sigma - \lambda \mathbf{I}| = 0$. Thus, the process of PCA can be concluded as below:

- Centralize X by row
- Calculate the covariance matrix of centralized X
- Implement SVD method to compute the eigenvalues and eigenvectors of the covariance matrix
- Rank the eigenvalues in a decreasing order, rearrange the corresponding top k eigenvectors as a new characteristic matrix $P \in \mathbf{R}^{k \times m}$
- Obtain PX as the new dataset

In experiments, when implementing PCA, k is not set as one pleases. The strategy of choosing k is by computing the contribution rate of components Z_i, which is defined as $\frac{\lambda_i}{\sum_{j=1}^{n}\lambda_j}$ (λ_i is the i_{th} eigenvalue), then choose k that makes the cumulative contribution rate $\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{j=1}^{n}\lambda_j}$ reaches a reasonable threshold.

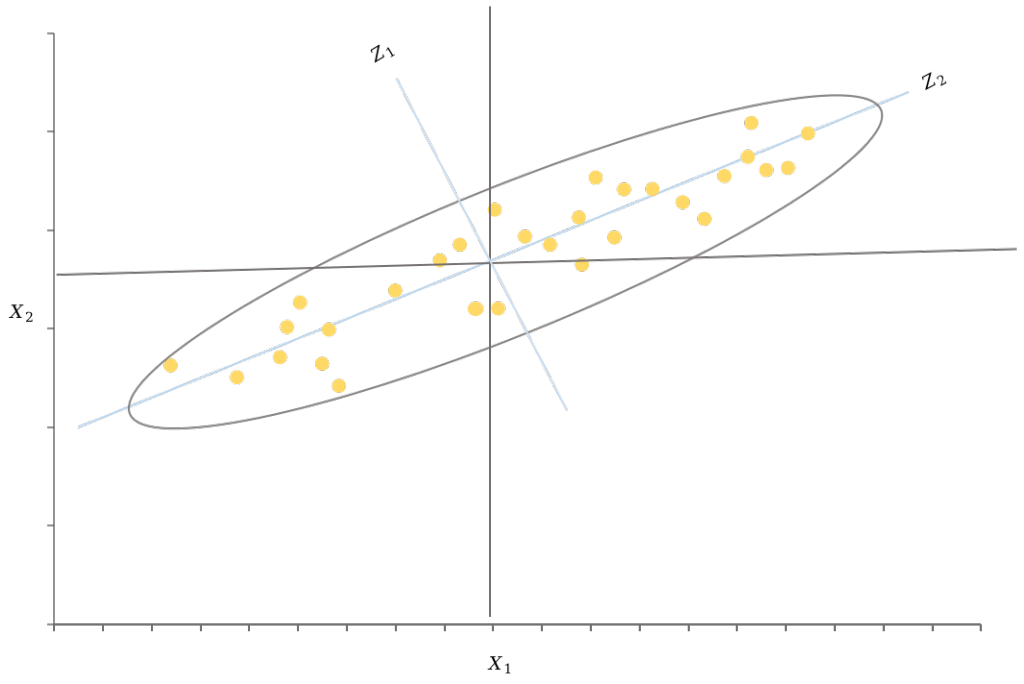


Figure 2.7: Geometrical meaning of principal component

2.3.7 RC Method with PCA Embedding

In the thesis, we combine PCA with RC methods to improve models' performances. PCA is implemented after all the states are updated, or after all the steps in the reservoir are finished, the process is shown as Figure 2.8. Target dimension of PCA is set manually.

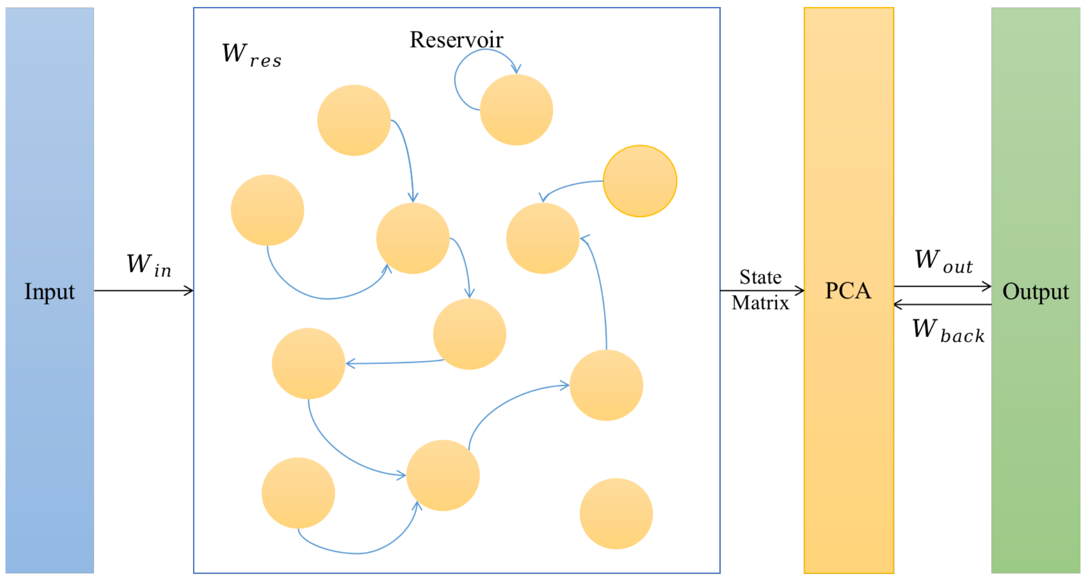


Figure 2.8: The overall structure of a RC method with feedback mechanism and PCA embedding

2.4 Evaluation Metric

After results are obtained, it is time to verify how good the model works based on predicted results. One classic metric to evaluate the quality of model is R^2. Similarly, in this thesis, we use an modified R^2 as the evaluation metric. The reason for using the modified model is that, unlike other prediction tasks, while forecasting forex rate, it is unrealistic to compare the predicted results with the real market data since the real market is almost unpredictable. Thus, we assume that the average alpha of a benchmark market is 0, indicating that the market is only influenced by supply and demand of currency pairs. Under this prerequisite, modified R square can be written as (2.4.1)

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - 0)^2} \tag{2.4.1}$$

Where y_i is real alpha and $\hat{y}_i$ is predicted alpha.

Instead of running the model for only one attempt. We want to make sure that it is universally robust. Thus, the algorithm is implemented over different training and testing dataset to obtain a series of R^2, some statistics of the R^2 values are then being checked.

First statistics is the mean of R^2, which is the main statistics that we will focus on. It is defined as:

$$\bar{R}^2 = \frac{1}{n} \sum_{i=1}^{n} R_i^2$$

Where R_i^2 is the i^{th} out-of-sample R^2, n is the time that the model is implemented.

Second statistics is the variance of R^2 values, which can showcase the stability of model. The formula writes as:

$$var(R^2) = \frac{\sum_{i=1}^{n} (R_i^2 - \bar{R}^2)}{n}$$

The third statistics is the skewness of R^2 sequence. Skewness used in this thesis is Fisher-Pearson coefficient of skewness, defined as:

$$skew(R^2) = \frac{E[(R^2 - \bar{R}^2)^3]}{(E[(R^2 - \bar{R}^2)^2])^{\frac{3}{2}}}$$

There are overall three conditions regarding a distribution's skewness. One is when skewness equals zero, the distribution is thought to be symmetric. While the skewness is smaller than 0, the distribution has a negative skew, vice versa. A distribution with positive skew has a longer tail on the right side and has its mode smaller than its median and its median is smaller than its mean. While a distribution with negative skew has a longer left tail and has an opposite relation among mode, median, and mean compared with distribution with positive skew.

