SCHOOL *of* GRADUATE STUDIES
EAST TENNESSEE STATE UNIVERSITY

East Tennessee State University
Digital Commons @ East Tennessee State University

Electronic Theses and Dissertations

12-2016

A Multi-Indexed Logistic Model for Time Series

Xiang Liu
East Tennessee State University

Follow this and additional works at: http://dc.etsu.edu/etd

 Part of the Applied Statistics Commons, and the Statistical Models Commons

Recommended Citation
Liu, Xiang, "A Multi-Indexed Logistic Model for Time Series" (2016). *Electronic Theses and Dissertations.* Paper 3140.
http://dc.etsu.edu/etd/3140

This Thesis - Open Access is brought to you for free and open access by Digital Commons @ East Tennessee State University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ East Tennessee State University. For more information, please contact dcadmin@etsu.edu.

A Multi-Indexed Logistic Model for Time Series

A thesis

presented to

the faculty of the Department of Mathematics

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Mathematical Sciences

---

by

Xiang Liu

December 2016

Jeff Knisley, Ph.D., Chair

Robert Price, Ph.D.

Chris Wallace, Ph.D.

Keywords: time series, logistic regression, statistics

ABSTRACT

Apply Multi-Index Logistic Model on Time Series

by

Xiang Liu

In this thesis, we explore a multi-indexed logistic regression (MILR) model, with particular emphasis given to its application to time series. MILR includes simple logistic regression (SLR) as a special case, and the hope is that it will in some instances also produce significantly better results. To motivate the development of MILR, we consider its application to the analysis of both simulated sine wave data and stock data. We looked at well-studied SLR and its application in the analysis of time series data. Using a more sophisticated representation of sequential data, we then detail the implementation of MILR. We compare their performance using forecast accuracy and an area under the curve score via simulated sine waves with various intensities of Gaussian noise and Standard & Poors 500 historical data. Overall, that MILR outperforms SLR is validated on both realistic and simulated data. Finally, some possible future directions of research are discussed.

Copyright by Xiang Liu 2016

All Rights Reserved

ACKNOWLEDGMENTS

I would like to express my gratitude to my thesis advisor Dr. Jeff Knisley for his unlimited patience and helpful guidance to me. Everything would be so difficult without his sincere instruction and tremendous kindness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Motivation of Thesis

For a stock analyst, it is crucial to make predictions of future pricing and volatility accurately. Both underestimates and overestimates of the predictive power of a forecast model can lead to a huge loss of money and time. Traditional ARMA models are good at dealing with stationary and linear time series. But in reality, transaction volume and pricing can be affected by transient and extreme events which disrupt the stationary assumption of an ARMA model. A machine learning classifier is therefore widely used for the analysis of delay prediction of irregular time series data [12, 2, 7, 8, 25].

Inspired by some well-studied simple logistic regression (SLR) models based on sequential data representation [24, 16, 10, 30], we develop a multi-indexed based logistic regression (MILR) model and hope it will outperform SLR. With multi-indexed data representations, information from a matrix representation is relational – which is often interpreted as graph-theoretic – and is more useful than initially sequential representation of the same data. In the long run, we hope our research can help people make more accurate predictions in a highly uncertain stock market.

1.2 Outline of Thesis

This thesis is organized as follows. Chapter 2 introduces several important background knowledge: time series and its frequently-used ARMA models, SLR and its maximum likelihood estimates, bias and variance definitions and their trade-offs in

making predictions, trajectory matrix illustration, and its role played in constructing an MILR model.

Chapter 3 illustrates how we develop MILR based on SLR with the hope of further reducing the loss function. Also, two different performance measures, the hit rate and an area under the curve, are introduced.

In Chapter 4, we present performance of SLR and MILR on both simulated noisy data and real stock historical data. The conclusions and discussions are given in Chapter 5.

2 BACKGROUND KNOWLEDGE

2.1 Definition of A Time Series

A time series is composed of a sequence of measurements indexed by a subset of the integers. The ordering of the index implies a sequential ordering of the observations, and indeed, a times series is often measured at successive points in time. It is mathematically defined as a set of vectors $x(t),\ t = 0, 1, 2, \cdots$ where t represents the time elapsed [1].

A time series can be either univariate or multivariate depending on the number of observations recorded at equally spaced time intervals. However, time is not always contiguous [29]. For example, stock prices are equispaced for all weekdays but are not available during weekends because the stock market is closed. A function $x(t)$ where t is a continuous variable is called a continuous time series. Examples include temperature readings, river flow, and change of height. On the other hand, many measurements are recorded at discrete points of time such as currency exchanges, the number of population growth and death. A continuous time series can be easily transformed into a discrete one by just grouping data within specific time intervals. Time series can be either linear or nonlinear depending on whether or not the dependent variable and all its lagged values appear in a linear fashion.

Definition 2.1 *A time series is **linear** when the next observation is a linear function of previous observations [20].*

Definition 2.2 *A time series is **non-linear** when it is not linear*

2.2 Time Series Decomposition

We can decompose a time series into the sum or the product of four different types of components: trend, cyclical, seasonal and irregular components. A trend is a tendency to increase, decrease or stagnate over an extended period. The cyclical variation accounts for non-periodic changes in time series caused by repeated events. The duration of a cycle extends over a longer period, usually two or more years [32]. Unlike cyclical which might involve some subjectivity in estimation, seasonal variations are more regular fluctuations that occur within a year during the season. It is quite essential for retailers to have a good estimation of seasonal differences for making a proper retail plan. The irregular piece of a time series is the random variation resulting from fluctuating influences. The irregular component is often assumed to be Gaussian distributed [28, p. 15]. Some time series show more periodic pattern whereas others display more fluctuations in their evolution. Depending on the underlying independence assumption of the four main components of time series, we can have either an additive model or a multiplicative model.

Definition 2.3 *A **multiplicative model** assumes the four main components of time series are not necessarily independent. Each observation Y_t can be written as a product of trend, cyclical, seasonal and irregular variation at time t : $Y_t = T_t \times C_t \times S_t \times I_t$ [23, p. 5].*

Definition 2.4 *An **additive model** assumes that the four components are independent of each other. Each observation Y_t can be written as a sum of trend, cyclical, seasonal and irregular variation at time t : $Y_t = T_t + C_t + S_t + I_t$ [23, p. 5].*

Time series in the real world are often only partially additive, as for example when the irregular component is additive to a multiplicative model of the other 3 (i.e., $Y_t = T_t \times C_t \times S_t + I_t$).

In time series forecasting, past observations are collected and analyzed to develop a mathematical model describing the underlying data generating process for the series [1]. After decomposition, we can proceed to make a prediction of each component. When detecting a seasonal component, we often assume it varies little in the future. Therefore, past seasonal effects are used to forecast seasonality in the future. Since a time series is non-deterministic in nature, which we cannot know for certain what will happen even short term, the analysis must reach beyond single-instant variable and instead deal with the joint probability distribution of time series components.

Definition 2.5 *Let X and Y be two continuous random variables. Then*

$$F(x, y) = P(X \leq x \bigcap Y \leq y)$$

is called the ***joint probability distribution*** *of X and Y*

The mathematical expression describing the probability structure of a time series is termed as a stochastic process [21].

Definition 2.6 *A* ***stochastic process*** *is a parametrized collection of random variables $\{X_t\}_{t \in T}$ defined on a probability space Ω and assuming values in $\mathbb{R}^n$ [31]. It is, in general, an n-dimensional joint probability distribution $p(X_1, X_2, \cdots, X_n)$*

For this reason, each time series $X = \{X_1, X_2, \cdots, X_n\}$ can be represented as a sampling realization of a stochastic process [17]. To capture the underlying dynamics

of time series, a time window is applied to derive a sequence of state vectors $Z_t = [X_{t-d+1}, \cdots, X_{t-1}, X_t]$ where d is the size of the vector. We could stack these state vector into a matrix which is called a trajectory of time series.

Definition 2.7 *Given a time series* $X = \{X_1, X_2, \cdots, X_t, \cdots\}$*, we apply a time window to derive a sequence of state vectors* $Z_t = [X_{t-d+1}, \cdots, X_{t-1}, X_t]$*, which leads to a* ***trajectory*** *of these state vectors* $T = [Z_1, Z_2, \cdots, Z_n]^T$ *[27]*

We hope to see more regularities that are implied in the trajectory space than in the originally sequential data space of the time series.

2.3 Classic Time Series Analysis and ARMA Model

As a specific realization of a stochastic process, we consider its forecast model of the form

$$y_t = f(X_t; \beta) + \epsilon_t$$

where $f(X_t; \beta)$ is a function of time t and unknown parameters β. The error term ϵ_t is often assumed to be uncorrelated, which suggests y_t to be independent as well. But in practice, this is rarely met. Therefore, we consider linear forecast models that capture the structure of time series, such as Autoregressive (AR)[8, 21, 11], Moving Average (MA) [8, 21, 14] and Autoregressive Moving Average (ARMA).

