VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE II

Master thesis

Forecasting of Financial Data

Done by:
Povilas Versockas signature

Supervisor:
dr. Tadas Meškauskas

Vilnius
2016

Contents

Abstract

In this work, stock market data is modelled using linear regression and multilayer perceptron models. Using these models, stock market prices are predicted. For each stock multilayer perceptron with different activation functions, size of the hidden and input layer are trained. Experiments show that, no model predicts best for all of the cases.

Prediction results are associated with nonlinear analysis measures, such as largest Lyapunov exponent and correlation dimension. Automatic systems for computing these measures are presented. The systems estimation accuracy is then checked with known chaotic maps and equations. Moreover, systems are validated with noisy time series.

Lastly, time series classification based on nonlinear measures is introduced. Classification is then used with stock market data. Only classification using largest Lyapunov exponent gives promising results, this is due to errors in computing correlation dimension. Results show that this type of classification could be used for filtering out stocks that are harder to predict.

Santrauka

Finansinės kilmės duomenų prognozavimas

Darbe prognozuojami akcijų biržų duomenys naudojantis tiesinės regresijos bei daugiasluoksnio perceptrono modeliais. Kiekvienai akcijai yra parenkama daugiasluoksnio perceptrono aktyvacijos funkcija, vidinio sluoksnio dirbtinių neuronų skaičius, įvesties taškų skaičius bei palyginami modelių rezultatai prognozuojant akcijų biržų kryptį.

Gauti prognozavimo rezultatai yra susiejami su netiesinės duomenų analizės rodikliais - didžiausia Liapunovo eksponente bei koreliacine dimensija. Liapunovo rodiklis yra panaudojamas klasifikuoti finansinės kilmės duomenims. Gauti rezultatai rodo, kad šią klasifikavimo sistemą galima taikyti norint atrinkti prognozuojamas akcijas.

Introduction

Accurate forecasting of market returns is a difficult problem. So far, experimental studies have shown that market returns data is very non-stationary, time-varying [47] and has nonlinear behaviour [32], [2], [45]. Also, there is some debate whether financial data may be chaotic or not [50].

McKenzie using close returns test showed that major national stock market indexes: S&P 100, S&P 500, Nikkei Dow, Straits Times, Swiss & Hang Seng indexes are chaotic [32]. Yousefpoor et al. using BDS test, largest Lyapunov exponent (LLE) method & close returns test showed that five stocks from Tehran stock exchange have sensitive dependence on initial conditions, but the data is not chaotic [50]. Das et al. used LLE method to investigate foreign exchange data during 2008 to 2009 recession and found that the data also exhibits sensitive dependence on initial conditions [10].

Efficient market hypothesis states that all available information in market is reflected in stocks price so that it is not possible to predict stock prices using the historical price date. But there are empirical evidence that markets often do not follow efficient market hypothesis. Fama and French, Ferson showed that financial variables could be forecasted using time series of financial data [13], [14]. Many technical analysts believe that most information about the stocks are reflected in historic prices and it's trends.

There are several approaches to modelling stock market return data. In the past statistical models have been used such as autoregressive integrated moving average, linear regression, multiple linear regression, Kalman filter (linear quadratic estimation). But the problem with these models is that they can only model linear relationship between input and output variables. So recently, more research has been developed to predict stock market data using artificial neural networks (ANN).

ANNs gained popularity because these methods can capture nonlinear relations between features and response variables; methods are data driven, no prior explicit assumptions about data are needed [36]. Atsalakis and Valavanis survey shows that soft computing methods outperform conventional methods in most cases [2]. Qiu et al. has used feed forward neural networks to predict returns of Japanese Nikkei 255 index [40]. Bernal et al. implemented echo state network to predict S&P 500 index and obtained better results than using Kalman Filter [5]. Guresen et al. has successfully used multilayer perceptron model to predict NASDAQ Stock Exchange index and compared results with other ANN models [19]. Although many papers have shown that ANN models achieve better performance than conventional methods, the financial time series prediction problem is still unsolved.

Combining nonlinear analysis techniques with flexible prediction methods can provide useful information about the problem. Using measures like correlation dimension that measures how complex is the system, which generated the time series data, or largest Lyapunov exponent that measures exponential divergence of close by trajectories, should help group similar complexity or predictability signals. This grouping should help to choose, which of the time series could be predicted or even select model's parameters.

In this work, predictive modelling is described in 1 section with linear regression model in 1.1 subsection, artificial neural networks and multilayer perceptron model in 1.2 subsection. Modelled financial time series and experiments on the series are described in 2, 3 sections. Experiments show, that no single model performs best on the stock market data. Also, that different historic price windows should be used in the input layer of multilayer perceptron model, as it turns out, that using only one day's before price is not enough for model to generalize well.

Nonlinear methods are presented in 4 section and used in automatic correlation dimension estimantion and automatic largest Lyapunov exponent estimation, which are in 5 section. These sections and 7.1 subsection are taken from scientific research project done before [48]. Automatic estimation systems accuracies are evaluated with known chaotic maps and equations. Additionally, systems are tested with noisy time series data. Automatic largest Lyapunov exponent estimation method shows quite good results even for noisy data, while presented automatic correlation dimension estimation method is sensitive to noise. Using these systems, classification system based on data binning (histogram) approach is described in section 7. The classification system is then evaluated on stock market data. The experiments in 7.2, 7.3 subsections show promising results, when using LLE measure. Grouping provided by data binning, helps to filter out stocks that are harder to predict.

1 Predictive modelling

Predictive modelling assumes that there is a relationship between some observed quantitative response y and **x** $= (x_1, x_2, x_n)$ different variables, which can be written in general form:

$$y = f(\mathbf{x}) + \epsilon, \tag{1.1}$$

where f is some unknown function of **x**, and ϵ is a random error term, which is independent of **x** and has a zero mean [20]. The goal of predictive modelling is to estimate function f, which represents the systematic information that **x** provides about y. In this setting, we assume that input variables **x** are available, but the output y cannot be easily obtained. Since the error term average to zero, we can predict y using

$$\hat{y} = \hat{f}(\mathbf{x}), \tag{1.2}$$

where $\hat{f}$ represents our estimate for unknown function f, and $\hat{y}$ is the resulting prediction values for y.

Usually, $\hat{f}$ will not be a perfect estimate for f, and this inaccuracy will result in some error. This error can be divided into reducible error and irreducible error. Reducible error is the error that can be improved, for example by using different machine learning models to estimate f. Meanwhile, y is also a function of ϵ, which, cannot be predicted using **x**. Therefore, variability associated with ϵ also affects the accuracy of our predictions. This is known as the irreducible error, because no matter how well we estimate f, we cannot reduce the error introduced by ϵ. The ϵ may contain unmeasured variables that would be useful for predicting y or the measurement errors, which accumulate while measuring the data. Generally, given an estimated function $\hat{f}$ and input variables **x**, it is possible to show that the prediction error is equal to:

$$E(y - \hat{y})^2 = E(f(\mathbf{x}) + \epsilon - \hat{f}(\mathbf{x}))^2 = (f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2 + var(\epsilon), \tag{1.3}$$

where $(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2$ is the reducible error, which represents the expected value, of the squared difference between the predicted and actual values of y and $var(\epsilon)$ is the irreducible error, which represents the variance of ϵ. It is important to understand that the irreducible error exists and it will always provide an upper bound on the accuracy of prediction.

Generally, predictive methods try to learn important characteristics from the data. Before using predictive model, data of $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ values needs to be collected and prepared. After that dataset is divided into training and test sets. Training set is used for "training" the model, while test set is for assessing the resulting model. Test set must only be used after model's training phase, so that model is assessed correctly. Usually, most of the dataset samples are in the training set. One common way of dividing dataset is to use 80% of data points for training and 20% for testing.

One of the big problems in predictive modelling is overfitting. This happens when model learns to mimic behaviour of training set, which does not generalize for test set. When model is overfit, it has a poor predictive power and it fails to capture dataset trends. Usually models, that have too many parameters, relative to the number of observations, have this problem. Techniques such as regularisation, cross-validation, early stopping may reduce the problem. Regularisation penalizes over-complex models by adding a weighted term, which depends on model parameters and does not allow model to completely fit the training data. Early stopping idea is to stop model's training phase before model's loss function's error minimum has been found. This can be done by limiting the number of iterations model is allowed to do over the training sample.

1.1 Linear regression

Linear regression is a simple statistical model used for regression type problems. This model can be used not only for prediction, but also for regression analysis, which is a method to discover the relationship between response variable y and input variables $x_1, x_2, ..., x_n$. For example, it could help to uncover causal relationship between y and $x_1, x_2, ..., x_n$, or find which of the input variables are more important to explain the response variable. Also, hypothesis testing could be used in order to statistically prove that there is no relationship between some input variable and resulting response.

There are many different variations of this model, but usually multiple linear regression is used. Multiple linear regression models function $\hat{f}$ as a linear combination of input variables and model's coefficients:

$$\hat{f}(\mathbf{x}) = \beta_0 + \beta_1 * x_1 + ... + \beta_n * x_n, \tag{1.4}$$

where $\beta_0, \beta_1, ..., \beta_n$ are regression coefficients, $\mathbf{x} = (x_1, ..., x_n)$ are the input variables. In the simplest case with $n = 1$, linear regression can be seen as a line with slope equal to β_1 and intercept equal to β_0, as seen in figure 1.

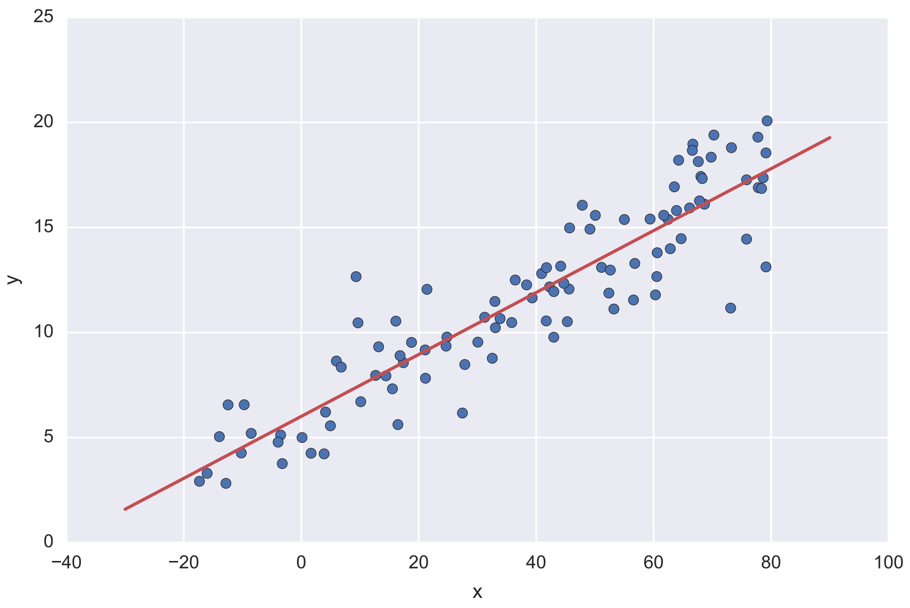

Figure 1. Linear regression model trained on a generated dataset. Red line shows the trained model, which is desribed by $\hat{f}(x_1) = 0.148 + 6.09x_1$.

In order to find corresponding parameters method of least squares is used. The method minimizes overall solution based on the sum of squared errors:

$$E_{total} = \sum_{i=1}^{m} (y_i - \hat{f}(\mathbf{x}_i))^2, \tag{1.5}$$

where m is the number of elements in the training sample, y_i is the actual value, $\hat{f}(\mathbf{x}_i)$ is the predicted value. Squared error E_{total} just accumulates models $\hat{f}$ error on each data point. Finding the correct $\beta_0, \beta_1, ..., \beta_n$ is just minimizing E_{total}. It is possible to solve this problem analytically using normal equations method. Normal equations describe a solution if $x_1, x_2, .., x_n$ are linearly independent. In that case parameters can be found using:

$$\beta = (X^T X)^{-1} X^T Y, \tag{1.6}$$

where X is a n by $m+1$ matrix with first column of all ones and then all the x values, Y is a vector of y values. Resulting vector β contains the $\beta_0, \beta_1, ..., \beta_n$ parameter values.

One of the key drawbacks of linear regression model, that it can only model linear relationship between input variables and y. This problem may be avoided by transforming the input variables, for example by raising x to different degrees of polynomials: $\hat{f} = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + ...$, which is also known as polynomial regression. Although it may solve this drawback, it makes the model more complex by needing to choose degrees of a polynomials for input variables.

Another problem is modelling interactions between the input variables. Linear regression model assumes that $x_1, x_2, ..., x_n$ are independent. If input variables are correlated, then model should also add terms to account for it, e.g. $\hat{f} = \beta_0 + \beta_2 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + ...$.

Generally, there are many more problems associated with this model and there are many variations to alleviate these problems. But the basic model defined by equation 1.4 gives a lot of information about the dataset and it serves as a good, understandable reference model.

1.2 Artificial neural networks

1.2.1 Introduction

Artificial neural networks are flexible computing methods that have the ability to capture complex patterns among different input variables. These methods have characteristics (mentioned in introduction section) that make them useful for classification and regression problems. ANNs were inspired by the biological understanding of the brain. Like in human brain ANNs are composed of many neurons, which take some input signal and resolve it into output signal.

Artificial neuron is a single computational unit, which takes some input variables and outputs a local decision based on its parameters. Artificial neuron computation can be defined as:

$$h(x) = g(b + w_1 x_1 + w_2 x_2 + ... + w_n x_n), \tag{1.7}$$

where g - activation function, w_i - ith connection weight, x_i - ith input object's attribute, b - bias unit. The computation is described by its parameters - weights, bias unit values and its activation function. If linear activation function is used such as $g(\mathbf{x}) = \mathbf{x}$, the resulting computation is a linear regression model. Usually artificial neurons are used with nonlinear activation function such as sigmoid function (shown in figure 2) or hyperbolic tangent, which allows multiple combined artificial neurons to learn complex functions.

When composing networks of artificial neurons, neurons are separated into layers. First layer is called the input layer, which does not do any computation, it just sends the input data values to the next layer. Last layer is called output, it is the predicted value of the model. All layers between input and output layer are hidden layers.

In order to use NNs for a particular problem, one should decide on the network topology, number of network layers, number of artifical neurons in each layer, activation function of the neurons and finally the learning algorithm.

Based on topology there are two types of NN methods: feed-forward, which only allows artificial neurons from one layer to be connected to the next without any loops, and recurrent, which

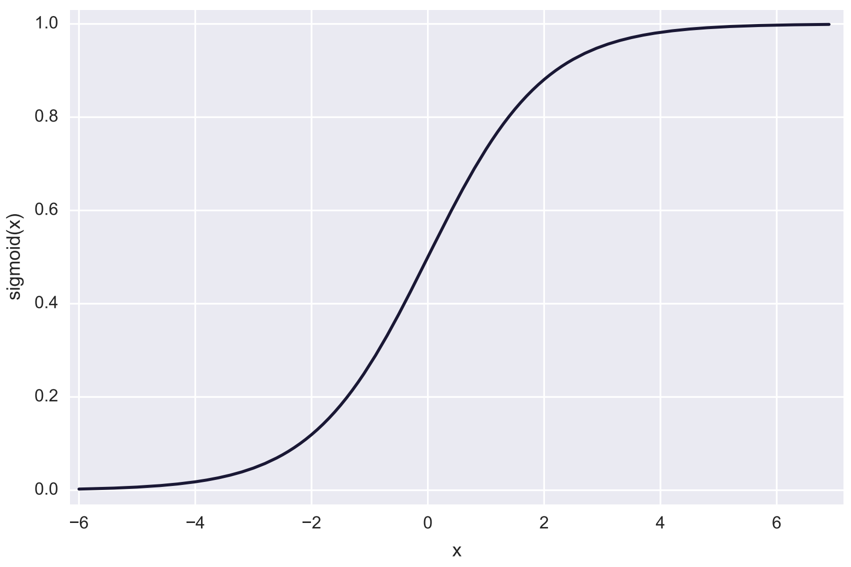

Figure 2. Sigmoid function, $S(t) = \dfrac{1}{1+e^{-t}}$

allows neurons to have looping connections in the network. Unlike other networks, RNN can better learn temporal dependencies on input variables, because these methods are capable of representing and encoding hidden states, in which a network's output depends on an arbitrary number of previous inputs. But these types of networks are known to be difficult to train [5]. Although RNNs have advantage in learning temporal dependencies, in financial time series forecasting, most of the researchers have used feed-forward networks [2, 31].