The last statistics is kurtosis, it reflects the flatness of a given distribution. The mathematical definition of it writes as:

$$kurtosis(R^2) = \frac{E[(R^2 - \bar{R}^2)^4]}{(E[(R^2 - \bar{R}^2)^2])^2}$$

The benchmark of kurtosis is the kurtosis of standard normal distribution, which equals 3. If a kurtosis of a distribution is larger than 3, then the distribution has a thick tail. If its kurtosis is smaller than 3, then the distribution has a thin tail.

Chapter 3

Experiment

This chapter contains an introduction to data used in experiments, results from selected models and intuitive explanations of the results.

3.1 Data

In this section, we will briefly review the structure of dataset we use and discuss some features constructed from the raw data.

3.1.1 Raw data

Our raw data is consisted of order book data and trade book data of USDMXN dating from 2022 January to 2022 December, recorded in unevenly divided time grid(tick). Order book data includes price, quantity, and count of bid and ask orders at different levels. Trade book data includes exchange rate of dealt trades and the time that a trade is give or paid. Size of limit order book data can reach around 500,000 to 700,000 in one trading day. Trade book data has a smaller size, but will later be merged with order book data. The overall size of raw data after merging is 22,829,398.

	Variable	Explanation
Order book	askRate_i	Exchange rate of ask order at level i
	askQty_i	Quantity of ask book at level i
	askCount_i	Amount of ask orders at level i
	bidRate_i	Exchange rate of bid order at level i
	bidQty_i	Quantity of bid book at level i
	bidCount_i	Amount of bid orders at level i
Trade book	price	Price of traded order
	side	State of dealt order (given or paid)
	time	Time of trades

Table 3.1: Order book and trade book variables at level $i = 0, 1, ..., 9$

3.1.2 Data Preprocessing

Data processing is the foundation of a robust model. In this thesis we implement single variable and multiple variable models based on time series data. Here, we will first showcase how the features of both models are constructed and then discuss their preprocessing methods.

Features & Target Data Construction

For all the models, alpha over a future time period is set as the target and alpha computed over a past time period is used as an input variable. For multiple variable models, some other features are constructed based on limit order book data and trade book data. The definitions of features and target variable are shown in Table 3.2

	Feature	Definition
Target	$\text{Return}_{future,t}$	$(\text{askRate}_{t+k} + \text{bidRate}_{t+k}) \times 0.5 - (\text{askRate}_t + \text{bidRate}_t) \times 0.5$
Features	$\text{Return}_{past,t}$	$(\text{askRate}_t + \text{bidRate}_t) \times 0.5 - (\text{askRate}_{t-\Delta t} + \text{bidRate}_{t-\Delta t}) \times 0.5$
	$\text{OFI}_{i,t}$	Order flow imbalance of order book data at level i, time t
	$\text{QI}_{i,t}$	Queue imbalance of order book data at level i, time t
	traded - mid	$\text{Traded price}_t - \text{Middle price}_t$

Table 3.2: Features and target data at level $i = 0, 1, ..., 9$

The underlying ideas of feature constructions are that:

- $\text{OFI}_{i,t}$: Order flow imbalance is constructed as [33]:

$$bOF_{i,t} = \begin{cases} v_t^{i,b}, & b_t^i > b_{t-1}^i \\ v_t^{i,b} - v_{t-1}^{i,b}, & b_t^i = b_{t-1}^i \\ -v_t^{i,b}, & b_t^i < b_{t-1}^i \end{cases}$$

$$aOF_{i,t} = \begin{cases} -v_t^{i,a}, & a_t^i > a_{t-1}^i \\ v_t^{i,a} - v_{t-1}^{i,a}, & a_t^i = a_{t-1}^i \\ v_t^{i,a}, & a_t^i < a_{t-1}^i \end{cases}$$

$$OFI_{i,t} = bOF_{i,t} - aOF_{i,t}$$

  Where $v_t^{i,b}$(respectively, $v_t^{i,a}$) is the volume of bid(respectively, ask) orders at time t, level i and b_t^i(respectively, a_t^i) is the bid(respectively, ask) price of orders at time t, level i. This feature is a stationary time series made out of a nonstationary time series [34]. It measures the imbalance of quantity between bid and ask orders, which can indicate the dynamics of given currency pair, thus having some predictability

- $\text{QI}_{i,t}$: Queue imbalance is defined as [35]:

$$QI_{i,t} = \frac{v_t^{i,b} - v_t^{i,a}}{v_t^{i,b} + v_t^{i,a}}$$

  Where $v_t^{i,b}$(respectively, $v_t^{i,a}$) is still the volume of bid(respectively, ask) orders. This feature describes the pressure of buying or selling a currency pair, which can also showcase the popularity of given currency pair

- traded - mid: Trade book and order book data are merged by time. If no exact matching time point exists, trade book data will be matched with order book data at a future time point with some tolerance. Blank spaces after merging are filled by zero. After merging, the feature can be constructed via simple substation. This feature indicates the difference between dealt price and market middle price. It illustrates the gap between the expectation of market and executed deal, which can be a potential indicator of the future path of the market. The underlying reason for its predictability is that executed deals can move the market thus changing the middle price

All the features are constructed without being carefully tested, some information they contain surely has some overlap. Thus, in experiments, multicollinearity among features will be examined to find the optimal combination of independent variables.

Missing Data

Missing values might exists due to versatile reasons. In order to avoid error caused by the null values while fitting models, historical data are used to fill in the blanks. If after filling there are still empty values, those blanks are then replaced with zero.

Data Matching for Simple Variable Model

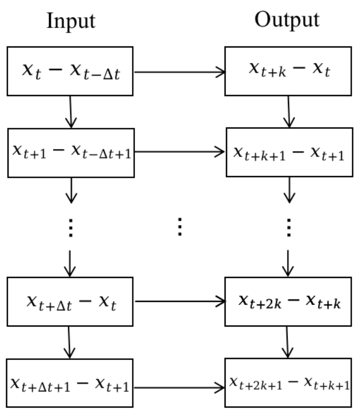


Figure 3.1: Process of data matching for single variable time series forecast

Matching of data is vital to time series modelling. Consider single variable models we implement, the corresponding data matching method is shown as Figure 3.1, where x_t stands for the middle rate of given currency pair at time t. Assume the current time point as t, the input is alpha over a past time horizon with length Δt and the output is alpha over a future time horizon with length k. The matching indicates that at time t, the model is designed to predict the alpha over the future k time steps based on current information and message from Δt tick time ago.

Data Preprocessing for Multiple Variables Model

For multiple variable models, data matching method is similar to the one used for single variable models. The only difference is that more features are included. After matching, data is standardized via MinMax standardization method, writes as (3.1.1)

$$X_{scaled} = \frac{X - min(X)}{max(X) - min(X)} \tag{3.1.1}$$

3.2 Modelling Details

3.2.1 Model Setup

Training & Testing Dataset

To find the optimal training and testing size, experiments are carried out on three different size of training datasets, whose sizes are shown in Table 3.3. The unit of size is one trading day, which is to make sure the model can obtain adequate intraday and overnight information.

	Small Size	Medium Size	Large Size
Training set	5	10	15
Testing set	1	1	1

Table 3.3: Different size of training and testing dataset

Forecasting Period Setup

For all the models, we set Δt and k in Figure 3.1 as (50, 50), (100, 100), and (500, 500), which approximates one second, one minute, eight minutes respectively.

Reservoir Computing Model Setup

RC methods' performance varies with the choice of its parameters, we want to investigate how different hyperparameters influence them. Thus, the parameters are set as follow:

- Rho(for ESN only): 0.95, 0.5, 0.2
- Number of neurons in the reservoir: 50, 100, 150, 200
- Number of runs for initialization: 100, 150, 200, 250
- Leaking decay rate: 0, 0.2, 0.5, 0.8, 1
- Regression method: 'linear', 'lasso', 'elastic'

Here Rho is a parameter used only in ESN to help scale the state matrix, number of runs for initialization is the number of runs that are implemented before the real training process starts, leaking decay rate controls how much past information is passed to the current state, and regression method is the regression algorithm used to train W_{out}.