2.3.1 ARMA Models

An ARMA(p, q) model is a combination of AR(p) and MA(q) models and is suitable for univariate time series modeling [1].

Definition 2.8 *An **AR(p)** model forecasts the future value of time series by using a linear combination of p past observations, a random error ϵ and a constant term that may or may not appear for the purpose of simplicity. Mathematically the AR(p) model can be expressed as [8, 21, 11]:*

$$\hat{y}_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \tag{1}$$

Definition 2.9 *An **MA(q)** model predicts the future value of time series by using past errors as the explanatory variables. Mathematically the MA(q) model can be expressed as [8, 21, 14]*

$$\hat{y}_t = \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t \tag{2}$$

Definition 2.10 *An **ARMA(p, q)** model can be mathematically represented as*

$$\hat{y}_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q} \tag{3}$$

An AR(p) process can always be written in terms of an MA(∞) process since each of the error terms can be represented as $y_t - \hat{y}_t$ where y_t is the actual observation and

$\hat{y}_t$ is the estimation at time t. For example, AR(1) can be rewritten as

$$\begin{aligned} y_t &= \phi_1 y_{t-1} + \epsilon_t \\ &= \phi_1(\phi_1 y_{t-2} + \epsilon_{t-1}) + \epsilon_t \\ &= \phi_1^2 y_{t-2} + \phi_1 \epsilon_{t-1} + \epsilon_t = \phi_1^2(\phi_1 y_{t-3} + \epsilon_{t-2}) + \phi_1 \epsilon_{t-1} + \epsilon_t \\ &= \phi_1^3 y_{t-3} + \phi_1^2 \epsilon_{t-2} + \phi_1 \epsilon_{t-1} + \epsilon_t \\ &= \cdots \\ &= \epsilon_t + \sum_{j=1}^{\infty} \phi_1^j \epsilon_{t-j} \end{aligned}$$

Let $\phi_1^j = \theta_j \ \forall j \geq 1$. Then we have

$$\begin{aligned} y_t &= \epsilon_t + \sum_{j=1}^{\infty} \theta_j \epsilon_{t-j} \\ &= \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_n \epsilon_{t-n} + \cdots \end{aligned} \tag{4}$$

which is the form of MA(∞).

2.3.2 Autocorrelation and Partial Autocorrelation Functions

Several mathematical terms must be defined before we go forward. The expectation of random variable X is a weighted summation over all possible values of the random variable and is denoted as E(X) [22]

Definition 2.11 *For a discrete random variable X with n values, the* ***expectation***

is

$$\mathrm{E}(X) = \sum_{i=1}^{n} x_i P(x_i)$$

For a continuous random variable with its value defined on a continuous sample space I_X*, the* ***expectation*** *is defined according to*

$$\mathrm{E}(X) = \int_{I_X} x f_X(x) dx$$

Where $P(x)$ is the probability of the outcome and $f_X(x)$ is the probability density function that describes a relative likelihood of a random variable X that takes on a given value x

The covariance is a measure of dependency between two random variables and is denoted as Cov [22]

Definition 2.12 *For two discrete random variables* X *and* Y *with joint sample space* S*, the* ***covariance*** *of* X *and* Y *is*

$$\mathrm{Cov}(X, Y) = \sum_{(x,y)\in S} \sum (x - \mathrm{E}(X))(y - \mathrm{E}(Y))P(x, y)$$

And if X *and* Y *are two continuous random variables with respective continuous sample space* I_X *and* I_Y*, the* ***covariance*** *of* X *and* Y *is*

$$Cov(X, Y) = \int_{I_Y} \int_{I_X} (x - \mathrm{E}(X))(y - \mathrm{E}(Y)) f(x, y) dx dy$$

Where $P(x, y)$ is the joint probability of a pair of outcome (x, y) and $f(x, y)$ is the joint probability function of X and Y that takes given values x and y.

The variance is a measure of how far a random variable is from its expectation and is denoted as Var [22]

Definition 2.13 *For a discrete random variable X with n values, the* ***variance*** *is*

$$\mathrm{Var}(X) = \sum_{i=1}^{n} (x_i - E(X))^2 P(x_i)$$

For a continuous random variable X with its value defined on a continuous sample space I_X, the ***variance*** *of X is*

$$\mathrm{Var}(X) = \int_{I_X} (x - E(X))^2 f_X(x) dx$$

The analysis of autocorrelation and partial autocorrelation functions can be used to determine the orders p and q of an ARMA(p, q) model for a given time series.

Definition 2.14 ***Autocorrelation*** *(ACF) measures how a time series is related to a time-shifted version of itself. Let γ_k, the autovariance at lag k, be the covariance of y_t and y_{t-k} [14, 21]. Then γ_0 is the variance of the time series with itself. We define ρ_k, the autocorrelation of y_t at lag k, to be the following:*

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\mathrm{Cov}(y_t, y_{t-k})}{\mathrm{Var}(y_t)} = \frac{\mathrm{E}[(y_t - \mu)(y_{t-k} - \mu)]}{\mathrm{E}(y_t - \mu)^2} \tag{5}$$

Identification of a MA model is often best done with the ACF, as the autocorrelation is significantly non-zero only at lags involved in the MA model [5].

Definition 2.15 *Partial Autocorrelation (PACF) is used to measure the correlation between observations spaced certain number of lags apart in time after accounting for their common dependence on the intermediate measurements [1]. The 1^{st} order* ***partial autocorrelation*** *will be defined to equal the 1^{st} order autocorrelation. Mathematically the k^{th} order partial autocorrelation is:*

$$\omega^k = \frac{\mathrm{Cov}(y_t, y_{t-k} \mid y_{t-1}, y_{t-2}, \cdots, y_{t-k})}{\sqrt{\mathrm{Cov}(y_t, y_t \mid y_{t-1}, y_{t-2}, \cdot, y_{t-k})\ \mathrm{Cov}(y_{t-k}, y_{t-k} \mid y_{t-1}, y_{t-2}, \cdots, y_{t-k})}} \tag{6}$$

Identification of an AR model is often best done with the PACF, as the number of non-zero partial autocorrelation gives the order of the AR model [5].

To build a proper ARMA(p, q) model, a time series requires stationary property.

Definition 2.16 *A time series is* ***stationary*** *when statistics such as mean and variance of data do not depend on time index [3].*

There are two types of stationary processes: Strongly Stationary and Weakly Stationary.

Definition 2.17 *A process* $\{x(t), t = 0, 1, 2, \cdots\}$ *is* ***Strongly Stationary*** *if the joint probability distribution function of* $x_{t-s}, x_{t-s+1}, \cdots, x_t, \cdots, x_{t+s-1}, x_{t+s}$ *is independent of* t *for all* s *[21, 14].*

Definition 2.18 *A process is said to be* ***Weakly Stationary*** *of order* k *if the statistical moments of the process up to that order depend only on time differences and not upon the time of occurrences of the data being used to estimate the moments [21, 14].*

ARMA(p, q) models are straight-forward and simple. However, sometimes there is no information in either the ACF or the PACF. Thus it can be difficult to estimate ARMA coefficients through ACF and PACF inspection, and sometimes time series is non-stationary if it violates those stationary assumptions. To overcome these drawbacks, several computational algorithms have been proposed in the literature [12, 2, 7, 25]. Logistic regression is one of the algorithms that works well on time series data [26, 6].

2.4 Time Series Forecasting Using Binary Logistic Regression

Logistic regression is widely used to model the outcome of a categorical dependent variable [15]. Logistic regression is often preferred to linear regression for numerous reasons:

1. Linear regression assumes errors to be normally distributed but sometimes errors are not normally distributed.

2. Linear regression maps data to continuous real numbers but sometimes the response variable is categorical

3. In linear regression, parameters are estimated via minimizing the sum of squared errors. However, in logistic regression, maximum likelihood estimation (MSE) is used to solve for the parameters to best fit the time series.

2.4.1 Binary Simple Logistic Regression Model

Binary SLR deals with situations in which the response variable has only 2 possible outcomes (e.g., 0 or 1). Suppose we have a binary response variable $Y \in \{1, 0\}$ and some independent features $X = (X_1, \cdots, X_k)$. Let each (y, X) be an independent observation at time $t = (1, \cdots, n)$ and denoted as (X_t, y_t). By convention, Y is said to be a "success" if it has a value of 1 and a "failure" otherwise (usually denoted by either 0 or -1). Let $N_{y=0}$ represent the number of failures and $N_{y=1}$ be the number of successes. Suppose we have k independent random variables (i.e., $X = X_1, \cdots, X_k$) and let $\pi = P(Y \mid X)$ be the probability of success or failure for a given observation with features $X = (x_1, \cdots, x_k)$. There is a column vector of length k such that for

each observation there is a corresponding relationship between each linear component $c + \sum_{i=1}^{k} x_i\beta_i$ and the probability of success (π_i) of this observation where c is a constant.