The number of hidden network layers and number of neurons depends on the complexity of problem. When predicting financial time series, most often researchers have used networks that have one or two hidden layers [2]. This is due to that there is no theoretical proof that suggest more hidden layer produces better forecasting [24].

Several researchers have proposed rules to choose number of hidden layers and number of artificial neurons . Azoff [3] in his work suggests that one hidden layer and $2N + 1$ hidden neurons is sufficient for N inputs. Gatley [17] purposes setting the number of hidden neurons to be equal to the total number of inputs and outputs. Martinez et al. [31] have set the number of neurons in the hidden layer equal to the square root of the product between the number of neurons in the input and output layers. Other researchers use trial and error; in their experiments they train a great number of neural networks with different configurations, and choose the best performing model. It is worth to note that none of these methods guarantee that the resulting model will be optimal.

Parameters such as learning algorithm and activation functions also need to be chosen for successful neural network prediction. Sigmoid and hyperbolic tangent functions have been widely used as activations for NNs [19]. Principe et al. [38] suggest using hyperbolic tangent over sigmoid functions.

1.2.2 Multilayer Perceptron

Multilayer perceptron (MLP) is a feed-forward neural network model, where artificial neurons from one layer are fully connected to the next layer. The key principle of the model is that MLP neurons use nonlinear activation functions. When using sigmoid activation functions and three or more layers MLP model is capable of approximating arbitrary functions [22], [9]. Due to this characteristic MLP are one of the most widely used artificial neural networks.

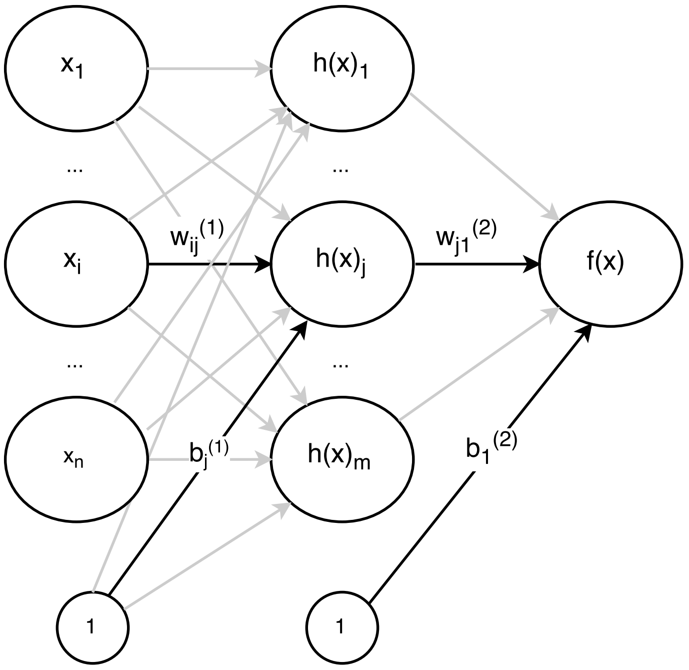

Figure 3. Multilayer Perceptron computation. x_i - ith input vector's attribute, $w_{ij}^{(h)}$ - weight value from input i to artificial neuron j in layer h; $b_j^{(h)}$ - bias value of artificial neuron j in layer h.

Figure 3 describes MLP computation. The model output value $\hat{f}(\mathbf{x})$ is described by input vector $\mathbf{x}$, weight values $w_{ij}^{(h)}$ and bias $b^{(h)}$. In order to make model have a predictive power, weights and biases must be chosen accordingly. The task of finding correct parameters is turn into optimisation problem.

General idea is to calculate model's loss function, which calculates the error (or distance) between the output $\hat{f}(\mathbf{x})$ and the desired value y. For regression problems, one of these loss functions are used:

- sum of squared errors: $E_{total} = \sum_{i=1}^{n}(y_i - \hat{f}(\mathbf{x}_i))^2$
- mean squared error: $E_{total} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\mathbf{x}_i))^2$
- mean absolute error: $E_{total} = \frac{1}{n}\sum_{i=1}^{n}(|y_i - \hat{f}(\mathbf{x}_i)|)$
- root mean squared error: $E_{total} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\mathbf{x}_i))^2}$

The method then modifies its internal adjustable parameters (weights and biases) to reduce this error.

To properly adjust these parameters, the learning algorithm computes gradient vector that indicates by what amount the error would increase or decrease if the weights were changed by a tiny amount. Weights are then adjusted in the opposite direction of the gradient. In practice, minimization of loss function is usually done using the steepest descent algorithm. Steepest descent algorithm is an iterative optimization method, which solves general optimization problem. For arbitrary function $f(x_1, x_2, ..., x_n)$, optimization methods search for parameters $x_1, x_2, ..., x_n$, which minimizes function f. The only requirement for this type of optimization methods is that function f must be differentiable. Although steepest descent is known to not find the global minimum, usually it produces quite good results and practically it should not be a problem [27].

Training MLP consists of showing the input layer few examples, computing the output value and the errors, computing the average gradient for those examples, and adjusting the weights and biases accordingly. The process is repeated for many small sets of examples from the training set with some number of iterations, or until the average of loss function stops decreasing.

MLP model's gradient – partial derivatives of loss function with respect to weights and biases are usually calculated using backpropagation rule (BP). The BP rule propagates prediction errors from the last layer to one before. For each artificial neuron partial derivatives $\frac{\partial E_{total}}{\partial w_{ij}^{(h)}}, \frac{\partial E_{total}}{\partial b_i^{(h)}}$ are calculated. This is done using chain rule for derivatives.

For example, output layer's $w_{11}^{(2)}$ partial derivative is:

$$\frac{\partial E_{total}}{\partial w_{11}^{(2)}} = \frac{\partial E_{total}}{\partial \hat{f}(\mathbf{x})} * \frac{\partial \hat{f}(\mathbf{x})}{\partial net^{(2)}} * \frac{\partial net^{(2)}}{\partial w_{11}^{(2)}}, \tag{1.8}$$

where $\hat{f}(\mathbf{x})$ - model's output, $net^{(2)} = b_1^{(2)} + w_{11}^{(2)} h(\mathbf{x})_1 + w_{12}^{(2)} h(\mathbf{x})_2 + ... + w_{1n}^{(2)} h(\mathbf{x})_n$.
Solving this partial derivative for $w_{11}^{(2)}$ using $E_{total} = \sum_{i=1}^{n} \frac{1}{2}\left(y_i - \hat{f}(\mathbf{x}_i)\right)^2$ loss function, would result in:

$$\frac{\partial E_{total}}{\partial \hat{f}(\mathbf{x})} = 2 * \frac{1}{2}\left(y_i - \hat{f}(\mathbf{x}_i)\right)^{2-1} * -1 = -\left(y_i - \hat{f}(\mathbf{x}_i)\right)$$

$$\hat{f}(\mathbf{x}) = \left(\frac{1}{1 + e^{-net^{(2)}}}\right)$$

$$\frac{\partial \hat{f}(\mathbf{x})}{\partial net^{(2)}} = \hat{f}(\mathbf{x}) * (1 - \hat{f}(\mathbf{x}))$$

$$\frac{\partial net^{(2)}}{\partial w_{11}^{(2)}} = h(x)_1$$

Overall solution for this partial derivative is:

$$\frac{\partial E_{total}}{\partial w_{11}^{(2)}} = -\left(y_i - \hat{f}(\mathbf{x}_i)\right) * \hat{f}(\mathbf{x}) * (1 - \hat{f}(\mathbf{x})) * h(x)_1 \tag{1.9}$$

Similarly, other partial derivatives can be found.

Usually loss function is modified with regularisation parameter, in order to avoid overfitting as described in 1 section. In this work, $\frac{1}{1000}(w_{ij}^{(h)})^2$ regularisation is used. This does not allow MLP to mimic training set, as it penalizes network for too small parameter values.

2 Financial time series

2.1 Introduction

Market analysis is the study of financial market features and attributes that influence prices of financial assets. The main goal of market analysis is to understand financial stock trends in order to help in the decision making process. Two main widely used approaches for analysing financial markets are fundamental and technical analysis. Both approaches have the primary goal of understanding the price movements and predict their future directions. However, the major difference between these two strategies is the choice of market attributes, which are considered. Fundamental analysts believe that the stock value is reflected by several political and economical factors which are both internal and external to the company. Several quantitative tools and indicators were developed to assist in studying the fundamentals of a company, such as marketing strategy, product innovation and management policy [26].

Technical analysis assumes that all attributes that influence the market movements are immediately reflected in the price [35]. Based on this strategy, technical analysts avoid the analysis of subjective economic factors. Key idea is to identify patterns in price, volume and trading activity, believing that this information is enough to determine the future value. Technical indicators, which are mathematical formulas applied to price or volume data, are built and used in order to model some aspect of financial market data.

2.2 Data

In this work, technical analysis variables are used with stock market data. Table 1 shows used financial assets. Data columns are explained in table 2. Different technological and bigger companies stocks with known exchange traded funds and indexes are used due to their high volatility and unpredictability. Financial data are gathered from finance.yahoo.com historic price section. Time series sampled once per day, all ending at 2016-03-24.

Listing 1 shows data sample used in experiments:

```
Date,Open,High,Low,Close,Volume,Adj Close
2016-03-08,199.320007,199.919998,198.210007,198.399994,121391000,198.399
2016-03-07,199.339996,201.070007,199.25,200.589996,95869500,200.589996
```

Listing 1. First 3 lines of SPY time series data file.

description	abbrevation	start date	number of rows
Standard & Poor's 500 index	^GSPC	1950-01-03	16664
NASDAQ Composite stocks	^IXIC	1971-02-05	11383
Russell 2000 stock market index	^RUT	1987-09-10	7194
BOE Interest Rate 10 Year T No options	^TNX	1962-01-02	13561
Volatility S&P 500 exchange traded fund	^VIX	1990-01-02	6610
Apple Inc. stocks	AAPL	1980-12-12	8897
Bank of America Corporation stocks	BAC	1986-05-29	7519
Direxion Daily Gold Miners Bear 3X ETF	DUST	2010-12-08	1332
Freeport-McMoRan Inc. stocks	FCX	1995-07-10	5215
MasterCard Inc. stocks	MA	2006-05-25	2475
Microsoft Corporation stocks	MSFT	1986-03-13	7572
CIGNA Corporation stocks	NYSE.CI	1965-12-31	12671
Opera Software ASA stocks	OPERA.OL	2004-03-11	3082
Oracle Corporation stocks	ORCL	1986-03-12	7573
China Petroleum & Chemical Corp	SNP	2000-10-18	3881
Standard & Poor's 500 exchange traded fund	SPY	1993-01-29	5831
SunEdison, Inc. stocks	SUNE	1995-07-13	5212
Tesla Motors, Inc. stocks	TSLA	2010-06-29	1445
Path S&P 500 VIX ST Futures ETN	VXX	2009-01-30	1800

Table 1. Description of time series used in experiments.

column name	type	description
Date	date	Date when stock's information was captured.
Open	numeric	Price, when market was opening.
High	numeric	Highest price of the day.
Low	numeric	Lowest price of the day.
Close	numeric	Price, when market was closing.
Volume	numeric	Total number of stocks available.
Adj Close	numeric	Price adjusted to stock's increase in volume and dividend payment.

Table 2. Data column description.

2.3 Data preprocessing

Data preprocessing may impact forecasting performance. In many cases input data has a large range of values reducing effectiveness of training procedures. This may be overcome by data normalization. In this work, z-score normalization is used. z-score normalization is described by equation:

$$f(x) = \frac{x - \bar{X}}{std(X)}, \tag{2.1}$$

where each data point $x \in X$ is normalized using $std(X)$ - standard deviation of time series X, $\bar{X}$ - mean of X. Mean and standard deviation is calculated only for training set. When normalizing test set, standard deviation and mean values are used from the test set normalization.

2.4 Evaluation

In order to evaluate the performance of the developed stock market predication models classification accuracy is used. True and predicted prices are converted into positive and negative classes: positive if current days value is greater or equal than the previous value, negative otherwise. So models are trained to predict stock's direction, not its exact price.
Financial data is split into training and test datasets, with 80% of data points used for training and 20% for testing. Datasets are not shuffled when splitting the data, so that models are not allowed to query data points that are in the future.

3 Experiments

3.1 Single input experiment

3.1.1 Description

Multilayer perceptron and linear regression models are trained to predict next day's adjusted closing price direction from current day's adjusted closing price.
MLPs are trained using following configurations:

- MLP using 3 hidden sigmoid artificial neurons,
- MLP using 3 hidden hyperbolic tangent neurons.

Number of hidden layer neurons are computed using $2N + 1$ rule described in section 1.2. Each model is trained once per each financial time series. Models are also compared with $f(x) = x$ model (next day's price is equal to current price).

3.1.2 Results

Table 3 shows results of the experiment. Results indicate that all models have same classification accuracy score. MLPs fail to extract information from single input variable. All models learn a mapping that is close to $f(x) = x$, in linear regression case slope coefficient β_1 is close 1.
Figures 4, 5 show the predictions and error differences for MLP with sigmoid activation function on S&P 500 and MLP with hyperbolic tangent activation function for Direxion Daily Gold Miners Bear 3X ETF. It can be seen that for some financial data, such as Direxion Daily Gold Miners Bear 3X ETF, rule $f(x) = x$ provides satisfactory results. In other cases the predicted and actual differences are quite big and volatile, such as in figure 4.

name	Sigm class.	Tanh class.	LR class.	$f(x) = x$ class.
AAPL	48.85	48.85	48.85	48.85
BAC	52.96	52.96	52.96	52.96
DUST	55.26	55.26	55.26	55.26
FCX	46.35	46.35	46.35	46.35
MA	54.05	54.05	54.05	54.05
MSFT	50.86	50.86	50.86	50.86
NYSE.CI	52.33	52.33	52.33	52.33
OPERA.OL	50.16	50.16	50.16	50.16
ORCL	51.92	51.92	51.92	51.92
SNP	53.22	53.22	53.22	53.22
SPY	50.26	50.26	50.26	50.26
SUNE	48.08	48.08	48.08	48.08
TSLA	52.78	52.78	52.78	52.78
VXX	49.03	49.03	49.03	49.03
^GSPC	52.79	52.79	52.79	52.79
^IXIC	49.78	49.78	49.78	49.78
^RUT	51.74	51.74	51.74	51.74
^TNX	50.96	50.96	50.96	50.96
^VIX	51.10	51.10	51.10	51.10

Table 3. Multilayer perceptron with sigmoid activation function and 3 hidden units, multilayer perceptron with hyperbolic tangent activation function and 3 hidden units, logistic regression and $f(x) = x$ are compared when trying to predict stock's direction. $t - 1$ day's price values are used to predict future (t) price. Classification accuracy is presented for the testing dataset.