In single(respectively, multiple) variable ESN, the input matrix is a sparse matrix generated via standard uniform random generator whose degree of sparsity equals 0.7(respectively, 0.5). Feedback matrix is initialized by standard random uniform generator, but scaled to $[-0.5, 0.5]$. Weight matrix between neurons is also a sparse matrix with 0.7(respectively, 0.5) degree of sparsity whose nonzero elements are obtained from a uniform generator ranging between -1 and 1, the matrix is then scaled by spectral radius of the uniform matrix and Rho. The initial state matrix is defined as a zero matrix.

For single variable(respectively, multiple) randomized signatures methods, A_1, $A_2^i,\ i = 1, 2, ..., d$ in (2.3.4) are set as 70%(respectively, 50%) sparse matrices with other nonzero elements randomly generated from standard normal distribution independently. The error terms ϵ_1 and $\epsilon_2^i,\ i = 1, 2, ..., d$

are generated from identical standard normal distribution independently.

Parameters for sparse regressions are provided in an array and the optimal one is found via cross validation, the corresponding Python functions are:

```
sklearn.linear_model.LassoCV
sklearn.linear_model.RidgeCV
sklearn.linear_model.ElasticNetCV
```

Within each RC method, the activation function is tanh, writes as :

$$tanh(x) = \frac{\mathbf{e}^x - \mathbf{e}^{-x}}{\mathbf{e}^x + \mathbf{e}^{-x}}$$

3.2.2 Implementation Method

Moving Block Bootstrap

To ensure the universal robustness of models, we use rolling block bootstrap to implement different models over the full dataset. The specific process can be expressed as Figure 3.2. For each step, training and testing R^2 are saved for results analysis.

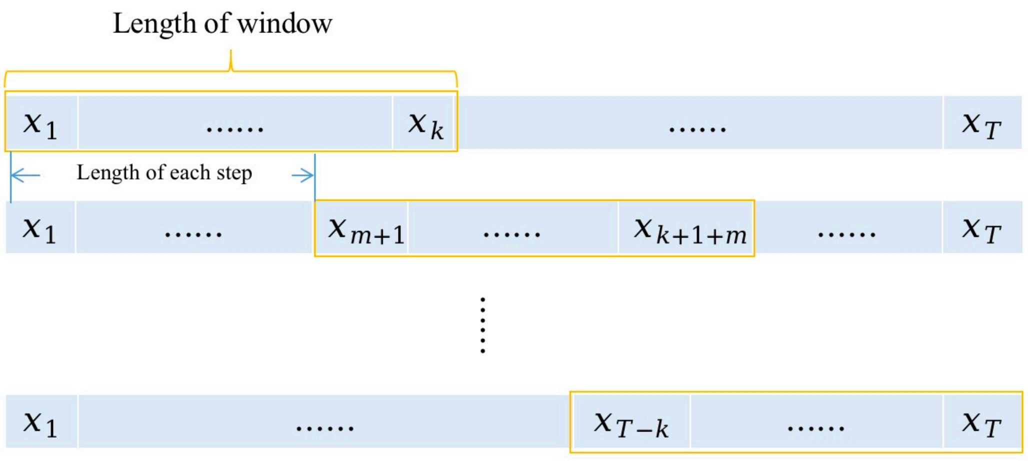

Length of window
x_1 …… x_k …… x_T
Length of each step
x_1 …… x_{m+1} …… x_{k+1+m} …… x_T
x_1 …… x_{T-k} …… x_T

Figure 3.2: Moving block bootstrap

In this graph, m is the length of step for each move, k is the length of training and testing dataset for each run. In our experiments, m is set as 1 trading day and k is set as $5 + 1$, $10 + 1$, $15 + 1$ trading days.[1]

3.3 Empirical Results

This section includes results obtained from different models in different scenarios respectively. Possible explanations are given towards the results.

3.3.1 Numerical Results of Single Variable Models

Literature works show that RC methods perform quite well when predicting time series data even the only feature is the target data itself. To investigate if this conclusion holds with data from

[1]5, 10, 15 trading days are set as the length of training data while 1 trading day is the length of testing data.

forex market. We implement several single variable RC methods on alpha data series. The results will be showcased and interpreted and an optimal model will be given at the end of section.

Shorter Horizon Helps Outperform

First, to make the testing process more efficient, a reasonable $(\Delta t, k)$ pair is needed. Results obtained from different models universally align with common sense. They all suggest that the shorter the length of the lag horizon(Δt and k in Figure 3.2), the better the model. This outcome can be intuitively explained as forecasting a further future is more difficult since the elements that might influence a further future can be more versatile compared with the ones that affect a nearer future. If explain the fact from aspect of modelling, the reason can be that the longer the prediction horizon, the more unrelated information the model will receive and the higher the level of noise, which leads to worse performances. Thus, in the following experiments, we set $(\Delta t, k)$ as (50, 50).

Choices of Hyperparameters

After selecting an optimal predicting horizon, we start to investigate the hyperparameters of ESN. First parameter to be set is Rho, which relates to matrix initialization. Based on empirical results, Rho is set as 0.5, which aligns with the rule that initialization parameter that adjust sate matrix should be smaller than one to maintain a stable model. In other words, a middle level forgetting speed of past information is preferred, and the echo ability of ESN might get harmed if Rho is too low, due to heavily faded memory of reservoir.

Next, let's take a careful look at the reservoir. While calculating hidden states, the amount of neurons in reservoir is vital to models' performance. Here, the number of neurons is chosen from 50, 100, 150, 200, and 250. Contrary to common sense, the final choice is 50, which suggests a small reservoir is preferred. The reason might be that to satisfy echo state property, the initialization matrix cannot be too large, if not, the maximum absolute value of eigenvalues will exceed 1. In addition, a model with smaller reservoir is less computationally expensive. Therefore, 50 neurons will be used in all the models.

Except for the overall structure, parameter between each state is also important. Considering ESN has a fading memory, its leaking decay rate should be carefully chosen. Commonly speaking, the larger the leaking decay rate, the more information the previous state will pass to the next one. For example, if leaking decay rate equals 1, then the t^{th} state will wholly incorporate the state generated from the $t-1^{th}$ state as its state. Oppositely, if it equals 0, the information in the n^{th} state will totally come from the updating process rather than simply intaking the previous state's information. The rate is chosen from 0, 0.2, 0.5, 0.8, 1. Our empirical experiments suggest that 0 is the optimal choice since variation of leaking decay rate does not influence the performance of models significantly. The malfunction of leaking decay rate indicates that states calculated by each neuron might have a large chance to be highly homogeneous, which aligns with the noisy property of given data, leading to the invalid of leaking decay rate.

The last parameter helps decide how many runs to take before collecting states produced by neurons. The purposes of running without recording results are to ensure the reservoir is well initialized and the model is able to reflect the properties of target data [36]. Here, to investigate how it works, we test the models on 100, 150, 200 and 250 runs. The results show that increment of initialization runs does not essentially improve the performance of models significantly but slows down the speed of computing. Thus, 100 runs is chosen consider the efficiency of models.

For randomized signature models, the hyperparameters mentioned above have almost the same effect on models' performances. Thus, for the convenience of comparison, same parameters are used.

Training Size Matters

After basic hyperparameters are set. We start to test models over different dataset. First, dataset with different training size are considered, all the models are trained over data over the past 5/10/15 trading days. Experiments are carried out using simple linear regression, RC methods utilizing simple linear regression, RC methods with lasso regression, and RC methods with elastic net regression. Results of RC methods using linear regression and lasso regression will not be displayed due to lack of significance. Linear regression's results will be presented regardless of its insignificance to provide a benchmark. Selected results are shown in Table 3.4.