Specifically, the SLR model equates the logit transform or the log-odds of the probability of a success, to the linear component:

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + x_1\beta_1 + x_2\beta_2 + \cdots + x_k\beta_k \tag{7}$$

$\frac{\pi}{1-\pi}$ is called the odds, which is the probability of success divided by the probability of failure. The logit is the log of the odds, $\log(\pi/(1-\pi))$. The $\beta_i, i \in \{0, \cdots, k\}$ are log-odds ratios. A positive value of β_i suggests an increased likelihood of "success" as the increment of feature x_i.

Definition 2.19 ***Log-odds ratio*** *is the logarithm of the odds ratio.* ***Odds ratio*** *(OR) represents the odds given the presence of a particular feature compared to the odds given the absence of that feature [18].*

Mathematically, for any $j \in 1, 2, \cdots, k$, the β_j can be represented as:

$$\begin{aligned}
\beta_j &= (\beta_0 + x_1\beta_1 + \cdots + 1\beta_j + \cdots + x_k\beta_k) - (\beta_0 + x_1\beta_1 + \cdots + 0\beta_j + \cdots + x_k\beta_k) \\
&= \text{logit}(\pi(x_j = 1)) - \text{logit}(\pi(x_j = 0)) \\
&= \log\left(\frac{\pi(x_j = 1)}{1 - \pi(x_j = 1)}\right) - \log\left(\frac{\pi(x_j = 0)}{1 - \pi(x_j = 0)}\right) \\
&= \log\left(\frac{\pi(x_j = 1)/(1 - \pi(x_j = 1))}{\pi(x_j = 0)/(1 - \pi(x_j = 0))}\right) \\
&= \log(OR)
\end{aligned} \tag{8}$$

Hence the OR of x_j is obtained by taking the exponential of β_j. SLR model also works for dependent variables having three or more categorical levels, but in this paper, we only focus on time series analysis in the context of binary response variables.

2.4.2 Maximum Likelihood Estimation of SLR

The goal of SLR is to find the proper values of the k parameters $\beta_1, \beta_2, \cdots, \beta_k$ such that the joint probability of obtaining the observed data is the greatest. Responses coming from distinct combinations of features are assumed to be from different populations. For example, older people compared to younger people, people with hypertension, compared to people without may have higher chance of "success" in diabetes.

Given a dataset with a sample size of M, let N represent the population size and let n_i denote the sample size from the i^{th} population such that $M = \sum_i^N n_i$. Let y_i represent the observed counts of the number of success in the i^{th} population. Since there are $\binom{n_i}{y_i}$ numbers of ways to arrange y_i success among n_i trails in each group, the joint probability density function of Y

$$f(y \mid \beta) = \prod_{i=1}^{N} \binom{n_i}{y_i} \pi_i^{y_i} (1 - \pi_i)^{n_i - y_i} \tag{9}$$

The joint probability density function in (9) expresses the values of y as a function of known, fixed values for β. But in reality, β is the unknown parameter whereas y is given. Therefore, the likelihood density function is of the same form as the probability

density function except that the parameters are conditioned upon the response

$$L(\beta \mid y) = \prod_{i=1}^{N} \binom{n_i}{y_i} \pi_i^{y_i} (1-\pi_i)^{n_i - y_i} \tag{10}$$

Since none of the π_i is involved in the binomial coefficients, we can treat each $\binom{n_i}{y}$ as a constant that can be ignored in maximizing the equation. The equation can be thus written as

$$L(\beta \mid y) \propto \prod_{i=1}^{N} \pi_i^{y_i} (1-\pi_i)^{n_i - y_i} = \prod_{i=1}^{N} \left(\frac{\pi_i}{1-\pi_i} \right)^{y_i} (1-\pi_i)^{n_i} \tag{11}$$

By exponentiation both sides of (7), we have

$$\left(\frac{\pi_i}{1-\pi_i} \right) = e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}$$

and

$$\pi_i = \left(\frac{e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}}{1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}} \right)$$

Substituting the above equations for (11) and we get the following representation for the likelihood function

$$L(\beta \mid y) \propto= \prod_{i=1}^{N} \left(e^{y_i(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)} \right) \left(1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)} \right)^{-n_i} \tag{12}$$

This is the kernel of the likelihood function to maximize [15].

Since the logarithm is monotonic, the maximum of the likelihood function will also generate the maximum of the log likelihood function. After applying the logarithm on both sides, the log likelihood function is

$$l(\beta) \propto= \sum_{i=1}^{N} y_i \left(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j \right) - n_i \cdot \log \left(1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)} \right) \tag{13}$$

It is beyond my ability to explain in this thesis, but it can be shown [33] that the log-likelihood has a global maximum at points where the first derivative with respect to each β simultaneously equal to zero

$$\begin{aligned}
\frac{\partial l(\beta)}{\partial \beta_j} &= \sum_{i=1}^{N} y_i x_{ij} - n_i \cdot \frac{1}{1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}} \cdot \frac{\partial}{\partial \beta_j} \left(1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}\right) \\
&= \sum_{i=1}^{N} y_i x_{ij} - n_i \cdot \frac{1}{1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}} \cdot e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)} \cdot \frac{\partial}{\partial \beta_j} \left(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j\right) \\
&= \sum_{i=1}^{N} y_i x_{ij} - n_i \cdot \frac{1}{1 + e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)}} \cdot e^{(\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j)} \cdot x_{ij} \\
&= \sum_{i=1}^{N} y_i x_{ij} - n_i \pi_i x_{ij}
\end{aligned} \tag{14}$$

The maximum likelihood estimates for β_j for $j \in \{1, \cdots, k\}$ can be found by setting each of the j equations in Eq. 14 equal to zero.

3.0.1 MILR Implementation

SLR model has the form

$$\log\left(\frac{P(Y=1|X)}{P(Y=0|X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k \tag{15}$$

Instead of taking the dot product of vector X and β to predict for the log-odds ratio of a particular event, MILR employs a trajectory of time series and introduces an additional multiplicative vector of parameters, δ. MILR model has the form

$$\log\left(\frac{P(Y=1|T)}{P(Y=0|T)}\right) = \beta_0 + \delta^T T\beta \tag{16}$$

with δ in $\mathbb{R}^l$, β in $\mathbb{R}^k$, and T in $\mathbb{R}^{l\times k}$ of the form

$$T = \begin{bmatrix} X_1 & X_2 & \cdots & X_k \\ X_2 & X_3 & \cdots & X_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ X_l & X_{l+1} & \cdots & X_{l+k-1} \end{bmatrix} \quad \delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_l \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

3.0.2 Why MILR

We hope MILR will capture more regularities in a given time series than SLR for the following reasons:

First we note each parameter of β in SLR model can be obtained through combinations of δ and β in MILR. For example, when we set $\delta = \begin{bmatrix} 0 & 1 & 0 & \cdots \end{bmatrix}$, both β_1 in MILR and β_2 in SLR are used to estimate the log-odds ratio of X_2.

Secondly, if we label the response variable as either 1 or -1, then for any binary logistic regression, we have

$$\pi(Y=1) = \frac{1}{1+\exp(-\beta^T X)}$$

and

$$\begin{aligned}\pi(Y=-1) &= 1-\pi(Y=1)\\ &= 1-\frac{1}{1+\exp(-\beta^T X)}\\ &= \frac{\exp(-\beta^T X)}{1+\exp(-\beta^T X)}\\ &= \frac{1}{1+\exp(\beta^T X)}\end{aligned}$$

Therefore we can model

$$\pi(y_i=\pm 1)=\frac{1}{1+\exp(-y_i\beta^T X)}$$

With a sample size of N, the log-likelihood function can be written as

$$l(X,Y,\beta)=\sum_i^N \log\frac{1}{1+\exp(-y_i\cdot\beta^T X_i)}=\sum_i^N -\log(1+\exp(-y_i\cdot\beta^T x_i)) \qquad (17)$$

Therefore to maximize (17) is equivalent to minimize $\sum_i^N \log(1+\exp(-y_i\cdot(\beta^T X_i))$, which is the form of logistic loss.

