上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

BACHELOR'S THESIS

论文题目	基于隐马尔科夫模型的股票收益率序列预测分析
学生姓名	高　天
学生学号	5120719083
指导教师	林建忠
专　　业	数学与应用数学（数学金融本硕贯通班）
学院（系）	数学科学学院

Submitted in total fulfilment of the requirements for the degree of Bachelor
in Mathematics and Applied Mathematics (Mathematics-Finance Class)

Prediction of stock return series with hidden Markov models

Tian Gao

Supervisor

Associate Professor Jianzhong Lin

School of Mathematical Sciences

Shanghai Jiao Tong University

Shanghai, P.R.China

Jun. 17, 2016

上海交通大学
毕业设计（论文）学术诚信声明

上海交通大学
毕业设计（论文）版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保　密 □，在 ________ 年解密后适用本授权书。

本学位论文属于

不保密 □。

(请在以上方框内打“✓”)

作者签名：____________________　　指导教师签名：____________________

日　期：________年____月____日　　日　期：________年____月____日

基于隐马尔科夫模型的股票收益率序列预测分析

摘　要

在量化金融领域，针对股票收益率序列的研究与分析是业界与学术界共同最关心的研究方向之一。股票收益率的分布通常具有尖峰厚尾的性质，而在不同的市场环境下，收益率分布的参数大相径庭。为了定量刻画这种收益率的市场状态依赖性，学者们建立了许多模型，隐马尔科夫模型则是其中较为重要的一种。

在本文中，我们首先对隐马尔科夫模型的定义及构成、主要统计量以及参数方法给出了详尽的介绍。本文的独创性贡献在于，我们在传统模型参数期望最大化算法、以及全局解码的维特比算法的基础上，综合了数据预处理、K 均值聚类算法等技术，搭建了一个完整的股票收益率序列预测及市场历史隐状态分析的系统。该系统完整封装了数据预处理、模型初始化、参数估计及模型校准、隐状态解码及分析、收益率序列预测以及数值结果输出等环节，且适用于一般的股票市场。我们利用 `Python` 编程对系统进行了实现，并且利用这一系统对中国股票市场的沪深 300 指数及美国股票市场的标准普尔 500 指数进行了详尽的实证分析与模型检验，完整地分析与对比了数值结果，并用 `R-ggplot2` 进行了可视化呈现。市场历史隐藏状态的分析结果符合一般认知，且系统在自适应收益率序列预测中有较好的表现。特别地，我们针对沪深 300 指数不同观测频率的数据作了进一步分析与对比，发现更长时间段数据的隐藏状态分析结果更准确，而更高频率数据的预测正确率更高。

此外，我们对模型构建及实证分析中的遗留问题也进行了更深入的探讨，并提出了潜在的解决方法，如指数加权期望最大化算法。我们还简要介绍了更一般化的隐马尔科夫模型，并引入了模型参数估计的粒子滤波方法，以期在未来的研究工作中加以解决。

关键词： 隐马尔科夫模型，股票收益率预测，期望最大化算法，维特比算法，K 均值聚类，粒子滤波

Prediction of stock return series with hidden Markov models

ABSTRACT

In the area of quantitative finance, research and analysis on stock return series is one of the most concentrated topics shared by both the industry and the academia. Distributions of stock returns usually present leptokurtosis and fat-tail. Parameters of these distributions, however, vary largely under different market states. Therefore, a lot of models have been proposed by scholars to quantitatively depict the market state dependency of stock returns, of which the hidden Markov model is a very important one.

In this thesis, we firstly provide analyses with details on the definition and formulation of the hidden Markov model, primary statistics, and methods to estimate the model parameters. The original contribution of the thesis is that, based on traditional expectation maximization algorithm for model estimation and Viterbi algorithm for global decoding, we combine the data-preprocessing and K-Means clustering techniques, in order to construct a complete system for stock return series prediction and historical market hidden states analysis. The system thoroughly encapsulates the modules of data-preprocessing, model initialization, parameters estimation and model calibration, hidden states decoding and analysis, return series prediction and results output, and it well applies to general stock markets. We accomplish the system realization through `Python` programming, and implement the system to perform empirical analyses and model validation on the CSI 300 Index from Chinese stock market and the S&P 500 Index from the U.S. stock market. We conduct thorough analyses and comparisons on the numerical results, along with presentations of results visualization through `R-ggplot2`. The results of market historical hidden states analyses match common acknowledgements, and the system performs well in adaptive stock return series predictions. Specifically, we further carry out analyses and comparisons on CSI 300 data with different observation frequency, finding that hidden states analyses are more accurate with data of longer observation periods and predictions have higher correctness with data of higher observation frequencies.

Additionally, we provide further discussions on unaddressed issues remained in the model construction part and empirical analysis part, and propose some potential solutions like exponentially weighted expectation maximization algorithm. We also briefly introduce the more general hidden Markov models, and the particle filter method for model estimation. We hope to solve the problems and do further researches in future works.

KEY WORDS: hidden Markov model, stock return prediction, expectation maximization, Viterbi algorithm, K-Means clustering, particle filter

Contents

List of Figures

List of Tables

Chapter 1 Introduction

We introduce the brief history and status quo of stock return series analysis and prediction in the very first chapter of this thesis. The topic has received heated discussion since the invention of exchange traded stocks. In Sec. 1.1 and 1.2 we briefly introduce the elementary concepts about financial time series and stock returns, and some mature and standard techniques for time series analysis. In Sec. 1.3 we introduce the novel hidden Markov model, and explain the background and aims of this thesis. Then in Sec. 1.4 we show the framework of the thesis, and list the notations, abbreviations and jargons that are used in the following chapters for the reader's convenience.

1.1 Financial time series and stock returns

A time series, by definition, is a sequence of data points that are recorded with a timeline. The observations can be made either over a continuous interval or at discrete time nodes; either regularly spaced or irregularly spaced. A financial time series is a special kind of time series of which observations are prices of financial assets like bonds, stocks or stock indices.

Of all kinds of financial markets, stock market is the most volatile and popular one. Analysis on stocks has been carried out since the very first day when exchange traded stocks were invented. In this work we focus our research on the stock markets; more specifically, the CSI 300 index in the Chinese market and the S&P 500 index in the U.S. market, which are weighted average of the most 300/500 stocks in the corresponding markets and considered to be the best indicators of the markets.

One of the most important and fundamental indicators of evaluating the performance of a financial asset is its return. Mathematically, the return r of a stock at time t is defined as

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1, \tag{1–1}$$

where P_t stands for the price of the stock at time t. Eq. 1–1 is also known as the simple return of a stock. For a k-period financial time series, the return at the end of the period is calculated cumulatively by the returns during the period

$$\begin{aligned} r_t(k) &= \frac{P_t - P_{t-k}}{P_{t-k}} \\ &= (1 + r_{t-k+1})(1 + r_{t-k+2}) \cdots (1 + r_t) - 1 \\ &= \prod_{i=t-k+1}^{t} (1 + r_i) - 1. \end{aligned} \tag{1–2}$$

In order to make the computations more easy and elegant, the log return has been introduced and the equations above can be transformed into the follows:

$$R_t = \log \frac{P_t}{P_{t-1}}, \tag{1–3a}$$

$$R_t[k] = \log \frac{P_t}{P_{t-k}} = \log \frac{P_t}{P_{t-1}} \cdot \frac{P_{t-1}}{P_{t-2}} \cdot \cdots \cdot \frac{P_{t-k+1}}{P_{t-k}} = \sum_{i=t-k}^{t} R_i. \tag{1–3b}$$

Eq. 1–3a is the definition of log return and Eq. 1–3b implements the additivity of logarithm.

1.2 Traditional time series analysis techniques

Time series analysis can be dated far back to early the last century, for different goals in different research areas.

In terms of researched domain, time series analysis can be divided into frequeny-domain analysis and time-domain analysis; the former includes spectral analysis and wavelet analysis, etc., while the latter is composed mainly of correlation analyses. In terms of statistical method, the models are further classified as parametric, semi-parametric and non-parametric. With respect to the observed variable(s) the models can be divided into linear and nonlinear, univariate and multivariate.

In 1951, Peter Whittle introduced the autoregressive–moving-average (ARMA) model in [1], and the method is popularized by George E. P. Box and Gwilym Jenkins. Later the model was developed to consider integrated time series and derived the autoregressive integrated moving average (ARIMA) model and autoregressive fractionally integrated moving average (ARFIMA) model. Extension of the models to deal with multi-variables, i.e. vector-valued variables, is then called the vector autoregression (VAR) model. Models related to heteroskedasticity of the time series were also developed, the most famous two being autoregressive conditional heteroskedasticity (ARCH) model and generalized autoregressive conditional heteroskedasticity (GARCH) model.

With development of Bayesian statistics, a kind of models named dynamic Bayesian network has been built for time series analysis, of which the simplest one is called hidden Markov model (HMM). HMM is a statistical Markov model, where there is an observed time series and an underlying series that is assumed to be an Markov chain with unobserved (hidden/latent) states. This thesis is concentrated on the very model.

1.3 Background and aims

In 1982, a private hedge fund investment company named Renaissance Technologies was founded in New York by Dr. James H. Simons, who is also a mathematician well-known in the area of pattern recognition. Employment of plenty of mathematicians and statisticians and widely applications of statistical models to quantitative finance, these elements help Renaissance become one of the most successful investment companies. The most famous fund of the company is the Medallion fund, and word is that the core model implemented in Medallion's quantitative investment strategies is the HMM. With great interest in the model itself and admiration to Dr. Simons, the author decided to study the HMM and try to figure out the secret to the success of Renaissance.

Despite that it is almost impossible to reproduce the investment strategies based on HMM, we can perform thorough financial time series analysis with the model, and construct a complete stock return series prediction system on top of the analysis results. Therefore, we aim to build such a system, in which we shall apply the hidden Markov model. We hope to achieve this goal by three sub-aims:

- **Aim 1.**

  Construct the entire system for stock return series prediction. The system should include data pre-processing, model construction and estimation, and eventually predictions with detailed analysis.

- **Aim 2.**

  Estimate the HMM parameters in order to calibrate the model with accessible market data.

- **Aim 3.**

  Carry out empirical analysis on both Chinese and U.S. stock markets. Verify the effectiveness of the prediction system and provide detailed analysis and comparisons.

Finally, we hope to realize the system with Python programming with each module encapsulated and flexible for different data and parameters.

1.4 Thesis organization

This thesis is composed of seven chapters and two appendices.

In Ch. 2 we introduce some prerequisite knowledge and models that will be later used in HMM introduction or system construction. Ch. 3 formally introduce the hidden Markov model, including model formulation, some key concepts and related algorithms. Then with knowledge of the model, we show construction of the full stock return series prediction system in Ch. 4, and then conduct empirical analysis with the system in Ch. 5, on both Chinese CSI 300 index and U.S. S&P 500 index. The Python codes for model realization are provided in Appendix B along with a short user manuscript. Part of the visualization results that are not presented in Ch. 5 are then listed in Appendix A. Ch. 6 concludes the thesis, and in Ch. 7 we make brief discussions on some remaining issues and thoughts for future works.

1.5 Notations, abbreviations and jargons

We list some notations and Greek letters that are used in this thesis for reference, see Table 1–1. The symbols appear mainly in Ch. 3. Some abbreviations are also included in the table.

Table 1–1 Notations and abbreviations

Notations/Abbreviations	Meanings
S_t	the latent variable (hidden state) at time t
s_t	the value of S_t
$\mathbf{S}^{(t)}, \mathbf{S}^{(t+1:T)}$	the vector $(S_1, S_2, \ldots, S_t)$ and $(S_{t+1}, \ldots, S_T)$
$\mathbf{s}^{(t)}, \mathbf{s}^{(t+1:T)}$	the value vetor of $\mathbf{S}^{(t)}$ and $\mathbf{S}^{(t+1:T)}$
X_t	the observed variable at time t
x_t	the value of X_t
$\mathbf{X}^{(t)}, \mathbf{X}^{(t+1:T)}$	the vector $(X_1, X_2, \ldots, X_t)$ and $(X_{t+1}, \ldots, X_T)$
$\mathbf{x}^{(t)}, \mathbf{x}^{(t+1:T)}$	the value vetor of $\mathbf{X}^{(t)}$ and $\mathbf{X}^{(t+1:T)}$
$\delta_i(t)$	the probability of state i at time t
$\boldsymbol{\delta}(t)$	the probability distribution of states at time t
$\mathbf{\Gamma}$	the Markov state transition matrix
γ_{ij}	the element at the i^{th} row and j^{th} column of $\mathbf{\Gamma}$
$\boldsymbol{\pi}$	the initial distribution of the states
$\boldsymbol{\alpha}_t$	the row vector of forward probabilities at time t
$\boldsymbol{\beta}_t$	the row vector of backward probabilities at time t
ξ_{ti}	the joint probability of state i at time t
$\mathbb{P}(\cdot)$	the probability function
$E(\cdot)$	the expectation function
w.r.t.	with respect to
iff	if and only if
PDF	probability density function
PMF	probability mass function
CDF	cumulative distribution function
Ch.	Chapter
Sec.	Section (including subsections and subsubsections)
Def.	Definition
Thm.	Theorem
Prop.	Proposition
Cor.	Corollary
Illus.	Illustration
Eq.	Equation

There are some terms that may be linguistically ambiguous for readers unfamiliar with the financial industry. Here we specify some terms that are used multiple times but may represent different things.

By **state transition matrix**, we refer to the state transition probability matrix of a Markov process.

By **conditional distribution**, we refer to the state-dependent distribution of the observed variable, depending on the context, or simply refer to a conditional probability distribution.

By **bear**, we mean the so-called bearish market or bear market for most of the time, and sometimes it refers to the name of the corresponding state in K-Means clustering of the hidden Markov model. Similar cases are for **intermediate** and **bull**.

By **rebounce**, we mean the reversion of a market trend. It can either refer to the rally after a period of fall, or the plunge following the rise.

By **prediction**, we refer to the stock return prediction and also the forecast step in a Markov chain of HMM.

Chapter 2 Preliminary Knowledge and Models

Before introduction of the hidden Markov models, we review some preliminary knowledge and models that are foundations of HMM. For starters, we introduce independent mixture distributions in Sec. 2.1. We then review the essential elements and properties of Markov chain in Sec. 2.2. We also present a brief introduction on K-Means clustering in Sec. 2.3, which is used for HMM initialization in the system that will be talked about in Ch. 4.

2.1 Independent mixture distributions

Often is the case that when we fit some real-world data with a particular statistical model, the sample variance is way larger than the sample mean, indicating existence of strong over-dispersion and thus inappropriateness of the model. One of the simplest examples is the multi-modality of some sample data, which is impossible to calibrate some common distributions like Poisson or normal distribution.

A universally used method is to implement a mixture distribution. Mixture distribution models represent presences of sub-populations of an overall population [2], and they shall derive the properties of the sub-populations and then the overall population.

The formulation of mixture distributions is simple and presented in Sec. 2.1.1. The most famous mixture model should be Gaussian mixture model (GMM) [3], which is a special case of more generalized exponential distribution family mixture models [4].

2.1.1 Model definition

Generally, a (finite) indepedent mixture distribution is composed of a finite number of component distributions. The mixing distribution is then formulated as the weighted average of the component distributions.

Say we have m continuous component distributions [1] (of random variables $Y_i, i = 1, 2, \ldots, m$), with probability density functions $p_i(x), i = 1, 2, \ldots, m$, and each PDF corresponds to probability δ_i which satisfies

$$\sum_{i=1}^{m} \delta_i = 1. \tag{2–1}$$

The mixture distribution (of random variable X) that consists of the component distributions is then formulated as:

$$p(x) = \sum_{i=1}^{m} \delta_i p_i(x). \tag{2–2}$$

Moments of the distribution can be easily inferred and given as:

$$E(X) = \sum_{i=1}^{m} \delta_i E(Y_i), \tag{2–3a}$$

$$E(X^k) = \sum_{i=1}^{m} \delta_i E(Y_i^k), \tag{2–3b}$$

and we can easily deduce the variance of X from the equations above.

2.1.2 Parameter estimation

Similar to traditional distribution parameter estimation methods, the parameters of a mixture distribution is estimated most frequently with maximum likelihood (ML) methods, and the estimation results are so called maximum likelihood estimations (MLE). In general, the likelihood of a mixture distribution is computed as:

$$L(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_m, \delta_1, \delta_2, \ldots, \delta_m \mid x_1, x_2, \ldots, x_n) = \prod_{j=1}^{n} \sum_{i=1}^{m} \delta_i p_i(x_j, \boldsymbol{\theta}_i), \tag{2–4}$$

where $x_i, i = 1, 2, \ldots, n$ are the n observations and $\boldsymbol{\theta}_i, i = 1, 2, \ldots, m$ are parameter vectors of the component distributions. Therefore, there are in total $\sum_{k=1}^{m} l_k + m - 1$ independent parameters to be estimated, where l_k stands for the length of a single parameter vector $\boldsymbol{\theta}_k$.

The method is quite ordinary and we do not expand the topic here. The reader can refer to [5]

[1]component distributions can be discrete as well, we avoid repetitions here and the results are similar

for details. It is worth mentioning that expectation maximization (EM) algorithms (see Sec. 3.2, which is an application of EM in the context of hidden Markov models) are as well carried out for parameter estimations (e.g. see [6] and [7]).

2.2 Markov chain

Named after Andrey Markov, the Markov chain is one of the most important kind of stochastic process in the world. Markov chain is a random process that transit from one state to another on a state space. The biggest feature of a Markov chain is that forecasts for the future depends only on the present but has nothing to do with the history, i.e. the 'future' is conditionally independent to the 'past' given information of the 'present'. This feature is known as the Markovian property or 'memorylessness', see Def. 2.1.

In terms of the continuousness of time, Markov chains can be further categorized as discrete-time Markov chains (DTMC) or continuous-time Markov chains. The state spaces employed by Markov chains differ from case to case, most of which are finite or countably infinite discrete.

Markov chains are applicable to many of the real-world industries, including engineering, statistics, physics, biology and economy, etc.

2.2.1 Definition and properties

For starters, we give the definition of the Markovian property and a discrete-time Markov chain.

Definition 2.1. Given a stochastic process $\{X_t \colon t \in \mathbb{T}\}$, if for any n time nodes $t_i, i = 1, 2, \ldots, n$, where $t_1 < t_2 < \cdots < t_n$, we have

$$\begin{aligned} &\mathbb{P}(X_{t_n} \leq x_n \mid X_{t_1} = x_1, X_{t_2} = x_2, \ldots, X_{t_{n-1}} = x_{n-1}) \\ =&\mathbb{P}(X_{t_n} \leq x_n \mid X_{t_{n-1}} = x_{n-1}), \end{aligned} \tag{2–5}$$

then we call $\{X_t \colon t \in \mathcal{T}\}$ a Markov process, the property described in Eq. 2–5 as Markovian property or memorylessness.

Definition 2.2. Given a stochastic process $\{X_n \colon n = 0, 1, 2, \dots\}$ and its corresponding state space $\mathbb{S} = \{S_1, S_2, \dots\}$, if for any non-negative integer k and $n_1 < n_2 < \cdots < n_l < m$, and $s_{n_1}, s_{n_2}, \cdots, s_{n_l}, s_m, s_{m+k} \in \mathbb{S}$, we have

$$\begin{aligned} &\mathbb{P}(X_{m+k} = s_{m+k} \mid X_{n_1} = s_{n_1}, \dots, X_{n_l} = s_{n_l}, X_m = s_m) \\ =&\mathbb{P}(X_{m+k} = s_{m+k} \mid X_m = s_m), \end{aligned} \tag{2–6}$$

then we call $\{X_n \colon n = 1, 2, \dots\}$ a Markov discrete-time Markov chain. The probability $\gamma_{ij}^{(k)}(m) = \mathbb{P}(X_{m+k} = j \mid X_m = i)$ is called the k-step transition probability of the chain at time m from state i to state j, and call the matrix $\mathbf{\Gamma}^{(k)}(m) = \left(\gamma_{ij}^{(k)}(m)\right)_{i,j \in \mathbb{S}}$ the k-step transition probability matrix.

It is implied in Eq. 2–6 that

$$\sum_{j=1}^{N} \gamma_{ij}^{(k)}(t) = 1, \ \forall k, t \text{ and } i = 1, 2, \dots, N. \tag{2–7}$$

With knowledge of the transition probability, we further define the time-homogeneity of a Markov chain.

Definition 2.3. If a Markov chain satisfies that

$$\mathbb{P}(X_{m+k} = j \mid X_m = i) = \mathbb{P}(X_k = j \mid X_0 = i) := \gamma_{ij}^{(k)}, \tag{2–8}$$

we call it a time-homogeneous Markov chain.

Without specific declarations, the Markov chains we discussed in this thesis are all assumed to be time-homogenous, and we denote the 1-step transition probability $\gamma_{ij}^{(1)} := \gamma_{ij}$.

The state transition probability matrix (t.p.m, also called state transition matrix) is one of the most important concepts in the area of Markov chains. We will discuss about it in details in Sec. 2.2.2. Before that, we define some other properties of a Markov chain.

Definition 2.4. If there exists a $n \geq 0$ such that $\gamma_{ij}^{(n)} > 0, i, j \in \mathbb{S}$, then we call the state j is accessible from state i, denoted as $i \to j$.

If $i \to j$ and $j \to i$, then we call state i communicates with state j, denoted as $i \leftrightarrow j$

Definition 2.5. Assume that $i \to j$. We define the probability that the process starts from state i and reaches state j with n steps as first-reach probability, written

$$f_{ij}^{(n)} := \mathbb{P}(X_n = j, X_m \neq j, m = 1, 2, \cdots, n-1 \mid X_0 = i). \tag{2–9}$$

Furthermore we define the probability that the process starts from state i and reaches state j sooner or later as

$$f_{ij} := \sum_{n=1}^{\infty} f_{ij}^{(n)}, \tag{2–10}$$

specifically $f_{ij}^{(0)} = 0$.

Definition 2.6. A state i is said to be transient if $f_{ii} < 1$, or it is saied to be recurrent or persistent, i.e. $f_{ii} = 1$.

Definition 2.7. The mean recurrence time at state i is the expected return time, denoted as

$$M_i = \sum_{n=1}^{\infty} n f_{ii}^{(n)}. \tag{2–11}$$

State i is said to be positive recurrent if $M_i < \infty$, or it is called null recurrent.

Definition 2.8. The period of a state is defined as

$$T = \text{gcd}\{n \colon \gamma_{ii}^{(n)} > 0\}, \tag{2–12}$$

where gcd stands for greatest common divisor. The state is said to be aperiodic if $T = 1$.

Definition 2.9. If a state is aperiodic and positive recurrent, it is said to be ergodic.

The properties defined above and some other theorems can be easily found on many books or Wiki websites (e.g. see [8–10]).