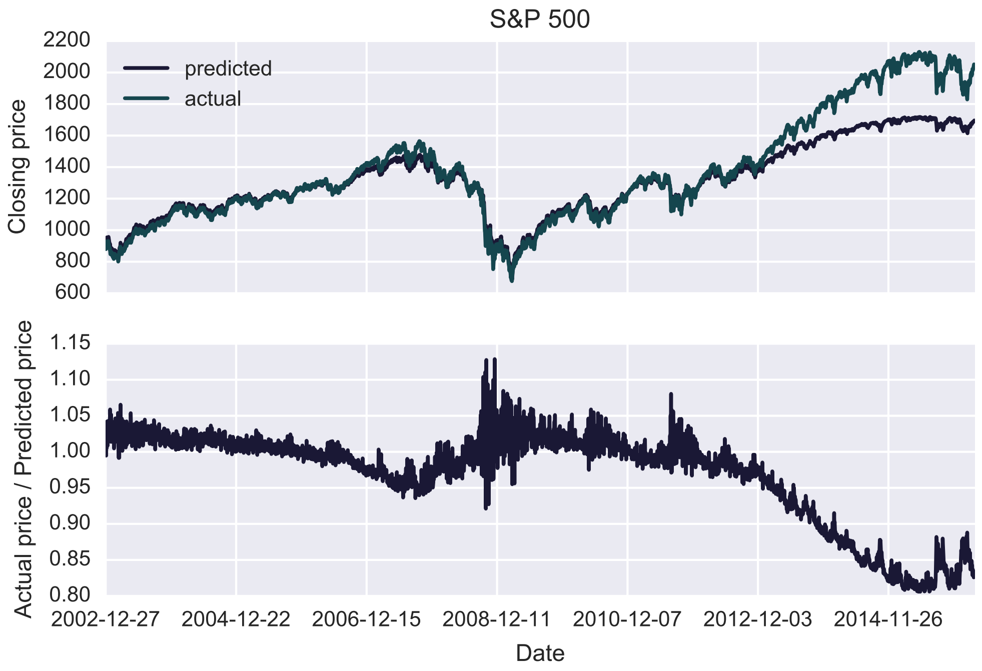

Figure 4. Above: Multilayer perceptron with sigmoid activation function prediction values and actual values for Direxion Daily Gold Miners Bear 3X ETF on testing set; Below: Relative difference between predicted and actual values for the figure above.

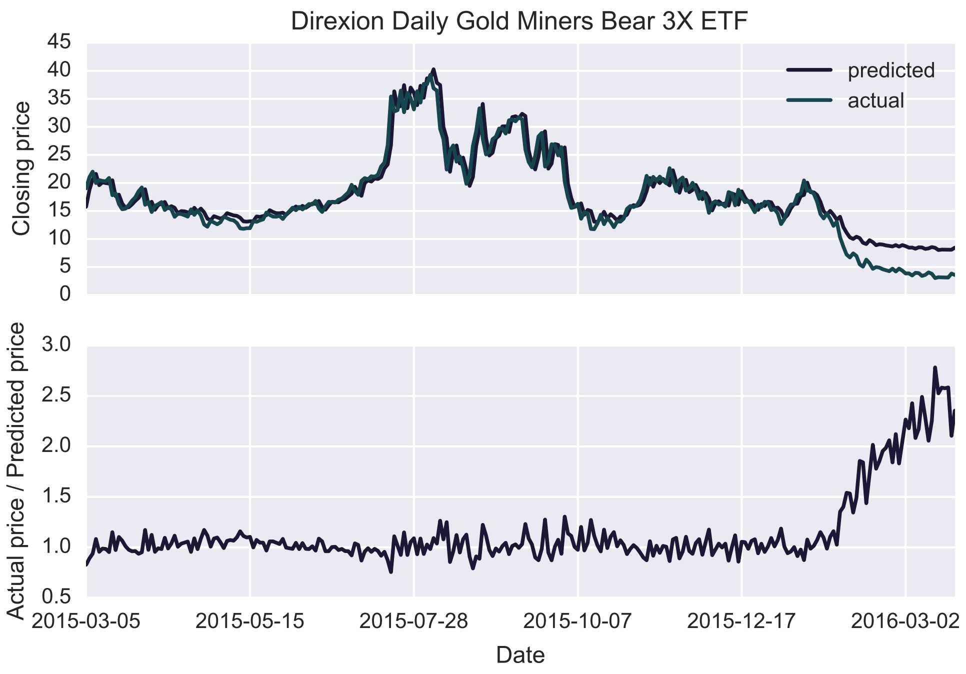

Figure 5. Above: Multilayer perceptron with hyperbolic tangent activation function prediction values and actual values for S&P 500 on testing set; Below: Relative difference between predicted and actual values for the figure above.

3.2 Multiple input experiment

3.2.1 Description

Multilayer Perceptron and Linear regression models are trained to predict next day adjusted closing price direction from $t-1, t-2, t-3$ day's adjusted closing prices.
MLPs are trained using following configurations:

- MLP using 7 hidden sigmoid artificial neurons,
- MLP using 7 hidden hyperbolic tangent neurons.

Number of hidden layer neurons are computed using $2N+1$ rule described in section 1.2. Each model is trained once per each financial time series. Models are also compared with $f(x) = x$ model (next day's price is equal to current price).

3.2.2 Results

Table 4 shows results of the experiment. When predicting direction MLP with sigmoid activation results has produced better results than linear regression for FCX, AAPL, OPERA.OL, SUNE, ^VIX, ^IXIC series; model with hyperbolic tangent produced better results for FCX, SUNE, TSLA, ^IXIC, AAPL. Maximum classification accuracy difference between MLP models & linear regression is 2.88% for FCX. Figure 6 shows the stock for which maximum positive difference was achieved. This shows how hard is to predict direction and only a little improvement over random prediction can be made.
For other stocks linear model has performed better than used neural network models, with maximum difference being 4.88%. Figure 7 shows the stock for which maximum negative difference was achieved.
Overall, results suggest that no single configuration can produce best results. In order to achive better prediction results, specific to financial stock neural network configuration should be chosen. Also, this experiment shows better results than experiment in section 3.1. This suggest using more input variables with MLP models.

name	Sigm class.	Tanh class.	LR class.	$f(x) = x$ class.
AAPL	51.04	48.79	48.73	48.85
BAC	49.57	51.50	53.10	52.96
DUST	50.38	54.92	56.44	55.26
FCX	49.23	48.65	46.35	46.35
MA	53.25	52.24	53.86	54.05
MSFT	49.60	50.07	50.53	50.86
NYSE.CI	52.05	51.54	52.53	52.33
OPERA.OL	51.30	49.84	50.81	50.16
ORCL	50.07	51.72	52.45	51.92
SNP	49.22	48.97	54.13	53.22
SPY	49.23	49.57	50.17	50.26
SUNE	48.94	50.10	48.46	48.08
TSLA	51.05	53.50	52.10	52.78
VXX	46.50	45.94	48.18	49.03
^GSPC	51.80	51.89	53.06	52.79
^IXIC	50.26	49.96	49.96	49.78
^RUT	49.86	50.07	52.23	51.74
^TNX	50.55	50.96	51.11	50.96
^VIX	51.78	50.95	50.80	51.10

Table 4. Multilayer perceptron with sigmoid activation function and 7 hidden units, multilayer perceptron with hyperbolic tangent activation function and 7 hidden units, logistic regression and $f(x) = x$ are compared when trying to predict stock's direction. $t-1, t-2, t-3$ day's price values are used to predict future (t) price. Classification accuracy is presented for the testing dataset.

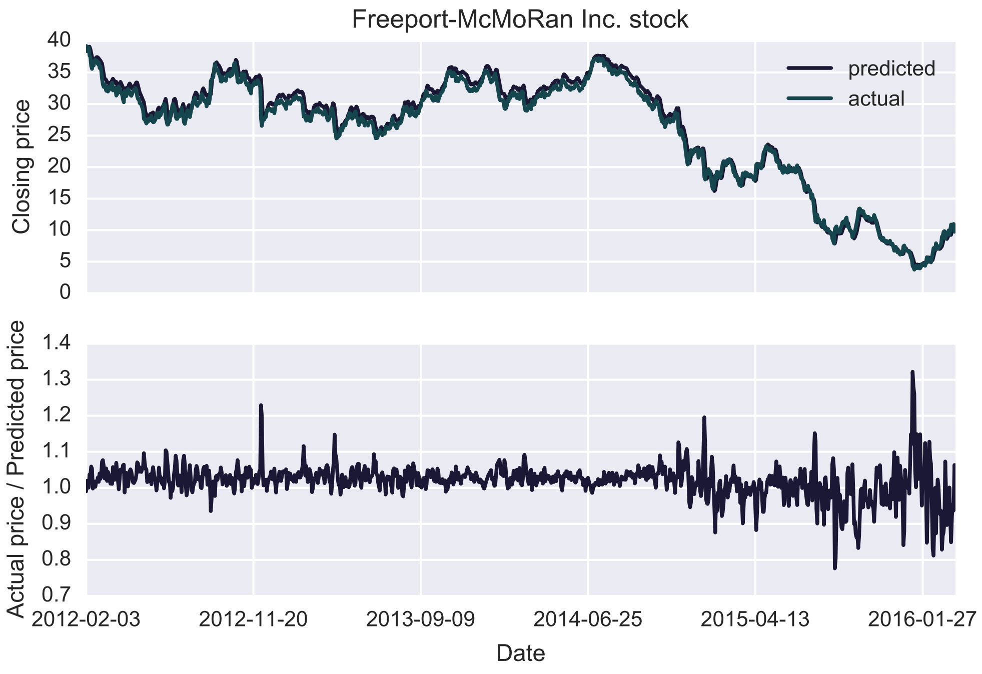

Figure 6. Above: Multilayer perceptron with sigmoid activation function prediction values and actual values for Freeport-McMoRan Inc. stock on testing set; Below: Difference between predicted and actual values for the figure above.

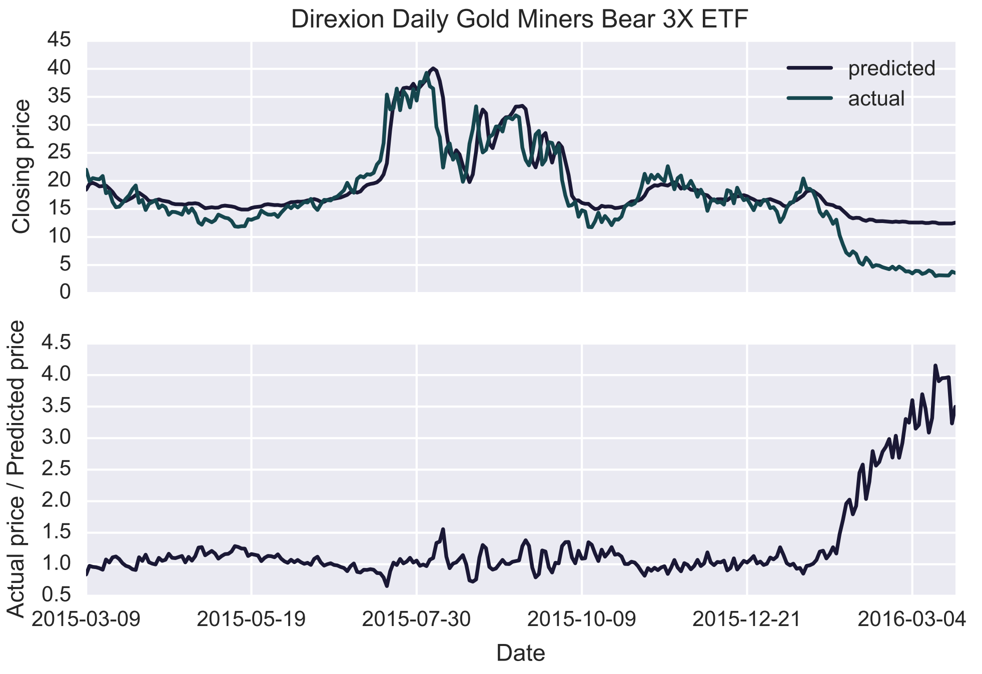

Figure 7. Above: Multilayer perceptron with sigmoid activation function prediction values and actual values for Direxion Daily Gold Miners Bear 3X ETF on testing set; Below: Difference between predicted and actual values for the figure above.

3.3 Number of input experiment

3.3.1 Description

This experiment, checks whether more input variables should be provided to MLP and linear regression models, so that better classification accuracy is achieved. All models are trained 10 times with inputs ranging from 1 to 10 day's adjusted closing prices.
MLPs are trained using following configurations:

- MLP using $2N + 1$ hidden sigmoid artificial neurons,
- MLP using $2N + 1$ hidden hyperbolic tangent neurons,

where N is the number of inputs. Each model is trained once per each financial time series. Also, models are compared with linear regression model.

3.3.2 Results

Table 5 shows results of the experiment. The results indicate that for each stock and model different number of inputs should be chosen.
Predicting direction with MLP model using sigmoid activation function produced better results than linear regression for FCX, ^VIX, AAPL, OPERA.OL, DUST, TSLA, ^TNX, SNP series; model with hyperbolic tangent produced better results for VXX, OPERA.OL, SUNE, SPY, MSFT, BAC, ^IXIC. Maximum classification accuracy difference between MLP models and linear regression is 6.55% for VXX stock, with classification accuracy being 55.81%. For other 3 stocks linear model has performed better than used neural network models, with maximum difference being 0.42%.
Also, table 5 shows the difference between the worst case scenario for that model against the best. This shows, what percentage of improvement potentially could be lost if number of input variables are not chosen according to stock/model. Average improvement is 4.28% for MLP with sigmoid activation function, 4.36% for MLP with hyperbolic tangent activation function and 1.08% for linear regression.
Overall, results suggest that no single number of input variables can produce best results & that specific neural network configurations should be chosen so that good prediction results are achieved.

name	Sigm class.	improv-ement %	number of inputs	Tanh class.	improv-ement	number of inputs	LR class.	improv-ement %	number of inputs
AAPL	51.58	3.83	10	50.73	2.59	10	49.41	0.76	5
BAC	53.57	4.27	5	53.64	4.51	5	53.10	0.33	3
DUST	57.59	10.69	10	56.06	12.00	3	56.44	2.18	3
FCX	52.37	6.13	8	49.66	4.11	4	47.16	0.81	6
MA	54.05	1.02	1	54.05	1.34	1	54.05	1.68	1
MSFT	51.49	2.19	8	52.12	3.98	7	50.86	0.56	1
NYSE	52.47	4.07	2	52.33	2.66	1	52.89	0.56	7
OPERA	52.44	4.50	3	52.44	5.79	3	50.81	2.46	3
ORCL	51.92	2.88	1	53.55	5.30	6	52.45	0.95	3
SNP	54.97	5.95	2	54.78	5.43	3	54.58	1.40	2
SPY	51.47	2.81	7	51.55	4.28	9	50.65	0.60	8
SUNE	50.29	3.57	7	51.74	4.63	9	49.18	1.24	10
TSLA	55.05	5.94	2	54.55	4.90	3	54.45	2.52	8
VXX	53.56	7.36	9	55.81	8.04	7	49.16	1.30	2
^GSPC	52.79	3.05	1	52.84	3.08	2	53.25	0.48	9
^IXIC	50.44	1.78	5	50.99	2.22	8	50.04	0.53	9
^RUT	51.95	4.12	5	51.81	2.34	5	52.23	1.04	3
^TNX	52.01	2.55	2	51.52	2.14	9	51.42	0.52	6
^VIX	53.95	4.74	4	52.92	3.57	5	51.21	0.72	2

Table 5. Multilayer perceptron with sigmoid activation function, multilayer perceptron with hyperbolic tangent activation function and linear regression are compared when trying to predict stock's direction. Best classification score is shown with it's number of inputs. Classification score is shown for the testing dataset. Also percentage improvement that potentially could be lost if number of input variables are not chosen according to stock/model is shown.