Suppose we select k features and implement a SLR model. β is then in $\mathbb{R}^k$ and X is in $\mathbb{R}^k$. The goal of SLR is to minimize

$$f(\beta,\beta_0)=\sum_{i=1}^N \log(1+\exp(-y_i(\beta_0+\beta^T X_i)))$$

By constructing a L by K evolution trajectory of this time series and implementing a MILR model, we have the logistic loss of MILR to be

$$f(\delta,\beta,\beta_0)=\sum_{i=1}^N \log(1+\exp(-y_i(\delta^T T_i\beta+\beta_0)))$$

By setting $\delta=\begin{bmatrix}1 & 0 & 0 & \cdots\end{bmatrix}$, the objective function changes to

$$f(\beta,\beta_0)=\sum_{i=1}^N \log(1+\exp(-y_i(\beta_0+\beta^T X_i))),$$

which has the same form of SLR objective function. Since this is only one of the many possible values of δ, it is guaranteed that there exists δ^* such that $f(\delta^*, \beta, \beta_0) \leq f(\beta, \beta_0)$. Therefore, MILR produces lower loss than SLR on the same data set.

Lastly, MILR yields smaller variance of errors in comparison to SLR implemented on the same data set with $\delta \in \mathbb{R}^l, \beta \in \mathbb{R}^k, X \in \mathbb{R}^k, T \in \mathbb{R}^{l \times k}$. With an initial value of $\delta = \begin{bmatrix} \frac{1}{l} & \frac{1}{l} & \cdots & \frac{1}{l} \end{bmatrix}$, the marginal value $\delta^T T \beta$ becomes:

$$\beta_1 \left(\frac{X_1 + X_2 + \cdots + X_l}{l} \right) + \cdots + \beta_k \left(\frac{X_k + X_{k+1} + \cdots + X_{l+k-1}}{l} \right)$$

Since $X_1, X_2, \cdots, X_{l+k-1}, \cdots, X_N$ are identically distributed random variables from time series, each random variable has the same mean and variance. Mathematically, we have

$$\mathbf{E}(X_i) = \mu \qquad \mathbf{Var}(X_i) = \sigma^2 \qquad \text{for all } i = 1, 2, \cdots, N$$

Let $\bar{X}_1 = \left(\frac{1}{l}\right)(X_1 + X_2 + \cdots + X_l)$, $\bar{X}_2 = \left(\frac{1}{l}\right)(X_2 + X_3 + \cdots + X_{l+1})$ and so on. The mean of $\bar{X}_i$ for all $i = 1, 2, \cdots, N$ is:

$$\begin{aligned}
\mathbf{E}(\bar{X}_i) &= \mathbf{E}\left[\left(\frac{1}{l} \right) (X_i + X_{i+1} + \cdots + X_{i+l-1}) \right] \\
&= \frac{1}{l} \mathbf{E}(X_i + X_{i+1} + \cdots + X_{i+l-1}) \\
&= \frac{1}{l} (\mathbf{E}(X_i) + \mathbf{E}(X_{i+1}) + \cdots + \mathbf{E}(X_{i+l-1})) \\
&= \frac{1}{l} (\mu + \mu + \cdots + \mu) \\
&= \frac{l}{l} \mu = \mu
\end{aligned}$$

the variance of $\bar{X}_i$ for all $i = 1, 2, \cdots, N$ is:

$$\begin{aligned}
\mathbf{Var}(\bar{X}_i) &= \mathbf{Var}\left[\left(\frac{1}{l}\right)(X_i + X_{i+1} + \cdots + X_{i+l-1})\right] \\
&= \left(\frac{1}{l}\right)^2 \mathbf{Var}(X_i + X_{i+1} + \cdots + X_{i+l-1}) \\
&= \frac{1}{l^2}(\mathbf{Var}(X_i) + \mathbf{Var}(X_{i+1}) + \cdots + \mathbf{Var}(X_{i+l-1})) \\
&= \frac{1}{l^2}(\sigma^2 + \sigma^2 + \cdots + \sigma^2) \\
&= \frac{l\sigma^2}{l^2} = \frac{\sigma^2}{l}
\end{aligned}$$

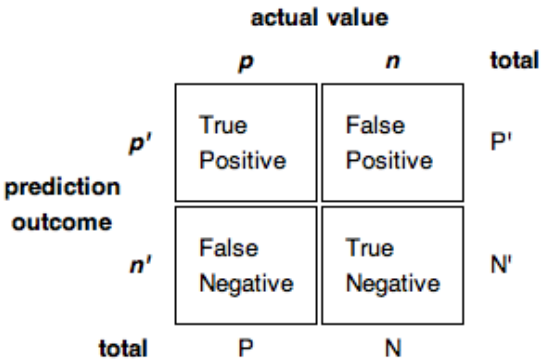
Since the average of X has the noise with the same mean and smaller standard deviation, i.e., $\sigma/\sqrt{l}$. MILR can do no worse than SLR on averaged data because of the smaller variance of noise fitting. Therefore, MILR model should perform at least no worse than SLR model in theory.

3.0.3 Receiver Operating Characteristic (ROC) Curve

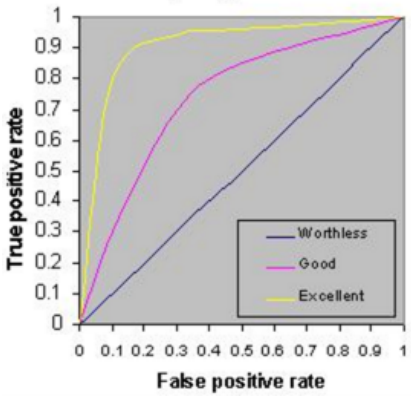
To test the validity of a forecasting model, it is fundamental to predict unused data accurately from the same population. To test the performance of a classifier, we draw the ROC curve and look at the area under curve (AUC) value.

As shown in Fig. 1a, TPR is also called sensitivity, the chance of accurately predicting the occurrence of "1" or success. While FPR is the percentage of making inaccurate prediction of "0" or failure, FNR is the complement of sensitivity, i.e., 1 - TPR. TNR is also called specificity which is the proportion of unsuccessful cases that are correctly classified as failures. We could obtain Hit rate or accuracy from Fig. 1a

$$\text{Hit Rate } = \frac{\text{TPR} + \text{TNR}}{\text{TPR} + \text{FPR} + \text{FNR} + \text{TNR}}$$



(a) TPR and FPR Table



(b) Comparing ROC Curves

Figure 1: Analysis of ROC Curve, Figures Extracted From [9]

TPR and FPR affect each other and are both under the influence of thresholds. All data is predicted to be "1" if threshold = 0. While we want TPR to be as large as possible and FPR to be as small as possible, we wouldn't see the efficiency of such a classification with all data are categorized to be "1" with TPR = 1 and FPR = 1 in this case. With the increment of threshold up to 1, there is less data predicted to be "1" and both TPR and FPR decrease down to 0. Overall, we want to increase TPR with less increase of FPR and hence drawing the ROC curve to visualize their relationship (see Fig. 1b). The more concave the curve, the better the classifier. An area of 0.5 under the curve represents a completely random classifier in which predictions are made via a coin toss. While AUC greater than 0.8 would be considered good.

4 APPLICATION: DATA AND RESULTS

This section depicts the performance of both SLR and MILR on three simulated noisy sine wave data and samples from the Standard & Poor's 500 historical data. The descriptions of data and results of model comparisons are presented with commentary.

In the analyses of simulated data, we start describing how we generate samples of sine wave for different intensities of Gaussian noise. Next we apply SLR model to a sequence of 3 lagged values and predict signs (i.e., $\pm$) of the pure sine wave of the same frequencies and amplitudes. Then by converting sequential data representation with a 3 by 3 trajectory matrix, we employ MILR to see if it outperforms SLR on the 2 important measures of forecast abilities: hitting rates of testing data and areas under the curve (AUC) values. The results of three simulated data analyses validate the applicability of MILR and inspire our investigation of a more complex, real stock data.

4.1 Analysis of Simulated Data

We sample 1500 data from a simulated lagged series of sine wave with a period of 400 and an amplitude of 1 (i.e., $\sin \frac{\pi t}{200}$). The first two-thirds is used to train a model and the rest is used to score the trained model. Responses are the signs of each data point. To investigate noisy measurements of SLR and MILR models and capture the randomness of real data, we add three different intensities $(20\%, 50\%, 80\%)$ of Gaussian noise to the signal. In SLR model, a sequence of 3 lagged values serves as the independent variable. In MILR, we create a 3 by 3 trajectory matrix on the basis of sequential data and plug it into the model. Two important measures of forecast

abilities, hitting rates of testing data and areas under the curve (AUC) values, are employed to test if MILR outperforms SLR.

4.1.1 Sine Wave with 20% Gaussian White Noise

Examining the time series plot in Figure 2. The straight line colored in black is the response variable, Direction. There is a clear shape of sine wave with little variation of the form.