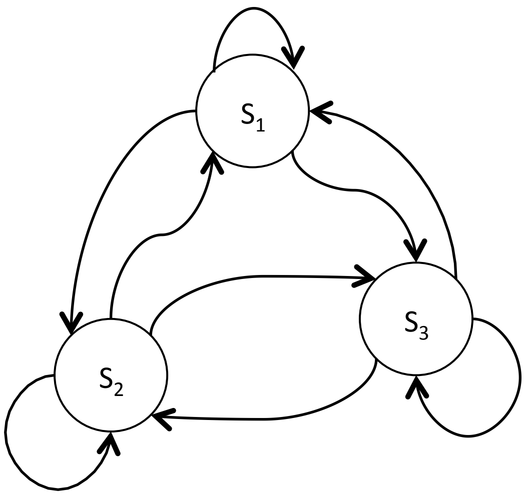


ILLUSTRATION 2–1 State transition process of a Markov chain

2.2.2 State transition probability

Illus. 2–1 visually shows a typical Markov chain. The state space in this case is finite (three in fact) and the process is the transition from the states to one another.

We have given the definition of the state transition matrix with Def. 2.2 in Sec. 2.2.1. For every finite-state time-homogeneous Markov chain, we have the following theorem.

Theorem 2.1. *For finite-state time-homogeneous Markov chains and any $t, u \in \mathbb{N}$, we have*

$$\boldsymbol{\Gamma}^{(t+u)} = \boldsymbol{\Gamma}^{(t)}\boldsymbol{\Gamma}^{(u)}. \tag{2–13}$$

Eq. 2–13 is known as Chapman-Kolmogorov equation.

If we denote the 1-step transition matrix $\boldsymbol{\Gamma}^{(1)}$ as $\boldsymbol{\Gamma}$, then we shall deduce the following corollary.

Corollary 2.2. *For every finite-state time-homogeneous Markov chain and any $t \in \mathbb{N}$, we have*

$$\boldsymbol{\Gamma}^{(t)} = \boldsymbol{\Gamma}^{t}, \tag{2–14}$$

that is, the k-step transition matrix is entirely decided by the 1-step transition matrix.

Now we give the definition of the initial distribution of a Markov chain, and we shall find the unconditional distribution of the chain at a given time.

Definition 2.10. We call the row vector

$$\boldsymbol{\delta}(t) = (\mathbb{P}(S_t = 1), \ldots, \mathbb{P}(S_t = N)) \tag{2–15}$$

the unconditional probabilities of a Markov chain at time t. In particular, we refer to $\boldsymbol{\delta}(1) := \boldsymbol{\pi}$ as the initial distribution of the Markov chain.

Then with the definition and Cor. 2.2, we have

$$\boldsymbol{\delta}(t+1) = \boldsymbol{\delta}(t)\boldsymbol{\Gamma} = \boldsymbol{\pi}\boldsymbol{\Gamma}^t. \tag{2–16}$$

The equation implies that the unconditional distribution relies fully on the initial distribution and the state transition matrix.

2.2.3 Stationary distributions

At last we introduce the idea of stationary distribution.

Definition 2.11. For a Markov chain, if there exists a row vector $\{v_j : j \in \mathbb{S}\}$ that satisfies

$$v_j \geq 0, j \in \mathbb{S}; \tag{2–17a}$$

$$\sum_{j \in \mathbb{S}} v_j = 1; \tag{2–17b}$$

$$v_j = \sum_{i \in \mathbb{S}} v_i \gamma_{ij}, \tag{2–17c}$$

then we call $V := \{v_j : j \in \mathbb{S}\}$ the stationary distribution of the Markov chain.

The equations above guarantees that V is simutaneously a probability distribution and stationary.

The introduction of stationary distribution is because that the state transition matrix will converge. We state the property as the following theorem.

Theorem 2.3. *For an ergodic finite-space time-homogeneous Markov chain, the limit distribution is stationary, i.e.*

$$\lim_{n\to\infty} \boldsymbol{\pi}\boldsymbol{\Gamma}^n = V. \tag{2–18}$$

One of the results of Thm. 2.3 is that forecast result in hidden Markov models converges with increasing forecast steps (see Sec. 3.3.1).

2.3 K-Means clustering

Data clustering is a popular technique in time series analysis. It enables us to cluster sub-populations with different properties so that more accurate analysis results could be available, considering the heterogeneity of the sample data.

K-Means, of all existing data clustering techniques, is the most common and universally acknowledged one. The name is firstly used in [11]. The idea originates from signal processing theories, which is known as vector quantization (VQ), and now is prevalent for cluster analysis and data mining.

The standard algorithm for K-Means was proposed by Stuart Lloyd in 1957 and by E. W. Forgy in 1965 [12] separately, which is known as Lloyd-Forgy algorithm. A more efficient algorithm is proposed by Hartigan in [13].

Clustering itself has been commonly implemented in financial data analysis (e.g. see [14–16]), while we introduce the K-Means algorithm merely to find the initial distribution of the hidden Markov model (see Sec. 4.2.1), thus we only provide a very brief introduction here.

2.3.1 Model formulation and aims

Firstly we state the K-Means clustering problem.

Given observation set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where $\mathbf{x}$ is a d-dimensional vector, we aim to partition the n observations into $k (k \leq n)$ sub-sets $\{S_1, S_2, \ldots, S_k\}$ in order to minimize the within-cluster sum of squares (WCSS), which is the sum of distance functions of each point in the cluster to the center. The formulation of this problem is given as follow:

$$\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2, \tag{2–19}$$

where $\boldsymbol{\mu}_i$ is the mean of points of S_i (center), and **S** is the k sub-sets.

The clustering centers found by the K-Means algorithm shall be appropriate guesses for the initial distribution.

2.3.2 The K-Means algorithm

The standard algorithm processes iteratively.

Given a set of initial guess of k means $\{m_1^{(1)}, m_2^{(1)}, \ldots, m_k^{(1)}\}$, we firstly assign each observation to the cluster whose mean yields the least WCSS, which is known as the assignment step.

$$S_i^{(t)} = \left\{ x_p \colon \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2, \forall j,\ 1 \leq j \leq k \right\}. \tag{2–20}$$

Each x_p is assigned to one and only one set S_i.

Then we calculate the new means to be the centroids of the observations in the new clusters, which is known as the update step.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j. \tag{2–21}$$

The two steps are performed in sequence until convergence of the result.

Notice that the algorithm only yields a local optimum but has no guarantees on finding the global one. The result is enough for the analysis we will carry out in empirical analysis. Since the algorithm is not the major concern of this thesis, the programming realization of K-Means implement the `sklearn.cluster.KMeans` module directly for convenience (see Appendix B).

Chapter 3 Hidden Markov Models

In this chapter we formally introduce the hidden Markov models (HMM). In Sec. 3.1 we examine the structure of HMM and some fundamental but significant statistics. A brief introduction to the history of HMM is also covered. The core part of this chapter lies in Sec. 3.2, i.e. how to estimate the model parameters for further analysis and predictions with the so-called Baum-Welch or Expectation Maximization (EM) algorithm. Principles and derivations are explained in details, and every module and step of the algorithm is specified. Sec. 3.3 firstly introduces the method to forecast the series based on model parameters deduced with the algorithm. Then we discuss about techniques to analyze the behavior of the hidden states, which is known as decoding.

3.1 Model overview

As is apparently indicated in its name, a hidden Markov model (HMM) is closely related to Markov chains but with additional existence of latent elements. Generally speaking, a HMM is a doubly stochastic process [17]. One stochastic process cannot be directly observed while the other sequence is observable, of which the probability distribution of the observed variable is conditional on the hidden states, i.e. the underlying and unobserved process. Specifically, the underlying process is considered to satisfy the Markovian property and HMM is thus named. HMM is also known as 'hidden Markov process' [18], 'Markov-dependent mixture' [19], 'Markov-switching model' [20], 'Markov mixture model' or 'models subject to Markov regime' [5], and these are just different names of HMM.

The model was firstly proposed by Ruslan L. Stratonovich in [21], who also firstly propose the forward-backward procedure (see Sec. 3.2.1). As we described in Sec. 2.2, states in a typical Markov chain are observable, thus the only parameters need estimating are the state transition probabilities (matrix). In a HMM, one of the two series is visible and each output is distributed conditional on the hidden states. The hidden states are distributed in a potential set with corresponding symbols. The directly observed sequence and the sequence of state symbols (which might need to be deduced from the observations) together provide information about the model.

HMM, due to its probabilistic property, is considered as the simplest form of a dynamic Bayesian network [22]. Besides its application in the financial industry (see Sec. 1.3), the model is mainly applied to pattern recognition such as speech recognition and part-of-speech tagging [22, 23], and bioinfomatics like health trajectory [24].

In more general HMMs, the latent variable is not limited to the form of hidden states represented with tokens. The latent variables themselves could be mathematically meaningful, either discrete or continuous. We will cover this part in Sec. 7.2. Currently in this section, we only deal with hidden states that are categorical.

3.1.1 Model formulation

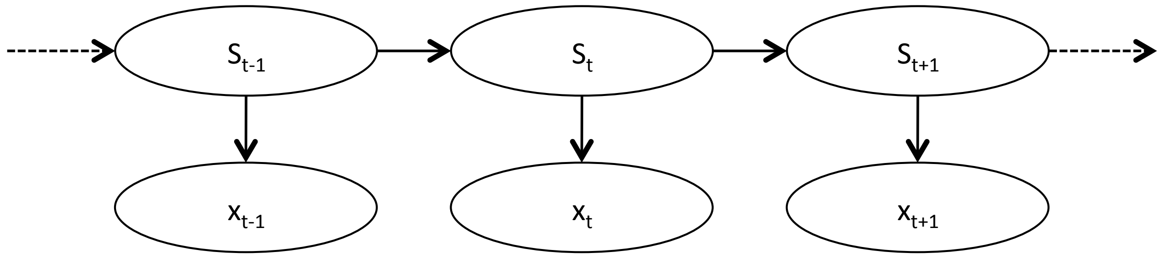


ILLUSTRATION 3–1 Directed graph of a typical HMM

Illus. 3–1 plots a typical HMM in the simplest way. The model is within the area dependent mixture models, with S_t representing the hidden states process and X_t representing the observed variable, i.e. the visible realizations of the other stochastic process.

The hidden states process $\{S_t \colon t = 1, 2, \dots\}$ is a Markov chain:

$$\mathbb{P}(S_t \mid \mathbf{S}^{(t-1)}) = \mathbb{P}(S_t \mid S_{t-1}),\ t = 2, 3, \dots, \tag{3–1}$$

and the observed variable has a probability distribution conditional only on the current hidden state:

$$\mathbb{P}(X_t \mid \mathbf{X}^{(t-1)}, \mathbf{S}^{(t-1)}) = \mathbb{P}(X_t \mid S_t),\ t = 1, 2, 3, \ldots. \tag{3–2}$$

Values of the hidden states are categorical tokens that qualitatively distinguishes the states. The number of symbols in the feasible set defines the name of the HMM, e.g. if there are in total N possible hidden states, the model is then referred to as a N-state HMM. The transition of each state from and to one another is exactly the same as one shown in Illus. 2–1

The probability $\mathbb{P}(X_t \mid S_t)$ in Eq. 3–2 can be used to derive the probability mass/density functions for discrete/continuous observations. We introduce the notation p_i, for $i = 1, 2, \ldots, N$, where

$$p_i(x) = \begin{cases} \mathbb{P}(X_t = x \mid S_t = i) & \text{, if } X_t \text{ is discrete,} \\ f_t(x \mid S_t = i) & \text{, if } X_t \text{ is continuous.} \end{cases} \tag{3–3}$$

We refer to the N distributions $\{p_i\}_{i=1}^N$ as state-dependent distribution, or simply conditional distributions, since the probability distributions are conditional on the states. The distributions are irrelevant with time t owing to their time homogeneity. In the following sections, we merely consider the case where X_t is discrete, as the observed stock return series that will be discussed about in Ch. 4 and 5 are discretely sampled.

3.1.2 Marginal distributions

With knowledge of Markov chain (Sec. 2.2) and Eq. 3–1,3–2 and 3–3, we are able to find the marginal distributions of the observed series X_t.

Consider the marginal probability

$$\begin{aligned} \mathbb{P}(X_t = x) &= \sum_{i=1}^{N} \mathbb{P}(X_t = x \mid S_t = i)\mathbb{P}(S_t = i) \\ &= \sum_{i=1}^{N} p_i(x)\delta_i(t). \end{aligned} \tag{3–4}$$

We can rewrite Eq. 3–4 in form of matrix:

$$\begin{aligned}\mathbb{P}(X_t = x) &= (\delta_1(t), \delta_2(t), \ldots, \delta_N(t)) \begin{pmatrix} p_1(x) & & 0 \\ & \ddots & \\ 0 & & p_N(x) \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \\ &= \boldsymbol{\delta}(t)\mathbf{P}(x)\mathbf{1}',\end{aligned} \tag{3–5}$$

where $\mathbf{P}(x)$ is the diagonalized matrix of $(p_1(t), p_2(t), \ldots, p_N(t))$.

Since $\boldsymbol{\delta}(t) = \boldsymbol{\pi}\boldsymbol{\Gamma}^{t-1}$ (Eq. 2–16), we shall rewrite Eq. 3–5 as

$$\mathbb{P}(X_t = x) = \boldsymbol{\pi}\boldsymbol{\Gamma}^{t-1}\mathbf{P}(x)\mathbf{1}'. \tag{3–6}$$

We also care about the bivariate distribution of X_t. Firstly we consider the joint probability expressed in conditional probabilities:

$$\mathbb{P}(X_t, X_{t+k}, S_t, S_{t+k}) = \mathbb{P}(S_t)\mathbb{P}(X_t \mid S_t)\mathbb{P}(S_{t+k} \mid S_t)\mathbb{P}(X_{t+k} \mid S_{t+k}), \tag{3–7}$$

which leads to

$$\begin{aligned}\mathbb{P}(X_t = v, X_{t+k} = w) &= \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{P}(X_t = v, X_{t+k} = w, S_t = i, S_{t+k} = j) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{P}(S_t = i)p_i(v)\mathbb{P}(S_{t+k} = j \mid S_t = i)p_j(w) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \delta_i(t)p_i(v)\gamma_{ij}(k)p_j(w) \\ &= \boldsymbol{\delta}(t)\mathbf{P}(v)\boldsymbol{\Gamma}^k\mathbf{P}(w)\mathbf{1}'.\end{aligned} \tag{3–8}$$

Thus, we can derive the expectation of X_t:

$$E(X_t) = \sum_{i=1}^{N} \delta_i(t)E(X_t \mid S_t = i). \tag{3–9}$$

Eq. 3–9 plays an important role in the series prediction that will be talked about later.

3.1.3 The likelihood

One of the most important statistics of a statistical model is its (log) likelihood. Therefore, we present a practical way to compute the likelihood L_T of the N-state HMM given observations $\{x_t\colon t = 1, 2, \ldots, T\}$. With knowledge of the likelihood, we shall estimate the model parameters through numerical maximization of the likelihood or other methods that require computable likelihood.

Consider a N-state HMM with initial distribution $\boldsymbol{\pi}$, transition matrix $\boldsymbol{\Gamma}$, and diagonalized conditional probability density functions p_i. The explicit form of the likelihood is then given by

$$L_T = \boldsymbol{\pi}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}'. \tag{3–10}$$

The result is quite elegant and we omit the derivation here, as it is not the main concern of this work. The reader shall refer to [5] for the proof of this conclusion. The reader should also notice that [5] provides details about the likelihood when observation data are missing at random or interval-censored. Corresponding adjustments should be taken to properly calculate the likelihood. Since the empirical analysis data that we will use in Ch. 5 are required to be complete and tidy, we avoid unnecessary explanations for these cases in this thesis.

3.2 Model estimation with Expectation Maximization algorithm

Knowing how to compute the likelihood of a HMM, we can estimate the model parameters by direct maximization of the likelihood, i.e. fulfill the task of model estimation with MLEs. Another commonly implemented method is the so-called Expectation Maximization (EM) algorithm.

The EM algorithm is specifically known as Baum-Welch algorithm within the area of HMMs, and the method is firstly proposed in [25–27]. The algorithm deals with Markovian hidden states series that is homogeneous but not necessarily stationary. Apart from the state transition matrix, the initial distribution $\boldsymbol{\pi}$ is also estimated and so is the conditional distribution probability density function. We will first introduce the forward and backward procedure in Sec. 3.2.1 and then present the full version of EM in Sec. 3.2.2.

3.2.1 Forward and backward procedure

Recall Eq. 3–10 for the form of likelihood:

$$L_T = \boldsymbol{\pi}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}',$$

and we provide the definitions of forward and backward probabilities as follows.

Definition 3.1. For $t = 1, 2, \ldots, T$, define the row vector $\boldsymbol{\alpha}_t$ as

$$\boldsymbol{\alpha}_t = \boldsymbol{\pi}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_t) = \boldsymbol{\pi}\mathbf{P}(x_1)\prod_{s=2}^{t}\boldsymbol{\Gamma}\mathbf{P}(x_s), \tag{3–11}$$

and refer to the elements as forward probabilities.

Definition 3.2. For $t = 1, 2, \ldots, T$, define the row vector $\boldsymbol{\beta}_t$ as

$$\boldsymbol{\beta}_t = \boldsymbol{\Gamma}\mathbf{P}(x_{t+1})\boldsymbol{\Gamma}\mathbf{P}(x_{t+2})\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}' = \left(\prod_{s=t+1}^{T}\boldsymbol{\Gamma}\mathbf{P}(x_s)\right)\mathbf{1}', \tag{3–12}$$

and refer to the elements as backward probabilities. The case $t = T$ yields $\boldsymbol{\beta}_T = 1$.

It is easy to find that we can rewrite Eq. 3–11 in the form of recursion that for $t = 1, 2, \ldots, T - 1, \boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t\boldsymbol{\Gamma}\mathbf{P}(x_{t+1})$. With derivation based on Bayesian formula (similar to 3–10,see [5]), we have the following proposition:

Proposition 3.1. *For $t = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, N$,*

$$\alpha_t(j) = \mathbb{P}(\mathbf{X}^{(t)} = \mathbf{x}^{(t)}, S_t = j). \tag{3–13}$$

The element $\alpha_t(j)$ is thus called the forward probability.

Similarly, we can rewrite Eq. 3–12 as that for $t = 1, 2, \ldots, T - 1, \boldsymbol{\beta}_t' = \boldsymbol{\Gamma}\mathbf{P}(x_{t+1})\boldsymbol{\beta}_{t+1}'$, and suggest the similar proposition:

Proposition 3.2. *For $t = 1, 2, \ldots, T-1$ and $i = 1, 2, \ldots, N$,*

$$\beta_t(i) = \mathbb{P}(\mathbf{X}^{(t+1:T)} = \mathbf{x}^{(t+1:T)} \mid S_t = i),\ \mathbb{P}(S_t = i) > 0. \tag{3–14}$$

The element $\beta_t(i)$ is thus called the backward probability.

Note that the forward probabilities are joint probabilities while backward probabilities are conditional probabilities.

With Eq. 3–10,3–13 and 3–14, we can rewrite the likelihood with forward and backward probabilities:

$$\alpha_t(i)\beta_t(i) = \mathbb{P}(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, S_t = i) \Longrightarrow L_T = \mathbb{P}(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \boldsymbol{\alpha}_t\boldsymbol{\beta}_t'. \tag{3–15}$$

The equation can be easily proved with Def. 3.1 and 3.2 and we omit the derivation here.

In order to apply the results to the EM algorithm, we also need the conditional probabilities of the hidden states given observations.

Proposition 3.3. *For $t = 1, 2, \ldots, T$,*

$$\mathbb{P}(S_t = j \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \frac{\alpha_t(j)\beta_t(j)}{L_T}, \tag{3–16}$$

and for $t = 2, \ldots, T$,

$$\mathbb{P}(S_{t-1} = j, S_t = k \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \frac{\alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)}{L_T}, \tag{3–17}$$

Results in Prop. 3.3 will be applied in both model estimation and decoding.

3.2.2 The EM algorithm

The traditional EM algorithm iteratively maximize the likelihood of the model and to find the MLEs under circumstances of missing data. Baum-Welch algorithm refers to this idea and consider the sequence of hidden states as missing data since they are unobserved.

We denote the model parameters as $\boldsymbol{\theta}$ and try to compute the complete-data log-likelihood (CDLL), i.e. the log-likelihood of parameters $\boldsymbol{\theta}$. Then EM can be carried out by two steps:

- **E step** calculates the expectations of (functions of) the missing data conditional on the observations and the current estimate of $\boldsymbol{\theta}$.
- **M step** maximizes the CDLL w.r.t. $\boldsymbol{\theta}$.

The iteration carries on until convergence, and the final $\boldsymbol{\theta}$ is the stationary point of the likelihood, which is the model parameter set that we want to figure out.

We define two indicators, of which the value is either zero or one.

$$u_j(t) = \begin{cases} 1 & \text{, iff } S_t = j \\ 0 & \text{, otherwise,} \end{cases} \quad \text{and} \quad v_{jk}(t) = \begin{cases} 1 & \text{, iff } S_{t-1} = j \text{ and } S_t = k \\ 0 & \text{, otherwise.} \end{cases} \tag{3–18}$$

Then we rewrite the likelihood of HMM, viewing hidden states $S_1, S_2, \ldots, S_T$ as missing data, in terms of the indicators and parameters:

$$\log\left(\mathbb{P}(\mathbf{x}^{(T)}, \mathbf{s}^{(T)})\right) = \underbrace{\sum_{j=1}^{N} u_j(1) \log \pi_j}_{\text{term 1}} + \underbrace{\sum_{j=1}^{N}\sum_{k=1}^{N}\left(\sum_{t=2}^{T} v_{jk}(t)\right) \log \gamma_{jk}}_{\text{term 2}} + \underbrace{\sum_{j=1}^{N}\sum_{t=1}^{T} u_j(t) \log p_j(x_t)}_{\text{term 3}}. \tag{3–19}$$

Iterative maximization of the CDLL will provide us information about both $\boldsymbol{\pi}$, the initial distribution and $\boldsymbol{\Gamma}$, the state transition matrix.

Apply Eq. 3–19 to the general EM and we can specify the two steps as

- **E step** replaces the two indicator vectors by their conditional expectations given observations $\mathbf{x}^{(T)}$ and the current parameter estimates

$$\hat{u}_j(t) = \mathbb{P}(S_t = j \mid \mathbf{x}^{(T)}) = \frac{\alpha_t(j)\beta_t(j)}{L_T}, \tag{3–20a}$$

$$\hat{v}_{jk}(t) = \mathbb{P}(S_{t-1} = j, S_t = k \mid \mathbf{x}^{(T)}) = \frac{\alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)}{L_T}. \tag{3–20b}$$

- **M step** maximizes the CDLL, Eq. 3–19, w.r.t. the initial distribution $\boldsymbol{\pi}$, the state transition matrix $\boldsymbol{\Gamma}$, and conditional distribution parameters which is embedded in $p_j(x_t)$ in term 3.

Notice that we have split the computation of CDLL into three terms so that the maximizations w.r.t. the three parameter sets are independent. Furthermore, the separation guarantees the existence of analytic solutions. We present the sub-problems and corresponding solutions as follows.