4 Nonlinear analysis

4.1 Phase space reconstruction

4.1.1 Introduction

Most nonlinear dynamics methods from chaos theory use multi-dimensional phase space, rather than time or frequency domain [21]. Usually phase space is reconstructed using method of delays [37]. For time series $x_1, x_2, ..., x_N$, method of delays creates matrix $\mathbf{X}$:

$$\mathbf{X} = [\mathbf{X_1}, \mathbf{X_2}, ...\mathbf{X_M}]^T, \tag{4.1}$$

where $\mathbf{X_i}$ is a row vector which represent system's state at time i and is defined by:

$$\mathbf{X_i} = [x_i, x_{i+L}, ...x_{i+(m-1)L}], \tag{4.2}$$

where L is **reconstruction delay** (lag), m is **embedding dimension**, and $M = N - (m-1)L$. Nonlinear algorithms are then calculated on the m-dimensional state vectors from the matrix $\mathbf{X}$. For example, figure 8 shows time series generated by logistic map as a function of time and it's reconstructed phase space.

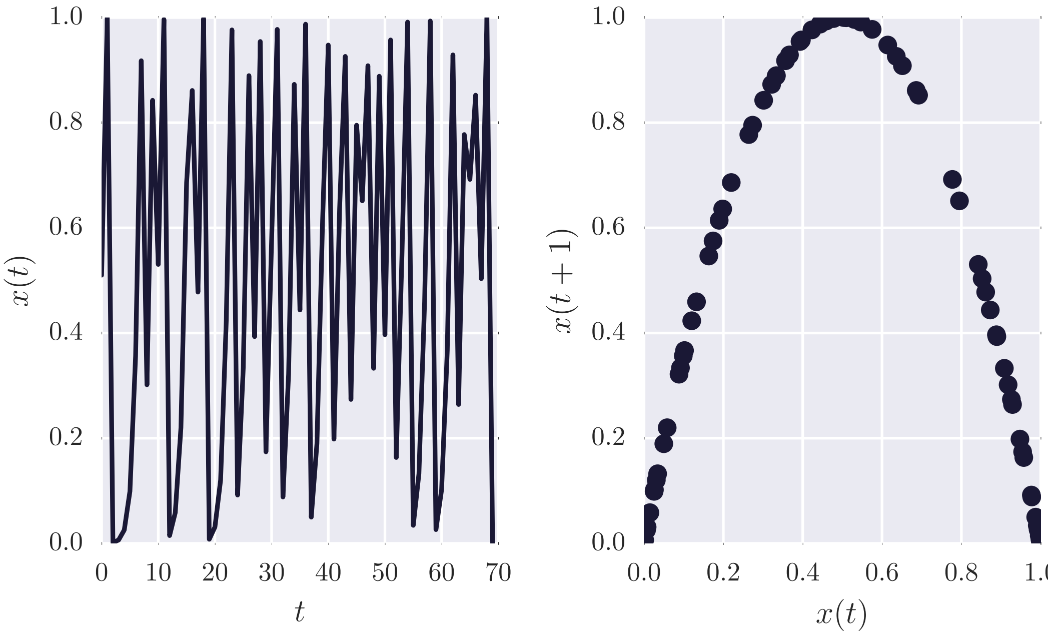

Figure 8. On the left: time series generated by logistic map ($N = 75$, $r = 4$, $x[0] = 0.51$). On the right: method of delays phase space reconstruction ($m = 1, L = 1$) on the data.

Theoretical foundation of method of delays is the **Takens-Mañé embedding theorem** [44, 29]. Takens-Mañé theorem assumes that time series $x_1, x_2, ..., x_N$ are generated by a dynamical system with ω variables. Dynamical system is acting on the state space $P \subseteq \mathbb{R}^\omega$ and it's behaviour is defined by a continuous map $\Psi : P \to P$, which for given state y finds the next state $\Psi(y)$. The idea of this theorem is that it is possible to infer properties of Ψ from delay vectors written in 4.2 equation. Even though dynamical system has ω variables, it is possible to reconstruct phase space using only one quantity, as showed by Packard et al. [37].

General difficulty in method of delays is determining m and L. Incorrect values of these parameters may lead to false results [8]. The choice may depend on data, e.g. choosing embedding parameters when studying low dimensional chaotic time series would not be optimal for a highly chaotic signal. Often it is not even possible to compute nonlinear measures from reconstructed phase space matrix $\mathbf{X}$. Delay matrix may be too small or won't approximate Ψ correctly.
There are many methods to estimate embedding dimension m [11, 16, 6, 25]. Cellucci et al. [8] have shown experimentally that reconstructing phase space using global false nearest neighbour (GFNN) method gave better results than Gao-Zheng, Schuster, characteristic length embedding methods for noisy time series.
To estimate reconstruction delay L generally only two methods are used. Most widely used method is finding time step t when value of autocorrelation function drops below certain threshold, such as 0, $\frac{2}{\sqrt{N}}$, $1 - \frac{1}{e}$ [8, 42, 34]. Some authors doubt this method, because it may produce wrong results when relationships in the data are not linear [1].
Mutual information function is a nonlinear analog to autocorrelation function. To find appropriate delay value one has to locate first minima of average mutual information function [15]. When estimating delay on noisy data this method needs filtering, because delay may be due to small fluctuations in the data [30].
Celluci et al. [8] showed that estimating reconstruction lag using mutual information produced better results. Although findings do not generally conclude that other methods are worse, but the global false nearest neighbour method with mutual information function can be used as a guideline.

4.1.2 Global false nearest neighbour method

Global false nearest neighbour method [25] is used to find minimum embedding dimension. Algorithm idea is to identify false nearest neighbours, points which are neighbours due to embedding dimension m being too small. Starting from d dimension, square Euclidean distance between state space point $x(n)$ and it's rth nearest neighbour $x^{(r)}(n)$ is defined by:

$$R_d^2(n,r) = \sum_{k=0}^{d-1}(x(n+kL) - x^{(r)}(n+kL))^2, \tag{4.3}$$

where L is reconstruction delay, n - point's index, r - closest neighbour's index. The distance after increasing embedding dimension by 1 is:

$$R_{d+1}^2(n,r) = R_d^2(n,r) + (x(n+dL) - x^{(r)}(n+dL))^2. \tag{4.4}$$

Simple metric to quantify how distance between the nearest neighbours increased after addition of a dimension would be:

$$\left(\frac{R_{d+1}^2(n,r) - R_d^2(n,r)}{R_d^2(n,r)}\right)^{1/2} = \left(\frac{|x(n+dL) - x^{(r)}(n+dL)|}{R_d^2(n,r)}\right)^{1/2} > R_{tol}, \tag{4.5}$$

where R_{tol} is a threshold. When the distance grows more than R_{tol}, the neighbour is marked as false. Numerically $R_{tol} = 15$ has been found to be a good choice [1]. It is enough to look for only one nearest neighbour point ($r = 1$) and go through all of the points in the state space $n = 1, 2, ..., M$ to find good embedding dimension [25]. When estimating embedded dimension on noisy signals, additional criterion to quantify false neighbours is needed. Kennel et al. purposed using:

$$\frac{R_{d+1}(n)}{R_A} > A_{tol}, \tag{4.6}$$

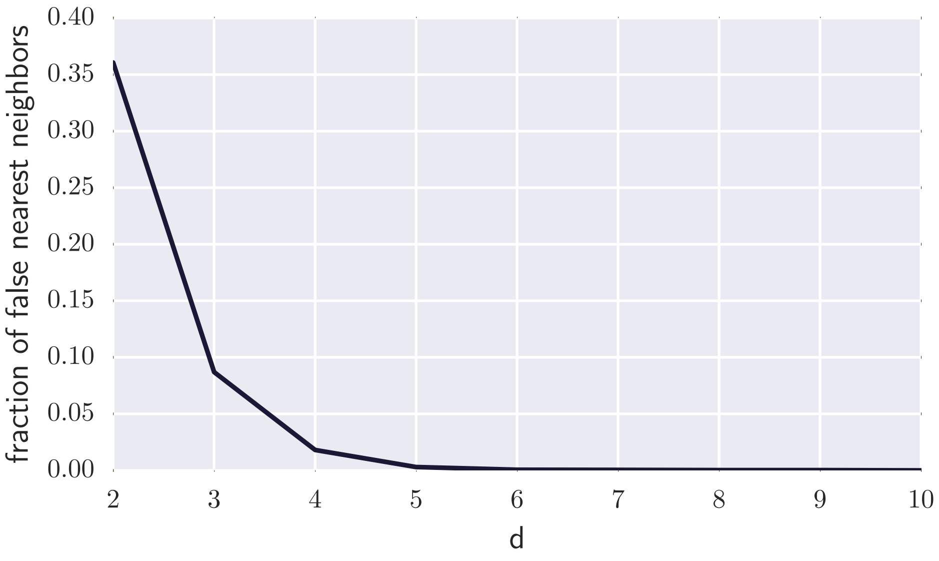

Figure 9. Global false nearest neighbour algorithm applied on S&P 500 stock market index. $d = 5$ is a good approximation of embedded dimension.

where A_{tol} is threshold, which in their work is used as a constant equal to 2, R_A is defined as:

$$R_A^2 = \frac{1}{M}\sum_{n=1}^{M}(x(n) - \bar{x}), \tag{4.7}$$

where $\bar{x} = \frac{1}{M}\sum_{n=1}^{M} x(n)$. Authors suggest to use these criteria together, if nearest neighbour satisfies both 4.5 and 4.6 criteria, it is flagged as false. Finally, computing the embedding dimension is just finding first dimension d for which number of false nearest neighbours is lowest (computing first minima).
For example, figure 9 shows fraction of false nearest neighbours as function of embeddding dimension for financial data (S&P 500 stock market index). It is seen that $d = 5$ would be a good approximation of embedding dimension.

4.1.3 Autocorrelation

Autocorrelation computes correlation coefficient between $x(t)$ and $x(t + d)$, that is correlation between signal and signal after d time steps. This can be seen as a measure of how "present" remembers "past" and how fast memory disappears increasing depth of memory d [34].
Autocorrelation for time series $x(t)$ is defined as:

$$r(d) = \frac{\sum_{t=0}^{N-d}(x(t) - \bar{x})(x(d+t) - \bar{X})}{\sqrt{\sum_{t=0}^{N-d}(x(t) - \bar{x})^2 \sum_{t=0}^{N-d}(x(d+t) - \bar{X})^2}}, d = 0, 1, ..., [N/2], \tag{4.8}$$

where N is time series length, d is the lag, $\bar{x} = \dfrac{1}{N-d+1}\sum_{t=0}^{N-d} x(t)$, $\bar{X} = \dfrac{1}{N-d+1}\sum_{t=0}^{N-d} x(t+d)$.

4.1.4 Mutual information

Given two discrete sets of messages $S = \{s_1, s_2, ..., s_n\}, Q = \{q_1, q_2, ..., q_n\}$ and their probabilities $\{P_S(s_1), P_S(s_2), ..., P_S(s_n)\}, \{P_Q(q_1), P_Q(q_2), ..., P_Q(q_n)\}$ mutual information of S and Q systems is number of bits on average of information is learned about q_j from s_i:

$$I(Q,S) = \sum_i \sum_j P_{SQ}(s_i, q_j) \log_2 \left(\frac{P_{SQ}(s_i, q_j)}{P_S(s_i) P_Q(q_j)} \right), \tag{4.9}$$

where P_{SQ} is joint P_Q and P_S probability distribution.
In order to estimate reconstruction delay for time series $x(t)$ using mutual information S needs to be set as a time series measurement at time t: $S = x(t)$; Q is set as a measurement after ΔT time $Q = x(t + \Delta T)$. Then ΔT needs to be chosen so that the value $x(t + \Delta T)$ is unpredictable from $x(t)$. Reconstruction lag is ΔT, which minimizes $I(S, Q)$. Figure 10 shows an example of computation of mutual information on financial market data.
Using this method with experimental data one needs to estimate P_{QS}. This can be done with Fraser-Swinney algorithm [15].

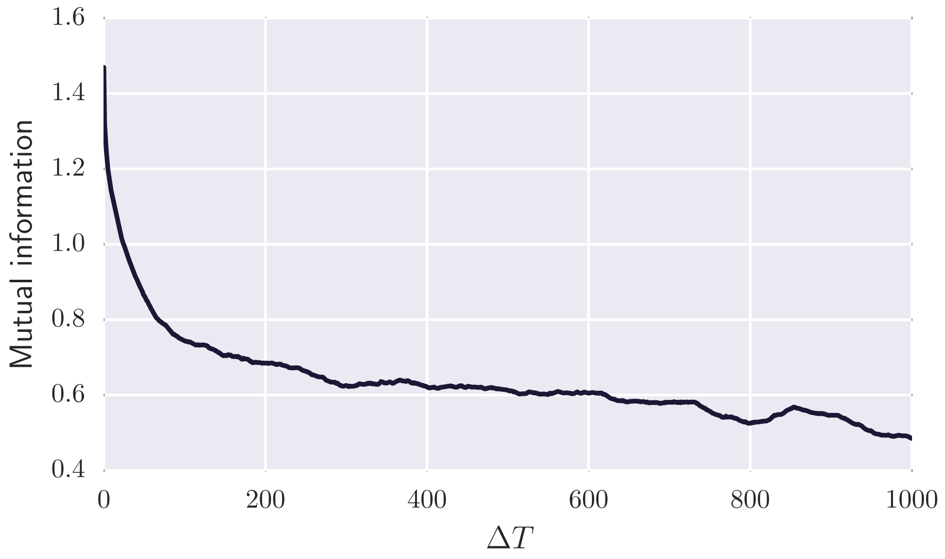

Figure 10. Mutual information measure applied on S&P 500 stock market index. $L = 300$ is a good approximation of reconstruction delay.

4.2 Correlation dimension

4.2.1 Introduction

Correlation dimension is one of the generalized fractal dimensions, which shows how many active modes there are in the deterministic dynamical system. Grassberger and Proccacia [18] proposed an algorithm to quantify correlation dimension using correlation integral. The general idea behind the algorithm is that if random points are put on a d-dimensional object, then number of points in the hypersphere of radius r should behave like r^d. So correlation integral $C(r)$ counts on average how many phase space vectors are close by (not further than distance r):

$$C(r) = \frac{1}{M(M+1)} \sum_{i,j=0,i\neq j}^{M} H(r - \|\mathbf{X_i} - \mathbf{X_j}\|), \tag{4.10}$$

where $H(n) = \begin{cases} 0 & , n < 0 \\ 1 & , n \geq 0 \end{cases}$ is heaviside step function, $\mathbf{X_i}$, $\mathbf{X_j}$ reconstructed phase space vectors, $||.||$ - euclidean distance metric. In the limit $M \to \infty$. On sufficiently small scales correlation dimension D_2 relates to correlation integral by:

$$C(r) \propto r^{D_2}. \tag{4.11}$$

Thus correlation dimension is equal to:

$$D_2 = \lim_{r\to 0} \frac{\log C(r)}{\log r}. \tag{4.12}$$

In order to compute D_2 on finite length time series one starts by reconstructing the phase space. Phase space is reconstructed with embedding dimension as some starting constant, e.g. 1. Reconstruction delay approximated by first minima of mutual information function or autocorrelation drop. There is no need to use embedding dimension algorithms such as false nearest neighbours, because this algorithm can estimate embedding dimension by itself.
Then correlation integral is computed using 4.10. From $\log C(r)$ vs $\log r$ plot one locates linearly scaling region and estimates region's slope with small enough r values. The slope with embedding dimension m is denoted as $\bar{D_2}^{(m)}$. An example of $\log C(r)$ vs $\log r$ plot is shown in figure 11. If $\bar{D_2}^{(m)}$ plotted as a function of m appear to reach a plateau for some large enough m values then D_2 is approximated by $\bar{D_2}^{(m)}$.