The outcomes show that for simple linear regression, the larger the training dataset, the better the results. However, for both ESN and randomized signature method utilizing elastic net regression, the optimal length of training dataset is 10 trading days, larger or smaller training size only diminishes RC methods' performance. These counterintuitive results suggest that the predictability of RC methods using elastic net regression reaches a peak when training dataset containing around 1.5 million datapoint. Once more data is added for training, the performance of model will drop due to incapable of dealing with overloaded noise.

Thus, considering the efficiency of testing, in the later sections, only results drawn from model with training size equals 10 trading days will be discussed.

	Mean	Variance	Skewness	Kurtosis	$R^2 > 0$ (%)
Simple Linear Regression (5,1)	-0.26802	0.91996	-2.0511	10.8830	41.1765
Randomized Signature with ElasticNet(5,1)	0.01122	0.03909	10.80684	140.8782	47.0588
ESN with ElasticNet (5,1)	0.00378	0.03944	10.3858	133.5544	43.1373
Simple Linear Regression (10,1)	-0.14091	0.39815	-1.5217	7.3696	43.2
Randomized Signature with ElasticNet(10,1)	0.02213	0.05634	10.5949	133.2179	46.4
ESN with ElasticNet (10,1)	0.01405	0.05632	10.6086	135.0786	42.4
Simple Linear Regression (15,1)	-0.07193	0.31445	-2.6091	20.1699	44.0816
Randomized Signature with ElasticNet(15,1)	0.01337	0.02568	7.3139	82.9464	47.7551

ESN with ElasticNet (15,1)	0.00458	0.02663	6.9764	78.7622	38.7755

Table 3.4: Mean, variance, skewness, and kurtosis of R^2 (in %) of simple linear regression and RC methods trained by different size training dataset

Nonlinearity Makes a Difference

As introduced before, the training process in RC methods contains embedding and regression processes, here we test how different regression and embedding methods affect single variable RC methods. Still, results of simple linear regression are presented as benchmarks and only significant results of nonlinear models are displayed.

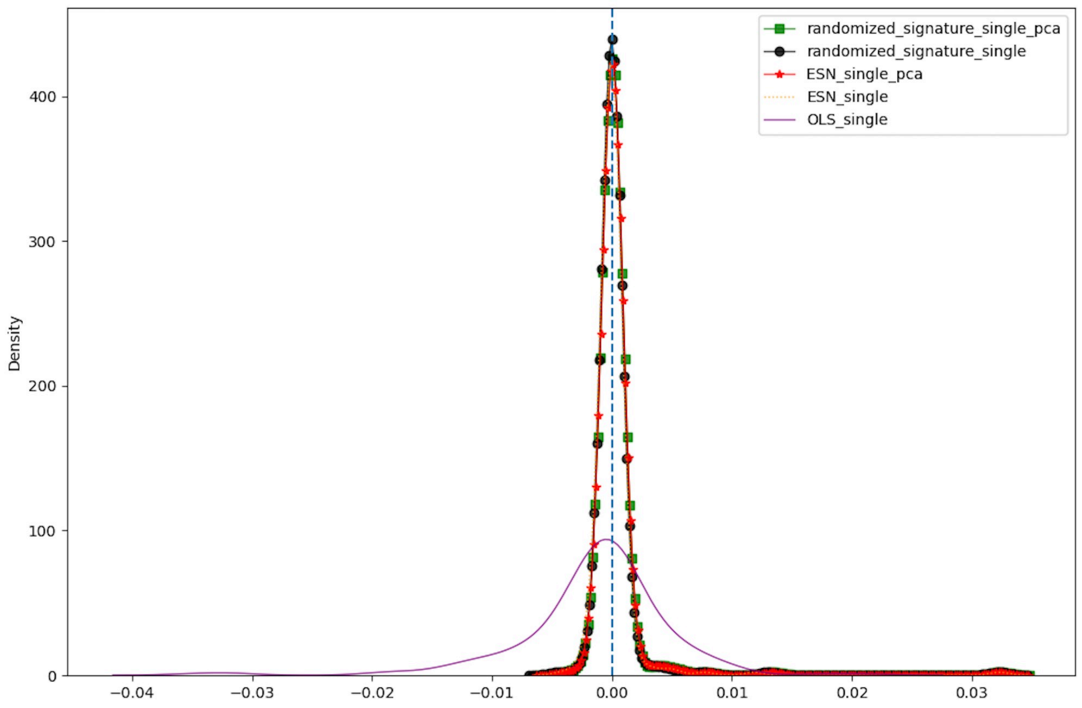
Statistics drawn from different models are shown in Table 3.5, the density of R^2 sequences are shown in Figure 3.3. It is clear that only RC methods utilizing elastic net regression can obtain positive average R^2, RC methods with other regression methods, for example, linear regression and lasso regression can not deliver satisfying results. This fact indicates that compare with regression without penalty, sparse regression does make a difference. Among sparse regression methods, only having the ability of compressing coefficients or sparsity is not enough, a combination of both characteristics is the key to success while implementing single variable model over a highly noisy dataset. The potential reason might be that after the input data being echoed in reservoir, the noise and useful message it contains will be amplified. Thus, dropping some noise and at the same time lower the density of repeated message can greatly improve the performance.

	Mean	Variance	Skewness	Kurtosis	$R^2 > 0$ (%)
Simple Linear Regression	-0.14091	0.39815	-1.5217	7.3696	43.2
ESN with ElasticNet	0.01405	0.05632	10.6086	135.0786	42.4
Randomized Signature with ElasticNet	0.02213	0.56340	10.5949	133.2179	46.4
ESN with ElasticNet & PCA	0.03280	0.05767	10.3085	127.1131	54.8
Randomized Signature with ElasticNet & PCA	0.03227	0.05763	10.3920	128.5606	55.6

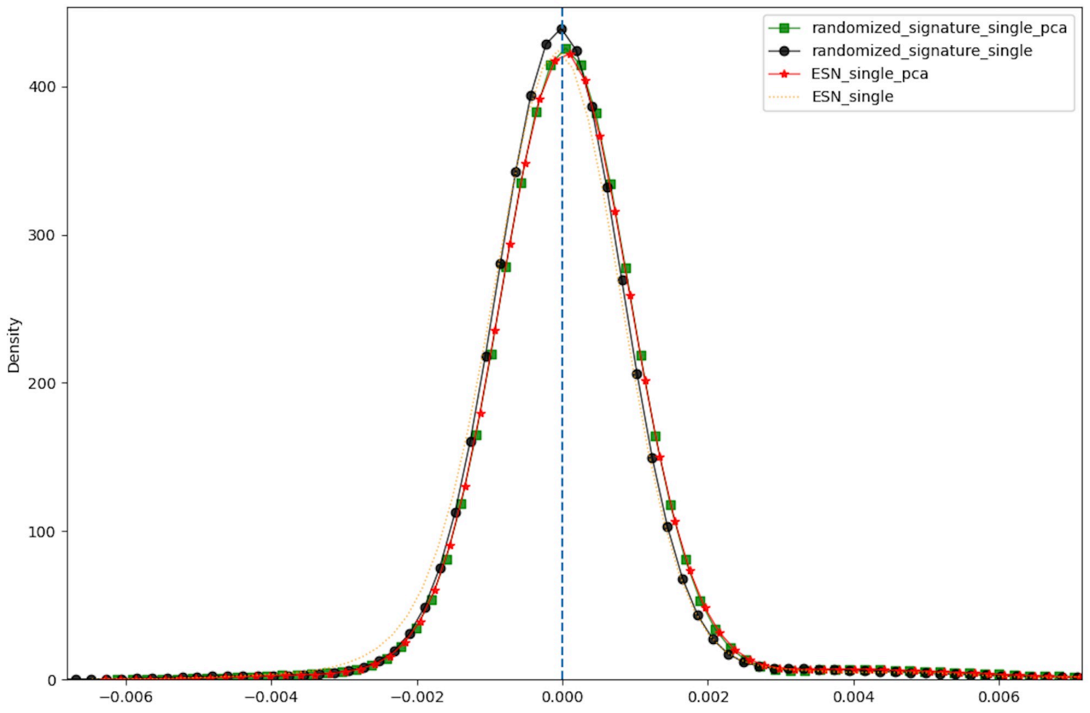
Table 3.5: Average, variance, skewness, and kurtosis of R^2(in %) obtained from simple linear regression and RC methods with different embedding methods

Aside from elastic net regression, dimensional reduction algorithm also helps improve models' performance. It can be witnessed that mean R^2 of RC methods with elastic net regression and PCA is significantly larger than the ones of RC methods using the same regression without conducting PCA.