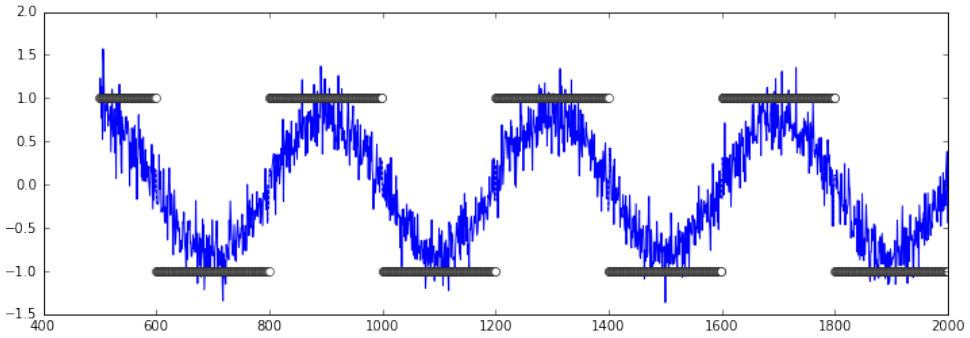


Figure 2: Time Series Plot of Sine Wave with 20% Gaussian White Noise

The organization and layout of a sample data set is shown in Table 1. Direction is the sign of current data point from noise free sine wave and Current is the corresponding value from the noisy data.

Table 1: Data Structures and Variables – Sine Wave with 20% Gaussian Noise

Index	Current	Lag1	Lag2	Lag3	Lag4	Lag5	Direction
1392	0.040	0.201	0.388	-0.128	0.285	-0.307	1.0
1393	0.350	0.040	0.201	0.388	-0.128	0.285	1.0
1394	0.482	0.350	0.040	0.201	0.388	-0.128	1.0
1395	0.187	0.482	0.350	0.040	0.201	0.388	1.0
1396	-0.161	0.187	0.482	0.350	0.040	0.201	1.0

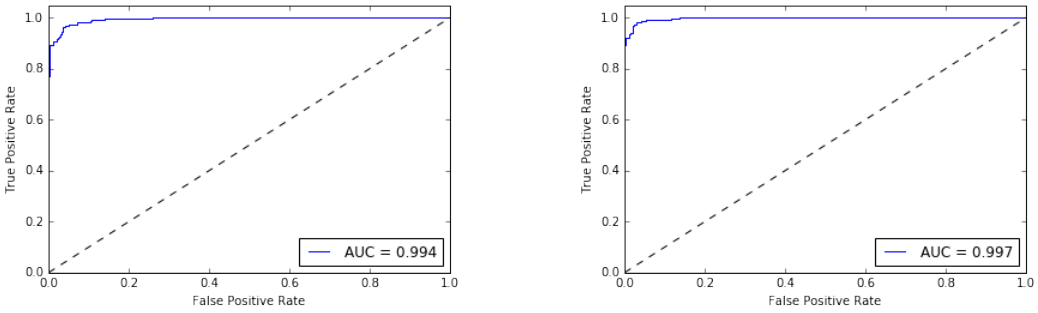
Because of the inadequate proportion of Gaussian noise (20%), they should match each other (i.e., Current $> 0 \Rightarrow$ Direction $= 1.0; \text{Current} < 0 \Rightarrow \text{Direction} = -1.0$) most of the time with few exceptions (see data at index 1396 for example). Lag1 $\sim$ Lag5 are the lagged values of Current variable from the noisy sine wave.

Overall measures of accuracies and AUC values are summarized in Table 2. Hit rates of both models are very good ($> 95\%$) on both training and testing data sets.

Table 2: Model Performance – Sine Wave with 20% Gaussian Noise

Data	Training Set		Testing Set	
Statistics	Hit rate	ROC	Hit rate	ROC
SLR	0.958	0.996	0.96	0.994
MILR	0.971	0.997	0.974	0.997

Indicated by AUC values, both models are considered to be excellent classifiers (AUC: 0.9 $\sim$ 1.0). The ROC curves for different trained models applied to forecast testing set are presented in Figure 3a and Figure 3b



(a) SLR (b) MILR

Figure 3: ROC Curve – Sine Wave with 20% Gaussian Noise

MILR slightly outperforms SLR in all measurements of model performance but both work well with simulated data with light noise.

4.1.2 Sine Wave with 50% Gaussian White Noise

Examining the time series plot in Figure 4. The straight line colored in black is the response variable, Direction. We can still see the form of a sinusoid with various length of spikes reaching out of the waveform.

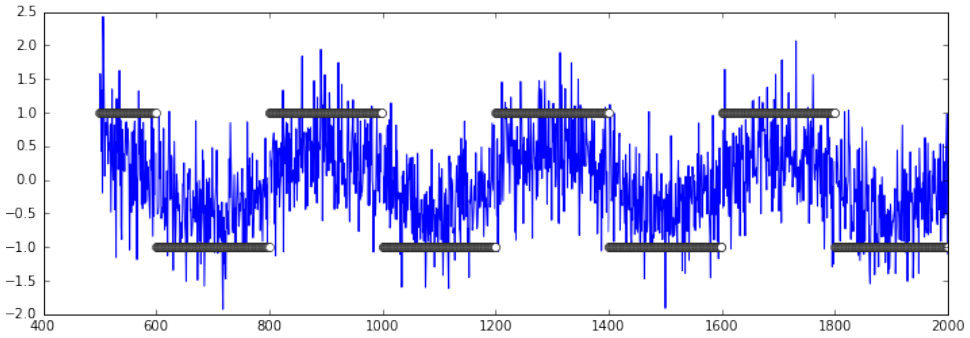


Figure 4: Time Series Plot of Sine Wave with 50% Gaussian White Noise

The organization and layout of a sample data set is shown in Table 3. As we add more noise to the signal, we see more cases of mismatch between signs of Current and Direction. However, we do not see values bouncing around zero that quickly.

Table 3: Data Structures and Variables – Sine Wave with 50% Gaussian Noise

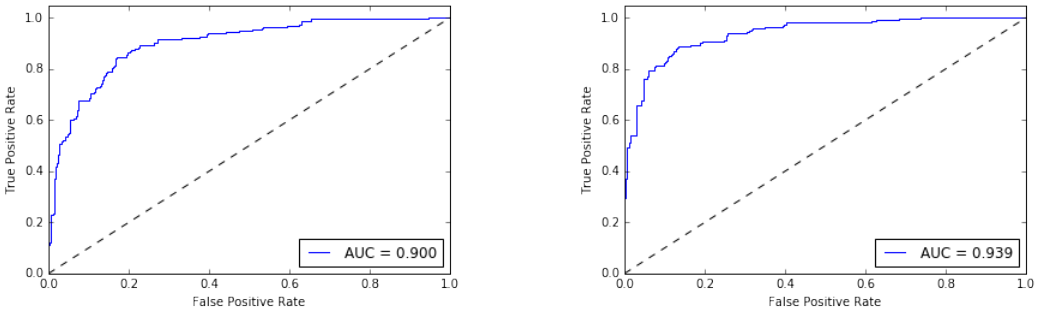
Index	Current	Lag1	Lag2	Lag3	Lag4	Lag5	Direction
628	0.290	0.360	-0.255	-0.295	0.592	1.020	-1.0
629	-0.679	0.290	0.360	-0.255	-0.295	0.592	-1.0
630	-1.344	-0.679	0.290	0.360	-0.255	-0.295	-1.0
631	-0.468	-1.344	-0.679	0.290	0.360	-0.255	-1.0
632	0.571	-0.468	-1.344	-0.679	0.290	0.360	-1.0

Overall measures of accuracies and AUC values are summarized in Table 4. Even with half as much noise, both SLR and MILR still generate over 80% hit rates in both training and testing data sets.

Table 4: Model Performance – Sine Wave with 50% Gaussian Noise

Data Statistics	Training Set		Testing Set	
	Hit rate	ROC	Hit rate	ROC
SLR	0.840	0.916	0.826	0.900
MILR	0.872	0.950	0.868	0.939

Indicated by AUC values, both models are still considered to be excellent classifiers (AUC: 0.9 $\sim$ 1.0). The ROC curves for different trained models applied to forecast testing set are presented in Figure 5a and Figure 5b



(a) SLR (b) MILR

Figure 5: ROC Curve – Sine Wave with 50% Gaussian Noise

Hit rate is improved by 5% with the implementation of MILR.

4.1.3 Sine Wave with 80% Gaussian White Noise

Examining the time series plot in Figure 6. The straight line colored in black is the response variable, Direction. We can barely see the sinusoid in the plot. Values fluctuate back and force between positive and negative values.