4.2.2 Correlation dimension estimation on experimental data

There are problems when using this algorithm with experimental data. Experimental data is usually noisy, has finite resolution and length. To solve these issues there are proposals of minimal length of time series, such as Tsonis criterion, which determines minimal time series length to be $N > 10^{2+0.4D_2}$. Other researchers provide some rules for correct dimension estimation. For example Martinerie et al. [30] purposed following rules:

- The plateu must be flat. The degree of varation of the derivative in the scaling region must not exceed specified standard.
- The scaling region must meet minimum-length requirement.

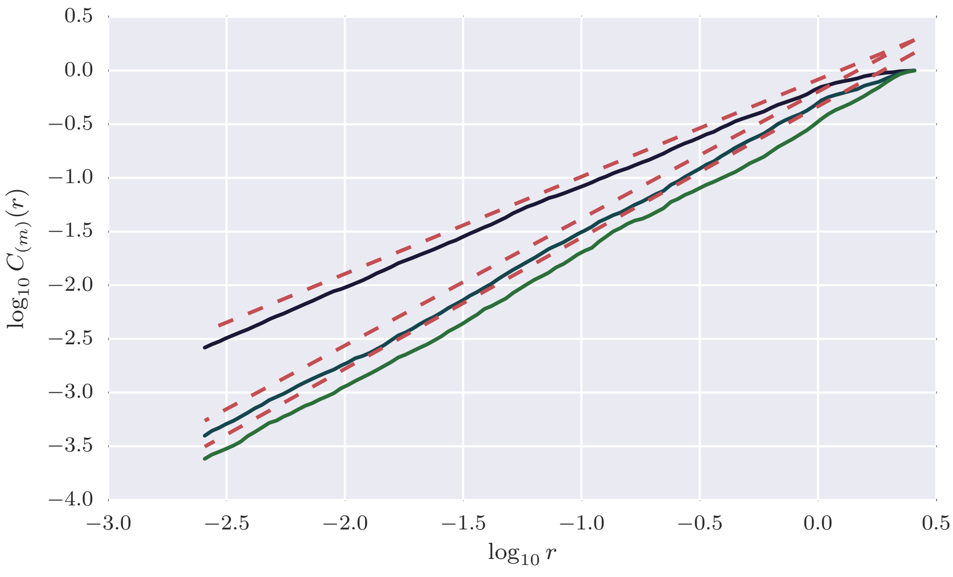

Figure 11. $\log C(r)$ vs $\log r$ plot of embedding dimensions $m = 1, 2, 3$ for time series generated from Henon map ($a = 1.4, b = 0.3, N = 1500$). Dashed lines indicate slope of scaling region which approximates correlation dimension.

- The estimated value of D_2 must be stable as embedding dimension increases.

Theiler [46] has suggested a minor change for Grassberger-Proccacia algorithm that estimates correlation dimension better for over-sampled time series or data with highly correlated noise. Autocorrelated noise produces "shoulder" like structure in $\log C(r)$ vs $\log r$ plot, which leads to bad estimation of dimension. Theiler proposed a change in correlation integral:

$$C(r) = \frac{1}{M(M+1)} \sum_{n=W}^{M} \sum_{i=0}^{M-n} H(r - ||\mathbf{X}_{i+n} - \mathbf{X}_i||), \tag{4.13}$$

where $X_i, M, H(x)$ are same as in equation 4.10 and W is **Theiler window**. When $W = 1$ the correlation integral 4.13 is equivalent to equation 4.10. Theiler in his work recommends minimum W to be chosen so that $W > \tau(2/N)^{2/m}$, where τ is autocorrelation time of time series, m is embedding dimension. Also W should be $w \ll N$. The change ensures that correlation integral counts only close state space vectors by accident, so that dimension estimate is not biased by pairs of vectors which are close in space because they are close in time.
Provenzale et al. [39] has suggested to use **space-time separation plot** to estimate correct Theiler window W. Space-time separation plots lines of constant probability $P(|x(t + \Delta t) - x(t)| < r)$, that is fraction of points closer than a distance r at a given time separation Δt. An example of space-time separation plot is shown in figure 12. Theiler window can be retrieved by value Δt corresponding to the first maximum joint of all the lines.

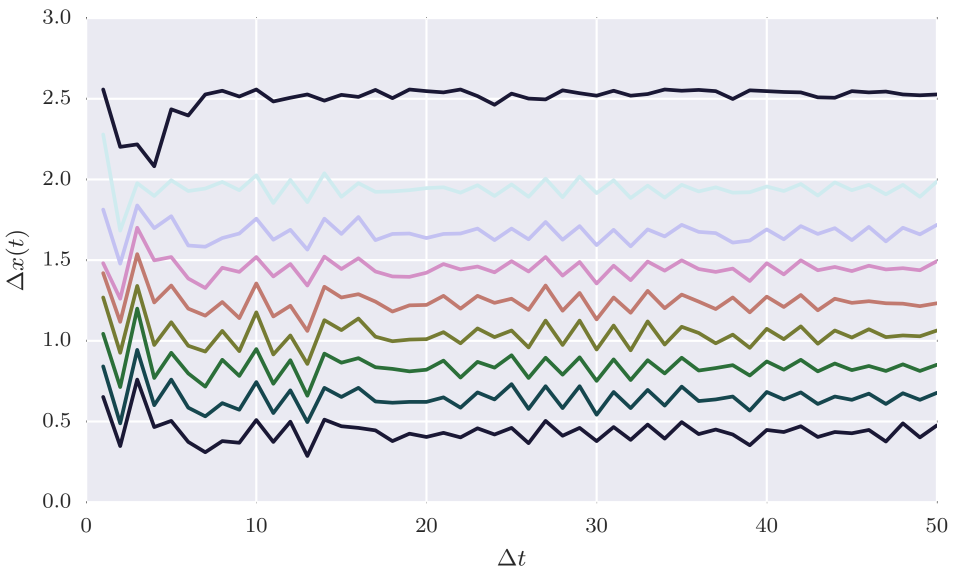

Figure 12. Space-time separation plot for time series generated from Henon map ($a = 1.4, b = 0.3, N = 1500$). From bottom to the top probability increases by 0.1 (starting with line of probability 0.1 up to 1.0).

4.3 Lyapunov exponent

4.3.1 Introduction

Lyapunov exponent quantifies systems exponential divergence rate of nearby trajectories in the phase space. In other words, this measure shows how system amplifies small errors. Systems which amplify errors exhibit property which is called **sensitive dependence on initial conditions**. For example if two nearby orbits after some number of iterations diverge, then the system is sensitive and largest Lyapunov exponent (LLE) is equal to the average exponential rate of divergence [12]:

$$|\Delta\mathbf{x}(t)| \approx |\Delta\mathbf{x}(0)|e^{\lambda t} \tag{4.14}$$

where $\Delta\mathbf{x}(0)$ is the difference between initial condition $\mathbf{x}(0)$ and initial condition perturbed with small error ϵ; $\Delta\mathbf{x}(t)$ is the difference after t iterations; λ is largest Lyapunov exponent. If λ value is positive then system has sensitive dependence on initial conditions, so infinitesimal errors tend to be magnified, if it is negative, then system shows stability, therefore, it tends to reduce small errors. When λ value is zero the system is between sensitivity and stability.
Dynamical systems can have move then one Lyapunov exponent and this concept is generalized in **Lyapunov exponent spectrum**. The number of elements in spectrum equals to the size of system's phase space. In the spectrum largest Lyapunov exponent is used quantify system's predictability, since exponential divergence (4.14) means that system where initial difference was infinitesimally small start to behave completely differently - predictability is rapidly lost [49]. The LLE magnitude shows how system process creates or destroys information and is expressed in bits/iteration or information/s. For example, if system has LLE value of 4.00 bit/iteration and initial conditions were specified by 20 bits of information, after about $\dfrac{20}{4.00} = 5$ iterations systems value could not be predicted. After this time, small initial errors will dominate the systems values.
It is possible to compute LLE for systems which have equations of motion [4], but there are problems when trying to estimate it from the time series data. Experimental data has errors, is finite length, usually contains external noise and is bound to experimental resolution. Several methods which estimate LLE from experimental time series have been created [49, 43, 42, 23]. Rosenstein and Kantz method have shown promising results when estimating LLE from noisy data. Since the creation of these algorithms, they have been widely used [50, 28, 41, 7]. Therefore, in this work these methods are used.

4.3.2 Rosenstein et al. algorithm

Firstly Rosenstein et al. algorithm [42] uses phase space reconstruction to approximate the attractor dynamics. In their work they use method of delays as explained in 4.1.1 section. Algorithm then precedes searching in the state space for nearest neighbour $\mathbf{X}_{\hat{j}}$ for particular reference point $\mathbf{X}_{\mathrm{j}}$:

$$d_j(0) = \min_{|j-\hat{j}|>\text{mean period}} \|\mathbf{X}_j - \mathbf{X}_{\hat{j}}\|, \tag{4.15}$$

where $\|\cdot\|$ denotes the Euclidean norm, mean period is estimated as the reciprocal of the mean frequency of the power spectrum. Given that distance between j-th pair of the nearest neighbours diverge approximately at a rate given by the largest Lyapunov exponent:

$$\ln d_j(i) \approx \lambda(i \cdot \Delta t) + \ln C_j, \tag{4.16}$$

where $d_j(i)$ is distance defined in 4.15 equation after i time steps, C_j is initial separation of neighbours. Largest Lyapunov exponents value can be found by using the least squares fit to linear region:

$$y(i) = \frac{1}{\Delta t}\frac{1}{M}\sum_{j=1}^{M} \ln d_j(i). \tag{4.17}$$

Slope of the linearly increasing region is LLE.
For example, calculating LLE on logistic map time series produces 13 figure. Logistic map's Lyapunov characteristic exponent value is known to be $\lambda = 0.693$ [12].

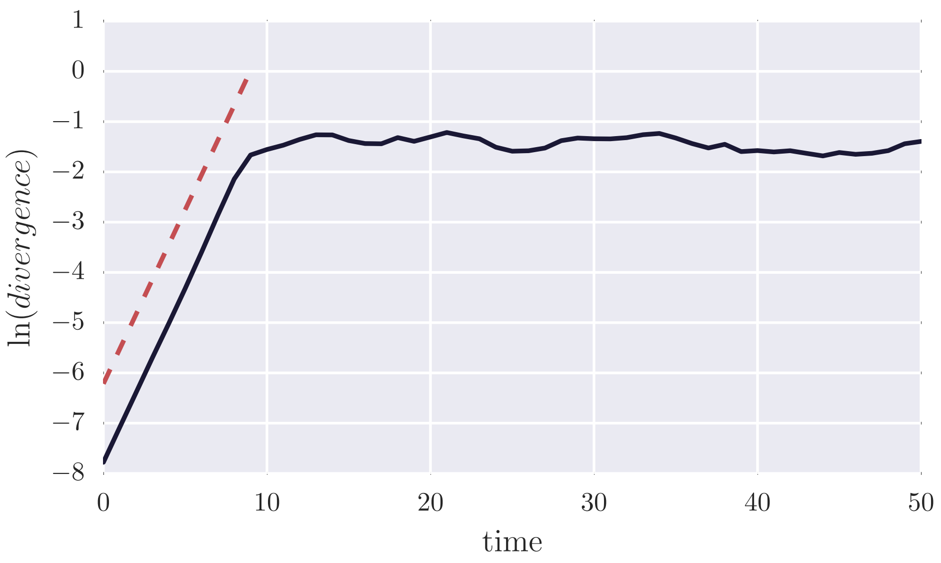

Figure 13. Rosenstein et al. algorithm used on logistic map time series ($N = 500$, $r = 4$, $x[0] = 0.51$). Phase space is reconstructed using $m = 1$ and $L = 1$. Slope of dashed line is $\lambda \approx 0.7076$.

4.3.3 Kantz algorithm

The general idea behind this algorithm, is that LLE can be found using:

$$\lambda = \lim_{t\to\infty}\lim_{\epsilon\to\infty}\frac{1}{t}\ln\left(\frac{|\mathbf{x(t)} - \mathbf{x}_\epsilon\mathbf{(t)}|}{\epsilon}\right), \\ |x(0) - x_\epsilon(0)| = \epsilon, \tag{4.18}$$

for almost all $\mathbf{x(0)} - \mathbf{x}_\epsilon\mathbf{(0)}$, where $\mathbf{x(t)}$ is the evolution of some initial condition $x(0)$ in the state space. To find LLE on experimental data, using 4.18 equation, phase space needs to be reconstructed as explained in 4.1.1 section. Kantz uses slightly different definition of state space vector:

$$\mathbf{X_i} = [x_{i-(m-1)}, x_{i-(m-2)}, ..., x_i]. \tag{4.19}$$

Algorithm searches for particular reference vector $\mathbf{X_t}$ all neighbourhood trajectories in $\mathcal{U}_t$. Neighbour vectors $\mathcal{U}_t$, are all state space points that fall into ϵ neighbourhood. After that method calculates distance between $\mathbf{X_j}$ trajectory and a neighbouring $\mathbf{X_i}$ trajectory after relative time τ:

$$dist(\mathbf{X_j}, \mathbf{X_i}; \tau) = |x_{j+\tau} - x_{i+\tau}|, \tag{4.20}$$

that is the absolute difference between τth scalar component of two trajectories.
For some intermediate range of τ, LLE can be found using:

$$\mathcal{S}(\tau) = \frac{1}{N}\sum_{\tau=1}^{N} \ln\left(\frac{1}{|\mathcal{U}_t|}\sum_{i\in\mathcal{U}_t} dist(\mathbf{X_j}, \mathbf{X_i}; \tau)\right). \tag{4.21}$$

LLE is a least squares line to linearly increasing $\mathcal{S}(\tau)$ region.
The benefit of this algorithm, that it can be used without explicit knowledge of embedded dimension m, i.e. you can start searching for neighbours in 2 dimensions, accumulate the distances $dist(\mathbf{X_j}, \mathbf{X_i}; \tau)$ and repeat by adding dimension. At the beginning slope will strongly increase due to false nearest neighbours, but at some large enough τ it will start behaving as true Lyapunov exponent. For this algorithm to work correctly, "good" range of embedding dimension values should be chosen.
Also, there is parameter ϵ which denotes the size of neighbourhood. The ϵ parameter depends on a noise level of the signal. For a noisy signal the typical distance between two neighbouring trajectories is in the order of ϵ. If ϵ is smaller than the noise amplitude, it will be hard to locate neighbours. Typically, when using this algorithm one should try different values of ϵ to adapt to noise.
For example, calculating LLE with Kantz algorithm with two different values of ϵ produces 14 figure. r

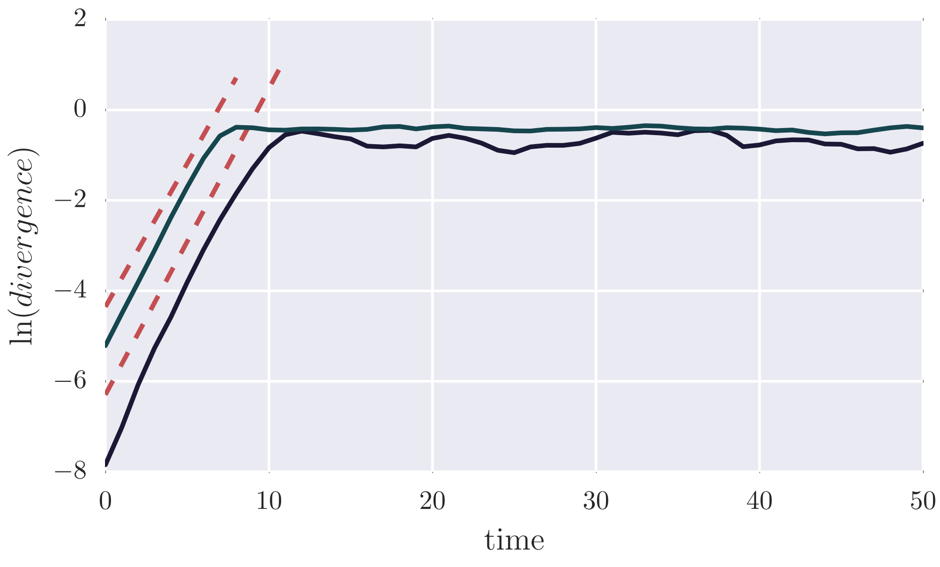

Figure 14. Kantz algorithm used on logistic map time series ($N = 500$, $r = 4$, $x[0] = 0.51$). Phase space is reconstructed using $m = 1$ and $L = 1$. Slope of dashed lines are $\lambda \approx 0.6771$ & $\lambda \approx 0.6331$.