3.2.2.1 Initial distribution

By term 1 in Eq. 3–19, consider the maximization problem

$$\max_{\boldsymbol{\pi}} \sum_{j=1}^{N} u_j(1) \log \pi_j, \tag{3–21}$$

and the solution is given as

$$\pi_j = \frac{\hat{u}_j(1)}{\sum_{j=1}^{N} \hat{u}_j(1)} = \hat{u}_j(1). \tag{3–22}$$

3.2.2.2 State transition matrix

By term 2 in Eq. 3–19, consider the maximization problem

$$\max_{\boldsymbol{\Gamma}} \sum_{j=1}^{N} \sum_{k=1}^{N} \left(\sum_{t=2}^{T} v_{jk}(t) \right) \log \gamma_{jk}, \tag{3–23}$$

and the solution is given as

$$\begin{aligned} \gamma_{jk} &= \frac{f_{jk}}{\sum_{k=1}^{N} f_{jk}}, \\ \text{where} \quad f_{jk} &= \sum_{t=2}^{T} \hat{v}_{jk}(t). \end{aligned} \tag{3–24}$$

3.2.2.3 Conditional distribution

By term 3 in Eq. 3–19, consider the maximization problem

$$\max_{\boldsymbol{\theta}} \sum_{j=1}^{N} \sum_{t=1}^{T} u_j(t) \log p_j(x_t; \boldsymbol{\theta}). \tag{3–25}$$

The solution to this sub-problem depends on the formulation of the conditional distribution, and is deduced through common MLE method. We provide solutions to two kind of distributions here, one for Poisson and the other for normal.

For Poisson distributions we have $p_j(x) = e^{-\lambda_j} \frac{\lambda_j^x}{x!}$, the solution is then given as

$$\hat{\lambda}_j = \frac{\sum_{t=1}^{T} \hat{u}_j(t) x_t}{\sum_{t=1}^{T} \hat{u}_j(t)}. \tag{3–26}$$

For normal distributions we have $p_j(x) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\{-\frac{(x-\mu_j)^2}{2\sigma_j^2}\}$, the solution is then given as

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} \hat{u}_j(t) x_t}{\sum_{t=1}^{T} \hat{u}_j(t)}, \tag{3–27a}$$

$$\hat{\sigma}_j = \frac{\sum_{t=1}^{T} \hat{u}_j(t) (x_t - \hat{\mu}_j)^2}{\sum_{t=1}^{T} \hat{u}_j(t)}. \tag{3–27b}$$

The results above are thus easy to compute and can be directly applied in the numerical realization procedures.

3.3 Prediction and decoding

The series prediction is essentially an application of the forecast distributions. With model parameters estimated, we can fulfill the prediction by one-step or multiple-step forecast. We cover this topic in Sec. 3.3.1.

Information of the historical hidden states distributions can be also deduced. The process of find the states occupied by the Markov chain is referred to as decoding, Decoding can be further classified as local ones and global ones in terms of the likelihood to be maximized. Both decoding procedures are discussed in this section. Specifically, we introduce the Viterbi algorithm that is implemented for global decoding in Sec. 3.3.3.

3.3.1 Forecast distributions

Consider h-step forecast distribution conditional on the observations

$$\begin{aligned}\mathbb{P}(X_{T+h} = x \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) &= \frac{\mathbb{P}(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, X_{T+h} = x)}{\mathbb{P}(\mathbf{X}^{(T)} = \mathbf{x}^{(T)})} \\ &= \frac{\boldsymbol{\alpha}_T \boldsymbol{\Gamma}^h \mathbf{P}(x)\mathbf{1}'}{\boldsymbol{\alpha}_T \mathbf{1}'}.\end{aligned} \tag{3–28}$$

In order to perform one-step prediction, we can simply assign h with value one and obtain the result.

Notice that when h increases, the result stated in Eq. 3–28 will converge due to the convergence of Markov chain state transition matrix, i.e.

$$\lim_{h\to\infty} \mathbb{P}(X_{T+h} = x \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \boldsymbol{\delta}^* \mathbf{P}(x)\mathbf{1}', \tag{3–29}$$

where $\boldsymbol{\delta}^*$ represents the stationary distribution of the Markov chain (see Sec. 2.2.3). One of the direct influences of the convergence is that we cannot take a too big h in our stock return series prediction system, or the prediction results would be smoothed due to the converging property.

3.3.2 State probabilities and local decoding

The concept of decoding is firstly adopted in speech recognition areas [28], and, in the context of HMM, refers to the procedure to find the most probable sequence of hidden states that generates the observation series. The idea ‘local decoding’ of the state at a specific time stands for the determination of the most possible state at that time node.

Recall Eq. 3–16

$$\mathbb{P}(S_t = j \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}) = \frac{\alpha_t(j)\beta_t(j)}{L_T},$$

we are able to compute the likelihood L_T with EM algorithm, and therefore able to calculate the state probabilities and decide the most probable state at each t:

$$S_t^* = \underset{j=1,2,\ldots,N}{\text{argmax}}\ \mathbb{P}(S_t = j \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}). \tag{3–30}$$

3.3.3 Global decoding and Viterbi algorithm

Local decoding only provides information of the most probable state for each separate time t but tells nothing about the most likely sequence of the states over the entire period. Global decoding solves a slightly different maximization problem as local decoding, and the result is often similar but not exactly identical.

The global decoding procedure enables us to find the conditional probability of the entire hidden state series given the observations, i.e.

$$\mathbb{P}(\mathbf{S}^{(T)} = \mathbf{s}^{(T)} \mid \mathbf{X}^{(T)} = \mathbf{x}^{(T)}). \tag{3–31}$$

The maximization problem can be solved with an algorithm named after Viterbi [29], which is a very effective dynamic programming technique. Define the probabilities ξ_{ti} as follows:

$$\begin{aligned} \xi_{1i} &= \mathbb{P}(S_1 = i, X_1 = x_1) = \pi_i p_i(x_1), \\ \xi_{ti} &= \max_{s_1,s_2,\ldots,s_{t-1}} \mathbb{P}(\mathbf{S}^{(t-1)} = \mathbf{s}^{(t-1)}, S_t = i, \mathbf{X}^{(T)} = \mathbf{x}^{(T)}), \end{aligned} \tag{3–32}$$

for $t = 2, 3, \ldots, T$. We can also rewrite Eq. 3–32 in a recursive form:

$$\begin{aligned} \xi_{1i} &= \mathbb{P}(S_1 = i, X_1 = x_1) = \pi_i p_i(x_1), \\ \xi_{tj} &= \left(\max_i(\xi_{t-1,i}\gamma_{ij})\right) p_j(x_t), \end{aligned} \tag{3–33}$$

for $t = 2, 3, \ldots, T$ and $i = 1, 2, \ldots, N$. The $T \times N$ matrix enables us to solve the maximization problem with computational complexity that is linear in T. The result can be solved recursively from:

$$\begin{aligned} S_T &= \underset{i=1,2,\ldots,N}{\text{argmax}}\ \xi_{Ti}, \\ S_t &= \underset{i=1,2,\ldots,N}{\text{argmax}}\ (\xi_{ti}\gamma_{i,S_{t+1}}), \end{aligned} \tag{3–34}$$

for $t = T-1, T-2, \ldots, 1$.

The maximization can also be rewritten in form of logarithms, and scaling problems should be noticed to avoid numerical underflows. This problem is explained in [5] with details.

Chapter 4 Stock Return Series Prediction System

This chapter explains in details about the composition of the stock return series prediction system and functionality of each module. Sec. 4.1 firstly present an overview of the system, introducing the entire data flow and module connection. Then, in Sec. 4.2, structure of each part is analyzed, including its intention, input and output, and settings specific to this thesis. Sec. 4.3 focuses on some of these settings and proposes several ideas concerning potential for improvement. The full version of the system realization with Python programming is appended to the thesis in Appendix B.

4.1 Overview

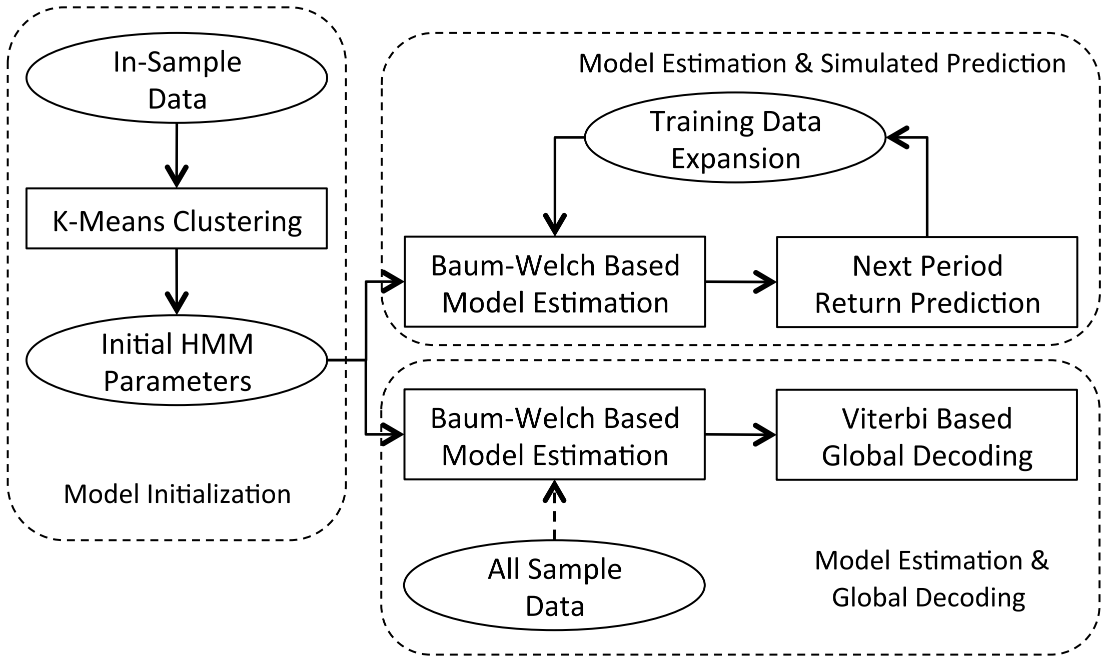

In-Sample Data
K-Means Clustering
Initial HMM Parameters
Model Initialization
Model Estimation & Simulated Prediction
Training Data Expansion
Baum-Welch Based Model Estimation
Next Period Return Prediction
Baum-Welch Based Model Estimation
Viterbi Based Global Decoding
All Sample Data
Model Estimation & Global Decoding

ILLUSTRATION 4–1 Overview of the stock return series prediction system

The stock return series prediction system is composed of two major parts, model initialization, and simulated prediction; the latter part can be further split into two sub-parts, model estimation with Baum-Welch (EM) algorithm and prediction based on the model. Apart from these two major parts, a global decoding module is also included in the system with purpose to conducted analysis of sample data within the entire observation period.

Illus. 4–1 is a brief overview of the system. The organization of the system are constructed and will be explained accordingly as follows

- **Model initialization,** including data pre-processing, initial guesses for parameters based on K-Means clustering.
- **Simulated prediction,** including two sub-parts:
  1. **EM based model estimation,** i.e. estimation of the state transition matrix and conditional distribution parameters with EM algorithm.
  2. **Prediction,** for stock return of the next period based on aforementioned model estimation results.
- **Global decoding,** including model estimation and decoding based on Viterbi algorithm. The model in this part takes in all sample data in order to provide analysis of the entire period and the most likely sequence of hidden states.
- **Results saving and visualization.** This part is significant to the system in practice, but needless to be explained in details. All numerical results are exported in the form of `.csv` files and visualizations are carried out on the results.

4.2 System composition and functionality

This section is focused on detailed descriptions of each part listed above in Sed. 4.1. The part of results saving and visualization is, however, omitted in this section since the process is more about convenience and intuition while it has very little to do with the system itself.

4.2.1 Model initialization

The data pre-processing work include data downloading and tidying. A complete input data file consists of two columns, the first standing for the timeline and the second for the observed closing prices. Timeline should be uniformed for each trading day (or trading year in daily cases), and thus missing value of closing prices should be filled. Here we fill the missing ones with previous values, but it is not the only method and sometimes linear interpolation or even spline interpolation are implemented for data filling (e.g. see [30, 31]).

In order to construct a K-Means model, further adjustment and transformation should be made on the raw data. Firstly we calculate the log return series out of the price series, and then estimate the standard deviation on each observing time node through calculating the rolling historical standard errors, as is shown in Eq. 4–1

$$\begin{aligned} \text{log return} : r_t &= ln(\frac{P_t}{P_{t-1}}),\ t = 1, 2, \dots, \\ \text{standard deviation} : \sigma_t &= \sqrt{\frac{1}{nf-1} \sum_{i=t-nf+1}^{nf} (r_i - \bar{r})^2},\ t \geq nf. \end{aligned} \tag{4–1}$$

where n is the number of rolling days and f represents the number of data within each trading day. For data with different observation frequency, f vary (e.g. $f = 1$ for daily data and $f = 24$ for 10min data) but we keep n the same; more specifically, we choose $n = 5$ in this system.

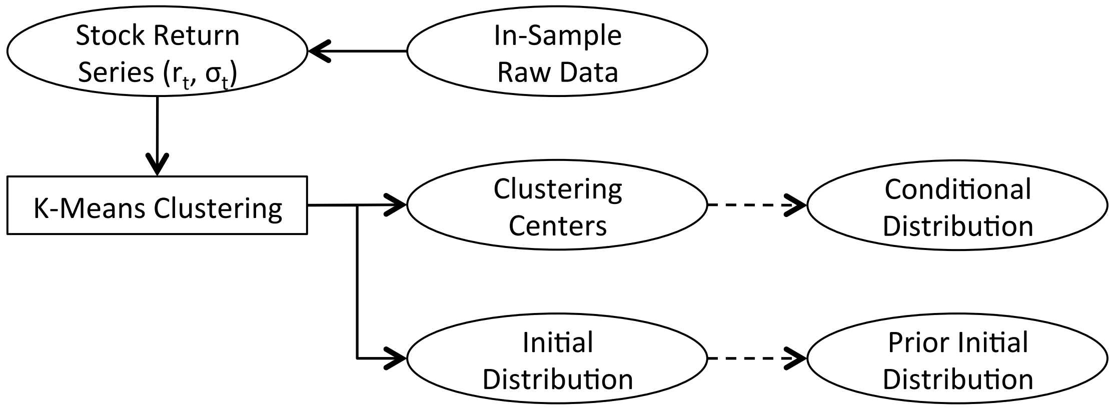


ILLUSTRATION 4–2 Model initialization module

Now K-Means model takes in a two dimensional time series, i.e. the stock return series and 5-day historical standard deviation series. Given the number of states, k, the model generates k clustering centers and corresponding probability distribution, which are then taken as initial guesses for HMM conditional distribution parameters and initial state distribution separately.

Initial guesses for each entry of the transition matrix is arbitrarily set as equal. The process follows the steps in Illus. 4–2

Notice that K-Means is a very common method to find the prior distribution of the parameters [32], while there are also some other methods to find the prior, e.g. generating random vector satisfying the numerical constraints [33], or introduction of machine learning techniques like artificial neural networks (ANN) and genetic algorithms (GA) [34].

4.2.2 Simulated prediction

Simulated prediction module processes with a time loop, containing two parts within each loop, model estimation and prediction. Every time a loop is finished, we incorporate the real historical return of the next period into the training data set and process the time horizon to the next period, estimate the model with the new data set in order to predict the return of one more next period. The iterations is visually described in Illus. 4–1.

4.2.2.1 EM based model estimation

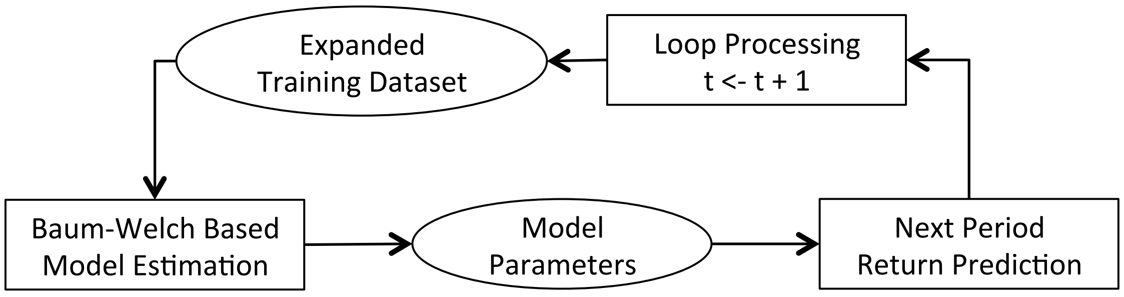

Expanded Training Dataset
Loop Processing t <- t + 1
Baum-Welch Based Model Estimation
Model Parameters
Next Period Return Prediction

ILLUSTRATION 4–3 EM based model estimation procedure

Model estimation refers to find the estimations of model parameters, including state transition matrix parameters (each entry of the matrix) and conditional distribution parameters (probability density distributions of observed variables conditional on each state). The method is fully explained in Sec. 3.2 so we avoid the unnecessary repetition here. Visual description of the procedure is shown in Illus. 4–3.

There is something though should be cleared. As we mentioned before, initial guesses for model parameters of the very first loop is found with K-Means clustering based on in-sample data. We use the same initial guesses for all of the following loops, i.e. initialization is carried out only once and we no more perform K-Means on the expanded training dataset. Besides, using the same initial guesses also means that the EM algorithm starts from the same parameters for different iterations. Another way to improve the initial guesses is to use the estimation results from the previous loop as the initial guess for the next one, which shall largely relieve the computational burden and reduce the running time of the system. However, the change of the training dataset is so small that it is reasonable to assume that model parameters change very little. Thus, it is possible that the initial guess is close enough to the final result so that the algorithm prematures due to precision limits, and that is why the system does not take this way to set the initial guesses.

4.2.2.2 Prediction

Prediction for the next return is calculated through the estimation results found with EM. The calculation is simply put as follows:

$$r_{t+1} = \boldsymbol{\delta}'\boldsymbol{\Gamma}\boldsymbol{\mu}, \tag{4–2a}$$

$$\sigma_{t+1} = \sqrt{\boldsymbol{\delta}'\boldsymbol{\Gamma}\boldsymbol{\sigma}^2}, \tag{4–2b}$$

where

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N) \tag{4–3}$$

are expectations of the conditional distributions, and

$$\boldsymbol{\sigma}^2 = (\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2) \tag{4–4}$$

are variances of the conditional distributions, and $\boldsymbol{\delta}$ represents the final distribution of the hidden state and $\boldsymbol{\Gamma}$ stands for the state transition matrix. Notice that an implied assumption of Eq. 4–2b is that the distributions conditional on hidden states are independent so the variances are added without covariance terms. The prediction results are then to be compared with the actual historical returns and corresponding standard deviations are used to find the confidence band.

There are two ways to plot the prediction results. The first one is to fix the price level at the last in-sample date, of which the stock price is denoted as P_0. We denote the predicted returns as $\hat{r}_i, i = 1, 2, \ldots, t$, and thus the prediction result of $\hat{P}_t$ is that

$$\hat{P}_t = P_0 e^{\sum_{i=1}^{t} \hat{r}_i}. \tag{4–5}$$

The second one is called the adaptive prediction, which incorporates not only the information of returns but also prices of the data. In order to predict the price at time t, we firstly predict the return $\hat{r}_t$ and compute the price given the price at time $t-1$, that is

$$\hat{P}_t = P_{t-1} e^{\hat{r}_t}. \tag{4–6}$$

In the first kind of prediction, errors of return series predictions accumulate, so it is very likely that the adaptive prediction performs much better. We will examine this idea in Ch. 5.

4.2.3 Global decoding

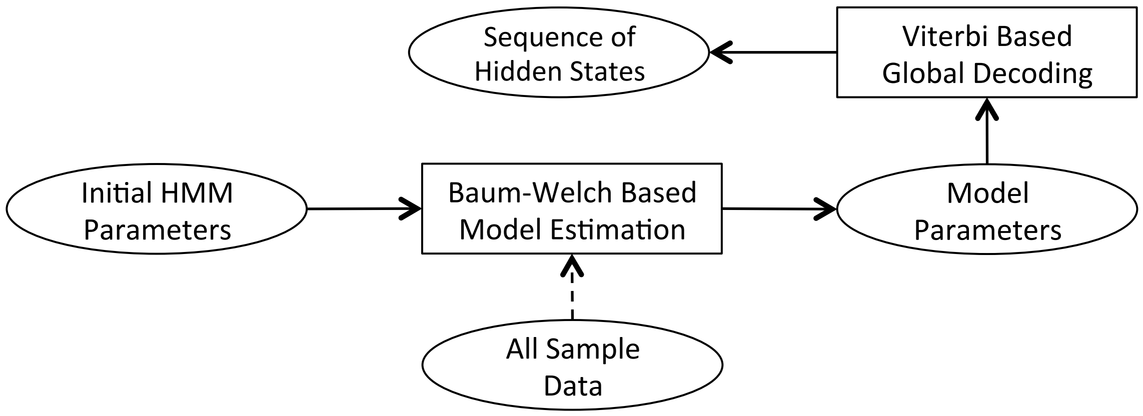

Sequence of Hidden States
Viterbi Based Global Decoding
Initial HMM Parameters
Baum-Welch Based Model Estimation
Model Parameters
All Sample Data

ILLUSTRATION 4–4 Global decoding module

The global decoding module is independent with the simulated prediction module, and is not the major focus of the prediction system. The aim of this module is to provide a HMM-based description of sample data over the entire observation period. Traditional methods enable us to know elementary statistics about the data (e.g. mean, variance, skewness, etc.), but with Viterbi algorithm we are able to know how the hidden states are distributed across the history and whether the result matches our conjectures and common senses.

The procedure is illustrated in Illus. 4–4. With initial HMM parameters found in the model initialization module (Sec. 4.2.1) we set up the model with all sample data [1]. The Viterbi algorithm based decoding is then conducted to deduce the most probable sequence of hidden states during the observation period.

4.3 Potential for improvement

Heretofore the system is complete to perform analysis and predictions for the stock return series. Well functioning as the system is, it can be further improved and perfected for better performances, e.g. more thorough description of the dataset and more correct and accurate predictions. We propose several perspectives where potential for improvement may exist.

4.3.1 Multiple observed variables

Currently the input for the system is a simple observation series on only one variable, i.e. the log return of the stock, derived from its closing price. However, the information embedded in the single closing price series is so little that it cannot capture many of the properties of the stock. One way to improve the case is to incorporating more observed variables, e.g. opening, low, high and closing prices, and to find the conditional observing probability with Gaussian mixture models (GMM) [32, 33, 35]. One of the formulations from [35] can be generalized as

[1] Initial parameters are found with in-sample data instead of all of them. Considering the length of out-of-sample data is much shorter than the in-sample, and the fact that initial guesses do not have large effect on the eventual optimal results, we choose to remain the initial guesses rather than perform another initialization based on all sample data

follows:

$$\mathbf{O}_t = (\frac{\text{close} - \text{open}}{\text{open}}, \frac{\text{high} - \text{open}}{\text{open}}, \frac{\text{open} - \text{low}}{\text{open}}) \tag{4–7a}$$

$$b_j(\mathbf{O}_t) = \sum_{n=1}^{N} c_{jn} N(\mathbf{O}_t, \boldsymbol{\mu}_{jn}, \boldsymbol{\sigma}_{jn}), \tag{4–7b}$$

where $\mathbf{O}_t$ is the vector of observed variables (observed vector), b is the observing probability of $\mathbf{O}_t$ conditional on the j^{th} state, c is the GMM weight and $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are GMM parameters.