The organization and layout of a sample data set is shown in Table 5. With high noise-to-signal ratio, signs of noisy data can no longer be reliable to indicate signs of

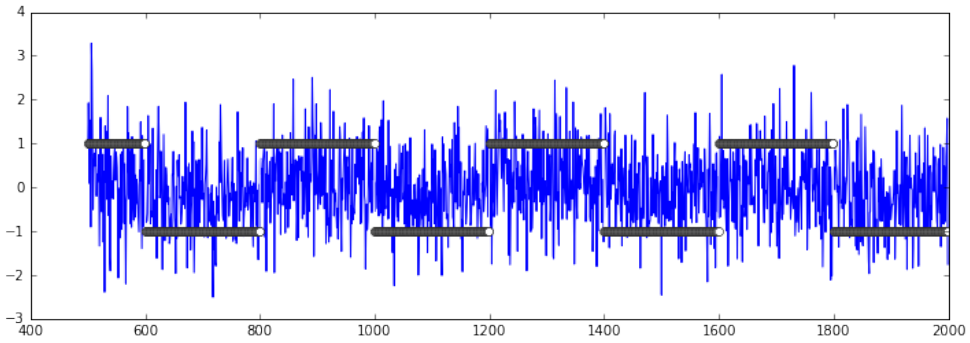


Figure 6: Time Series Plot of Sine Wave with 80% Gaussian White Noise

value from pure sine wave. Also, it becomes almost impossible to indicate if the next data point is above or below zero by the value of current data point.

Table 5: Data Structures and Variables – Sine Wave with 80% Gaussian Noise

Index	Current	Lag1	Lag2	Lag3	Lag4	Lag5	Direction
1974	0.913	0.368	0.328	-0.734	-0.425	-0.984	-1.0
1975	-0.097	0.913	0.368	0.328	-0.734	-0.425	-1.0
1976	0.443	-0.097	0.913	0.368	0.328	-0.734	-1.0
1977	0.701	0.443	-0.097	0.913	0.368	0.328	-1.0
1978	-0.048	0.701	0.443	-0.097	0.913	0.368	-1.0

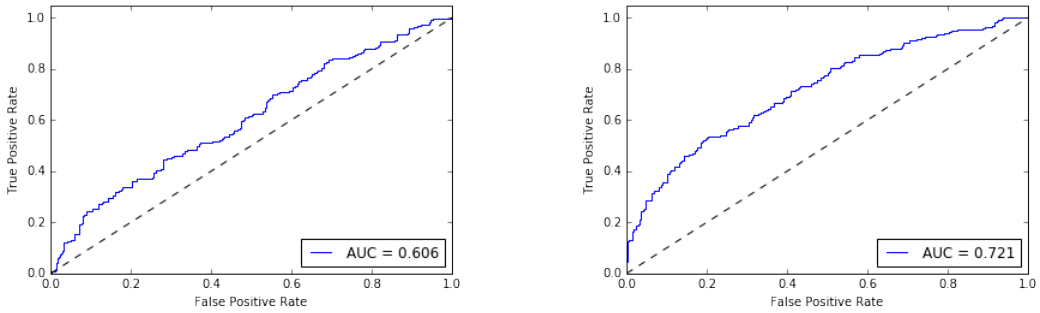
Overall measures of accuracies and AUC values are summarized in Table 6. There are still 70% values from the training set and 66% from the testing set that can be correctly classified by MILR model.

Table 6: Model Performance – Sine Wave with 80% Gaussian Noise

Data	Training Set		Testing Set	
Statistics	Hit rate	ROC	Hit rate	ROC
SLR	0.628	0.673	0.570	0.606
MILR	0.703	0.775	0.660	0.721

Indicated by AUC values, SLR almost fails to work as a classifier (AUC: $0.5 \sim 0.6$).

The hit rate is improved by more than 10% with the implementation of MILR. The ROC curves for different trained models applied to forecast testing set are presented in Figure 7a and Figure 7b



(a) SLR (b) MILR

Figure 7: ROC Curve – Sine Wave with 80% Gaussian Noise

Throughout analyses of three simulated data with various intensities of Gaussian noise, we find the MILR is more advantageous to SLR when the signal to noise ratio decreases. The results of simulated data validate the applicability of MILR and inspire our investigation of a more complex, real stock data.

4.2 Analysis of Standard & Poor's 500 Historical Data

The Standard & Poor's 500 index data is obtained from Yahoo Finance. Standard & Poor's 500 is a weighted index of the 500 publicly traded corporations in the US stock market [4]. Standard & Poor's 500 index is the asset basis of many market derivative products. We try to predict the signs of the percentage return of today based on the price information of the past. The formula for percentage return is the difference of closing price between today and yesterday divided by yesterday's closing price. If the percentage return is positive, the stock price increases that day and if

the percentage return is negative, the investment on the stock loses its value.

To make our study comparable to what Michael Halls-Moore did [19], we analyse the same training and testing data. Specifically, the data before January 1st, 2005 and after January 1st, 2001 is training set and the data in year 2005 is used for testing our models. We have a total of 996 training data and 250 testing data. As for SLR model, settings of variables and parameters remain the same as it is in Michael's work. In MILR, a 4 by 2 trajectory matrix whose first row is the features inputs in SLR serves as the independent variable.

The general trend of adjusted closing price is shown in Figure 8. Points colored in red have positive percentage returns. There is a general decreasing trend before 2003 and an upward trend after 2003. No clear cycles and seasonal effects are seen in the plot.

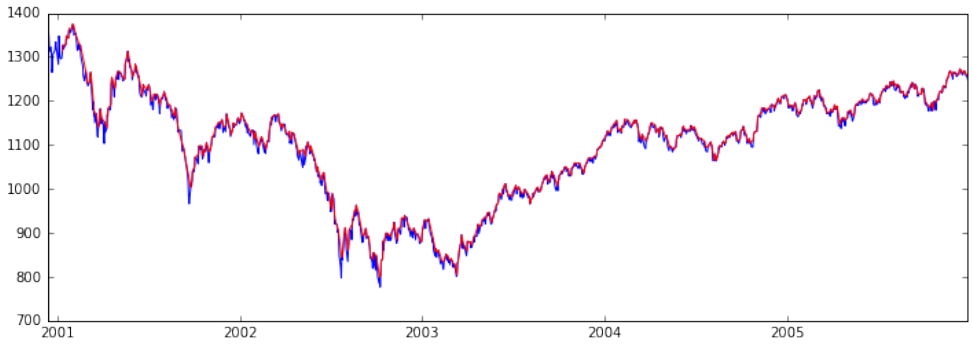


Figure 8: Time Series Plot of Standard & Poor's 500 Historical Data

The organization and layout of a sample data set is shown in Table 7. We could not see a clear pattern for telling the signs of current value by the values of the past.

Table 7: Data Structures and Variables – Standard & Poor's 500 Index

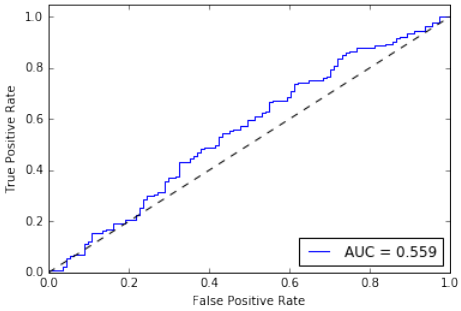
Date	Current	Lag1	Lag2	Lag3	Lag4	Lag5	Direction
2001-01-10	0.959	0.381	-0.192	-2.624	-1.055	5.010	1.0
2001-01-11	1.032	0.959	0.381	-0.192	-2.624	-1.055	1.0
2001-01-12	-0.623	1.032	0.959	0.381	-0.192	-2.624	-1.0
2001-01-16	0.614	-0.623	1.032	0.959	0.381	-0.192	1.0
2001-01-17	0.213	0.614	-0.623	1.032	0.959	0.381	1.0

Overall measures of accuracies and AUC values are summarized in Table 8. Both models seem to be under-fitted as the training accuracy is moderately lower than testing accuracy.

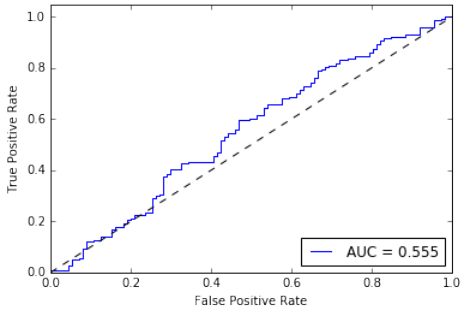
Table 8: Model Performance – Standard & Poor's 500 Stock

Data	Training Set		Testing Set	
Statistics	Hit rate	ROC	Hit rate	ROC
SLR	0.516	0.527	0.560	0.559
MILR	0.514	0.527	0.579	0.555

Although by the standard of AUC values, both models fail to identify a correct class, MILR outperforms SLR by its improved testing accuracy (58% vs 56%). The ROC curves for different trained models applied to forecast testing set are presented in Figure 9a and Figure 9b



(a) SLR



(b) MILR

Figure 9: ROC Curve – Standard & Poor's 500 Stock

5 DISCUSSION

The work is done with the aim of providing better tools for financial data forecasting. Recent years have seen the growing use of forecasting algorithms in analyzing financial time series data [12, 2, 7, 8, 25]. It is quite important to capture the regular pattern among the overall uncertain financial markets. Existent stationary time series models like ARMA can hardly tell apart the noisy inputs and reliable predictions. Simple logistic regression (SLR) model, as an example of a generalized linear model, is found to be very useful in the context of non-stationary time series [13].