5 Automatic estimation

5.1 Introduction

Usually estimation of correlation dimension and LLE is done manually. Researchers examine linearly increasing regions in Grassber-Procaccia, Rosenstein et al. or Kantz algorithm calculations. This is done because there may be spurious effects which would appear in the plots, for example "shoulder" like structure in $\log C(r)$ vs $\log r$ plot.
Manually examining large amounts of time series data can take a lot of time and resources. Automating this task can help researchers to do nonlinear analysis faster. Also, automatic estimation would allow to quickly check whether system has some nonlinear properties and then manually inspect the plots for spurious estimation.
Systems purposed in this project, allows automatically to calculate correlation dimension and largest Lyapunov exponent. In this work, popular TISEAN [21] project is used. This project implements many nonlinear analysis algorithms such as false nearest neighbours, mutual information, autocorrelation, Grassberger-Procaccia.

5.2 Estimation of correlation dimension

In this work automatic estimation of correlation dimension is done in the following way:

1. Reconstruction delay L can be estimated from autocorrelation or filtered mutual information function.
   (a) When estimating using autocorrelation, first d is found which satisfies $r(d) < 1 - \frac{1}{e}$. d is the estimated reconstruction delay.
   (b) When estimating using filtered mutual information function, before calculation of mutual information $x(t)$ is filtered using simple moving average with chosen window size w. Then first local minima of computed filtered mutual information function is found and resulting ΔT is used as reconstruction delay.
2. Embedding dimension m is estimated using global false nearest neighbours method. Method is used with estimated delay value L and some chosen value m_{max}, which is maximum embedding dimension.
3. Theiler window W is found from space-time separation plot using estimated delay value L. First local maxima is computed for every line in space-time separation plot. The most common ΔT is used as Theiler window W.
4. (a) Correlation dimension is computed using L and W. Starting from embedding dimension $m = 1$ correlation integrals are computed and m is increased by 1 until $\bar{D_2}^{(m)}$ no longer increases (Stopping criteria is satisfied).
   (b) The scaling region is found using Ramer–Douglas–Peucker (RDP) algorithm. Line is chosen with biggest Δy from the lines found using RDP. The distance dimension in RDP is approximated by standard deviation of time series divided by some constant. The default value for distance dimension is standard deviation devided by 2. Such value is chosen to account for variation in time series.

(c) Stopping criteria is computed using percentage difference between $\bar{D_2}^{(m-1)}$ and $\bar{D_2}^{(m)}$, $\bar{D_2}^{(m+1)}$ and $\bar{D_2}^{(m)}$. If these two differences are smaller than ϵ, calculation is stopped and D_2 is estimated using $\bar{D_2}^{(m)}$. We use $\epsilon = 0.05$ as default value.

5.3 Experimental results

The system defined in 5.2 is tested using time series generated from: Ikeda map using $a = 0.4, b = 6.0, c = 0.9$, logistic map using $r = 4$, Henon map using $a = 1.4, b = 0.3$, Lorenz equations using $r = 28, \sigma = 10, b = \frac{8}{3}$, taking 100 sample points per unit time, Rössler system using $a = 0.1, b = 0.2, c = 5.7$, integrating with first order forward Euler method. For all generated time series only x coordinate is used. Published estimated correlation dimension values for these systems are presented in 6 table.

series	D_2
Henon	1.21 ± 0.01
Ikeda	1.67
logistic	0.98
Lorenz	2.05 ± 0.01
Rössler	1.986

Table 6. Published correlation dimension values.

Correlation dimension is calculated with two types of delay reconstruction methods: autocorrelation, filtered mutual information using window size $w = 10$ and time series of lengths 1000, 5000, 10000, 50000. The results of this experiment are shown in 7, 8 tables.
Mostly autocorrelation and filtered mutual information delay estimation show similar results for generated time series. The exception is for Lorenz & Rössler systems data, where filtered mutual information function produced wrong estimates of delay value. This suggests that different type of filters or different filter parameters should be used, when filtering mutual information function.
When choosing length of time series it is seen that best results are for time series from 5000 up to 10000 length. Too small time series could show about $20\% - 35\%$ errors in D_2 estimates, e.g. Lorenz & Rössler systems. Too large (or oversampled) time series produce $1\% - 10\%$ errors. This suggest that length and oversampling of experimental data should be considered before applying this automated system.
Another important aspect is noise. Experimental data is usually noisy, so how does system estimate correlation dimension when estimating on noisy data? To account for this effect, Gaussian noise of 1% and 5% is added to generated time series. Resulting computations are in 9, 10, 11, 12 tables. Results show, that presented system is sensitive to noise, even for 1% Gaussian noise errors are from 1% up to 107%. Comparing filtered mutual information and autocorrelation reconstruction delay methods, filtered mutual information function produces better results for Lorenz & Rössler systems. These results are due to filtering, which removed some noise and allowed to approximate delay time better. 5% level noise produces even worse results, the errors are up to 205%.

series	length	D2	% error
Henon	1000	1.21	-0.3
Henon	5000	1.20	0.5
Henon	10000	1.21	-0.1
Henon	50000	1.22	-0.5
Ikeda	1000	1.82	-8.9
Ikeda	5000	1.78	-6.4
Ikeda	10000	1.81	-8.2
Ikeda	50000	1.80	-7.6
logistic	1000	1.03	-5.5
logistic	5000	1.09	-11.2
logistic	10000	1.10	-11.8
logistic	50000	1.09	-11.3
Lorenz	1000	2.60	-26.9
Lorenz	5000	2.20	-7.2
Lorenz	10000	2.18	-6.4
Lorenz	50000	2.02	1.3
Rössler	1000	1.55	22.1
Rössler	5000	1.96	1.5
Rössler	10000	1.96	1.4
Rössler	50000	1.76	11.2

Table 7. Results of applying system defined in 5.2 using autocorrelation estimation method on generated time series.

series	length	D2	% error
Henon	1000	1.21	-0.3
Henon	5000	1.20	0.5
Henon	10000	1.21	-0.1
Henon	50000	1.22	-0.5
Ikeda	1000	1.82	-8.9
Ikeda	5000	1.78	-6.4
Ikeda	10000	1.81	-8.2
Ikeda	50000	1.80	-7.6
logistic	1000	1.03	-5.5
logistic	5000	1.09	-11.2
logistic	10000	1.10	-11.8
logistic	50000	1.09	-11.3
Lorenz	1000	1.46	28.6
Lorenz	5000	1.42	30.9
Lorenz	10000	1.39	32.4
Lorenz	50000	1.37	33.0
Rössler	1000	1.52	23.7
Rössler	5000	1.02	48.4
Rössler	10000	0.87	56.2
Rössler	50000	0.88	55.6

Table 8. Results of applying system defined in 5.2 using filtered mutual information estimation method on generated time series.

series	length	D2	% error
Henon	5000	2.06	-70.3
Henon	10000	2.10	-73.5
Ikeda	5000	2.20	-31.5
Ikeda	10000	2.37	-42.1
logistic	5000	1.84	-88.2
logistic	10000	2.02	-106.2
Lorenz	5000	2.41	-17.3
Lorenz	10000	2.52	-23.2
Rössler	5000	2.18	-9.8
Rössler	10000	2.61	-31.5

Table 9. Experimental results with 1% added gaussian noise of system defined in 5.2 using autocorrelation estimation method.

series	length	D2	% error
Henon	5000	2.06	-70.3
Henon	10000	2.10	-73.5
Ikeda	5000	2.20	-31.5
Ikeda	10000	2.37	-42.1
logistic	5000	1.84	-88.2
logistic	10000	2.02	-106.2
Lorenz	5000	2.18	-6.2
Lorenz	10000	2.22	-8.5
Rössler	5000	1.96	1.1
Rössler	10000	1.93	2.9

Table 10. Experimental results with 1% added gaussian noise of system defined in 5.2 using filtered mutual information estimation method.

series	length	D2	% error
Henon	5000	2.72	-125.0
Henon	10000	2.81	-132.3
Ikeda	5000	3.00	-79.8
Ikeda	10000	3.36	-101.5
logistic	5000	2.81	-186.9
logistic	10000	2.98	-204.4
Lorenz	5000	2.84	-38.7
Lorenz	10000	3.18	-55.1
Rössler	5000	2.64	-32.9
Rössler	10000	2.89	-45.3

Table 11. Experimental results with 5% added gaussian noise of system defined in 5.2 using autocorrelation estimation method.

series	length	D2	% error
Henon	5000	2.72	-125.0
Henon	10000	2.81	-132.3
Ikeda	5000	3.00	-79.8
Ikeda	10000	3.36	-101.5
logistic	5000	2.81	-186.9
logistic	10000	2.98	-204.4
Lorenz	5000	2.56	-24.9
Lorenz	10000	2.99	-45.6
Rössler	5000	2.48	-25.1
Rössler	10000	2.63	-32.7

Table 12. Experimental results with 5% added gaussian noise of system defined in 5.2 using filtered mutual information estimation method.

5.4 Estimation of largest Lyapunov exponent

Estimation of largest Lyapunov exponent is done in the following way:

1. 1, 2, 3 steps are computed from section 5.2.
2. (a) Largest Lyapunov exponent λ can be estimated using Rosenstein or Kantz method.

   (b) When estimating using Rosenstein method, the LLE is found on a phase space reconstructed using m embedding dimension and L reconstruction delay. The scaling region is found using RDP (procedure is the same as in 5.2 section, 4B step). Slope of the scaling region is the resulting LLE.

   (c) When estimating using Kantz method, only reconstruction L is used for phase space reconstruction. Theiler window value is used as a minimum neighbour separation in time. From $m = 2$ up to some maximum constant m_{max}, plots are computed with different ϵ sizes of neighbourhoods (from $\frac{|\text{data interval}|}{1000}$ up to $\frac{|\text{data interval}|}{100}$). For each plot, scaling regions are found using RDP (procedure is the same as in 5.2 section, 4B step). Median of the computed slope values is the resulting LLE.

5.5 Experimental results

The system defined in 5.4 is tested using time series generated from: Ikeda map using $a = 0.4, b = 6.0, c = 0.9$, logistic map using $r = 4$, Henon map using $a = 1.4, b = 0.3$, Lorenz equations using $r = 45.92, \sigma = 16, b = 4$, taking 100 sample points per unit time. For all generated time series only x coordinate is used. Published estimated LLE values for these systems are presented in 13 table.

LLE is calculated with two types of delay reconstruction methods: autocorrelation, filtered mutual information using window size $w = 10$; time series of lengths 1000, 5000, 10000, 50000 and LLE

series	LLE
Henon	0.418
Ikeda	0.505
logistic	0.693
Lorenz	1.500

Table 13. Published largest Lyapunov exponent values.

approximation methods: Rosenstein et al., Kantz. The results of this experiment are shown in 14, 15, 16, 17 tables.

Both Kantz and Rosenstein et al. has failed to identify LLE for Lorenz system. Algorithms did not found linearly increasing region in their plots, the spurious results are due to "bump" in the corresponding plots. Figure 15 shows "bump" in the plot, when calculating LLE using Rosenstein et al. method. Results from Lorenz system are excluded from further consideration.

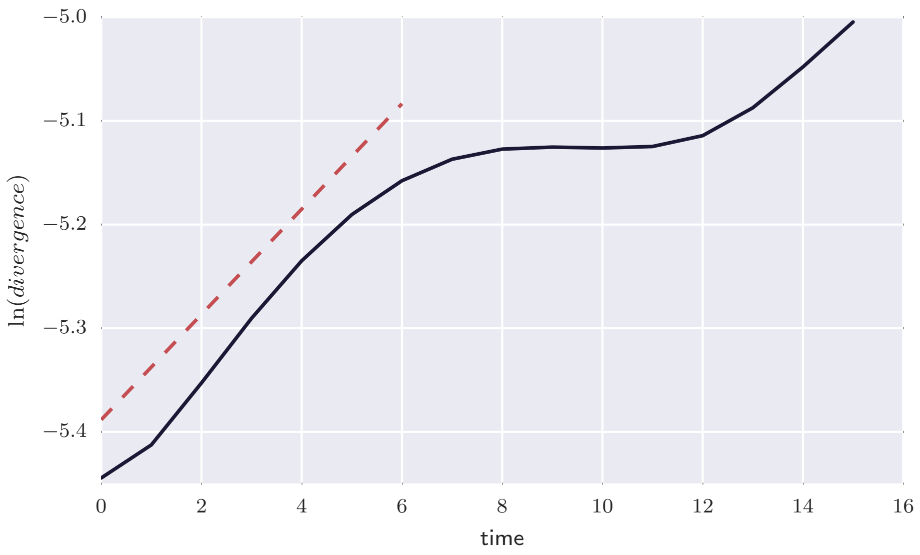

Figure 15. Rosenstein et al. algorithm used on Lorenz equation time series. Phase space is reconstructed using $m = 3$, $L = 12$. Slope of dashed line is $\lambda \approx 0.05$. There is no linearly increasing region, so LLE value is not correctly identified.

Filtered mutual information delay estimation method show better results than comparing with autocorrelation for both LLE estimation methods.

Autocorrelation method produce errors from 0.3% up to 25.2%, on average error is 6.28%, while filtered mutual information method produce errors from 0.3% up to 49.7%, on average error is 6.19%.

When choosing length of time series it is seen that best results are for time series from 5000 up to 10000 length. Too small time series could show about $6\% - 25\%$ errors in LLE estimates. Too large (or oversampled) time series produce $1\% - 14\%$ errors. This suggest that length and oversampling of experimental data should be considered before applying this automated system.