The idea could be furthered to incorporate more factors apart from the observing prices, such as financial indicators or economic indicators. For example, [36] presents a way to include Boltzman chain, a generalized HMM, into the multi-factor binding genome model. Including more observed variable may require some other statistical techniques since the Gaussian distribution assumption could fail for these variables. Principal component analysis (PCA) and machine learning theories like neural network (NN) are commonly implemented methods [34, 37, 38].

4.3.2 Multiseries HMM

In order to include multiple observed variables, the conditional distributions should be accordingly designed and modified. Another way to make multiple observed variables compact with the original Gaussian distribution is to introduce the multiseries HMM model, where the observation is a series of vectors rather than one-dimensional time series. See [5] for more details.

4.3.3 Multiple latent variables

Apart from improvements from the angle of observed variables, the latent variable, i.e. the hidden states, can also be improved.

The latent variable we use in this thesis is categorical, which means it merely represents the set of hidden states, while the states are just labels without realistic statistical meanings. However, the hidden states can be expressed in other ways that they are mathematically meaningful, and so the latent variable is included in the conditional distribution function of the observed variables as a parameter, either discrete or continuous.

For example, time-variation properties are considered for the discrete latent variables in [39]; properties and theories related to continuous latent variables are introduced in [40]. We will further discuss about the topic in Sec. 7.2.

Like observed variables, the number of latent variables can be over one so can the dimension of it, and we do not expand the topic here.

Chapter 5 Empirical Analysis on Stock Market Indices

In this chapter we present the numerical results of HMM application for real-world data. empirical analysis is conducted on both Chinese and U.S. stock market indices. We first make several base assumptions on the model and realizations, and propose some conjectures on the results, including model effectiveness, prediction correctness, and possible differences caused by diverse data frequency and observation time period, etc. We then perform empirical analysis on Standard & Poor's 500 Index (S&P 500) and present both numerical and visual results in Sec. 5.2. Similar analysis is also carried out on Chinese CSI 300 Index in Sec. 5.3. We take a further look at the Chinese market and apply the model for data with higher frequencies. An overall analysis is presented in Sec. 5.6 along with comparisons among different data groups.

5.1 Assumptions and conjectures

In this section we specify some basic but necessary assumptions for the model application and empirical analysis, and also propose some conjectures on the prediction result, including differences in behaviors, potential causes of prediction errors, etc.

5.1.1 Assumptions

The main parameters of a hidden Markov model include the state transition matrix and conditional distribution density functions, of which dimensions are predetermined by the number of hidden states and values are computed through model estimation procedure.

Assumption 5.1. *The U.S. and Chinese market both have and only have three hidden states, namely bear, intermediate and bull.*

This assumption is intuitively easy to explain. Usually we tend to describe the market as a bull if the index keeps rising and a bear if it goes down all the time. Sometimes the trend of the index is not that obvious, daily returns dangle around zero and market volatility is low, this is when we consider the market as in an intermediate state. Assuming the market have exactly three states suits customs and has realistic economic meanings.

However, taking the number of states as three might not be statistically optimal, and we will talk about it in Sec. 5.6.5.

Assumption 5.2. *Stock returns conditional on the market state are normally distributed. Distribution parameters (expectation μ and variance σ^2) vary across hidden states and time.*

Assumption 5.2 follows traditional economic and financial ideas and is accepted in most models. Besides, the normal distribution assumption guarantees the convergence of EM algorithm and the existence of analytic solution to iterative log-likelihood maximizations (see Sec. 3.2).

Some more advanced models have been proposed, considering the skewness and kurtosis of real-world returns' sample density distribution, and [41] presents detailed analysis and comparisons of them. The topic is beyond the scope of this work so no further discussions will be made, and we stick to the normal distribution assumption throughout this thesis.

5.1.2 Conjectures

Based on the system described in Sec. 4.2, we propose some conjectures about the model calibration and simulated prediction results.

Conjecture 5.1. Prediction results are good at first, and might worsen as processing. More specifically, the prediction result may present similar trend and pattern as the historical data that is used for initial model calibration, which can be viewed as time lag effect, and that is because the model incorporates too much out-dated information.

Conjecture 5.1 is one of the potential sources of prediction errors (and maybe the largest). As described in Sec. 4.2.2, every time the prediction processes to the next loop, i.e. the next out-of-sample time node, it takes one more point into the model, but not replaces the oldest one. Thus, all information in the past still exist, and worse, all data takes the same weight in the model. The data from two years ago is as significant as one from yesterday when predicting the return of tomorrow, and that is quite against intuition. Too much weight on out-dated data shall be reflected as time lags in prediction result. Potential improvement for this problem will be proposed in Sec. 7.1.

Conjecture 5.2. Prediction correctness vary with time periods of the data, i.e. model calibrated for different time periods will have differences in prediction accuracy (ceteris paribus[1]).

Conjecture 5.2 is proposed from the perspective of time. Although market states switch to each other, each state lasts for some time and the endurings are not necessarily short. Hence, it is very common (almost sure) that the distribution of states vary every year (maybe less or more frequently); even though there should be economy cycles, hidden market states changed and differed from previous years. So it is possible that the model have different effectiveness and prediction correctness when calibrated with market data from different time period. We will try to catch the difference by separating data into different time period groups and applying the model to each group. Detailed result analysis and comparisons will be given in Sec. 5.6.3.

Another thing worth mentioning is that the differences between the time groups could also be supportive evidence for Conjecture 5.1. Distinguishing the groups also means cutting off data from the previous time period, and the results from the sub-periods and those from the entire sample period may differ. However, the "support" here can hardly meet the requirement of ceteris paribus. We merely provide a point of view here and further analysis and experiments should be conducted if to prove this opinion, which is not the major concern of this thesis.

Conjecture 5.3. Global decoding results may vary with data frequency. For a certain time period, hidden states sequence may be found different using data with diverse frequency.

Conjecture 5.4. Prediction results may vary with data frequency. More specifically, data with higher frequency may generate better prediction results.

Conjecture 5.3 and 5.4 are concerned with data frequency within the same time period. For data within the same time period, difference in observation frequencies represents different amount

[1]Meaning holds all other conditions the same. It is important to do so for results comparisons and to draw meaningful conclusions.

of information. Usually is the case that the higher the frequency is, the more information the data contains. Thus we can also examine the differences due to various observation frequencies. To be more specific, we suppose that prediction correctness increase with data frequency. Examination of these two conjectures will be given in Sec. 5.6.4.

5.2 U.S. S&P 500 Index daily return series

S&P 500 is one of the oldest and the most important stock market indices in the world, and it is safe to say that it has the most typical behaviors of a mature stock market. We firstly apply our model to S&P 500 and then CSI 300 so as to compare the effectiveness of our model under different market maturity.

5.2.1 Data preprocessing and description

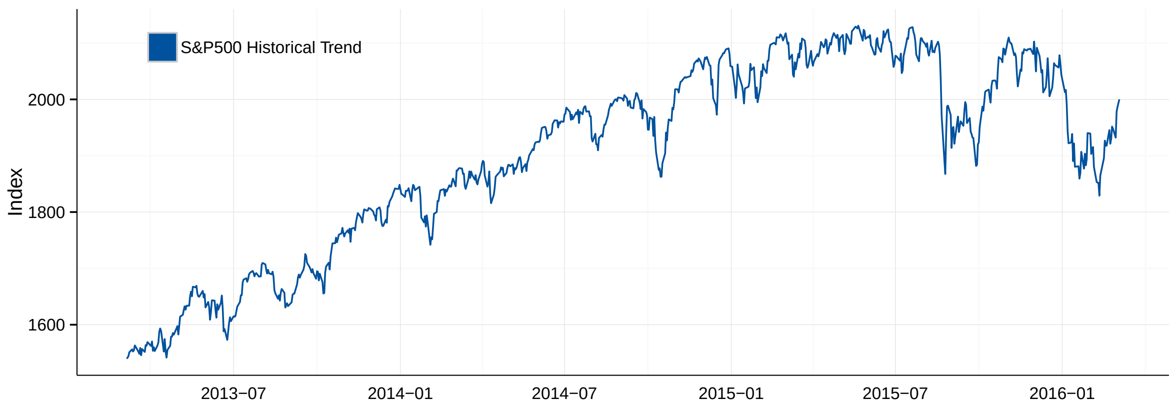


ILLUSTRATION 5–1 S&P 500 historical prices

Sample data of S&P 500 is composed of its daily close between Mar. 5^{th}, 2013 and Mar. 4^{th}, 2016 [2], i.e. 757 prices within three years in total. Of all 757 records, two thirds of them (data before Mar. 6^{th}, 2015) are used as in-sample data for K-Means model training and HMM initialization. The remaining one third are reserved to compare with the simulated prediction results. Illus. 5–1 presents the historical prices and trends of S&P 500 during all three years. The index

[2]Data source: Yahoo Finance

had been going up steadily, though with fluctuations, since the very beginning until around July, 2015. It then suffered a large downturn followed by a short period of turbulence, and then a large upgoing trend. Generally speaking, the index presented quite different behaviors, in terms of both trend and volatility, between in-sample data and out-of-sample data.

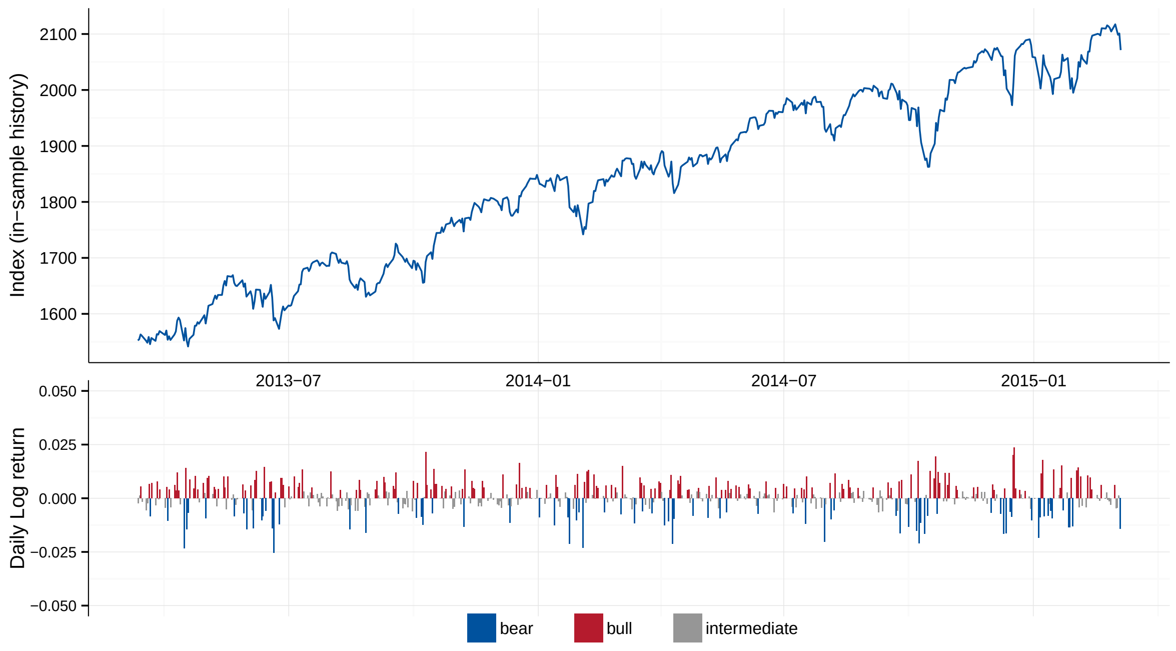


ILLUSTRATION 5–2 S&P in-sample trend with daily return bars

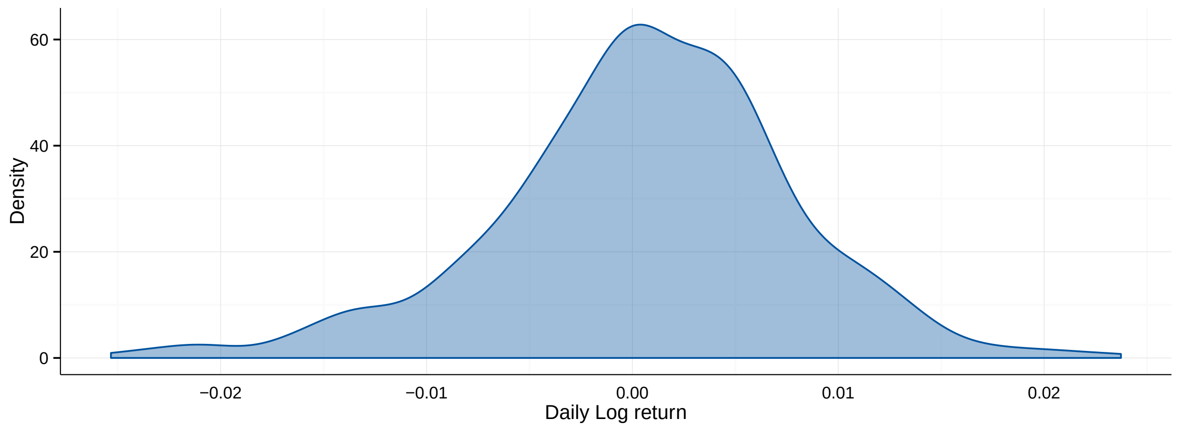


ILLUSTRATION 5–3 Density distribution of S&P in-sample daily returns

Now take a look at the in-sample period. Density distribution of daily returns is almost bell-curve shaped, with slight skewness to the right (see Illus. 5–2 and 5–3). Almost all returns are within the range $[-2.5\%, 2.5\%]$ and with 5-day volatility less than 2% (daily basis, indicated in Illus. 5–4).

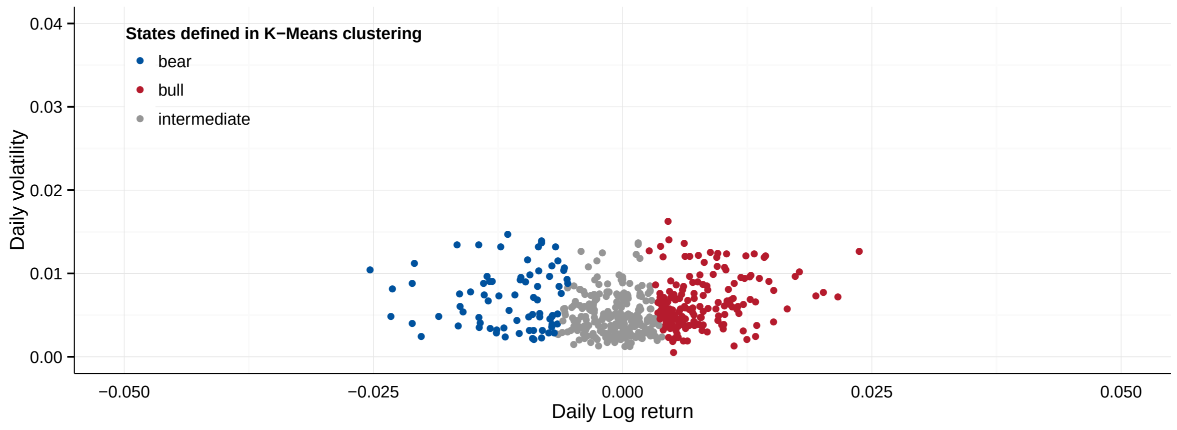


ILLUSTRATION 5–4 K-Means result of S&P 500 in-sample returns

The result of K-Means clustering is pretty straightforward as it classifies the lowest one third of the returns as bear, the highest one third as bull and the remaining as in the intermediate state. Note that the clustering of K-Means method is purely based on quantitative differences of the data (daily return and corresponding 5-day volatility in this case), and has nothing to do with the market states then. The definition and names of the states are artificially given to merely distinguish the three clustering centers.

5.2.2 State transition parameters

The three by three state transition matrix has nine unknown parameters in total, estimated by EM algorithm through iteration. Illus. 5–5 presents four chord plots of the state transition matrix for S&P 500 in-sample data.

The first plot in Illus. 5–5 visualizes a complete state transition matrix. Three different colors each stands for a hidden state. Every chord links two arcs: the arc with the same color as the chord represents the current state, and the other one represents the next state. Since HMM allows for stays of the latent variable, the state transition matrix may have non-zero diagonal entries, leading to chords that travels from and to the same arc.

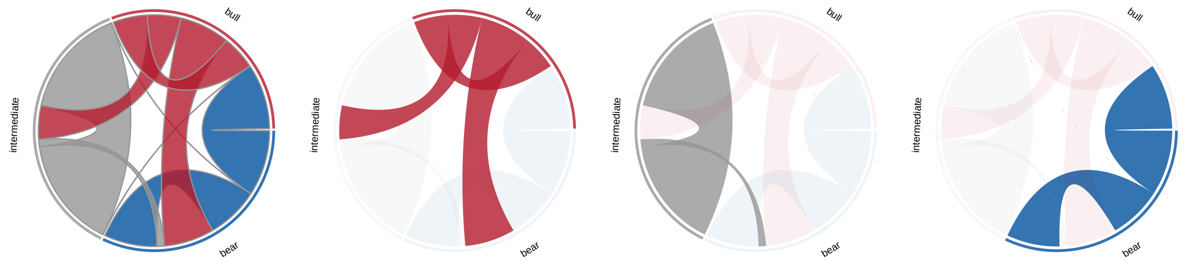


ILLUSTRATION 5–5 State transition for S&P 500 in-sample data

The latter three plots each shows the transition vector conditional on the current state. As can be seen, a bull market is most likely to enter a bear state (with probability of 42%), and to stay put or transit into an intermediate state with the same probability (of 29%). However, a market at intermediate state stays at intermediate with great chance (about 93%), but is almost impossible to change into a bull market (with probability less than 10^{-5}). A bear market, on the other hand, is more likely to transit into a bull than stay still with a slightly higher probability (55% vs. 45%) and has very little chance to go intermediate (with probability less than 10^{-6}). Transitions between extreme cases (bull and bear) seem more likely to happen, compared to transitions between the intermediate state and an extreme case.

These patterns indicate a highly volatile market, where it is more possible to meet with a rebounce rather than a trend (except the intermediate state, which might represent lack of trends). This fact can be indirectly proved by Illus. 5–2. The index kept going up for two years and the trend seems quite obvious; but upturns and downturns, not only small waves but also large ones, are densely distributed within the period. The volatility of the market will be more clear if one takes into account inflation.

5.2.3 Conditional distribution parameters

Another type of unknown parameters are those of the Gaussian distributions conditional on each hidden state. Conditional distribution parameters include three pairs of expectation and variance (or standard deviation). The optimal estimation during each iteration can be analytically computed through EM (see Sec. 3.2).

The estimation result of S&P 500 in-sample is given as follow:

$$\begin{pmatrix} -1.13\% & -0.08\% & 0.78\% \\ 0.70\% & 0.49\% & 0.65\% \end{pmatrix}. \tag{5–1}$$

Each column in order represents the bear state, the intermediate and the bull. Entries in the first row are corresponding expectations and ones in the second row are standard deviations. It is clear that returns during the bear and bull have higher variances and non-zero expectations, while returns during the intermediate are closer to zero and have lower volatility.

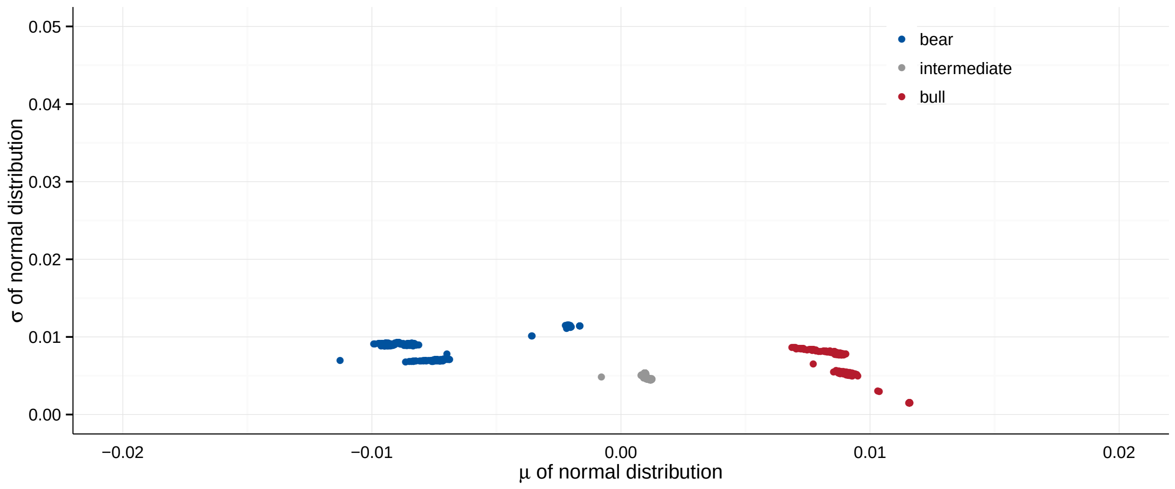


ILLUSTRATION 5–6 Conditional distribution parameters across time

We also consider time-variation of the parameters. As a matter of fact, when the simulated prediction is processing, more information has been incorporated into the model and parameters change with time. Illus. 5–6 and 5–7 are to describe the conditional distribution parameters time series.

The aforementioned patterns about the parameters remain across time. Bear(bull) states have expected returns significantly below(over) zero and higher volatility, while intermediate states have expectations around zero and a relatively lower volatility. Notice that some distributions conditional on bull states also have smaller variance, indicating a slower and milder rise. Clearly bears always come more strongly.