In this paper, we focus on developing a generalized scope of SLR by introducing an extension of multi-index representation of sequential data. We call this model MILR (multi-indexed logistic regression). MILR, compared to SLR, is more versatile (e.g., incorporating SLR as a special case) and informative (e.g., averaging away uncertainties). Our research is driven to support the application of MILR.

Three simulated data with various intensities of Gaussian random noise are used to test our hypothesis. Results consistently support our prediction that MILR achieves better forecast power than SLR especially when the data is more noisy. Moving on to test the validity of MILR in forecasting real financial stock data, we expect to see a similar effect of MILR, compared to SLR, on increasing the predictive performance. This effect could probably be smaller since real data is more unpredictable and less assumptions we made in simulation are met. Although both SLR and MILR are seen to be poor classifiers by the standard of AUC score, MILR outperforms SLR with higher hit rates (58% vs 56%).

We believe the insight and methodology used in our research could benefit future

work in the field of multi-dimensional data and time series. This study only examines simulated Gaussian noisy data and volatile stock data. More varieties of time series data (e.g., Natural events, electroencephalogram (EEG) recordings, industrial production indices, etc.) can be tested to verify our hypothesis and to refine MILR. Also the length of δ can be arbitrary and more research questions should be asked in regards to the impact of δ on the performance of MILR. Furthermore, The relationship of MILR with other important mathematical algorithms (e.g., Singular Value Decomposition, Markov Chains, etc) can be explored.

BIBLIOGRAPHY

[1] R. Adhikari and R. Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[2] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010.

[3] R. M. Alvarez, J. N. Katz, et al. Aggregation and dynamics of survey responses: The case of presidential approval. *docu mento de trabajo, California, Institute of Technology*, 2000.

[4] C. Ang. *Analyzing Financial Data and Implementing Financial Models Using R*. Springer, 2015.

[5] S. A. Baker and B. M. Iqelan. Comparative approach of artificial neural network and arima models on births per month in gaza strip using r. 2015.

[6] D. J. Barr and A. F. Frank. Analyzing multinomial and time-series data. In *Workshop on ordinary and multilevel modeling at 2009 CUNY conference on sentence processing, UC Davis*, 2009.

[7] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne. Machine learning strategies for time series forecasting. In *Business Intelligence*, pages 62–77. Springer, 2013.

[8] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[9] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[10] F. Briggs, P. Ramsay, E. Madden, J. Norris, V. Holers, T. R. Mikuls, T. Sokka, M. F. Seldin, P. Gregersen, L. A. Criswell, et al. Supervised machine learning and logistic regression identifies novel epistatic risk factors with ptpn22 for rheumatoid arthritis. *Genes and immunity*, 11(3):199–208, 2010.

[11] L. D. Brown, R. L. Hagerman, P. A. Griffin, and M. E. Zmijewski. Security analyst superiority relative to univariate time-series models in forecasting quarterly earnings. *Journal of Accounting and Economics*, 9(1):61–87, 1987.

[12] L.-J. Cao and F. E. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *Neural Networks, IEEE Transactions on*, 14(6):1506–1518, 2003.

[13] Y. Chang, B. Jiang, and J. Y. Park. Nonstationary logistic regression1. 2005.

[14] J. H. Cochrane. Time series for macroeconomics and finance. *Manuscript, University of Chicago*, 2005.

[15] S. A. Czepiel. Maximum likelihood estimation of logistic regression models: theory and implementation. *Available at czep. net/stat/mlelr. pdf*, 2002.

[16] S. Dreiseitl and L. Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5):352–359, 2002.

[17] P. A. Griffin. The time-series behavior of quarterly earnings: preliminary evidence. *Journal of Accounting Research*, pages 71–83, 1977.

[18] S. M. Hailpern and P. F. Visintainer. Odds ratios and logistic regression: further examples of their use and interpretation. *interpretation*, 318(134):0–356, 2003.

[19] M. Halls. Forecasting financial time series - part i, 2014.

[20] J. Hao. Input selection using mutual information–applications to time series prediction. *Helsinki University of Technology, MS thesis, Dep. of Computer Science and Engineering*, 2005.

[21] K. W. Hipel and A. I. McLeod. *Time series modelling of water resources and environmental systems*, volume 45. Elsevier, 1994.

[22] R. M. Howard. *A Signal Theoretic Introduction to Random Processes*. John Wiley & Sons, 2015.

[23] D. Jain and B. Jhunjhunwala. Chapter 8: Analysis of time series. *Business Statistics for B. Com (Hons)*, 2007.

[24] A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[25] D. M. Kline and G. Zhang. Methods for multi-step time series forecasting with neural networks. *Neural networks in business forecasting*, pages 226–250, 2004.

[26] K.-Y. Liang and S. L. Zeger. A class of logistic regression models for multivariate binary time series. *Journal of the American Statistical Association*, 84(406):447–451, 1989.

[27] B. Liu, H. Chen, A. Sharma, G. Jiang, and H. Xiong. Modeling heterogeneous time series dynamics to profile big sensor data in complex physical systems. In *Big Data, 2013 IEEE International Conference on*, pages 631–638. IEEE, 2013.

[28] I. B. MacNeill and G. Umphrey. *Time Series and Econometric Modelling: Advances in the Statistical Sciences: Festschrift in Honor of Professor VM Joshis 70th Birthday*, volume 3. Springer Science & Business Media, 2012.

[29] S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork. Comparison of different methods for univariate time series imputation in r. *arXiv preprint arXiv:1510.03924*, 2015.

[30] W. Shankle, S. Mani, M. Pazzani, and P. Smyth. Use of a computerized patient record database of normal aging and very mildly demented subjects to compare classification accuracies obtained with machine learning methods and logistic regression. *Computing Science and Statistics*, pages 201–209, 1997.

[31] X.-C. Tai, K.-A. Lie, T. F. Chan, and S. Osher. *Image Processing Based on Partial Differential Equations: Proceedings of the International Conference on PDE-Based Image Processing and Related Inverse Problems, CMA, Oslo, August 8-12, 2005*. Springer Science & Business Media, 2006.

[32] F. Transforms. Engineering mathematics–iii.

[33] N. Wang, X. Qi, J. Zhang, and Q. Xu. Location optimization problem with multi refineries and multi stations. 2015.