A possible reason for the boost might be that even sparse regression method can drop some noise, the overall size of state matrix is still too large for a regression method. For instance, in our experiment, elastic net regression has to take 51 features transformed from one piece of information at each time step. In this case, reducing dimension of state data before training regression can



(a) Density of R^2 from linear and non linear models



(b) Density of R^2 from RC models.

Figure 3.3: Density of R^2 from Different Models

help linear model concentrate more on useful information.

One thing worthy to be mentioned is that even though linear regression sometimes obtains more positive R^2 compared with ESN using elastic net regression. ESN still demonstrates higher average R^2 and less scattered results. The fact indicates that alpha over the past 50 tick time is not informative enough for linear regression. Thus, compared with ESN, who can amplify input information, much smaller R^2s are obtained by linear regression, leading to a smaller average R^2 regardless of a higher percentage of positive R^2.

Randomized Signature Improves

From results shown in Table 3.4 and Table 3.5, it is clear that in most cases, randomized signature outperforms ESN significantly, which illustrates the outstanding predictability of randomized signature. The improvement may come from information carried by non-scaled input data and emphasis of previous information.

Optimal Single Variable Model

As Figure 3.3 and tables show, it is clear that compared with simple linear regression, RC methods generally have higher average R^2, higher tendency to have positive results, more stable outcomes and less extreme negative R^2 values.

Among RC methods, in general, randomized signature methods with sparse regression and PCA outperform most of others. Consider the average R^2, ESN using elastic net and PCA trained over past 10 trading days has the highest average R^2, but not the highest chance of achieving positive R^2, which indicates its strong ability in obtaining larger positive R^2 compared with randomized signature method implemented in the same scenario.

Based on these facts, the optimal single variable model is selected as ESN with elastic net regression and PCA trained over past 10 trading days.

3.3.2 Numerical Results of Multiple Variable Models

As shown in previous sections, when lots of efforts are made on single variable models only focus on alpha data, RC methods can have some predictability. However, single variable models' predictive power is limited and the cost of gaining positive R^2 is high. Thus, more features are added to help models achieve better results. This section includes results drawn from multiple variable models and discusses the causes of outcomes.

Choices of Features

Firstly, to avoid invalid linear regression model(invalid benchmark model) caused by multicollinearity, VIF of features is computed, results are shown in Table 3.6. The results show that severe multicollinearity exists among order flow imbalance from different levels. Thus, features that contain overlap information are discarded. To describe the process in a quantitative way, the standard can be set as features whose VIF is larger than 5 are dropped while conducting experiments.

Feature	VIF
$\text{Return}_{future,50}$	1.01102
QI_0	1.00328
QI_1	1.00978
QI_2	1.00453
QI_3	1.00465
QI_4	1.00508
QI_5	1.00446
QI_6	1.00538
QI_7	1.01022
QI_8	1.01525
traded-mid	1.00080
$\text{OFI}_{0,t}$	1.83546
$\text{OFI}_{1,t}$	6.00176
$\text{OFI}_{2,t}$	11.60641
$\text{OFI}_{3,t}$	16.39904
$\text{OFI}_{4,t}$	22.97832
$\text{OFI}_{5,t}$	31.34739
$\text{OFI}_{6,t}$	39.70684
$\text{OFI}_{7,t}$	46.98449
$\text{OFI}_{8,t}$	32.35030

Table 3.6: VIF of features

Choices of Hyperparameters

Similar to single variable models, multiple variable models are also tested in different scenarios. The empirical results show that when choosing Rho from 0.95, 0.5, 0.2, the optimal choice is 0.95, indicating that the model performs well when its memory fades slowly. The result is somewhat different from the one drawn from single variable model, indicating that the added features might essentially contain some useful information.

For leaking decay rate, the multiple variable models draw the same conclusion on it as single variable models do. The results indicate that the homogeneity property of states remains unchanged. Thus, 0 is chosen as leaking decay rate of multiple variable model.

Regarding the runs for initialization, same results hold. No significant improvement is seen from different situations. Thus, for the efficiency of model, the number of runs is chosen as 100 again.

Last but not least, multiple variable models also prefer a smaller reservoir considering the efficiency and computational cost of algorithm. Thus, 50 neurons are used in the reservoir.

Specific Model Performances

To find the optimal training size, multiple variable models are tested over training dataset in different sizes, results are shown in Table 3.7.

Results of simple linear regression show stable improvement with the increment of training size.

However, for RC methods, the positive correlation between training size and performance of models vanishes. Empirical outcomes suggest that all the RC methods perform the best while trained over 10 trading days. Larger and smaller training sets bring slight falls on average R^2 of RC methods, indicating that too large training set contains too much noise while too small training set cannot provide enough information for RC models. In this scenario, the optimal training and test ratio might be 10:1.

If we focus on ESN with different regression methods, some surprising results show that in most cases, ESN using lasso regression fails to outperform other models, including simple linear regression, which is not within our expectation since lasso regression commonly functions well when multiple variables are given. Since ESN with other regression methods perform far better than ESN using lasso regression, the reason is surely on lasso. Thus, we simply test different regression methods over the same dataset and find that lasso regression can not outperform linear regression nor elastic net regression, the results are shown in Table 3.8.

	Lasso	OLS	Elastic Net
Training size=5	0.33545	0.71410	0.87883
Training size=10	0.20030	0.78418	0.92941
Training size=15	0.13328	0.81244	0.92128

Table 3.8: Average R^2(in %) of different regression models

Based on the definition of regression methods, one major difference between those regressions is the ability of feature selection, so we take a careful look at the chosen features of different regression methods. The results are as follow:

- Lasso regression: queue imbalance at level 0, 1 and order flow imbalance t level 0 are chosen
- Elastic net regression: all the features are selected
- Linear regression: can not choose features

It is obvious that even if we have already dropped features with high multicollinearity, lasso regression still abandoned a decent amount of features. This is because lasso regression is greatly influenced by grouping effect, thus will wrongly drop features even though they should have been included. Expanding this finding into RC method, we know that the input of lasso regression in RC method at time t can be written as follow:

$$\hat{z}_t = \begin{pmatrix} \hat{z}_{t,1} \\ \hat{z}_{t,2} \\ \dots \\ \hat{z}_{t,p} \end{pmatrix}$$

Where $\hat{z}_{t,i},\ i = 1, 2, ..., p$ is the i^{th} principal component of state $\hat{s}_t = (\hat{s}_{t,1}, \hat{s}_{t,2}, ..., \hat{s}_{t,n})$, p is the number of principal components and n is the dimension of features. As defined by PCA, each $\hat{z}_{t,i}$ is a linear transform of $\hat{s}_t$. We can deduce that when implementing lasso regression on $\hat{z}_t$, there must be some $\hat{z}_{t,i}$ that have high correlation with each other considering they are all from the same information. Thus, due to clustering effect, main part of $\hat{z}_{t,i}$ will be dropped by lasso regression in ESN, leading to an omission of useful information. To prove the guess, we include 0 into the choosing range of lasso regression's[2] penalty coefficient. Results show that 0 is the optimal

[2]In ESN

choice, indicating that lasso is not operating in this scenario. Thus, in the later discussions, lasso regression will not be included consider its incapability of dealing the given features.