Illus. 5–7 shows the density distribution of these parameters. The multi-modality of σ matches

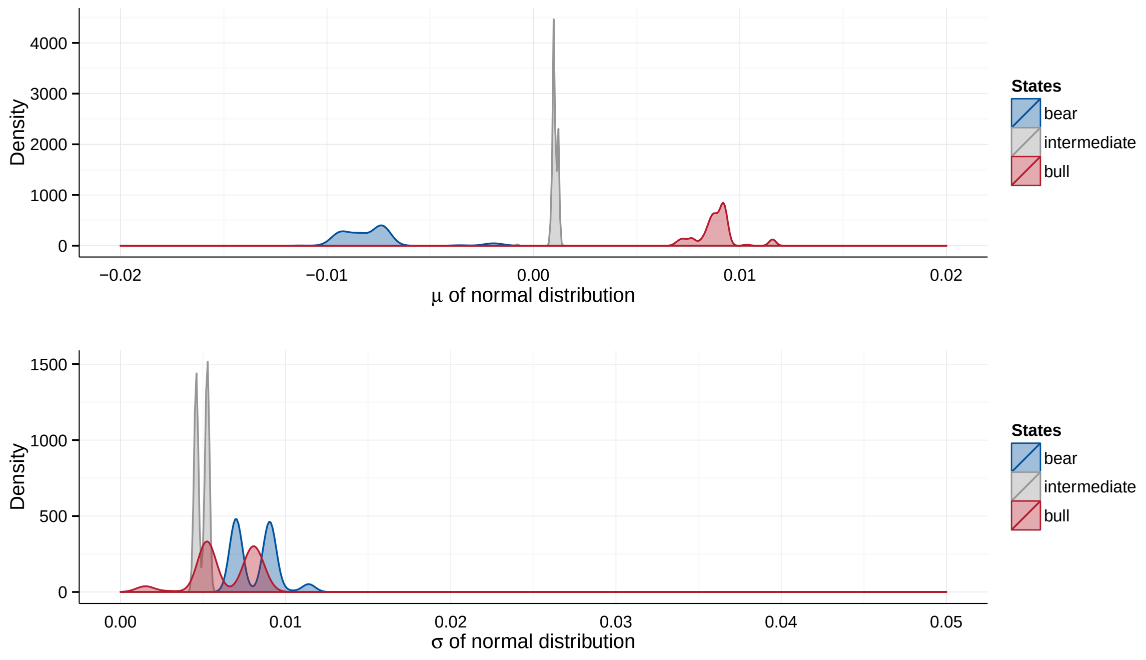


ILLUSTRATION 5–7 Density distribution of conditional distribution parameters

the multi-clustering-center in Illus. 5–6. Therefore, chances are that even the same hidden state presents different behaviors. It means it might be necessary to increase the number of hidden states in the model, which is currently against Assumption 5.1 and will be discussed about in Sec. 5.6.5.

5.2.4 Global decoding and hidden states

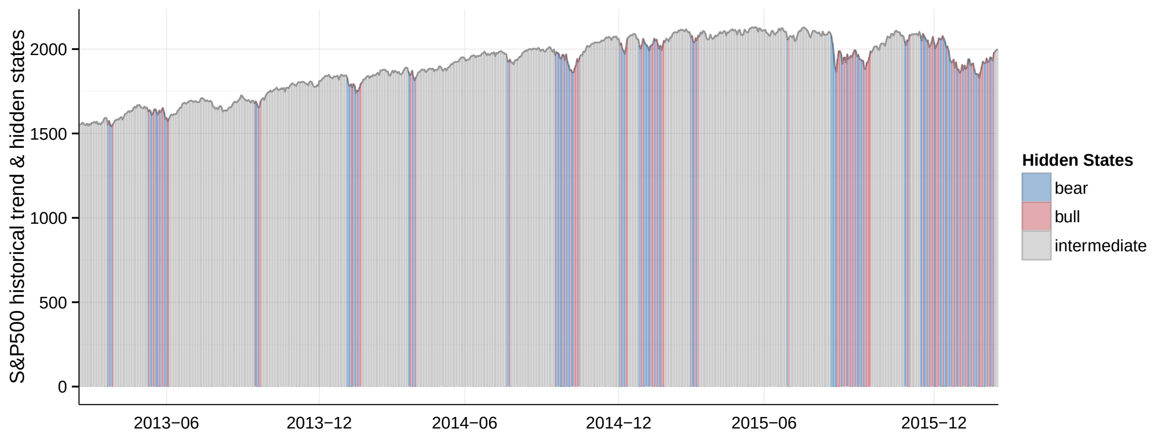


ILLUSTRATION 5–8 S&P 500 in-sample trend and states sequence

Global decoding based on Viterbi algorithm enables us to find the most probable sequence of hidden states, which shall help us learn about the market with ability to identify the market states during the observation period.

Illus. 5–8 provides knowledge about the sequence of hidden states of S&P 500 in-sample data. The color gray, standing for the intermediate state, makes up the majority of the figure. Bears and bulls are far less frequent than intermediates and often come together. This phenomenon is better expressed by Illus. 5–9

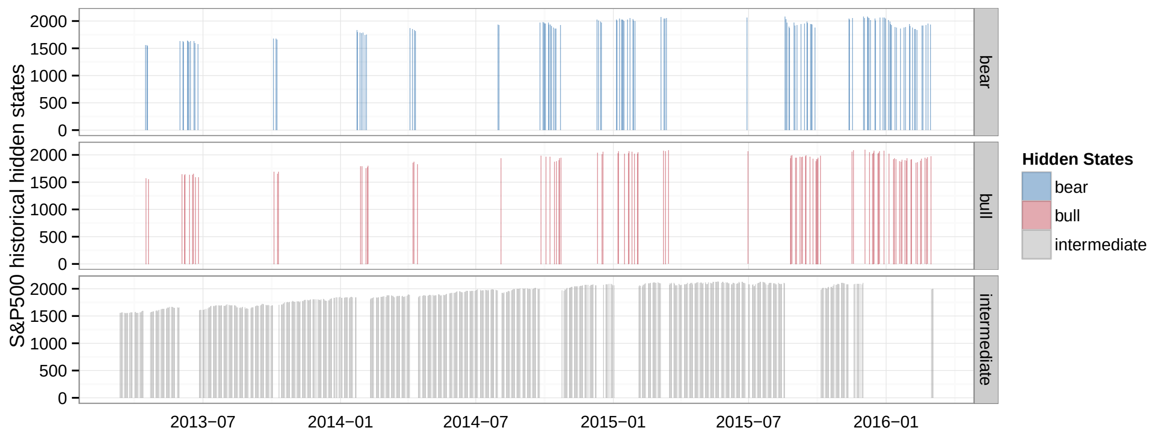


ILLUSTRATION 5–9 S&P 500 historical hidden states

The upper two parts of Illus. 5–9 are almost the same except for color labels, which proves the closeness of bears and bulls.

Actually the global decoding result also supports, from a visual perspective, our findings mentioned in Sec. 5.2.2, that extreme states tend to transit into each other instead of the intermediate.

Illus. 5–10 shows the states of in-sample returns implied by HMM and K-Means. As we mentioned before, K-Means distinguish the states based merely on quantitative properties of the data, not taking into account their correlations and transitions. Hence, most of the differences lie in the bull zone in K-Means, which are identified as intermediate in HMM. Statistically speaking, these returns are relatively higher, but they are more likely to appear during an intermediate state according to HMM. It is similar for those which are labeled as bear in K-Meas but intermediate in HMM. There also exist some points classified as intermediate in K-Means but considered to

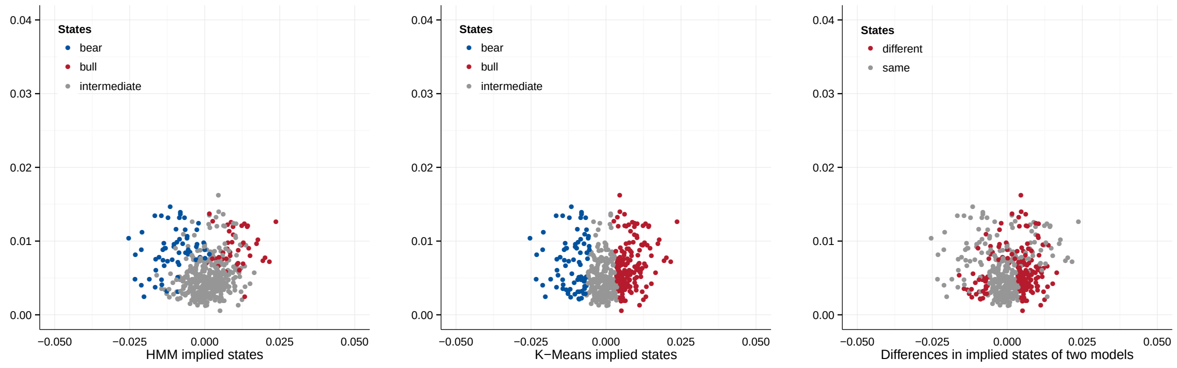


ILLUSTRATION 5–10 Comparison between K-Means states and HMM hidden states

be in one of the extreme states in HMM. These points may be viewed as small adjustments (or small rebounces) under a big trend, even though they are close to zero.

5.2.5 Simulated prediction results

Simulated prediction is carried out on out-of-sample data and the model is updated on a daily basis. Information of the real historical daily return is incorporated into the current model in order to predict the return for the next day. Illus. 5–11 presents the result of the prediction over the entire observation period (in-sample and out-of-sample). Predicted returns are cumulatively summed and taken exponential, multiplied by the first closing price of the out-of-sample data.

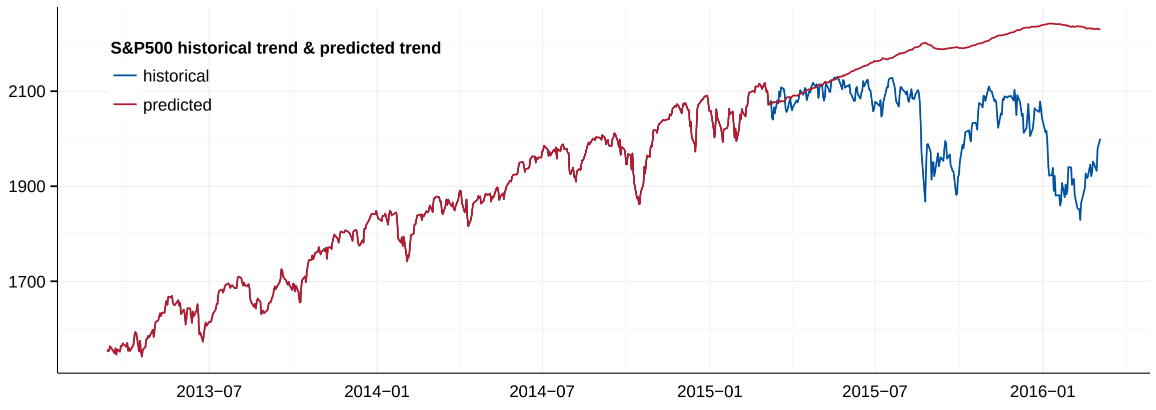


ILLUSTRATION 5–11 S&P 500 simulated prediction result

Visually speaking the prediction result is extremely poor. The predicted result preserves the trend of the in-sample data and has started to deviate from the real history since June 2015.

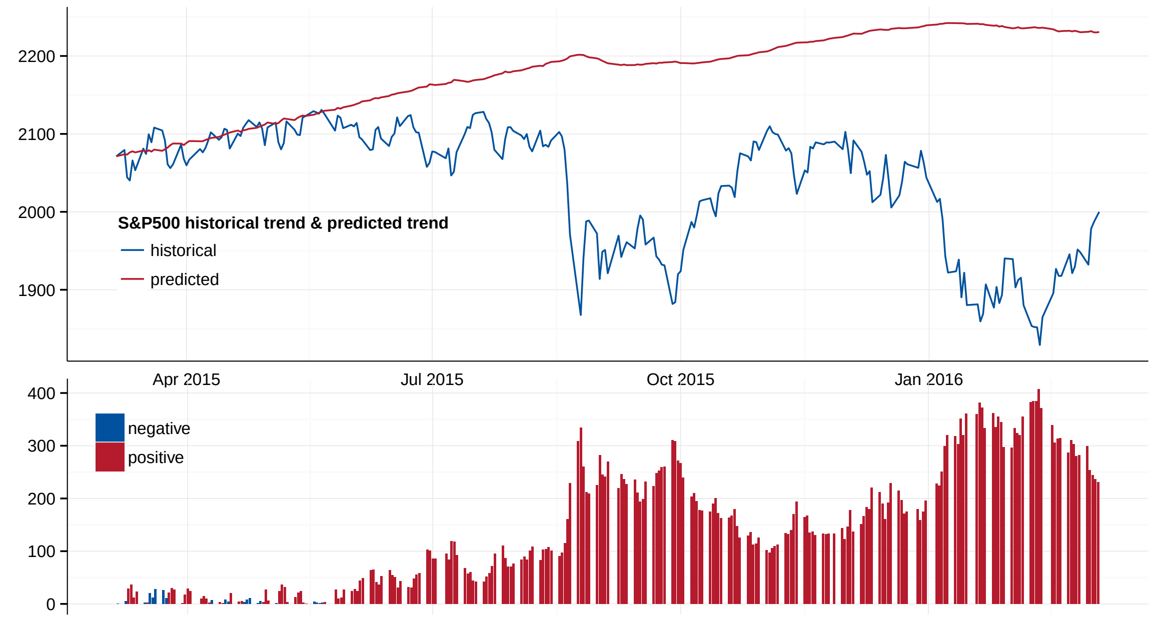


ILLUSTRATION 5–12 S&P 500 simulated prediction result out-of-sample part

Illus. 5–12 shows both the trend and differences between the historical value and the prediction result. Difference shot up when the index changed from a up-going trend to a downward shape. Illus. 5–13 indicates that even considering the standard deviations cannot help with the prediction.

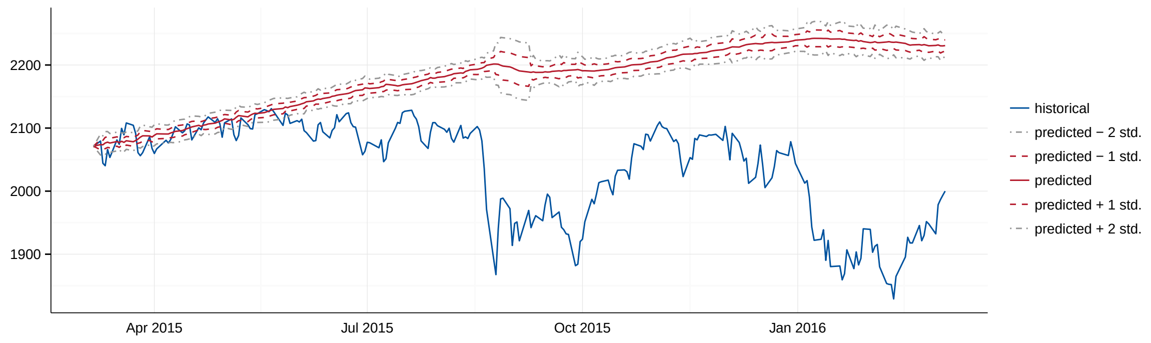


ILLUSTRATION 5–13 S&P 500 simulated prediction result with confidence band

However, the simulated prediction result has a win ratio of 50.4%, which means the model has predicted the right direction of the index with probability slightly over a half.

Besides, as we mentioned in the very beginning of Sec. 5.2.5, the predicted curve is based on the first closing price of the out-of-sample data but does not consider the current price of the index. The most direct and obvious result of such a method is that prediction errors accumulate, leading to larger and larger deviation from the real history.

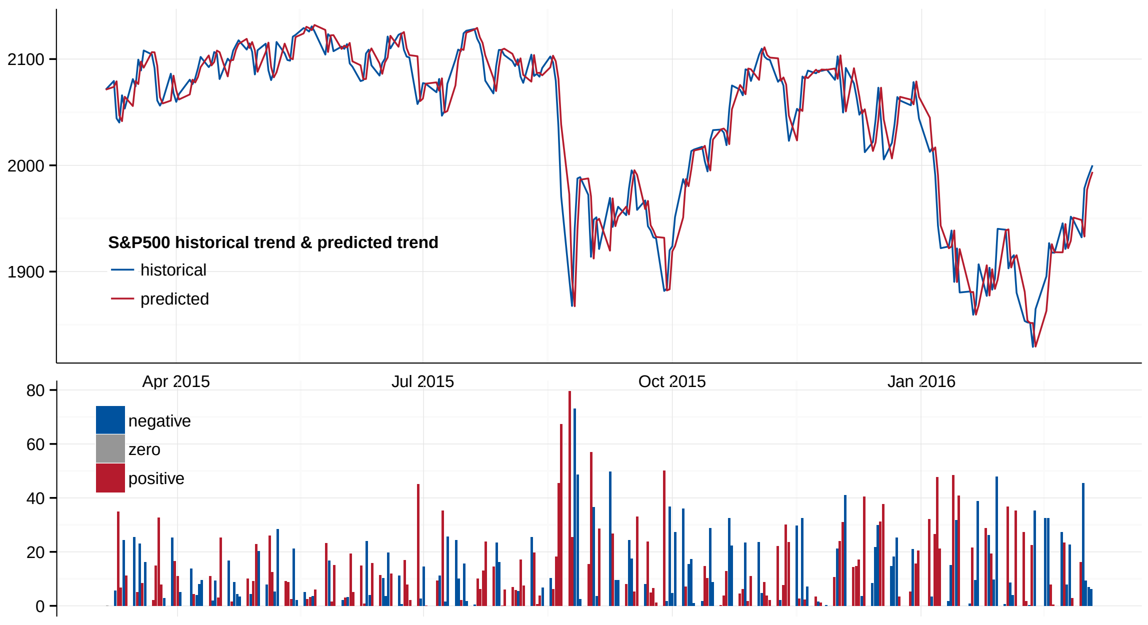


ILLUSTRATION 5–14 S&P 500 adaptive prediction

Therefore, it is reasonable to fully use the information provided by the data, including both real returns and real closing prices. We base prediction for the next day on both return and closing price at the standing point, and the result seems much better, as shown in Illus. 5–14. Illus. 5–15 plots the result and index along with $\pm\sigma$, $\pm 2\sigma$ bands. The real index is almost entirely covered by the $\pm 2\sigma$ band, which stands for a 95.45% confidence interval of prediction.

It is worth noticing that there are clearly lags in the prediction results compared to the real index. The extension of previous trend in Illus. 5–11 can be viewed as a stronger version of such lag. Furthermore, the lagging effect is not relived as time goes by, but, on the opposite, exacerbated.

Thus, Conjecture 5.1 seems reasonable to explain the phenomenon, that too much outdated information is included and causes the lag and error. Previous trends extend and too few weights have been allocated for the status quo. More detailed analysis and explanations will be discussed in both Sec. 5.6.1 and Sec. 7.1.

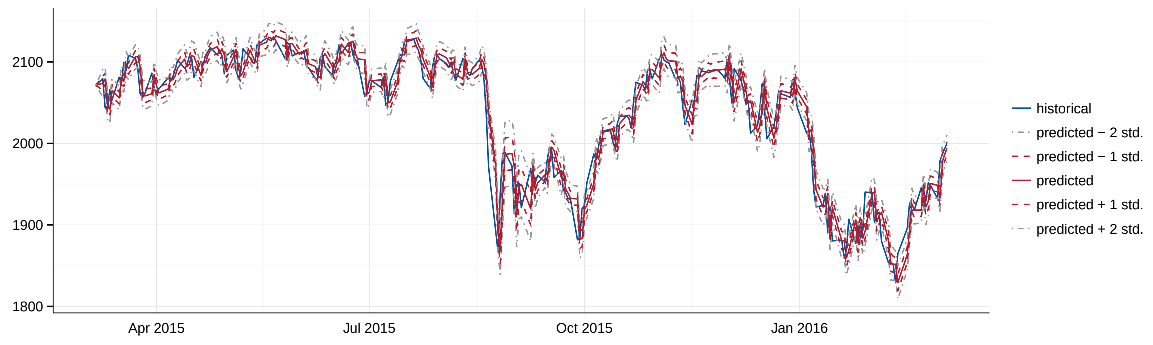


ILLUSTRATION 5–15 S&P 500 adaptive prediction with confidence band

5.3 Chinese CSI 300 Index daily return series

Of all stock market indices like Shanghai Composite Index, CSI 500 and CSI 800, etc., CSI 300 is the most typical and used for analysis.

5.3.1 Data preprocessing and description

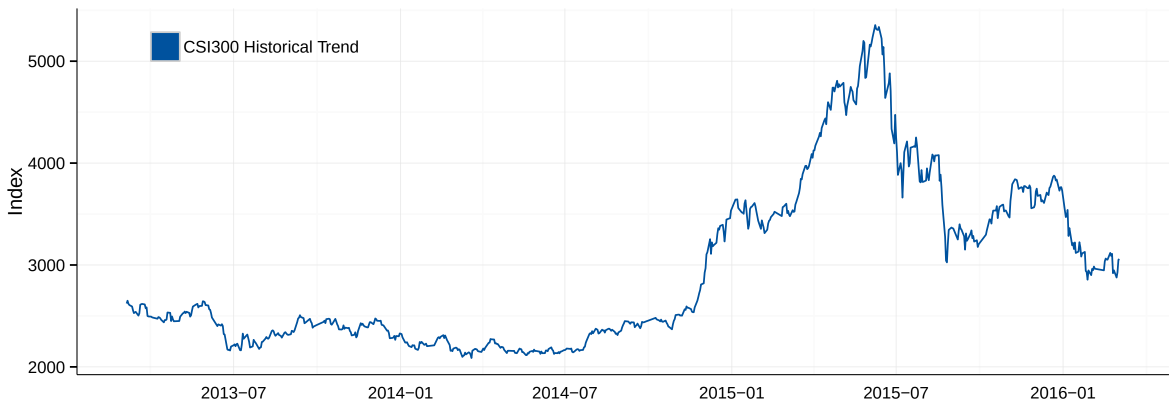


ILLUSTRATION 5–16 CSI 300 historical prices

Sample data of CSI 300 is selected between Mar. 5^{th}, 2013 and Mar. 3^{rd}, 2016 [3], totally 729 prices within three years, of which two thirds are used as in-sample, similar to the scheme in Sec. 5.2. The time period is chosen as the same to the S&P 500 case as well so that the results are comparable.

[3]Data source: Wind. All data of CSI 300, with different frequencies, are from the same data source

As is shown in Illus. 5–16, the index was almost flat during 2013 and 2014 and met its big upturn at the end of 2014. The big rise came in two parts and reached peak in May 2015, followed by a rapid and large fall. Rebounce occurred around September 2015 and then came another downturn in December.