APPENDIX: Python Code

```
***Python Package Import***
import pandas as pd
from pandas import DataFrame
from pandas.io.data import DataReader
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.linear_model import LogisticRegression
import theano
import theano.tensor as T
***Loading data***
# Create global data that variable in function can be used
global NofLag; NofLag = 3
global stock; stock = "^GSPC" #S&P 500
# Input is a stock data between 2001 and 2015 year
# Dependent variabe is adjusted closing price of today
snpret, ts_adj_close = create_lagged_series(stock, datetime.
    datetime(2001,1,10), datetime.datetime(2005,12,31), lags=
    NofLag)
# print (snpret[:5].to_latex(float_format='%.4f', bold_rows =
```

```
    False))
# Training and testing data
X = snpret[["Lag%d" %i for i in range(1,NofLag+1)]]
y = snpret["Direction"]
start_test = datetime.datetime(2005,1,1)
X_train = X[X.index < start_test]
X_test = X[X.index >= start_test]
global y_train; y_train = y[y.index < start_test]
global y_test; y_test = y[y.index >= start_test]
***trajectory matrix***
global T_train; T_train = [] #initiate a two-dimensional
    trajectory list
for i in range(len(X_train)-NofLag+1):
    T_train.append(X_train[::-1][i:i+NofLag])
T_train = T_train[::-1] # training trajectory matrix
print("The first two training trajectory input data: \n \n%s
    \n%s" %(T_train[:1], T_train[1:2]))
print()
global T_test; T_test = [] #initiate a two-dimensional
    trajectory list
```

```
for i in range(len(X_test)-NofLag+1):
    T_test.append(X_test[::-1][i:i+NofLag])
T_test = T_test[::-1]
print("The first two testing trajectory input data: \n \n%s \
    n%s" %(T_test[:1], T_test[1:2]))
***Theano Logistic Optimization with Ordinary logistic model
    ***
# Declare Theano symbolic variables
x = T.dmatrix("x")
y = T.dvector("y")
# declare logistic regression coefficient
s = zeros(3); s[0] = 1
w = theano.shared(s, name="w")
# initialize the bias term
b = theano.shared(0., name="b")
# Construct Theano expression graph
p_1 = 1 / (1 + T.exp(-T.dot(x, w) - b))    # Probability that
    target = 1
prediction = p_1 > 0.5                    # The prediction
    thresholded
```

```
xent = -y * T.log(p_1) - (1-y) * T.log(1-p_1) # Cross-entropy
    loss function
# xent = T.log(1+ T.exp(y*(-T.dot(x, w) - b)))
cost = xent.mean() # The cost to minimize
gw, gb = T.grad(cost, [w, b])
# Compile the training function and predict function(
    initialization)
train = theano.function(
          inputs=[x,y],
          outputs=[prediction, xent, w],
          updates=((w, w - 0.1 * gw), (b, b - 0.1 * gb)))
predict = theano.function(inputs=[x], outputs=prediction)
predict_prob = theano.function(inputs=[x], outputs=p_1)
delta = array([1,0,0])
# Because we start with value of delta to tune value of beta,
    we need to dot product of delta and X
deltaTx_train = squeeze([dot(delta, x.values) for x in
   T_train])
deltaTx_test = squeeze([dot(delta, x.values) for x in T_test
   ])
```

```
# Dependent Variable (convert 1/-1 binary variable to 1/0
    binary variable)
train_y = y_train.copy()
for i, x in enumerate(train_y):
    if x == -1: train_y.__setitem__(i, 0)
test_y = y_test.copy()
for i, x in enumerate(test_y):
    if x == -1: test_y.__setitem__(i, 0)
# Note: lag1, lag2 and lag3 is ralative to the index of each
    row. For example, the bottom right corner of the first
# training data (-2.624..) is lag3 of 2010-01-10 and
    therefore lag5 of 2001-01-12. And this input would be used
     to
# predict the trend(upward/downward) of stock on 2010-01-12
# Model Fit
cost_old = 100000
print('{:<15s} {:>15s} {:>25s} {:>12s} {:>12s}'.format('# of
    fitting','Beta','cost value','train acc', 'test acc'))
for i in range(150+1):
    pred, err, coef = train(deltaTx_train, train_y[NofLag
```

```
        -1:])
    beta = coef
    cost = mean(err)
#       if (cost_old - cost < 0.000001): break
    train_acc = mean(predict(deltaTx_train) == train_y[NofLag
        -1:])
    test_acc = mean(predict(deltaTx_test) == test_y[NofLag
        -1:])
    # Model Prediction
    set_printoptions(precision=4, suppress = True)
    if( not (i % 10) ):
        cost_old = cost
        print('{:<15d} {:<25s} {:^20s} {:>0.4f} {:>12.3f}'.
            format(i, str(beta), str(cost.round(5)), train_acc
            , test_acc))
***ROC curve***
thresholds = np.linspace(1,0,101)
# This is the model's prediction on the test data.
T = predict_prob(deltaTx_test)
ROC = np.zeros((101,2))
```

```
Y = test_y[NofLag-1:]
for i in range(101):
    t = thresholds[i]
    # Classifier / label agree and disagreements for current
       threshold.
    TP_t = np.logical_and( T > t, Y==1 ).sum()
    TN_t = np.logical_and( T <=t, Y==0 ).sum()
    FP_t = np.logical_and( T > t, Y==0 ).sum()
    FN_t = np.logical_and( T <=t, Y==1 ).sum()
    # Compute false positive rate for current threshold.
    FPR_t = FP_t / float(FP_t + TN_t)
    ROC[i,0] = FPR_t
    # Compute true  positive rate for current threshold.
    TPR_t = TP_t / float(TP_t + FN_t)
    ROC[i,1] = TPR_t
AUC = 0.
for i in range(100):
    AUC += (ROC[i+1,0]-ROC[i,0]) * (ROC[i+1,1]+ROC[i,1])
AUC *= 0.5
if plot:
```

```
    # Plot the ROC curve.
    fig = plt.figure(figsize=(6,6))
    plt.plot(ROC[:,0], ROC[:,1], lw=2)
    plt.xlim(-0.1,1.1)
    plt.ylim(-0.1,1.1)
    plt.xlabel('$FPR(t)$')
    plt.ylabel('$TPR(t)$')
    plt.grid()
    plt.title('ROC curve, AUC = %.4f'%AUC)
    plt.show()
***Multi-index logistic model***
***function defined***
def get_beta(delta, train_in, test_in, train_out, test_out,
   model = LogisticRegression()):
    deltaTtrain, deltaTtest = deltaTx(delta, train_in,
       test_in)
    model.fit(deltaTtrain, train_out); intercept = model.
       intercept_
    beta = model.coef_; cost = compute_cost(squeeze(beta),
       intercept, deltaTtrain, train_out)
```

```
    acc_train = model.score(deltaTtrain, train_out)
    acc_test = model.score(deltaTtest, test_out)
    return [beta, delta, acc_train, acc_test, cost, intercept
        , 'beta']
def get_delta(beta, train_in, test_in, train_out, test_out,
    model = LogisticRegression()):
    trainTbeta, testTbeta = xTbeta(beta, train_in, test_in)
    model.fit(trainTbeta, train_out); intercept = model.
        intercept_
    delta = model.coef_; cost = compute_cost(squeeze(delta),
        intercept, trainTbeta, train_out)
    acc_train = model.score(trainTbeta, train_out)
    acc_test = model.score(testTbeta, test_out)
    return [beta, delta, acc_train, acc_test, cost, intercept
        , 'delta']
def deltaTx(delta, train_in, test_in):
    X_train = squeeze([dot(delta, x.values) for x in train_in
        ])
    X_test = squeeze([dot(delta, x.values) for x in test_in])
    return (X_train, X_test)
```

```
def xTbeta(beta, train_in, test_in):
    X_train = squeeze([x.values.dot(beta.T) for x in train_in
        ])
    X_test = squeeze([transpose(dot(x.values, beta.T)) for x
        in test_in])
    return (X_train, X_test)
def update_beta_and_delta(beta_init, cost_init):
    beta, delta, acc_train, acc_test, cost, intercept, coef =
        [[0] for i in range(7)]
    beta[0] = beta_init; cost[0] = cost_init; delta[0] =
        array([1]+[0]*(NofLag-1))
    List = [beta, delta, acc_train, acc_test, cost, intercept
        , coef]
    for i in range(200): #200 times updating
        res = get_delta(beta[-1], T_train, T_test, y_train[
            NofLag-1:], y_test[NofLag-1:])
        for x,y in enumerate(List): y.append(res[x])
        res = get_beta(delta[-1], T_train, T_test, y_train[
            NofLag-1:], y_test[NofLag-1:])
        for x,y in enumerate(List): y.append(res[x])
```

```
    i = cost.index(min(cost))
    return beta, delta, acc_train, acc_test, cost, intercept,
        coef, i
#   return beta[i], delta[i], acc_train[i], acc_test[i], cost
    [i], coef[i]
"""Find the best model by iterating the above algorithms"""
beta, delta, acc_train, acc_test, cost, intercept, coef,
    index = update_beta_and_delta(beta_init, cost_old)
***ROC curve***
thresholds = np.linspace(1,0,101)
# This is the model's prediction on the test data.
T = sigmoid(intercept[index] + squeeze([delta[index].dot(x).
    dot(beta[index].T) for x in T_test]))
Y = y_test[NofLag-1:]
ROC = np.zeros((101,2))
for i in range(101):
    t = thresholds[i]
    # Classifier / label agree and disagreements for current
        threshold.
    TP_t = np.logical_and( T > t, Y==1 ).sum()
```

```
        TN_t = np.logical_and( T<=t, Y==0 ).sum()
        FP_t = np.logical_and( T > t, Y==0 ).sum()
        FN_t = np.logical_and( T<=t, Y==1 ).sum()
        # Compute false positive rate for current threshold.
        FPR_t = FP_t / float(FP_t + TN_t)
        ROC[i,0] = FPR_t
        # Compute true  positive rate for current threshold.
        TPR_t = TP_t / float(TP_t + FN_t)
        ROC[i,1] = TPR_t
    AUC = 0.
    for i in range(100):
        AUC += (ROC[i+1,0]-ROC[i,0]) * (ROC[i+1,1]+ROC[i,1])
    AUC *= 0.5
    if plot:
        # Plot the ROC curve.
        fig = plt.figure(figsize=(6,6))
        plt.plot(ROC[:,0], ROC[:,1], lw=2)
        plt.xlim(-0.1,1.1)
        plt.ylim(-0.1,1.1)
        plt.xlabel('$FPR(t)$')
```

```
plt.ylabel('$TPR(t)$')

plt.grid()

plt.title('ROC curve, AUC = %.4f'%AUC)

plt.show()
```

VITA

XIANG LIU

Education:	B.S. Psychology, Beijing Normal University, Beijing, China 2013 M.S. Mathematical Sciences, concentration in Statistics, East Tennessee State University Johnson City, Tennessee 2016
Professional Experience:	Graduate Assistant, East Tennessee State University Johnson City, Tennessee, 2015–2016