Except for the abnormal results of ESN using lasso regression, outcomes from other RC methods all outperform benchmark model in different degree. Also, randomized signature and ESN show very similar performances, which aligns with our expectation. The outcomes also display that all the RC methods can slightly outperform the regression method they use. However, it can be witnessed from Table 3.8 and Table 3.7 that RC methods using linear regression bring higher improvement of average R^2 over its regression method while the high average R^2 of RC methods using elastic net rely less on RC methods' structures. Besides, since ESN using elastic net regression can always outperform ESN using linear regression, we can deduce that elastic net regression is more suitable for dealing with the amplified information compared with OLS regression. Specific reasons for the difference will be discussed in details in the next subsection.

3.3.3 Investigative Experiments & Optimal Multiple Variable RC Methods

Source Power of RC Methods

Table 3.7 shows that in each scenario, RC methods can outperform benchmark model, among the two RC methods, ESN always has the highest average R^2 among all the models, it can even slightly outperform randomized signature stably. The outstanding performances of RC methods raise our curiosity: what drives them to outperform? Experiments are then implemented to investigate the magic power that helps RC methods outperform.

First of all, we start from the initialization processes of W_{res} and W_{in} in ESN. Here the initialization methods of matrices are changed by generating W_{res} via standard uniform distribution rather than from uniform distribution ranging from -1 to 1, the process lower the spectral radius of W_{res}. Then we set W_{in} as an identity matrix. Results trained over small training set(with the most significant differences) are shown in Table 3.9. It can be witnessed from the results that an inappropriately initialized W_{res} can even lead to negative average R^2 value. For W_{in}, results show that setting it as identity matrix can help improve the performances of ESN, but the improvement brought by W_{in} cannot offset the negative impact brought by a less appropriate W_{res}. Thus, we conclude that W_{res} is the main reason for a good ESN(results over larger training set demonstrate the same conclusion). The outcomes also prove that when there is no zero in the input matrix, the restriction of spectral radius $|\lambda_{max}| < 1$ can be relaxed. For randomized signature method, the initialization methods of matrix is nearly fixed and does not allow much space for investigation.

Even though we have found that initialization of W_{res} is vital to ESN's success. It is still stunning that when trained over small training set, inappropriate W_{res} can turn average R^2 value from positive to negative. The abnormal behaviour of ESN is obviously an overfitting. By looking at time-wise R^2 values in Figure 3.4 of two ESN models: ESN using elastic net and OLS with W_{res} initialized by standard uniform distribution, it can detected that the disastrous overfitting of ESN over small training set is caused by some abnormally negative R^2 values, which totally cancel the effect of all positive R^2 values. From this fact, we can deduce that properly initialized initialization matrices can help ESN capture useful information more efficiently from a small training set and thus preventing overfitting.

	Mean (%)	Variance	Skewness	Kurtosis	$R^2 > 0$ (%)
ESN & PCA & OLS Original $\|\lambda_{max}\| > 1$	0.85357	1.76306	-1.85425	15.50343	92.15686
ESN & PCA & Elastic Net Original $\|\lambda_{max}\| > 1$	0.90639	0.36971	2.13331	19.29729	96.86275
ESN & PCA & OLS Standard Uniform $\|\lambda_{max}\| > 1$	-0.37136	57.21940	-6.97449	50.41778	96.27451
ESN & PCA & Elastic Net Standard Uniform $\|\lambda_{max}\| > 1$	-0.27034	53.75463	-7.40584	54.82481	90.19608
ESN & PCA & OLS Standard Uniform $\|\lambda_{max}\| \approx 0.86$	-0.05076	27.55537	-6.62018	49.56922	87.05882
ESN & PCA & Elastic Net Standard Uniform $\|\lambda_{max}\| \approx 0.86$	0.00581	25.6	-7.48734	60.55925	89.80392
ESN & PCA & OLS Standard Uniform $\|\lambda_{max}\| \approx 0.86$ W_{in} Identity	-0.02075	23.45740	-6.03584	41.09481	86.66667
ESN & PCA & Elastic Net Standard Uniform $\|\lambda_{max}\| \approx 0.86$ W_{in} Identity	0.031469	21.56926	-6.86233	50.28206	89.41176

Table 3.9: Average, variance, skewness, and kurtosis of R^2(in %) obtained from simple linear regression and RC methods with different embedding methods

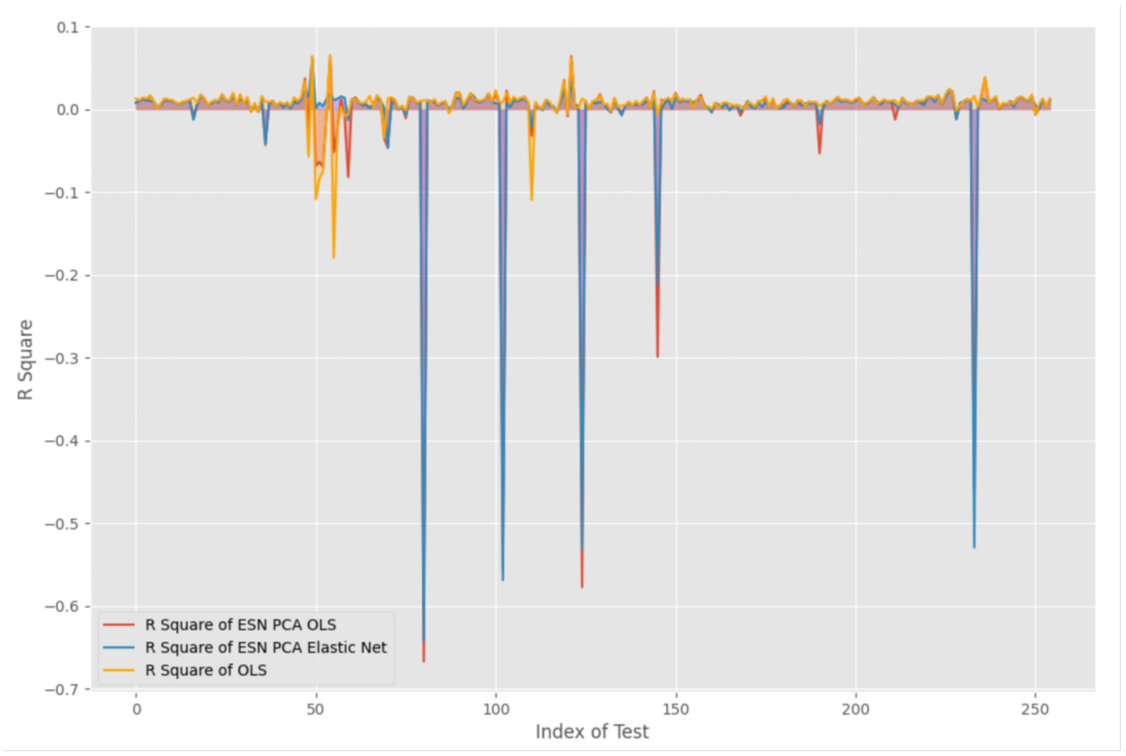


Figure 3.4: Step wise R^2 value of OLS & ESN models initialized by standard uniform distribution

Besides initialization matrices, with a closer look at performances of different models, we might also be curious about if the outperforming ability of RC methods over the regression method it uses partly comes PCA method. To figure the answer, we carry out experiments without PCA, results are shown in Table 3.7. It is clear that without PCA, performances of RC methods drop significantly. More surprisingly, ESN methods without PCA can not outperform benchmark model in many cases. These facts reveal that PCA plays a crucial role in the outperforming of RC methods over linear regressions, which might due to the property that reservoir will extend the dimension of data: from D to N, in our experiment, is from 12 to 50. The extension process does not essentially make the features more informative, but hugely increases the dimension of them. Thus, considering linear regression's limited ability in dealing with large amount of features, using PCA to extract more informative factors and reduce the dimension of features that are to be fed to regression is of great importance.