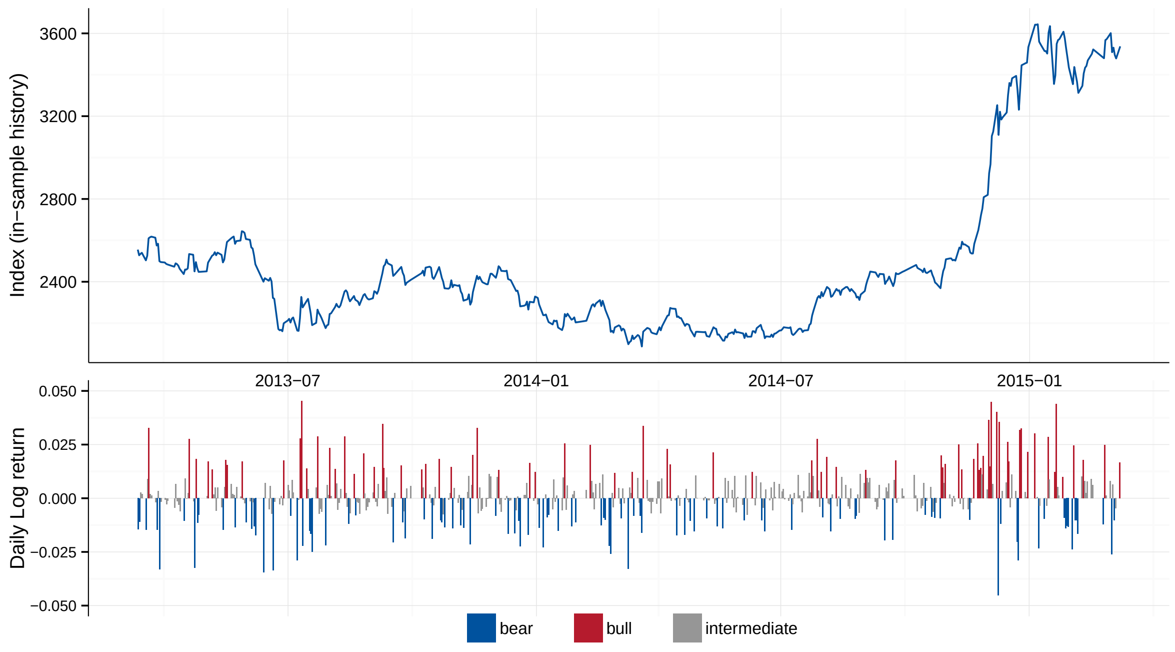


ILLUSTRATION 5–17 CSI 300 in-sample trend with daily return bars

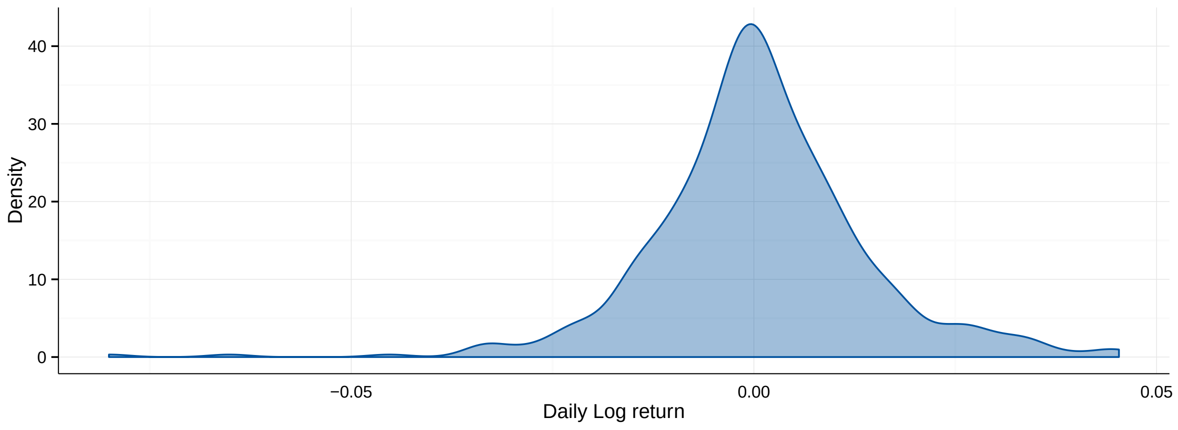


ILLUSTRATION 5–18 Density distribution of CSI 300 in-sample daily returns

The long period of fluctuation and almost half of the upgoing period makes up the in-sample period. Density distribution of daily returns has a lower skewness than S&P 500 in-sample data, but a longer tail on the left.

5.3.2 State transition parameters

State transition matrix of CSI 300 data is quite similar to the one in Sec. 5.2.2 but different in exact numbers.

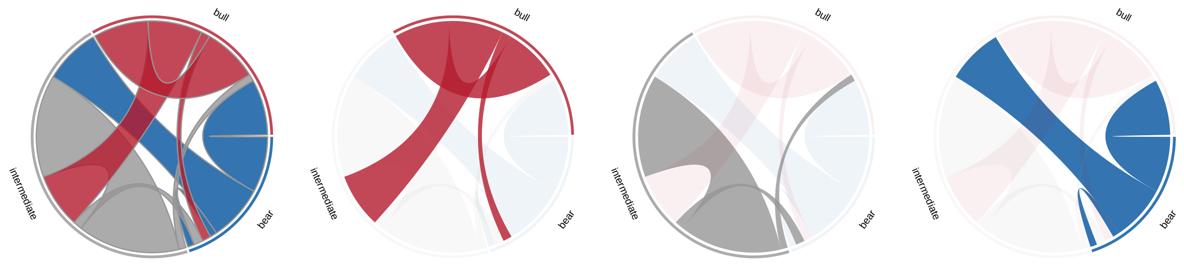


ILLUSTRATION 5–19 State transition for CSI 300 in-sample data

Illus. 5–19 implements a similar chord diagram to visually present the transition matrix. A bull market tends to stay at bull or transit into an indermediate with the same probability (46%) and is relatively unlikely to go into a bear directly. An intermediate market has the chance of 86% to stay put and a slightly higher probability to go into bear (7.7%). However, a bear market is very likely to get off the status quo, with a probability of 47% to turn into an intermediate and 46 to even go into a bull directly. This pattern is quite different from one in the U.S. market.

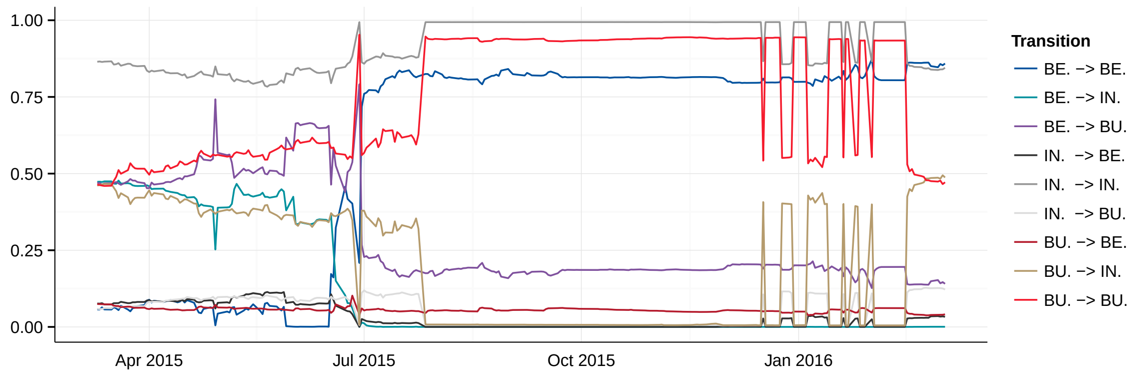


ILLUSTRATION 5–20 Evolution of state transition matrix entries

To be more thorough, we also examine evolution of the state transition matrix as the prediction processes, which leads to increases in data used to estimate parameters. Illus. 5–20 plots the

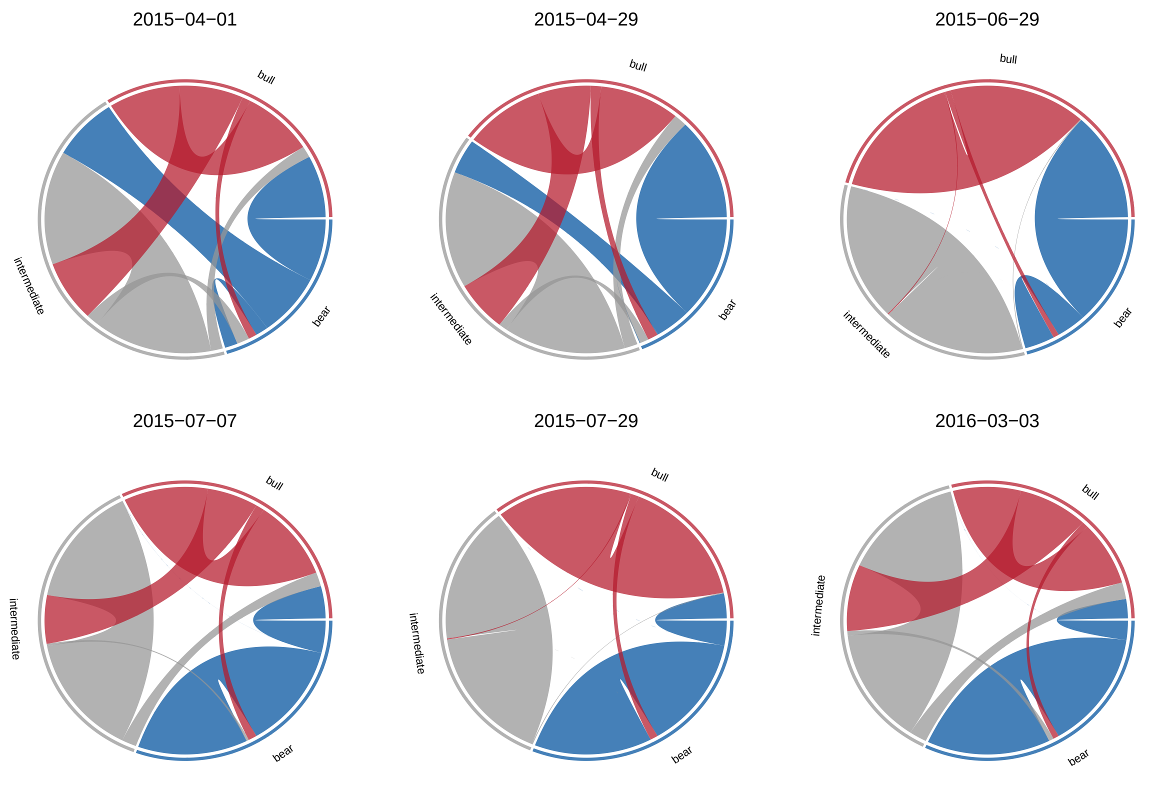


ILLUSTRATION 5–21 Typical state transition matrices during the entire period

dynamic evolution of the matrix as size of in-sample data grows. Dense fluctuations appear in June to July 2015, when the market has just passed its peak, and in early this year, when another downturn took place. It seems that falls of the market have greater impacts on the parameters than rises, and that may be related to the long tail we mentioned in Sec. 5.3.1.

Illus. 5–21 selects six chord diagrams within the entire observation period. Market states tend to remain the same with more chance after the big fall and hence the probability to transit into other states goes down, that is, the market has a bigger tendency to behave like previous trading days.

5.3.3 Conditional distribution parameters

The estimation result of conditional distribution parameters for CSI 300 in-sample is given as follow:

$$\begin{pmatrix} -1.66\% & 0.13\% & 2.13\% \\ 1.02\% & 0.99\% & 1.23\% \end{pmatrix}. \tag{5–2}$$

Columns stand for states, the first row for expectation and the second for standard deviation. The pattern resembles the one in 5.2.3 and clearly makes sense.

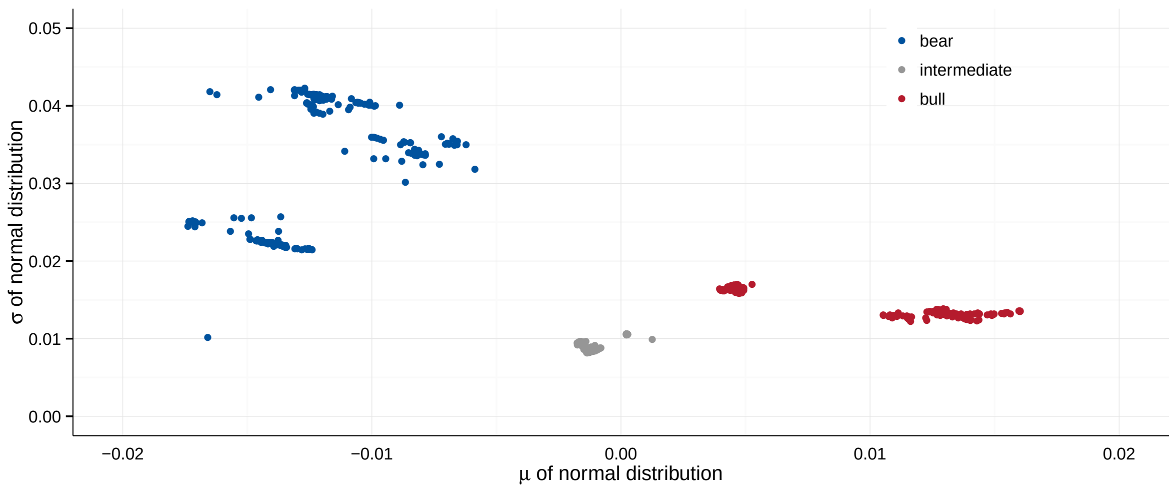


ILLUSTRATION 5–22 Conditional distribution parameters across time

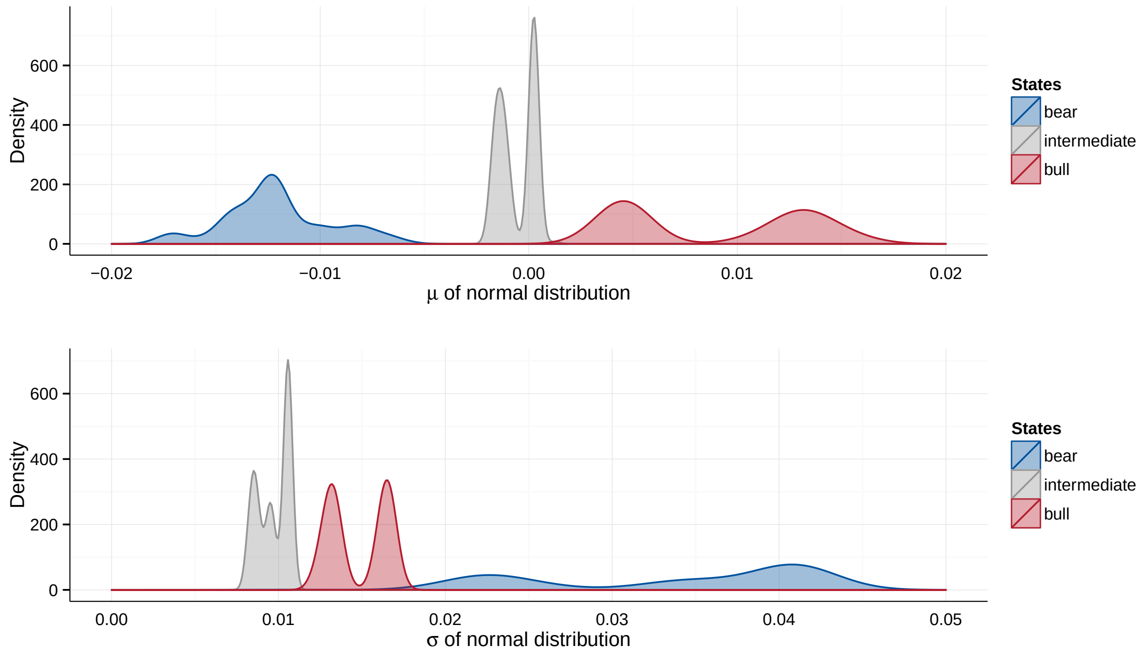


ILLUSTRATION 5–23 Density distribution of conditional distribution parameters

Illus. 5–22 and 5–23 give the time-variation of distribution parameters. High volatility of bears seems more significant than S&P 500 case, but standard deviations are dispersed, which is reflected as flatness of the blue line in the downer part of Illus. 5–23. Multi-modality exists, indicating changes of the states and a potential need to increase the number of hidden states in the model.

5.3.4 Global decoding and hidden states

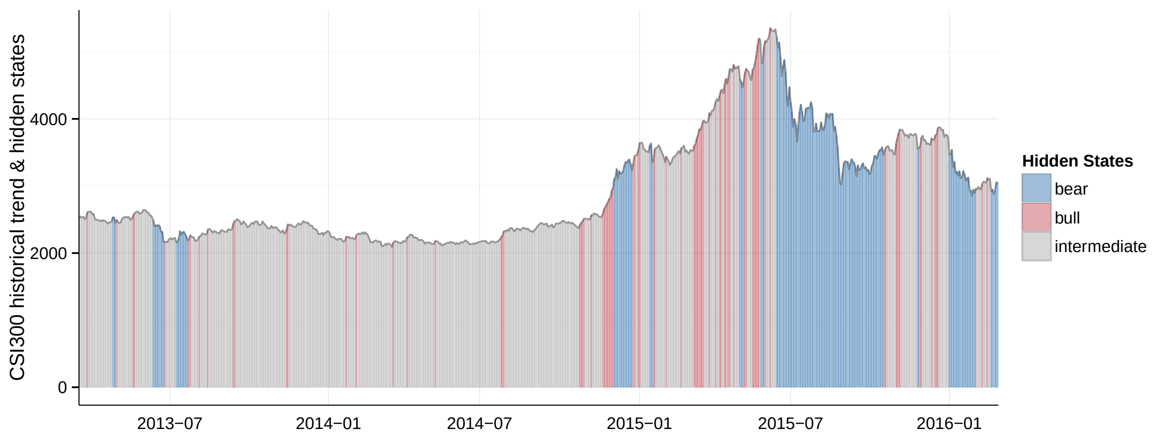


ILLUSTRATION 5–24 CSI 300 in-sample trend and states sequence

Global decoding result meets our expectation well. The intermediate state lasts for almost two years, followed by frequent occurences of bulls. Then there came the large bear until the rebounce, and then the bear hit the market again.

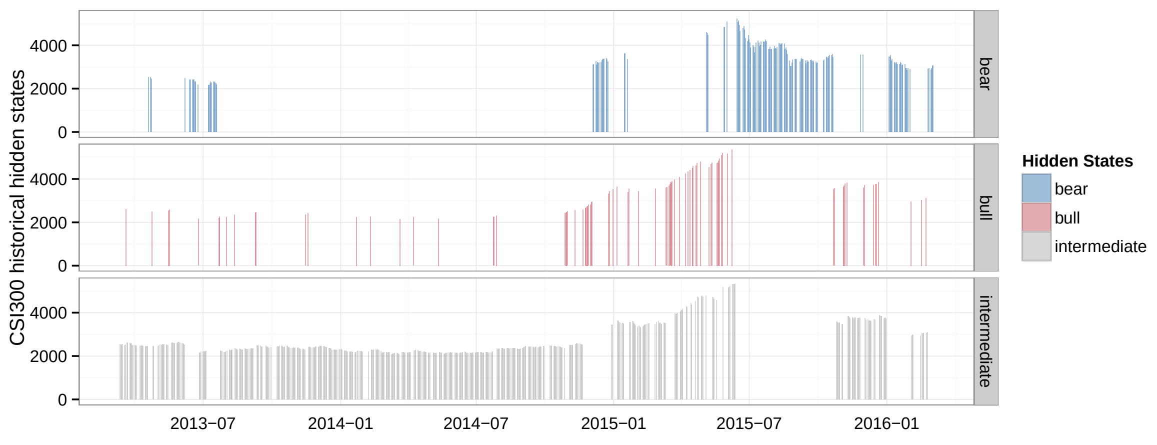


ILLUSTRATION 5–25 CSI 300 historical hidden states

The bull states have not appeared consecutively like the bear. We suppose that is because when the market went up, the returns were relatively small, and small positive returns accumulated to form a long-term bull with help of time, while the bear came with large negative returns and conquered the market rapidly. Results visualizations are presented in Illus. 5–24 and 5–25.

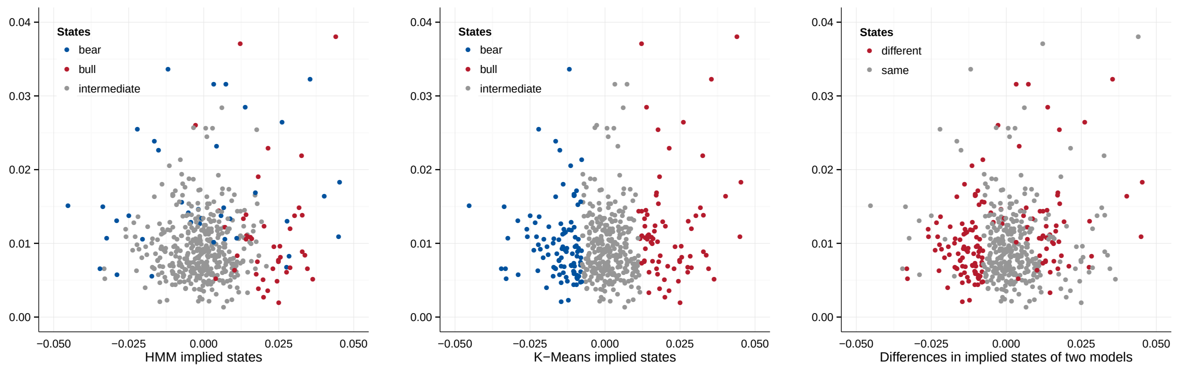


ILLUSTRATION 5–26 Comparison between K-Means states and HMM hidden states

Similarly, differences between HMM states and K-Means states are examined (see Illus. 5–26). Differences occur mainly in the bear zone, some in the bull, and few in the intermediate. The pattern resembles one in Sec. 5.2.4 and we no longer repeat it here.

5.3.5 Simulated prediction results

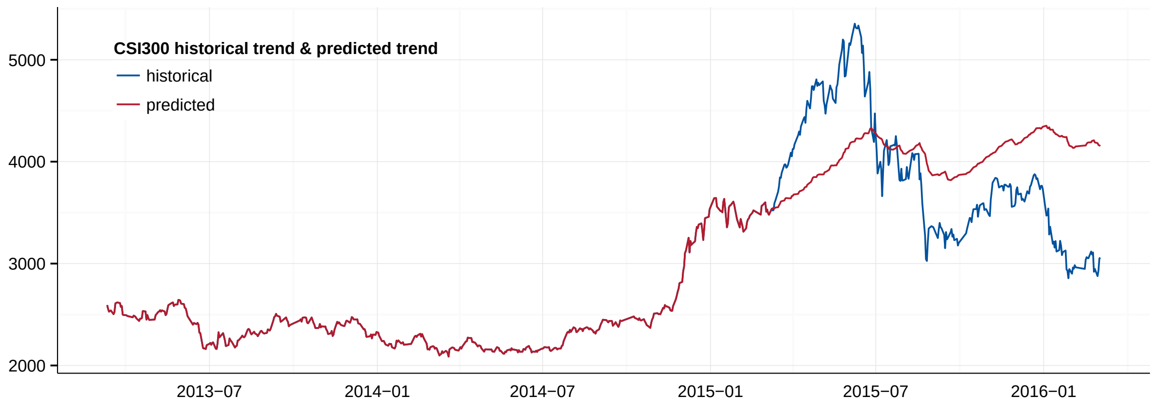


ILLUSTRATION 5–27 CSI 300 simulated prediction result

Simulated prediction result along with CSI 300 in-sample data is plotted in Illus. 5–27. The out-of-sample part is separatedly presented with daily return bars in Illus. 5–28

The win ratio of the prediction is 50.2%, almost the same as in S&P case, and the extension and lagging pattern is more clearly conveyed. At the very first of the out-of-sample period, the

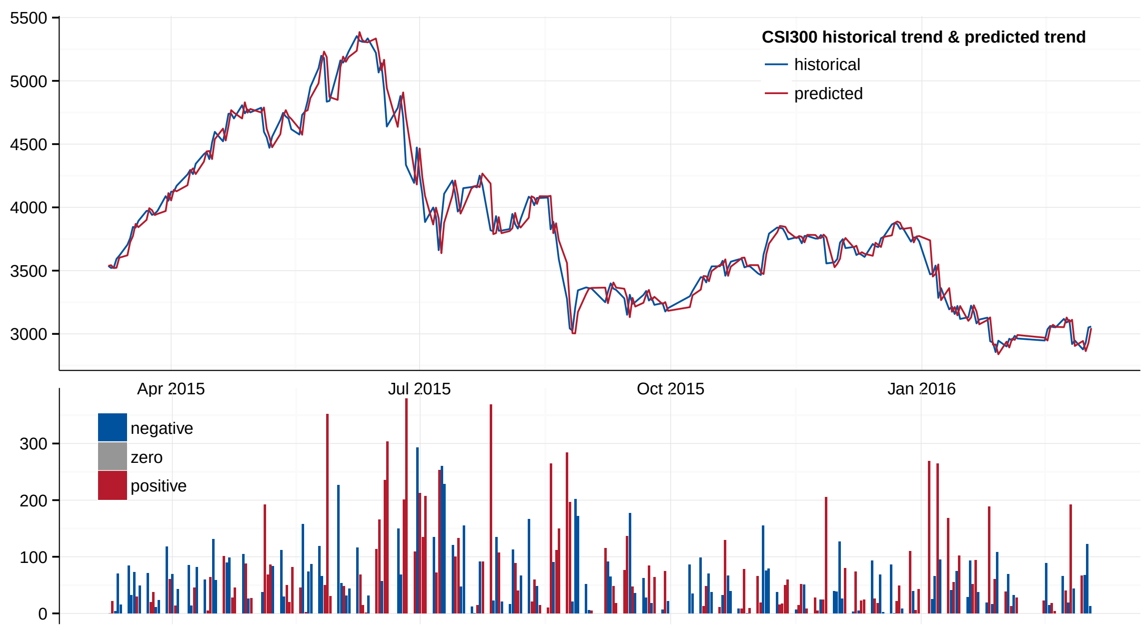


ILLUSTRATION 5–28 CSI 300 simulated prediction result out-of-sample part

predicted index remains going up straight but with a lower slope, which extends the trend before and also influenced by recent adjustments. Downturn appears after the real fall takes place and the lag is quite obvious. Prediction behaviors are similar in the rest part. Confidence bands and adaptive prediction results are also visualized and the formats are similar as in S&P case. In order to avoid unnecessary information, we do not present the illustrations here and they will be listed in Appendix A in case the reader is interested [4].