Kantz method show smaller errors from 0.3% to 7.7% for intermediate length time series, while Rosenstein et al. method produces errors from 0.4% to 14.4%. These results are gathered using filtered mutual information function reconstruction delay estimation method.

series	length	LLE	% error
Henon	1000	0.39	6.1
Henon	5000	0.40	4.3
Henon	10000	0.41	2.7
Henon	50000	0.37	10.3
Ikeda	1000	0.38	25.2
Ikeda	5000	0.43	14.3
Ikeda	10000	0.44	12.8
Ikeda	50000	0.46	8.5
logistic	1000	0.68	1.5
logistic	5000	0.69	0.7
logistic	10000	0.66	4.5
logistic	50000	0.66	5.2
lorenz	1000	0.04	97.2
lorenz	5000	0.03	97.8
lorenz	10000	0.01	99.1
lorenz	50000	0.02	98.3

Table 14. Results of applying system defined in 5.4 using autocorrelation delay reconstruction method and Rosenstein et al. LLE estimation method.

series	length	LLE	% error
Henon	1000	0.41	2.6
Henon	5000	0.41	1.4
Henon	10000	0.41	1.3
Henon	50000	0.38	10.1
Ikeda	1000	0.44	13.5
Ikeda	5000	0.44	12.6
Ikeda	10000	0.46	9.6
Ikeda	50000	0.43	14.4
logistic	1000	0.68	1.5
logistic	5000	0.69	0.7
logistic	10000	0.66	4.5
logistic	50000	0.69	0.4
lorenz	1000	0.39	74.0
lorenz	5000	0.36	75.8
lorenz	10000	0.39	73.7
lorenz	50000	0.21	86.1

Table 15. Results of applying system defined in 5.4 using filtered mutual information delay reconstruction method and Rosenstein et al. LLE estimation method.

series	length	LLE	% error
Henon	1000	0.36	13.5
Henon	5000	0.39	6.7
Henon	10000	0.39	5.8
Henon	50000	0.40	3.5
Ikeda	1000	0.51	-1.2
Ikeda	5000	0.50	1.3
Ikeda	10000	0.47	7.7
Ikeda	50000	0.45	10.3
logistic	1000	0.72	-3.3
logistic	5000	0.69	-0.3
logistic	10000	0.70	-0.9
logistic	50000	0.70	-0.3
lorenz	1000	0.09	93.9
lorenz	5000	0.03	98.1
lorenz	10000	0.03	97.9
lorenz	50000	0.01	99.0

Table 16. Results of applying system defined in 5.4 using autocorrelation delay reconstruction method and Kantz LLE estimation method.

series	length	LLE	% error
Henon	1000	0.43	-2.9
Henon	5000	0.41	1.0
Henon	10000	0.40	3.6
Henon	50000	-	-
Ikeda	1000	0.76	-49.7
Ikeda	5000	0.49	2.0
Ikeda	10000	0.48	4.7
Ikeda	50000	0.47	7.7
logistic	1000	0.72	-3.3
logistic	5000	0.69	-0.3
logistic	10000	0.70	-0.5
logistic	50000	0.70	-0.3
lorenz	1000	0.08	94.6
lorenz	5000	0.05	96.4
lorenz	10000	0.07	95.3
lorenz	50000	0.70	53.2

Table 17. Results of applying system defined in 5.4 using filtered mutual information delay reconstruction method and Kantz LLE estimation method.

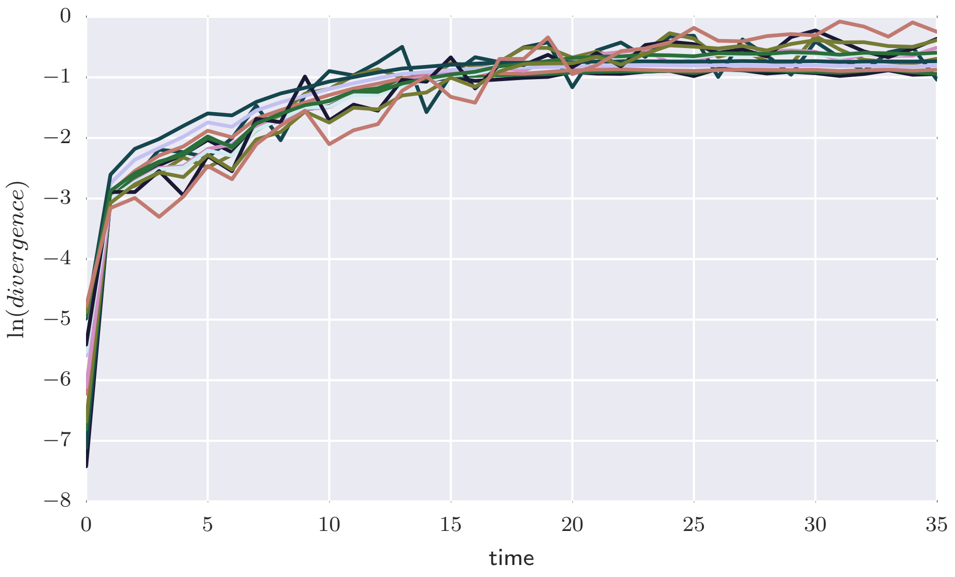

Figure 16. Kantz algorithm used on Lorenz equation time series. Phase space is reconstructed using $m = 6$, $L = 6$. Slope of dashed lines are $\lambda \approx 2.84$. Linearly increasing region has many fluctuations (region between 1 and 12 timesteps).

To model noisy experimental data, Gaussian noise of 1% and 5% is added to generated time series. Resulting computations are in 18, 19, 20, 21 tables.
For small data perturbation Kantz method produces smaller errors, which ranges from 0.8% to 14.5%, while Rosenstein et al. method produces errors from 15.9% to 19.6%. These results are gathered using filtered mutual information function reconstruction delay estimation method. Systems behave totally differently, than used with 5% Gaussian noise. Rosenstein et al. method produced errors between 18.4% to 36.6%, while Kantz methods results there from 68.5% to 369.1%. Spurious results produced using Kantz method are due to wrong linearly increasing region detection. When time series are noisy, there are many flucations in the scaling region as shown in figure 16.

series	lle method	length	LLE	% error
Henon	Kantz	5000	0.37	12.2
Henon	Kantz	10000	0.37	12.4
Ikeda	Kantz	5000	0.47	7.8
Ikeda	Kantz	10000	0.41	19.6
logistic	Kantz	5000	0.62	9.9
logistic	Kantz	10000	0.66	4.3
Henon	Rosenstein	5000	0.35	15.3
Henon	Rosenstein	10000	0.34	17.9
Ikeda	Rosenstein	5000	0.39	22.3
Ikeda	Rosenstein	10000	0.40	20.2
logistic	Rosenstein	5000	0.56	19.6
logistic	Rosenstein	10000	0.56	19.3

Table 18. Experimental results with 1% added Gaussian noise of system defined in 5.4 using autocorrelation estimation method.

series	lle method	length	LLE	% error
Henon	Kantz	5000	0.36	14.5
Henon	Kantz	10000	0.37	10.3
Ikeda	Kantz	5000	0.50	0.8
Ikeda	Kantz	10000	0.49	3.7
logistic	Kantz	5000	0.62	9.9
logistic	Kantz	10000	0.66	4.3
Henon	Rosenstein	5000	0.35	15.9
Henon	Rosenstein	10000	0.35	17.3
Ikeda	Rosenstein	5000	0.41	18.2
Ikeda	Rosenstein	10000	0.41	19.1
logistic	Rosenstein	5000	0.56	19.6
logistic	Rosenstein	10000	0.56	19.3

Table 19. Experimental results with 1% added Gaussian noise of system defined in 5.4 using filtered mutual information estimation method.

series	lle method	length	LLE	% error
Henon	Kantz	5000	1.09	-160.0
Henon	Kantz	10000	0.70	-68.5
Ikeda	Kantz	5000	2.18	-331.9
Ikeda	Kantz	10000	2.37	-369.1
logistic	Kantz	5000	1.79	-158.2
logistic	Kantz	10000	2.12	-206.4
Henon	Rosenstein	5000	0.34	18.4
Henon	Rosenstein	10000	0.32	24.5
Ikeda	Rosenstein	5000	0.32	36.6
Ikeda	Rosenstein	10000	0.33	35.2
logistic	Rosenstein	5000	0.50	28.3
logistic	Rosenstein	10000	0.48	31.0

Table 20. Experimental results with 5% added Gaussian noise of system defined in 5.4 using autocorrelation estimation method.

series	lle method	length	LLE	% error
Henon	Kantz	5000	1.51	-260.7
Henon	Kantz	10000	1.49	-256.3
Ikeda	Kantz	5000	2.31	-357.9
Ikeda	Kantz	10000	1.32	-161.5
logistic	Kantz	5000	1.79	-158.2
logistic	Kantz	10000	-	-
Henon	Rosenstein	5000	0.34	19.4
Henon	Rosenstein	10000	0.33	21.9
Ikeda	Rosenstein	5000	0.32	36.6
Ikeda	Rosenstein	10000	0.33	34.0
logistic	Rosenstein	5000	0.50	28.3
logistic	Rosenstein	10000	0.48	31.0

Table 21. Experimental results with 5% added Gaussian noise of system defined in 5.4 using filtered mutual information estimation method.

6 Implementation details

Linear regression (sec. 1.1), multilayer perceptron (sec. 1.2), automatic estimation systems (sec. 5.2, 5.4) were implemented in Python programming language. Project relies on scientific computing package NumPy (http://www.numpy.org) & SciPy (https://www.scipy.org) for linear algebra support (vectors, matrices, matrix manipulations), matplotlib (http://matplotlib.org) and seaborn (http://seaborn.pydata.org) for plotting support, pandas (http://pandas.pydata.org) for financial data loading into data frames.

Multilayer perceptron with sigmoid and hyperbolic tangent activation functions were implemented. The correctness of backpropagation rule (partial derivative computation) was validated with derivatives computed numerically. Small errors from 10^{-10} to 10^{-12} where achieved comparing with leap frog gradient.

Automatic estimation systems are built into seperate python package. This package builds on top of TISEAN package [21]. The code is written to be reusable, some TISEAN low level details are abstracted away. Moreover, more chaotic time series such as henon map or ikeda map and methods such as Ramer–Douglas–Peucker algorithm is added. In the future, this python package is going to be open sourced.

Around 2100 lines of code are written for prediction methods and experiments with financial data and 2300 lines of code for automatic estimation system package.

7 Classification

7.1 Introduction

Unsupervised classification based on LLE and correlation dimension is presented here. Automatic calculation of LLE or correlation dimension measures are done according to 5.2, 5.4 sections. Reconstruction delay is computed using filtered mutual information. Automatic LLE estimation is done using Rosenstein method.
Classification is done using data binning (histogram) approach. When computed measure value falls into given interval it is considered to be a part of that group. All values for which measure could not be computed are grouped into a single group.
The number of groups (bins) is a parameter of this method, which may be different for particular problem. For example, if using this system with data intensive prediction methods one could use equal density binning, so that training data would be split equally; or when trying to evaluate difference of signals visually one could try choosing values manually. The benefit of this classification method is that groups have labels (intervals of LLE values or D_2 values). LLE groups suggest how far signals may be predicted and correlation dimension groups by the complexity of the signals.

7.2 Experiment on 2 section data

The classification system is applied on financial data described in 2.2 section. Computed correlation dimension and LLE values are in tables 23, 24. The approximation of correlation dimension is incorrect, this is due to high noise or volatility of the data. This problem was described in section 5.2.
Different results are seen for classification using LLE values. Table 22 shows how sigmoid MLP, hyperbolic tangent MLP, linear regression classification accuracy and improvement relates to LLE values. This relationship is quantified using Pearson correlation coefficient, which shows the linear dependence between two variables. For a sample, correlation coefficient is computed using:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}. \tag{7.1}$$

The computed absolute values of Pearson correlation coefficient are expressed in percentage. Results show that LLE values correlate with classification accuracy and improvement of analysed models.

	LLE
Sigm class.	64.42
Sigm improvement %	45.18
Tanh class.	33.32
Tanh improvement %	45.67
LR class.	33.66
Linear improvement %	64.72

Table 22. Absolute values of Pearson correlation coefficient for time series. Coefficient values are expressed in percentage

Manually choosing to classify time series into 4 groups, produces one group with empty values (^IXIC,SUNE stocks) and others with LLE:

- 0.02 to 0.18 - ^TNX, ORCL, AAPL, NYSE.CI, VXX, ^GSPC, MSFT;
- 0.38 to 0.82 - TSLA, FCX, SNP, SPY, BAC, ^RUT;
- 1.24 to 2.17 - DUST, ^VIX, OPERA.OL, MA.

This grouping could provide useful information, when trying to predict the time series.
Using these classes with prediction results from 3.3 section shows that first group on average has an increase of 2.7% classification rate for MLP with sigmoid and 3.4% with hyperbolic tangent activation function. 0.02 to 0.18 LLE group on average has an increase of 3.7% for MLP with sigmoid and 3.97% with hyperbolic tangent activation function. 0.38 to 0.82 LLE group on average has an increase of 4.87% for MLP with sigmoid and 4.26% with hyperbolic tangent activation function. Last group's increase varied from 1.02% to 10.69% for sigmoid MLP and 1.64% to 12% for hyperbolic tangent MLP.
These results suggest that 0.02 to 0.18 LLE group could be classified using MLP with hyperbolic tangent activation function and 0.38 to 0.82 group with sigmoid activation function. First group has a low improvement rate so these time series should be discarded, while last group has a big improvement variability, this making them hard to predict.

name	D_2
DUST	0.99
^VIX	0.98
SUNE	0.98
TSLA	0.92
^GSPC	0.91
MA	0.87
^RUT	0.86
^TNX	0.86
SPY	0.85
FCX	0.84
SNP	0.84
OPERA.OL	0.81
^IXIC	0.68
AAPL	0.67
MSFT	0.67
ORCL	0.57
VXX	0.55
NYSE.CI	0.53
BAC	-

Table 23. Automatic correlation dimension estimation system applied to financial time series.

name	LLE
DUST	2.17
^VIX	1.75
OPERA.OL	1.74
MA	1.24
TSLA	0.82
FCX	0.61
SNP	0.57
SPY	0.56
BAC	0.39
^RUT	0.38
^TNX	0.18
ORCL	0.15
AAPL	0.14
NYSE.CI	0.14
VXX	0.13
^GSPC	0.06
MSFT	0.02
^IXIC	-
SUNE	-

Table 24. Automatic largest Lyapunov exponent estimation system applied to financial time series.

7.3 Experiment on 4026 stocks

The classification system using LLE value is applied on 4026 stocks from finance.yahoo.com. The results are shown in figure 17 . It can be seen that LLE values are largely distributed with values ranging from 0.004 to 13.759. Only 1669 out of 4026 stocks are shown, because for all the other stocks LLE could not be computed.

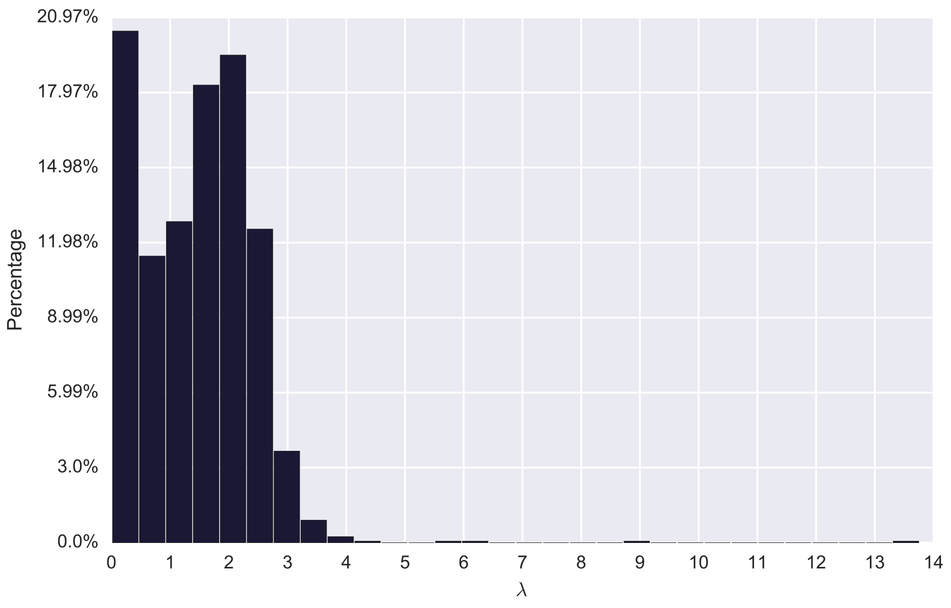

Figure 17. Classification of financial time series based on LLE(λ) value. LLE values are from 0.004 to 13.759.