Another interesting finding from results of RC methods without PCA is that even though ESN without PCA underperforms OLS from time to time, randomized signature without PCA can still stably outperform OLS under the same condition. It can be deduced from the fact that when dealing with the given forecasting task, randomized signature has a reservoir structure that functions better than ESN's.

Last, regressions used in RC methods are considered. Table 3.7 shows that under the same condition, RC methods implementing elastic net regression outperforms RC methods using linear regression. This fact can be potentially explained by RC methods' capability of memorizing information as elastic net regression can lower the density of information by compressing coefficients.

In all, for ESN, its predictive ability comes from the combination of a carefully structured initialization matrices in ESN, PCA and a strong sparsity regression method, no single factor can take the credit of ESN's outperforming. While for randomized signature, its reservoir structure and a suitable regression method seem to be more important in model's predictability.

Remarks from Exploratory Experiments

In the previous subsection, we find that ESN requires careful cultivation to achieve its optimal performance. However, randomized signature methods achieve stable outputs more effortlessly. To investigate what contribute to this unfair difference, when α is set as 0, we modify randomized signature as:

$$s_t = f(A_1 s_{t-1} + \epsilon_1) + \sum_{i=1}^{d} f(A_2^i s_{t-1} + \epsilon_2^i) \times f(x_t^i) \tag{3.3.1}$$

Experiments are then implemented using reservoir constructed via (3.3.1). Results are shown in Table 3.10. The outcomes reveal that when trained over small training set, randomized signature scaled by tanh function overfits at some points, which makes the model deliver very similar overall results to ESN with inappropriately chosen W_{res}, indicating that tanh function can lead to overfitting of RC methods over small training set when the initialization matrices are not strictly set. However, while trained over larger training set, randomized signature performs similar to or even outperform the optimal ESN model. These abnormal facts may demonstrate that over larger training set, when overfitting is no longer a problem, the combination of tanh function and the sum of features($\sum_{i=1}^{d} f(A_2^i s_{t-1} + \epsilon_2^i) \times f(x_t^i)$) can help randomized signature method obtain some surprising outcomes.

	Model	Mean (%)	$R^2 > 0$ (%)
Training size = 5	RS & PCA &OLS Scaled by tanh	-0.03099	86.66667
	RS & PCA &Elastic Net Scaled by tanh	0.02201	89.41176
Training size = 10	RS & PCA &OLS Scaled by tanh	0.91525	92.4
	RS & PCA &Elastic Net Scaled by tanh	0.82485	94
Training size = 15	RS & PCA &OLS Scaled by tanh	0.88068	91.83673
	RS & PCA & Elastic Net Scaled by tanh	0.92601	98.77551

Table 3.10: Average R^2(in %) and percentage of positive R^2 values(in %) obtained from models trained with different dataset. RS stands for randomized Signatre method

To sum up, we deduce from the results that not scaled x_t might be the main reason for randomized signature's universally stability, while the summation in randomized signature's structure has a high chance to contribute to its outperforming. However, the evidences are not solid enough and more experiments await to be undertaken for validation.

Optimal Multiple Variable Model

Based on all the results shown above, it can be witnessed that all the models can always achieve positive R^2 values and are able to predict the future alpha in more than 90% of cases, indicating the predictive power of chosen features. For multiple variable reservoir computing models, it is clear that a combination of appropriate regression methods, PCA and the structure of RC method are the key factors that help RC methods outperform. Thus, optimal RC method is chosen from randomized signature using PCA and elastic net and ESN with the same dimension reduction and regression algorithm.

We can conclude from the results that carefully initialized ESN can obtain the highest overall average R^2 values. However, this achievement relays on initialization matrices in an incredibly degree, making the outstanding performance of ESN fragile. While for randomized signature methods, even though in some cases they slightly underperform ESN models, but during the experimental processes, they all show very low requirements on their initialization processes. The simplicity in hyperparameter choosing showed by randomized signature methods can be a huge advantage over ESN since one of the reason reservoir computing is chosen in applications is because of its nature of simplicity. To sum up, if we emphasize the overall performance of models, ESN with PCA and elastic net regression might be the optimal choice. However, if highly efficient hyperparameters choosing process is more preferred, then randomized signature using PCA and elastic net regression should be considered.

	Model	Mean	Variance	Skewness	Kurtosis	$R^2 > 0$ (%)
Training size = 5	Simple Linear Regression	0.71410	4.08259	-5.05760	37.93469	90.98039
	ESN with OLS	0.60810	1.74539	-1.56914	15.09503	87.45098
	ESN with Elastic Net	0.67637	0.37807	2.30931	20.84556	92.94118
	ESN with OLS & PCA	0.85357	1.76306	-1.85425	15.50343	92.15686
	ESN with Lasso & PCA	0.33450	0.123257	1.04523	50.41778	86.27451
	ESN with Elastic Net & PCA	0.90639	0.36971	2.13331	19.29729	96.86275
	RS with OLS	0.80240	1.76931	-1.82143	15.39583	90.98039
	RS with Elastic Net	0.85560	0.36797	2.18499	19.91063	96.47059
	RS with OLS & PCA	0.84576	1.76832	-1.86651	15.46317	91.76471
	RS with Elastic Net & PCA	0.90042	0.36929	2.07640	18.54016	96.47059
Training size = 10	Simple Linear Regression	0.78418	5.74158	-6.25835	87.34904	92.4
	ESN with OLS	0.78306	2.47393	2.16640	41.42622	91.2
	ESN with Elastic Net	0.81238	0.42095	5.03256	50.47135	96.4
	ESN with OLS & PCA	0.91919	2.50205	1.98683	39.90611	92.4
	ESN with Lasso & PCA	0.20022	0.06927	1.33238	1.38901	76
	ESN with Elastic Net & PCA	0.94302	0.41447	4.90828	48.63432	97.6
	RS with OLS	0.89587	2.49576	2.01074	39.98387	92.4
	RS with Elastic Net	0.91848	0.41400	4.89676	48.82737	98
	RS with OLS & PCA	0.91342	2.51462	1.91330	39.57926	92.4
	RS with Elastic Net & PCA	0.93697	0.41626	4.81827	47.71563	98
Training size = 15	Simple Linear Regression	0.81244	2.60409	-2.05322	19.97967	90.20408
	ESN with OLS	0.77822	2.10817	-1.20040	20.98608	91.02041
	ESN with Elastic Net	0.82366	0.25478	2.33369	15.34193	97.95918
	ESN with OLS & PCA	0.881253	2.10768	-1.30237	20.88781	91.93548
	ESN with Lasso & PCA	0.133257	0.04512	1.67819	3.07716	71.42857
	ESN with Elastic Net & PCA	0.92634	0.25701	2.18059	13.76337	98.77551
	RS with OLS	0.86576	2.14961	-1.33169	20.57463	91.42857
	RS with Elastic Net	0.91098	0.257305	2.17494	13.98130	98.36735
	RS with OLS & PCA	0.87707	2.14093	-1.30451	20.65664	91.83673
	RS with Elastic Net & PCA	0.92247	0.25784	2.17310	13.86121	98.77551

Table 3.7: Average(in %), variance, skewness, kurtosis, and percentage of positive values(in %) of R^2 obtained from models trained with different dataset. RS stands for randomized Signatre method

Chapter 4

Possible Improvements

Looking back at the models, there are some modifications that can potentially improve their performances. The first is the choice of hyperparameters of ESN. In our experiments, hyperparameters of ESN are not chosen by a very large parameter search grid. Thus, the hyperparameters in our experiments have a great chance to not be the optimal ones. We assume that if more parameters are tested, possibly better model performances can be obtained.