However, results are quite good for adaptive predictions. Though the time lag effect still exists, it eliminates the error accumulation on the absolute level of the index.

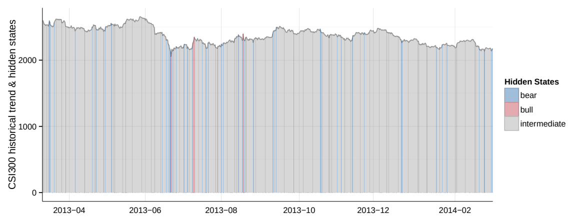
5.4 Chinese CSI 300 Index medium frequency (60min) return series

CSI 300 medium frequency (60min) return data are selected within the same observation period as before. The only difference is the observation frequency. Higher frequency indicates that the number of data increase during the same period, so we split the entire period into three parts,

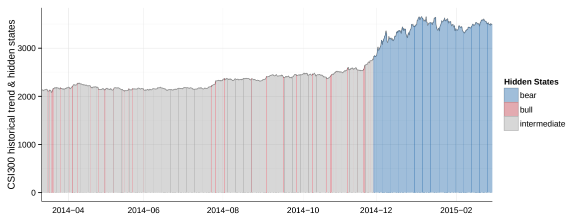
[4]Similar prediction and analysis are also carried out for CSI 300 data with medium frequencies (see Sec. 5.4 and Sec. 5.5, and the figures are also presented in Appendix A)

each has length of about one year (from March to March).

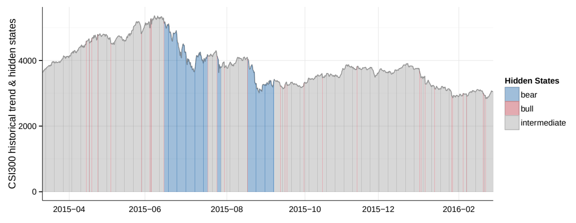
5.4.1 Global decoding and hidden states



(a) Mar. 2013 ∼ Mar. 2014



(b) Mar. 2014 ∼ Mar. 2015



(c) Mar. 2015 ∼ Mar. 2016

ILLUSTRATION 5–29 CSI 300 60min data global decoding results

Decoding results in year 2013 to 2014 matches our expectation, while the result in the latter two sub-periods is quite astonishing.

The last part in Part (b) of Illus. 5–29 presents the trend of a lasting rise but is identified as bear. The conditional distribution parameters (60min basis [5]) in the period are

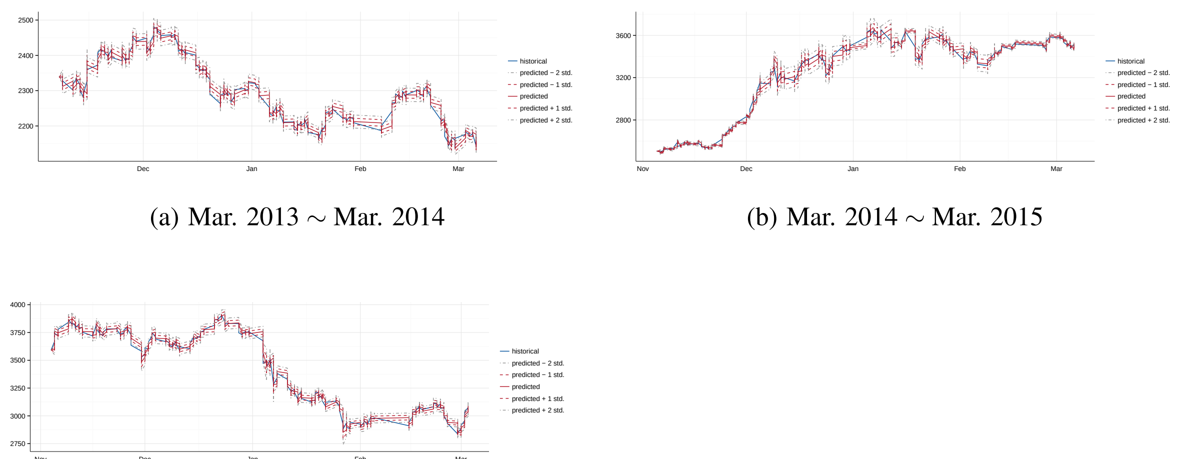
$$\begin{pmatrix} 0.108\% & 0.019\% & 0.115\% \\ 1.207\% & 0.324\% & 0.827\% \end{pmatrix}. \tag{5–3}$$

The parameter pair corresponding to bear states has a positive expectation and a relatively big standard deviation. Thus, the “bear” state here has actually a higher expected return than the intermediate state and is classified as bear because its volatility is higher and expected return is lower than the bull.

[5]The returns are calculated on a 60min basis and we do not annualize the data or change the scale to the same as former cases, since compared to the actual value of the parameters, we care more about their differences, orders and decoding and prediction results.

As we mentioned before, the name of states here are artificially given to the states based on customs. During Mar. 2014 to Mar. 2015, the market is at a (conventionally acknowledges) bull state almost all the time and does not actually have a bear state, if not consider the small rebounces. Therefore, although the labels do not meet our expectations, the actual distributions are not that unexpected.

5.4.2 Simulated prediction results



(a) Mar. 2013 ~ Mar. 2014

(b) Mar. 2014 ~ Mar. 2015

(c) Mar. 2015 ~ Mar. 2016

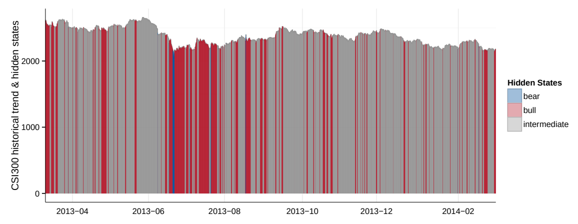
ILLUSTRATION 5–30 CSI 300 60min data simulated prediction results

Win ratios in the three year are 51.7%, 55.5% and 50.5% separately, which are all better than former cases. Time lag effects and error accumulations exist for long-term prediction. Adaptive prediction results are much better.

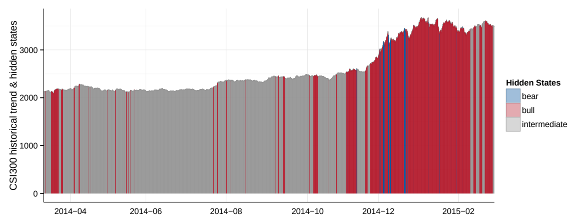
5.5 Chinese CSI 300 Index medium frequency (10min) return series

CSI 300 medium frequency (10min) return data have exactly the same structure as 60min data except higher observation frequency. This only difference enables us to examine the effect of observation frequency, or data density within a given time period, on the decoding results and prediction results.

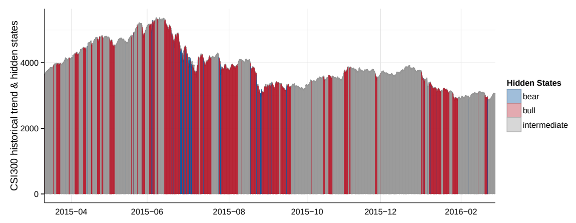
5.5.1 Global decoding and hidden states



(a) Mar. 2013 ∼ Mar. 2014



(b) Mar. 2014 ∼ Mar. 2015



(c) Mar. 2015 ∼ Mar. 2016

ILLUSTRATION 5–31 CSI 300 10min data global decoding results

Anomaly in 10min case occurs in year 2015 to 2016 (instead of 2014 to 2015 in 60min case). Conditional distribution parameters (10min basis) are

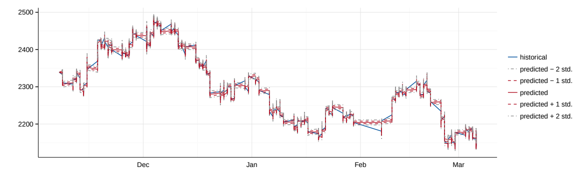
$$\begin{pmatrix} -0.117\% & 0.026\% & -0.031\% \\ 1.627\% & 0.269\% & 0.515\% \end{pmatrix}. \tag{5–4}$$

So it is actually very similar to the 60min case, just that the "bull" has a negative expected return and higher volatility here while the anomaly in 60min case is related to the "bear" then.

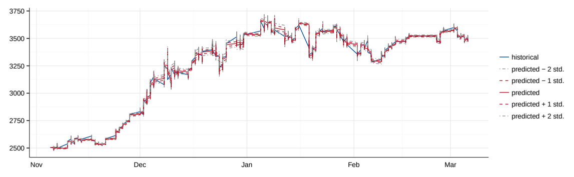
5.5.2 Simulated prediction results

Win ratios in the three year are 51.7%, 57.5% and 52.0% separately, better than ones in the 60min cases and of course the daily case.

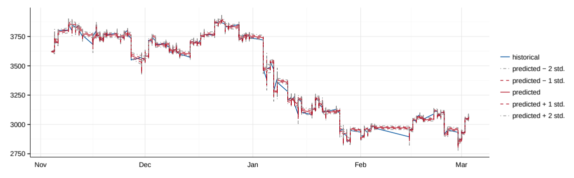
Patterns are similar to 60min cases. The slight increase in correctness is certainly not enough to cause revolutionary improvements of curve prediction and fitting, yet still the increase is worth noticing and will be discussed about in Sec. 5.6.4.



(a) Mar. 2013 ~ Mar. 2014



(b) Mar. 2014 ~ Mar. 2015



(c) Mar. 2015 ~ Mar. 2016

ILLUSTRATION 5–32 CSI 300 10min data simulated prediction results

5.6 Result analysis and comparisons

Analysis of the results and various comparisons are carried out in this section. Firstly we analyze all the prediction results and potential causes of errors. Then comparisons from three different perspectives (market, time period, data frequency) are conducted. At last we discuss about setting of the number of hidden states, as supplementary explanation to Assumption 5.1.

5.6.1 Prediction correctness and error analysis

So far we have performed eight complete stock return series predictions in total (one for S&P 500, seven for CSI 300 with different observation frequencies). All results have a win ratio greater than 50% (at least 50.2%), so theoretically we can construct a trading strategy based on the model and to profit through very frequent (not necessarily high-frequency) trades. However, in practice, this ratio is quite low for real-life trading strategies since all kinds of transaction costs occur along with each trade, whatever win or lose. Thus the prediction results heretofore has only theoretical values but not enough to apply to realistic problems.

As we found in previous sections, The biggest problem of the predictions is the time lag, which proves Conjecture 5.1 reasonable and very likely to be right. Influences of outdated information is the main cause of prediction errors, while this kind of errors can be reduced with sample reweighting or rolling window methods. Potential methods to deal with time lags are presented in Sec. 7.1 and no further discussions are given here.

Besides, the prediction method, given the current distribution of states, state transition matrix and corresponding conditional distribution parameters, lower the correctness since it takes expectation (see Sec. 4.2.2). But it is statistically logical and meaningful, and reduces prediction variance for the next node. We provide this point of view only to be thorough about potential causes of the errors.

5.6.2 Comparisons between U.S. market and Chinese market

Global decoding results and prediction correctness for both markets (daily data case) are similar. There are not obvious contradictions to our expectation, nor obvious differences in prediction effectiveness. Thus currently it is safe to say that the performance of our model does not depend on the market.

However we should notice that the data we select have a very long length and they cover all (conventionally acknowledged) states that can be easily identified with our eyes. One of the advantages of such data is that parameters distinguish clearly from one another. Therefore, a more rigorous statement here is that, when sample data are long enough to cover all traditional states (big rises, big falls, and long periods of fluctuations), the model has no differences in performance for different markets.

5.6.3 Comparisons among different time periods

According to results in Sec. 5.4 and 5.5, performances differ in terms of observation period. Prediction correctness tends to increase during time when the trends are consistent, e.g. bull all the time. Actually this phenomenon seems overlap with Conjecture 5.1 again, since inconsistency in trends actually means the ineffectiveness of old data.

5.6.4 Comparisons among data of different frequencies

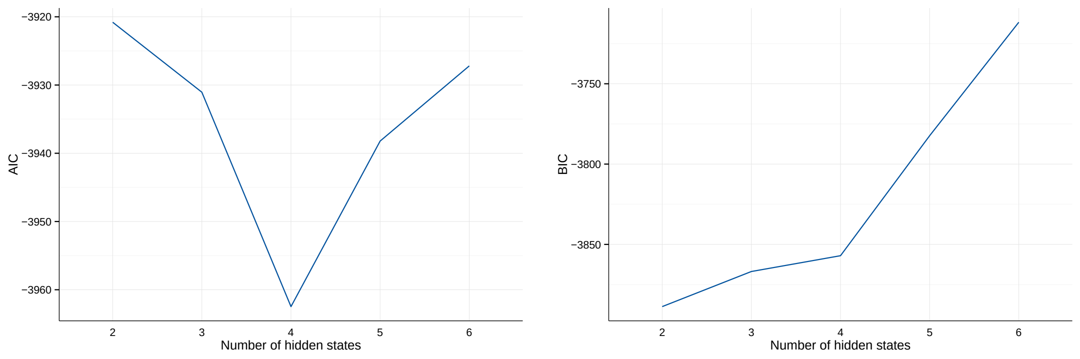
Differences in performances with respect to data frequency are analyzed also based on Sec. 5.4 and 5.5. Prediction correctness in 10min cases are no smaller than in 60min cases (51.7%, 57.5% and 52.0% against 51.7%, 55.5% and 50.5%). Hence, apparently, higher observation frequency within the same period leads to higher prediction accuracy, since it contains more information with other conditions hold still.

Table 5–1 Brief description of all prediction results

Target Index	Data Frequency	Time Period	Data Length	Win Ratio
S&P 500	daily	2013∼2016	757	50.4%
CSI 300	daily	2013∼2016	729	50.2%
CSI 300	60min	2013∼2014	968	51.7%
CSI 300	60min	2014∼2015	976	55.5%
CSI 300	60min	2015∼2016	972	50.5%
CSI 300	10min	2013∼2014	5808	51.7%
CSI 300	10min	2014∼2015	5856	57.5%
CSI 300	10min	2015∼2016	5832	52.0%

Table 5–1 present a brief description of the eight groups on data frequency, time periods, data length (the number of records) and win ratios.

5.6.5 Comparisons among different number of hidden states



(a) AIC vs. N

(b) BIC vs. N

ILLUSTRATION 5–33 Goodness of fit for different number of hidden states

We state in Assumption 5.1 that the markets have only three hidden states and we name them as bear, intermediate and bull. As is conventionally acknowledged, there are bull markets and bear markets. However there are also intermediates that do not have obvious trends and fluctuations dominate during the period, where only two states are not enough to describe the markets. Certainly more market states could be introduced to the model (e.g. four to represent big rises, small rises, small falls and big falls), but more states also bring higher risk of over-fitting and higher possibility to lack in realistic economic meanings.

Illus. 5–33 and Table 5–2 indicate that two is the optimal parameter of the number of states based on BIC and four optimal according to AIC. Thus we choose three as the parameter, which is the most widely used (see [5, 39, 42]), statistically (nearly) optimal and also economically meaningful. Yet still the reader should notice that it is possible there are more optimal parameter selections.

Table 5–2 Numerical results of goodness of fit for different number of states

Number of States	AIC	BIC
2	-3920.80	-3888.71
3	-3931.05	-3866.86
4	-3962.47	-3857.02
5	-3938.21	-3782.33
6	-3927.20	-3711.71

Chapter 6 Conslusion

In this chapter we summarize all contents above of this thesis and draw some conclusions on the stock return series prediction work we have done.

Up to now, we have formally introduced the hidden Markov model (HMM) in Ch. 3, including the model formulation and important statistics, model estimation methods, and ways to perform forecasts and decoding. As a simple dynamic Bayesian network, HMM is easy to be realized, and it excavates the information not only about the stock returns themselves but also the classifications of them. which enables us to analyze the data from the angles of points and also intervals. The number of model parameters in under control of the number of hidden states, at the level of $\mathcal{O}(N^2)$ (N represents the number of states), which is usually small in empirical analysis. From these perspectives, HMM is quite suitable to apply for stock return series analysis and prediction, due to its easy implementation, appropriate complexity and small number of parameters.

In Ch. 4 we have constructed the stock return series prediction system, which combines model initialization with K-Means, model estimation, historical analysis (with HMM global decoding) and visualization. The system is adaptive to different data populations (w.r.t. target index, time horizon, observation frequency, etc.) and exogenous parameters (e.g. the number of hidden states). It is also complete, encapsulated and user-friendly (see Appendix B for the user manuscript and source codes).

In Ch. 5 we have carried out empirical analysis on both U.S. (the S&P 500 Index) and Chinese (the CSI300 Index) stock markets. With some modest assumptions, the system has been well functioning for all the analysis. From the perspective of prediction correctness, the system is effective and always has a win ratio greater than 50%, and sometimes approaches 60% under certain circumstances. We can also draw the conclusion that data populations with higher observation frequencies tend to outperform those with lower frequencies in this system due to the difference in the amount of information contained in the data. From the view of global decoding, we come up to the conclusion that data populations with longer time horizon (i.e. longer observation period) tend to have (better) global decoding results that more fit our usual acknowledgements. Potential for improvement of the HMM-based system lies in the incorporation of resampling or sample reweighting, which shall largely ameliorate the time lag effect. This part will be briefly introduced in Ch. 7.

To sum up, we have successfully constructed the HMM-based stock return series prediction system and have accomplished all the aims of this thesis. We want to reinstate that this thesis in aimed to fulfill a complete implementation of stock return prediction with HMM, while we have no intention to realize the improvements of model-level techniques. Finally note that the model is imperfect and there exist many unaddressed issues. We cover some of them in the last chapter and hope to incorporate such improvements in future works.

Chapter 7 Future work

In the final chapter, we briefly talk about some remaining issues mentioned in the previous chapters and potential directions for future research. In Sec. 7.1 we discuss about the time lag effect that has been mentioned multiple times in Ch. 5, and we propose some (possibly) feasible solutions to the problem. In Sec. 7.2 we introduce particle filters, which is currently a preferred method for HMM estimations [43]. We address these issues here merely for the reader's information, and we hope to deal with them in future works.

7.1 Dealing with time lags

As we stated in Conjecture 5.1, incorporation of too much out-dated information leads to large prediction errors, and the results worsen as time goes by, i.e. with the iterative (loops of out-of-sample data) predictions process.

The essential issue embedded in the problem is that all data sample take up equal weight when they are used to fit the model. The latest data point has the same significance as the one from a year ago, which is intuitively problematic. Therefore, reweighting of the sample observations are necessary to deal with the time lag effect.

We introduce two different ways in this section. The first one described in Sec. 7.1.1 implements a rolling window, which makes cut-off of the data that are too old. This method is mostly used in industries due to its simplicity and flexibility. The second one presented in Sec. 7.1.2, more popular in academia, is the exponentially weighted EM algorithm. The algorithm remains all sample data from the population, and the only difference with the traditional EM is that the sample observations are reweighted.

7.1.1 Prediction with rolling windows

The rolling window method, as is indicated from the name itself, refers to the idea that the data sample (set) is updated with arrival of new observations while the size of the sample remain the same. Eq. 7–1 shows the dynamic process of data set changing.

$$\dots, \underbrace{x_{t-l+1}, \rlap{$\overbrace{\phantom{x_{t-l+2}, \dots, x_{t-1}, x_t, x_{t+1}}}^{\text{sample}_{t+1}}$} x_{t-l+2}, \dots, x_{t-1}, x_t}_{\text{sample}_t}, x_{t+1}, \dots \tag{7–1}$$

Every time when the system processes to the next loop, the first (in time) sample point is discarded and the newly come observation is included, and the length of observation series used to fit model remains l. Simply put, we cut the tail of the data sample whenever a new observation is made. Essentially the rolling window is a reweighting method, only the weight for all out-dated observations are arbitrarily zero and ones remained in the sample are equally weighted.

Due to its easiness to implement, the rolling window method is commonly used in real-world financial analysis. The only changeable parameter in the method is the size of the sample. Usually the parameter is chosen to have some realistic meanings, e.g. for analysis on daily return data, l is usually chosen as $5, 10, 21, 63$, which separately stands for a week, half-month, a month and three months, counting in trading days. Optimization within the small pool helps us to find a (relatively) good choice of the parameter, meanwhile reduces the risk of over-fitting.

7.1.2 Exponentially weighted EM algorithm

Considering the change of importance over the time, a few weighting methods are created based on the moving average (MA) concept. The methods are then adopted beyond MA and used along with many other techniques, e.g. the EM algorithm.

The exponentially weighted expectation maximization (EWEM) algorithm, inspired by the idea of exponentially weighted moving average (EWMA), performs reweighting of the sample observations.