Same experiment as in 3.3 section is applied on the 4026 stocks. MLP and linear regression models are trained 10 times with inputs ranging from 1 to 10 day's adjusted closing prices. MLPs are trained using following configurations:

- MLP using $2N + 1$ hidden sigmoid artificial neurons,
- MLP using $2N + 1$ hidden hyperbolic tangent neurons,

where N is the number of inputs.
Pearson correlation was computed using equation 7.1 on results from models classification accuracy, improvement and computed LLE values. Table 25 shows the absolute values of Pearson correlation coefficient, which are expressed in percentage. It can be seen that LLE values weakly correlate with classification accuracy and improvement of analysed models. This result is expected as Pearson correlation quantifies only linear dependence of two variables, so LLE with model's prediction accuracy and improvement could relate in nonlinear way.
Manually time series are grouped into 4 groups with LLE:

- non computed - 2357 stocks;
- 0 to 0.5 - 350 stocks;
- 0.5 to 1 - 238 stocks;
- more than 1 - 1081 stocks.

	LLE
Sigm class	24.16
Sigm improvement %	20.25
Tanh class	25.67
Tanh improvement %	17.14
LR class	20.18
LR improvement %	29.05

Table 25. Absolute values of Pearson correlation coefficient for time series. Coefficient values are expressed in percentage.

Suggested classes are relatively close to grouping described in 7.2 section.
Most of the stocks fall into non computed category. This happens when automatic LLE system fails to compute LLE value due to embedding dimension or reconstruction delay being too large, so that not enough points are available in the phase space to correctly compute nonlinear measure. Interestingly stocks which could not be computed also have very low mean classification accuracy of $36.36\% - 37.21\%$ as seen in 29 table. This grouping, allows discarding stocks for which LLE could not be computed, because these stocks are too complex to predict.
Tables 26, 27, 28 show prediction performance of stocks for which LLE was computed successfully. Stocks that are in LLE range 0 to 0.5 have low classification accuracy of $45.25\% - 46.05\%$ and quite large standard deviation of $14.30 - 14.36$. Also, the improvement, when choosing number of input variables for stocks is quite big ($11.26\% - 13.60\%$), statistics suggest that maybe some other model or different parameters could be used with these stocks to get better prediction results.
MLP model seem to excel when trying to predict stocks for which LLE values are in 0.5 to 1 range. Mean classification accuracy is 53.77% to 53.84% for MLP models with quite low standard deviation (5.35).
Stocks, which fall into last category, MLP models can produce predictions slightly better than random guess ($50.71\% - 50.81\%$). For these stocks Lyapunov exponent is larger than one and small changes in the surrounding environment can hugely affect the price direction. So, trying to predict these stocks should be done with caution.
Generally, from 4026 only 588 stocks could be potentially predicted (2nd and 3rd groups), with MLP models predicting stocks from 3rd group. Classification based on nonlinear methods in financial time series prediction seem to provide useful information. Groupings provided by classification system described in section 7 are very useful when trying to filter out unpredictable time series.

	Sigm class.	Sigm increase %	Tanh class.	Tanh increase %	LR class.	LR increase %	LLE
mean	46.05	13.60	46.05	13.59	45.25	11.26	0.18
std	14.30	6.78	14.29	6.91	14.46	8.07	0.14
min	0.00	0.00	0.00	0.00	0.00	0.00	0.01
max	62.87	33.87	67.37	30.99	63.27	28.51	0.50

Table 26. Prediction statistics for stocks, which LLE lies between 0 and 0.5. Totally 350 stocks fall into this category.

	Sigm class.	Sigm increase %	Tanh class.	Tanh increase %	LR class.	LR increase %	LLE
mean	53.77	9.41	53.84	9.61	51.94	4.33	0.78
std	5.35	4.37	5.35	4.56	6.15	5.25	0.15
min	0.00	0.00	0.00	0.00	0.00	0.00	0.50
max	68.00	35.24	69.79	36.76	100.00	100.00	1.00

Table 27. Prediction statistics for stocks, which LLE lies between 0.5 and 1. Totally 238 stocks fall into this category.

	Sigm class.	Sigm increase %	Tanh class.	Tanh increase %	LR class.	LR increase %	LLE
mean	50.81	11.48	50.71	11.41	49.22	8.04	2.00
std	10.13	5.86	10.00	5.91	10.14	6.86	0.66
min	0.00	0.00	0.00	0.00	0.00	0.00	1.00
max	61.86	27.58	59.55	29.70	62.04	28.24	13.76

Table 28. Prediction statistics for stocks, which LLE measure is bigger than 1. Totally 1081 stocks fall into this category.

	Sigm class.	Sigm increase %	Tanh class.	Tanh increase %	LR class.	LR increase %
mean	37.21	15.62	37.16	15.73	36.36	13.85
std	19.22	9.35	19.12	9.56	19.03	9.26
min	0.00	0.00	0.00	0.00	0.00	0.00
max	80.00	71.43	90.00	90.00	81.82	66.67

Table 29. Prediction statistics for stocks, which LLE was not calculated. Totally 2357 stocks fall into this category.

Conclusions and Recommendations

Multilayer perceptron and linear regression models are presented in 1 section. MLP models show promising results when trying to predict stock market data, although these models do not perform better for all of the stocks. Also experiments show that different historic price windows should be used in the input layer of multilayer perceptron model, as it turns out, that using only one day's before price is not enough for model to generalize well.
Automatic correlation dimension estimation system presented in 5.2 section is suitable for non-noisy and non-oversampled time series data. When using this estimation method with experimental time series data one should be aware of possibility of spurious results.
Automatic Largest Lyapunov exponent estimation method presented in 5.4 section is suitable for experimental time series data with some level of noise. When testing with time series with Gaussian noise from 1% to 5%, system showed relatively small errors.
Altough, automatic estimation systems presented here show quite good results, resulting measures should not be used as accurate estimations. The purpose of these methods are for applying on large datasets, where manual work would take a lot of time and for use cases where errors are acceptable.
Classification based on nonlinear measures is presented in 7 section. Applying this system using largest Lyapunov exponent on financial data results in interesting observations about the data. This grouping allows to filter out time series that are too complex to be predicted. Also, experimental results show that MLP models perform well predicting financial time series for which Lyapunov exponent lies between 0.5 and 1.

Future work guidelines

Forecasting stock market prices is going to continue be actively researched by many scientists and investors that try to outperform the market. Models such as neural networks will continue to improve as machine learning and artificial intelligence are getting more popular every day. Using different neural networks to forecast stock market prices, such as recurrent neural networks could give better performance than multilayer perceptron model used in this work.

Also, improving nonlinear measures automatic estimation is very important. Nonlinear measures are used in many domains such as medicine, meteorology, physics and are still computed manually. Improving automatic estimation described in this work could be done in many ways. One of the ways is improving phase space reconstruction methods which could improve nonlinear measures accuracy. Recently, false strands method was introduced, which improves global false nearest neighbour method and approximates embedding dimension better for noisy and autocorrelated data.

Automatic correlation dimension system could be estimated with Gausian kernel estimator or Takens-Theiler estimator, which should produce more accurate results for noisy time series. Automatic correlation dimension estimation could be compared with recently published systems, such as [33]. Also, estimation methods reliability could be improved by using surrogate data methods, which should help distinguish low dimensional chaos from stochastic processes.

References

[1] Henry D. I. Abarbanel, Reggie Brown, John J. Sidorowich, and Lev Sh. Tsimring. The analysis of observed chaotic data in physical systems. *Rev. Mod. Phys.*, 65:1331–1392, Oct 1993.

[2] George S. Atsalakis and Kimon P. Valavanis. Surveying stock market forecasting techniques – part II: soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941, 2009.

[3] E. Michael Azof. *Neural Network Time Series Forecasting of Financial Markets*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[4] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: Theory. *Meccanica*, 15(1):9–20, 1980.

[5] Armando Bernal, Sam Fok, and Rohit Pidaparthi. Financial Market Time Series Prediction with Recurrent Neural Networks. pages 1–5, 2012.

[6] Liangyue Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Phys D Nonlinear Phenom*, 110(1-2):43–50, 1997.

[7] J. -Y. Carré, A. Høst-Madsen, K. L. Bowes, and M. P. Mintchev. Dynamics of the level of deterministic chaos associated with gastric electrical uncoupling in dogs. *dical & Biological Engineering & Computing*, 39(3):322–329, 2001.

[8] CJ Cellucci, AM Albano, and PE Rapp. Comparative study of embedding methods. *Phys Rev E*, 67(6), 2003.

[9] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, dec 1989.

[10] Atin Das. Chaotic analysis of the foreign exchange rates during 2008 to 2009 recession. *AFRICAN JOURNAL OF BUSINESS MANAGEMENT*, 6(15), 2012.

[11] Ding, Grebogi, Ott, Sauer, and JA Yorke. Plateau onset for correlation dimension: When does it occur? *Phys. Rev. Lett.*, 70(25):3872–3875, 1993.

[12] J. P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617–656, Jul 1985.

[13] Eugene F. Fama and Kenneth R. French. Business conditions and expected returns on stocks and bonds. *Journal of Financial Economics*, 25(1):23–49, 1989.

[14] Wayne E Ferson. Changes in expected security returns, risk, and the level of interest rates. *The Journal of Finance*, 44(5):1191–1217, 1989.

[15] AM Fraser and HL Swinney. Independent coordinates for strange attractors from mutual information. *Phys Rev A Gen Phys*, 33(2):1134–1140, 1986.

[16] Jianbo Gao and Zhemin Zheng. Local exponential divergence plot and optimal embedding of a chaotic time series. *Phys Lett A*, 181(2):153–158, 1993.

[17] Edward Gately. *Neural Networks for Financial Forecasting*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[18] Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. *Phys D Nonlinear Phenom*, 9(1-2):189–208, 1983.

[19] Erkam Guresen, Gulgun Kayakutlu, and Tugrul U. Daim. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, 2011.

[20] Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2009. Autres impressions : 2011 (corr.), 2013 (7e corr.).

[21] Rainer Hegger, Holger Kantz, and Thomas Schreiber. Practical implementation of nonlinear time series methods: The tisean package. *Chaos*, 9(2):413–435, 1999.

[22] K Hornik, M Stinchcombe, and H White. Kornick et. al. *Neural Networks*, 2:359–366, 1989.

[23] Holger Kantz. A robust method to estimate the maximal lyapunov exponent of a time series. *Physics Letters A*, 185(1):77–87, 1994.

[24] Vojislav Kecman. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, Cambridge, MA, USA, 2001.

[25] MB Kennel, Brown, and HD Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Phys. Rev., A*, 45(6):3403–3411, 1992.

[26] Monica Lam. Neural network techniques for financial performance prediction: Integrating fundamental and technical analysis. *Decision Support Systems*, 37(4):567–581, 2004.

[27] Yann LeCun, Yoshua Bengio, Geo rey Hinton, Lecun Y., Bengio Y., and Hinton G. Deep learning. *Nature*, 521(7553):436–444, 2015.

[28] Francesco Lisi and Vigilio Villi. CHAOTIC FORECASTING OF DISCHARGE TIME SERIES: a CASE STUDY1. *JAWRA Journal of the American Water Resources Association*, 37(2):271–279, 2001.

[29] Ricardo Mañé. *Dynamical Systems and Turbulence, Warwick 1980: Proceedings of a Symposium Held at the University of Warwick 1979/80*, chapter On the dimension of the compact invariant sets of certain non-linear maps, pages 230–242. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.

[30] J. M. Martinerie, A. M. Albano, A. I. Mees, and P. E. Rapp. Mutual information, strange attractors, and the optimal estimation of dimension. *Phys. Rev. A*, 45:7058–7064, May 1992.

[31] Leonardo C. Martinez, Diego N. Da Hora, João R. De, Wagner Meira, and Gisele L. Pappa. From an artificial neural network to a stock market day-trading system: A case study on the BM&F BOVESPA. *Proceedings of the International Joint Conference on Neural Networks*, pages 2006–2013, 2009.

[32] Michael D McKenzie. Chaotic behavior in national stock market indices new evidence from the close returns test. *Global Finance Journal*, 12(1):35–53, 2001.

[33] Alexey Mekler. Calculation of EEG correlation dimension: Large massifs of experimental data. *Comput Meth Prog Bio*, 92(1):154–160, 2008.

[34] Tadas Meškauskas, Algimantas Juozapavičius. *Vaizdų ir signalų analizė ir apdorojimas*. 2011.
Sygnal analysis part availabe: http://www.mif.vu.lt/~meska/SAA/Tadas_Meskauskas_-_Signalu_Analize_Ir_Apdorojimas_-_Mokymo_Priemone.pdf.

[35] John J. Murphy and John J. Murphy. *Technical analysis of the financial markets*. New York Institute of Finance, Fishkill, N.Y., 1999.

[36] Seyed Taghi Akhavan Niaki and Saeid Hoseinzade. Forecasting S&P 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1):1, 2013.

[37] Norman H Packard, James P Crutchfield, Doyne J Farmer, and Robert S Shaw. Geometry from a time series. *Physical Review Letters*, 45(9):712, 1980.

[38] Jose C. Principe, Neil R. Euliano, and W. Curt Lefebvre. *Neural and Adaptive Systems: Fundamentals Through Simulations with CD-ROM*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.

[39] A. Provenzale, L.A. Smith, R. Vio, and G. Murante. Distinguishing between low-dimensional dynamics and randomness in measured time series. *Phys D Nonlinear Phenom*, 58(1-4):31–49, 1992.

[40] Mingyue Qiu, Yu Song, and Fumio Akagi. Application of artificial neural network for the prediction of stock market returns: The case of the japanese stock market. *Chaos, Solitons & Fractals*, 85:1–7, 2016.

[41] Radhakrishna K.A Rao and Vikram Kumar Yeragani. Decreased chaos and increased nonlinearity of heart rate time series in patients with panic disorder. *Autonomic Neuroscience*, 88(1-2):99–9108, 2001.

[42] Michael T Rosenstein, James J Collins, and Carlo J De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1-2):117–134, 1993.

[43] Shinichi Sato, Masaki Sano, and Yasuji Sawada. Practical methods of measuring the generalized dimension and the largest lyapunov exponent in high dimensional chaotic systems. *Progress of Theoretical Physics*, 77(1):1–5, 1987.

[44] Floris Takens. *Dynamical Systems and Turbulence, Warwick 1980: Proceedings of a Symposium Held at the University of Warwick 1979/80*, chapter Detecting strange attractors in turbulence, pages 366–381. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.

[45] T. Z. Tan, C. Quek, and G. S. Ng. Brain-inspired genetic complementary learning for stock market prediction. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2653–2660 Vol. 3, Sept 2005.

[46] Theiler. Spurious dimension from correlation algorithms applied to limited time-series data. *Phys Rev A Gen Phys*, 34(3):2427–2432, 1986.

[47] W.L. Tung and C. Quek. Financial volatility trading using a self-organising neural-fuzzy semantic network and option straddle-based approach. *Expert Systems with Applications*, 38(5):4668 – 4688, 2011.

[48] Povilas Versockas. Scientific research project: Forecasting of financial data. 2016.

[49] Alan Wolf, Jack B. Swift, Harry L. Swinney, and John A. Vastano. Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285 – 317, 1985.

[50] P. Yousefpoor, M.S. Esfahani, and H. Nojumi. Looking for systematic approach to select chaos tests. *Applied Mathematics and Computation*, 198(1):73–91, 2008.