Second, empirical results show that informative features are key factors to model's predictability. Therefore, more carefully constructed features are worthy to be considered in further research. For example, the order flow imbalance factors we use in experiments are constructed symmetrically, some researches show that asymmetric features might be more informative, so trying asymmetric order flow imbalance features might help boost R^2.

Third, even though this thesis has conducted some experiments on what make the gap between ESN and randomized signature, more systematical investigations should be applied to deliver more convincing results.

Besides, it can be witnessed that randomized signature does not have feedback mechanism, so we boldly assume that adding a feedback path in randomized signature may be an option to be considered for further improvement.

Last but not least, it is obvious that dimension reduction or feature selection methods can adding some value to model performances. Based on these facts, we think some more complex methods such as attention mechanism can also be applied in RC methods for boosting model's performance.

Conclusion

In this thesis, we implement two RC methods on order book and trade book data. The empirical results of single variable model shows that while using alpha as the only feature, simple linear regression suggests alpha as one that's not informative. However, by implementing RC methods with sparse regression method and PCA over the same feature, we managed to predict alpha with some degree of accuracy, revealing some predictability of RC methods. Among all the single variable RC methods, randomized signature shows the strongest ability in foresting and prove itself as a robust method according to its stable performances in all the scenarios.

Outcomes from multiple variable models show that with more informative features, all the models experience significant increment in average R^2 value. Specifically speaking, RC methods can stably outperform benchmark model in all the scenarios. Among RC methods, ESN using PCA and elastic net regression has the highest average R^2 while randomized signature has universally robustness over different hyperparameters. We also find that the predictive ability of ESN used in the thesis roots from its overall structure rather than one part of it. While randomized signature outperforms because its reservoir's structure and regression method(elastic net regression) used. Also, during the process of hyperparameter choosing, randomized signature shows lower demand while ESN is more sensitive to hyperparameters. Based on all the facts, when choosing models, we can start from specific needs. For example, if high average R^2 is the ultimate goal, then ESN with PCA and ealstic net can be used. Otherwise, if a quick hyperparameter choosing is prioritized, and slight drop in R^2 values is acceptable, then randomized signature is the best option.

In all, we can conclude that RC methods function well in forex market prediction and the specific model choice dependents on actual needs.

Bibliography

[1] Marc Levinson et al. *The economist guide to financial markets: Why they exist and how they work*. The Economist, 2014.

[2] Triennial Central Bank Survey. Otc foreign exchange turnover in april 2022: Preliminary global result. 2022,10,27.

[3] Michael Ayitey Junior, Peter Appiahene, Obed Appiah, and Christopher Ninfaakang Bombie. Forex market forecasting using machine learning: Systematic literature review and meta-analysis. *Journal of Big Data*, 10(1):9, 2023.

[4] Salman Ahmed, Saeed-Ul Hassan, Naif Radi Aljohani, and Raheel Nawaz. Flf-lstm: A novel prediction system using forex loss function. *Applied Soft Computing*, 97:106780, 2020.

[5] Deniz Can Yıldırım, Ismail Hakkı Toroslu, and Ugo Fiore. Forecasting directional movement of forex data using lstm with technical and macroeconomic indicators. *Financial Innovation*, 7:1–36, 2021.

[6] Svetoslav Zhelev and Dimiter R Avresky. Using lstm neural network for time series predictions in financial markets. In *2019 IEEE 18th international symposium on network computing and applications (NCA)*, pages 1–5. IEEE, 2019.

[7] Mohammad Zoynul Abedin, Mahmudul Hasan Moon, M Kabir Hassan, and Petr Hajek. Deep learning-based exchange rate prediction during the covid-19 pandemic. *Annals of Operations Research*, pages 1–52, 2021.

[8] Svitlana Galeshchuk and Sumitra Mukherjee. Deep learning for predictions in emerging currency markets. In *ICAART (2)*, pages 681–686, 2017.

[9] Charles Cao, Oliver Hansch, and Xiaoxin Wang. The information content of an open limit-order book. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 29(1):16–41, 2009.

[10] Tarun Chordia, Richard Roll, and Avanidhar Subrahmanyam. Order imbalance, liquidity, and market returns. *Journal of Financial economics*, 65(1):111–130, 2002.

[11] Orawan Chantarakasemchit, Siranee Nuchitprasitchai, and Yuenyong Nilsiam. Forex rates prediction on eur/usd with simple moving average technique and financial factors. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 771–774. IEEE, 2020.

[12] AS Babu and SK Reddy. Exchange rate forecasting using arima. *Neural Network and Fuzzy Neuron, Journal of Stock & Forex Trading*, 4(3):01–05, 2015.

[13] Juergen Schmidhuber. Annotated history of modern ai and deep learning. *arXiv preprint arXiv:2212.11279*, 2022.

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

[15] Lina Ni, Yujie Li, Xiao Wang, Jinquan Zhang, Jiguo Yu, and Chengming Qi. Forecasting of forex time series data based on deep learning. *Procedia computer science*, 147:647–652, 2019.

[16] Zhiwen Zeng and Matloob Khushi. Wavelet denoising and attention-based rnn-arima model to predict forex price. In *2020 International joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2020.

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

[18] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[19] David Verstraeten, Benjamin Schrauwen, Michiel D'Haene, Dirk Stroobandt, et al. A unifying comparison of reservoir computing methods. *Neural Networks*, 20:391–403, 2007.

[20] Leandro Maciel, Fernando Gomide, David Santos, and Rosangela Ballini. Exchange rate forecasting using echo state networks for trading strategies. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 40–47. IEEE, 2014.

[21] Xiaowei Lin, Zehong Yang, and Yixu Song. Short-term stock price prediction based on echo state networks. *Expert systems with applications*, 36(3):7313–7317, 2009.

[22] Taehwan Kim and Brian R King. Time series prediction using deep echo state networks. *Neural Computing and Applications*, 32:17769–17787, 2020.

[23] Patrick Kidger, Patric Bonnier, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] Peter K Friz and Martin Hairer. *A course on rough paths*. Springer, 2020.

[25] Ben Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, pages 109–167, 2010.

[26] Christa Cuchiero, Lukas Gonon, Lyudmila Grigoryeva, Juan-Pablo Ortega, and Josef Teichmann. Expressive power of randomized signature. In *The Symbiosis of Deep Learning and Differential Equations*, 2021.

[27] Enea Monzio Compagnoni, Luca Biggio, Antonio Orvieto, Thomas Hofmann, and Josef Teichmann. Randomized signature layers for signal extraction in time series data. *arXiv preprint arXiv:2201.00384*, 2022.

[28] Chenxi Sun, Moxian Song, Shenda Hong, and Hongyan Li. A review of designs and applications of echo state networks. *arXiv preprint arXiv:2012.02974*, 2020.

[29] Bernhard Schäfl, Lukas Gruber, Johannes Brandstetter, and Sepp Hochreiter. G-signatures: Global graph propagation with randomized signatures. *arXiv preprint arXiv:2302.08811*, 2023.

[30] Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[31] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.

[32] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[33] Petter N Kolm, Jeremy Turiel, and Nicholas Westray. Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. *Available at SSRN 3900141*, 2021.

[34] Rama Cont, Arseniy Kukanov, and Sasha Stoikov. The price impact of order book events. *Journal of financial econometrics*, 12(1):47–88, 2014.

[35] Martin D Gould and Julius Bonart. Queue imbalance as a one-tick-ahead price predictor in a limit order book. *Market Microstructure and Liquidity*, 2(02):1650006, 2016.

[36] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the" echo state network" approach. 2002.