Recall Eq. 3–19 and we rewrite the iteratively maximization problem in form of function of the parameter $\boldsymbol{\theta}$:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i-1)}) = E\left[\log p(\mathbf{x}, \mathbf{s} \mid \boldsymbol{\theta}) \mid \mathbf{x}, \boldsymbol{\theta}^{(i-1)}\right], \tag{7–2}$$

and we shall introduce a time-dependent weight η into Eq. 7–2:

$$\begin{aligned} \hat{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i-1)}) &= E\left[\log \eta p(\mathbf{x}, \mathbf{s} \mid \boldsymbol{\theta}) \mid \mathbf{x}, \boldsymbol{\theta}^{(i-1)}\right] \\ &= E\left[\eta \log p(\mathbf{x}, \mathbf{s} \mid \boldsymbol{\theta}) \mid \mathbf{x}, \boldsymbol{\theta}^{(i-1)}\right] \\ &= \sum_{s \in \mathbb{S}} \eta \log p(\mathbf{x}, s \mid \boldsymbol{\theta}) p(s \mid \mathbf{x}, \boldsymbol{\theta}^{(i-1)}) \end{aligned} \tag{7–3}$$

The full description and analysis of the method (including definition, implementation, convergence result, etc.) is explained in details in [44], and we do not provide further steps here. The two equations above are enough for the introduction to the general idea.

The EWEM algorithm is more complex than the rolling window method and much more difficult to add to the traditional EM algorithm, thus it is mostly discussed about in academia while seldom implemented for industrial research. Yet the idea is very enlightening and we hope to study further on the sample reweighting of HMM in the future.

7.2 Particle filters

Particle filters (PF) are also known as sequential Monte Carlo (SMC) methods, which is a set of genetic-type particle Monte Carlo methodologies to solve the filtering problem [45], and is firstly proposed in [46, 47].

Firstly we re-formulate the HMM problem in a uniform way:

$$\mathbf{x}_n = m_n(\mathbf{s}_n, \epsilon_n), \tag{7–4a}$$

$$\mathbf{s}_n = h_n(\mathbf{s}_{n-1}, \eta_n), \tag{7–4b}$$

where $\mathbf{x}$ is the observed variable (vector) and $\mathbf{s}$ the state variable (vector). In previous chapters, we redeem that both $\mathbf{s}$ and $\mathbf{x}$ are discrete and $\mathbf{s}$ is even categorical, while they may be continuous, discrete or even combined of the two [40]. The functions m_n and h_n are very likely to be nonlinear and with unknown forms.

Similar to before, we write some important statistics as follows:

- joint smoothing distribution

$$p(\mathbf{s}_{0:n} \mid \mathbf{x}_{1:n}; \theta) = \frac{p(\mathbf{s}_{0:n}, \mathbf{x}_{1:n}; \theta)}{p(\mathbf{x}_{1:n}; \theta)}; \tag{7–5}$$

- transition density

$$p(\mathbf{s}_n \mid \mathbf{x}_{1:n-1}; \theta) = \int p(\mathbf{s}_n \mid \mathbf{s}_{n-1}; \theta) p(\mathbf{s}_{n-1} \mid \mathbf{x}_{1:n-1}; \theta)\, d\mathbf{s}_{n-1}; \tag{7–6}$$

- filtering distribution

$$\begin{aligned} p(\mathbf{s}_n \mid \mathbf{x}_{1:n}; \theta) &= \frac{p(\mathbf{x}_n, \mathbf{s}_n \mid \mathbf{x}_{1:n-1}; \theta)}{p(\mathbf{x}_n \mid \mathbf{x}_{1:n-1}; \theta)} \\ &= \frac{p(\mathbf{x}_n \mid \mathbf{s}_n; \theta) p(\mathbf{s}_n \mid \mathbf{x}_{1:n-1}; \theta)}{\int p(\mathbf{x}_n \mid \mathbf{s}_n; \theta) p(\mathbf{s}_n \mid \mathbf{x}_{1:n-1}; \theta)\, d\mathbf{x}_n}; \end{aligned} \tag{7–7}$$

- forecast distribution

$$p(\mathbf{x}_n \mid \mathbf{x}_{1:n-1}; \theta) = \int p(\mathbf{x}_n \mid \mathbf{s}_n; \theta) p(\mathbf{s}_n \mid \mathbf{x}_{1:n-1}; \theta)\, d\mathbf{x}_n; \tag{7–8}$$

where $theta$ is the set of model parameters. In order to estimate the probabilities and integrals, we can implement the Monte Carlo (MC) methodologies, which are simulation-based techniques to find estimates of them.

The reason we introduce the approximation method is that analytic solutions to model estimation only exist for specific models, such as discrete and categorical latent variable models (as in our case) and linear-Gaussian observed variable models like Kalman Filter.

Variance reduction techniques are necessary for MC in order to accelerate the convergence and relieve the computational burden. Importance sampling (IS) is one of the most commonly implemented method. The method is proposed in [48, 49]. It generates random particles from a importance (or proposal, or biased) distribution and reweighting the samples to have the unbiased estimates of integrals. Search for the importance distribution is, however, not easy. Several methods have been proposed, e.g. efficient importance sampling in [50] and cross-entropy (CE) method [51, 52], which minimizes the Kullback-Leibler divergence (KLD). The methods have not been applied to HMM problems yet and we consider them very potential.

Computational cost is high to adopt standard IS in HMM problems, thus sequential importance sampling (SIS) method has been proposed to draw random particles from a sequence of conditional distributions. Furthermore, resampling algorithms are also incorporated in SIS and the new sequential importance sampling with resampling (SISR) algorithm mitigates the degeneracy problem so that the method shall function much more efficient than the former one, see [53, 54]. With the methods above, PF is largely implemented to solve HMM problems and has become a standard tool for them.

As for our problem, it is possible to further modify our problem formulation and then solve the more complex problem with PF. For example, at present we assume the hidden states to be discrete, categorical and finite. We can view the conditional distribution parameters as our latent variable and solve directly for them under certain assumptions, such as certain function forms:

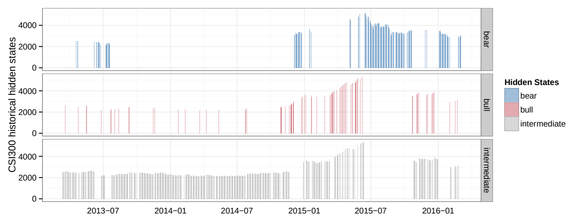
$$\mathbf{s}_t = (\mu_t, \sigma_t) = \begin{cases} \mu_t \sim \text{ARMA}(p, q), \\ \sigma_t \sim \text{GARCH}(m, n). \end{cases} \tag{7–9}$$

We propose the equation above only to offer some possible ideas. It remains to be examined whether Eq. 7–9 is feasible for the problem.

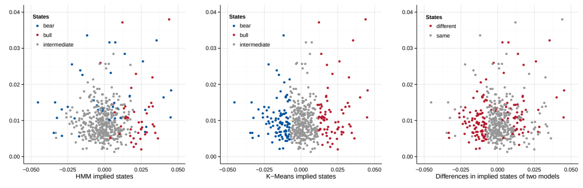
Besides PF, there are also some other more advanced methods to solve for HMM problems, e.g. approximate Bayesian computation (ABC) method [55, 56], an exact, online and plug and play method based on PF named SMC^2 [43]. The topic is quite beyond the scope of this thesis and we do not make further discussions here.

Appendix A Visualization of Simulated Prediction Results

A.1 CSI 300 daily return series

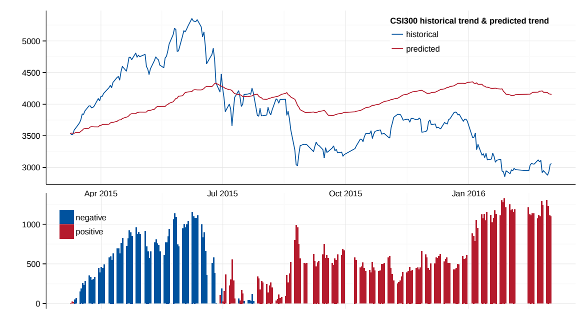


(a) CSI 300 historical hidden states

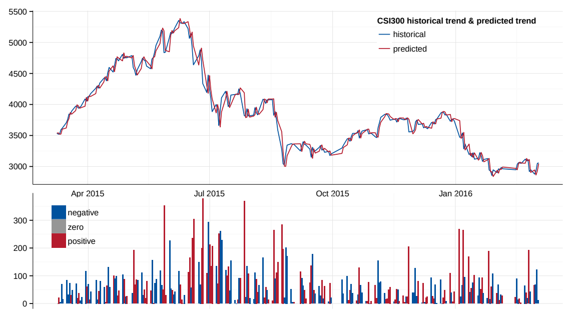


(b) Comparison between K-Means states and HMM hidden states

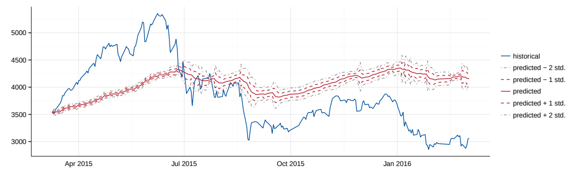
ILLUSTRATION A–1 CSI 300 daily data global decoding results



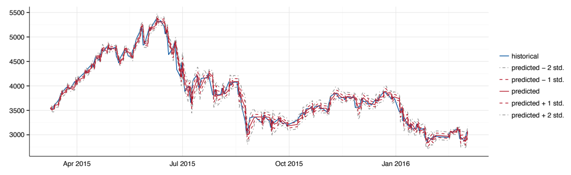
(a) CSI 300 simulated prediction result out-of-sample part



(b) CSI 300 adaptive prediction



(c) CSI 300 simulated prediction result with confidence band

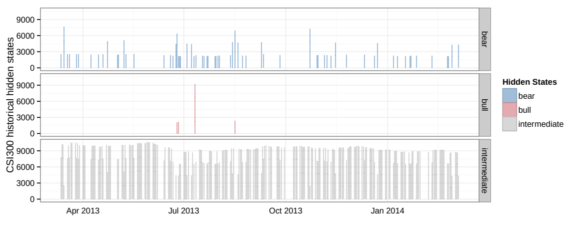


(d) CSI 300 adaptive prediction with confidence band

ILLUSTRATION A–2 CSI 300 daily data simulated prediction results

A.2 CSI 300 60min return series

A.2.1 2013 - 2014


(a) CSI 300 historical hidden states

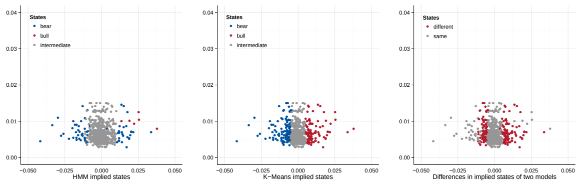
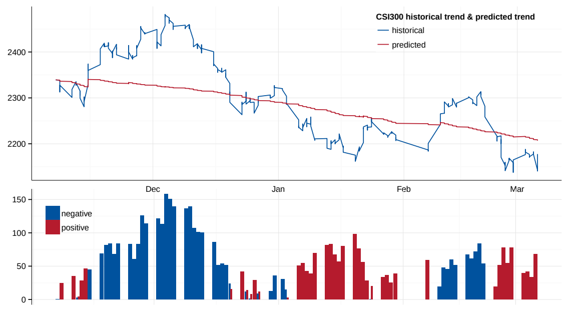
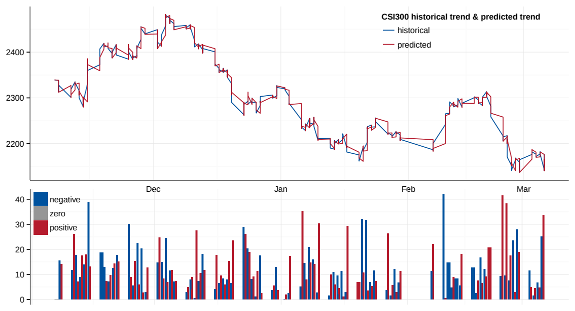
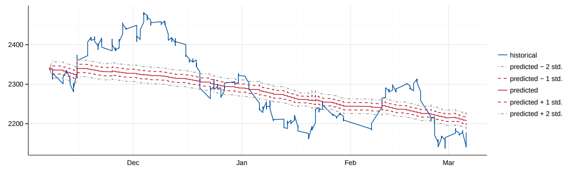

(b) Comparison between K-Means states and HMM hidden states

ILLUSTRATION A–3 CSI 300 60min data global decoding results - 2013∼2014


(a) CSI 300 simulated prediction result out-of-sample part


(b) CSI 300 adaptive prediction


(c) CSI 300 simulated prediction result with confidence band

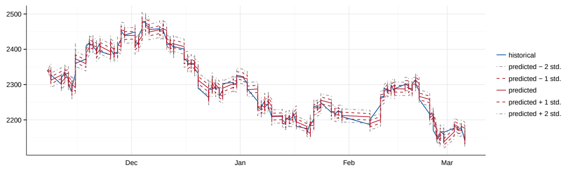
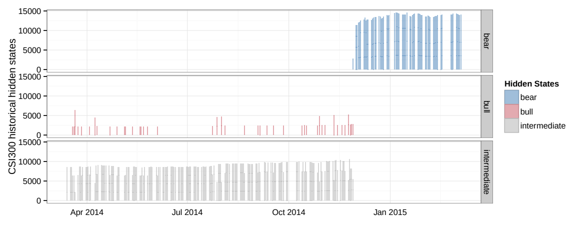
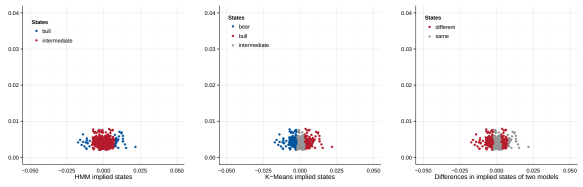

(d) CSI 300 adaptive prediction with confidence band

ILLUSTRATION A–4 CSI 300 60min data simulated prediction results - 2013∼2014

A.2.2 2014 - 2015

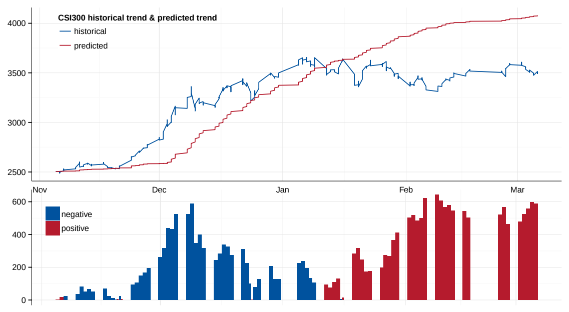


(a) CSI 300 historical hidden states

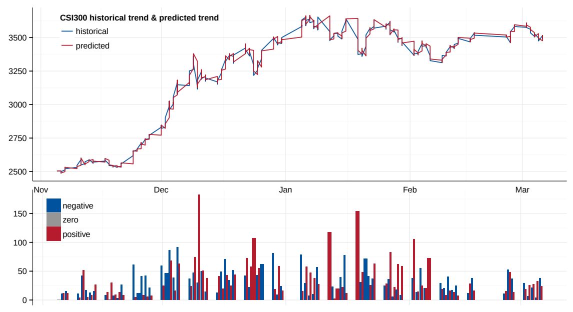


(b) Comparison between K-Means states and HMM hidden states

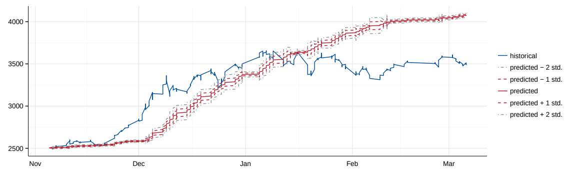
ILLUSTRATION A–5 CSI 300 60min data global decoding results - 2014∼2015



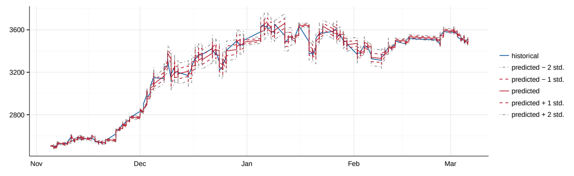
(a) CSI 300 simulated prediction result out-of-sample part



(b) CSI 300 adaptive prediction



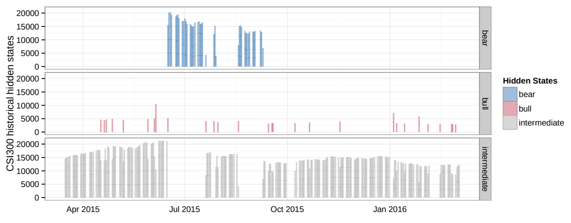
(c) CSI 300 simulated prediction result with confidence band



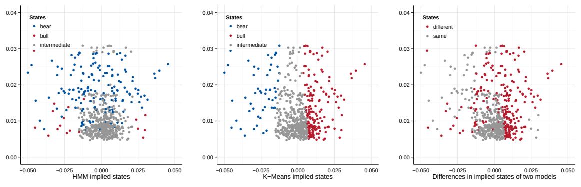
(d) CSI 300 adaptive prediction with confidence band

ILLUSTRATION A–6 CSI 300 60min data simulated prediction results - 2014∼2015

A.2.3 2015 - 2016

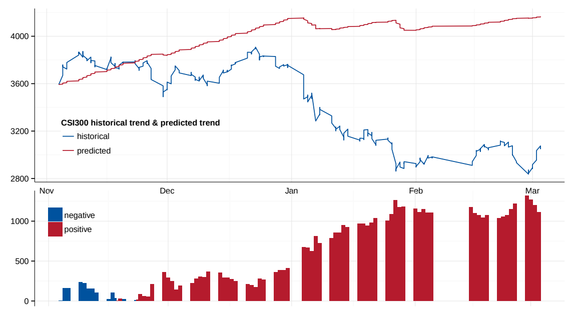


(a) CSI 300 historical hidden states

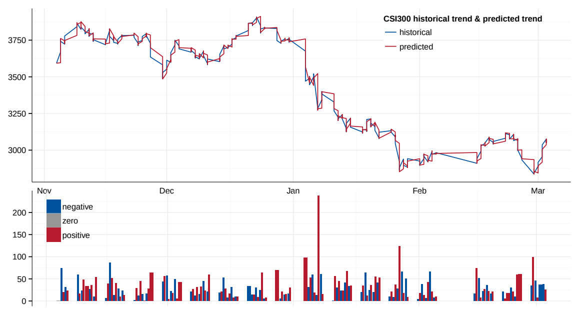


(b) Comparison between K-Means states and HMM hidden states

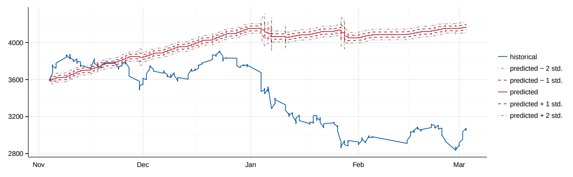
ILLUSTRATION A–7 CSI 300 60min data global decoding results - 2015∼2016



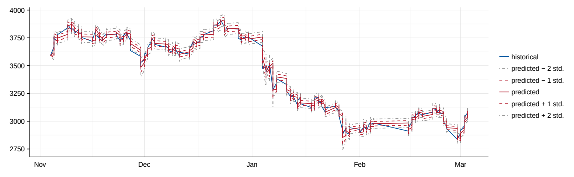
(a) CSI 300 simulated prediction result out-of-sample part



(b) CSI 300 adaptive prediction



(c) CSI 300 simulated prediction result with confidence band

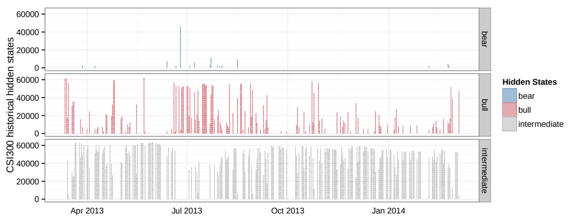


(d) CSI 300 adaptive prediction with confidence band

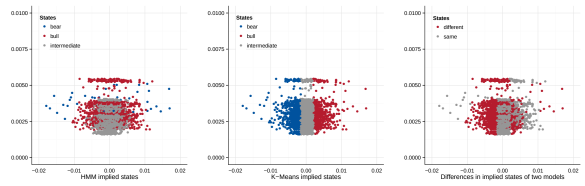
ILLUSTRATION A–8 CSI 300 60min data simulated prediction results - 2015∼2016

A.3 CSI 300 10min return series

A.3.1 2013 - 2014

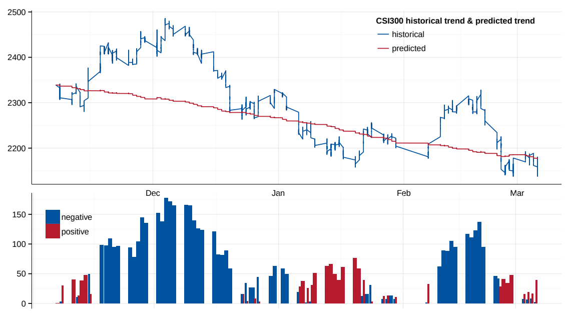


(a) CSI 300 historical hidden states

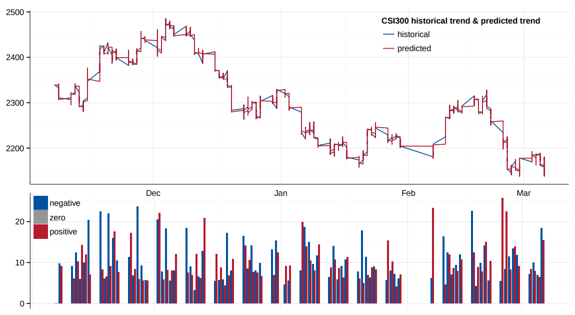


(b) Comparison between K-Means states and HMM hidden states

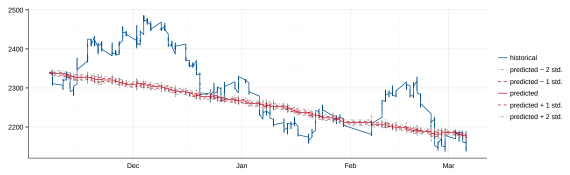
ILLUSTRATION A–9 CSI 300 10min data global decoding results - 2013∼2014



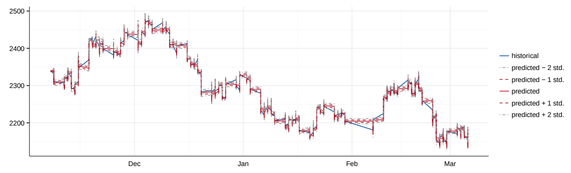
(a) CSI 300 simulated prediction result out-of-sample part



(b) CSI 300 adaptive prediction



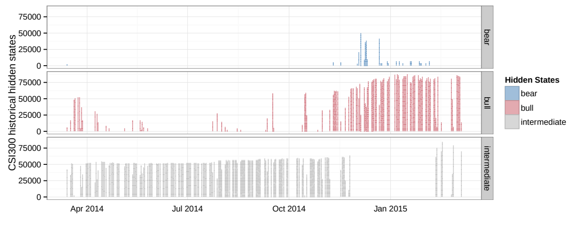
(c) CSI 300 simulated prediction result with confidence band



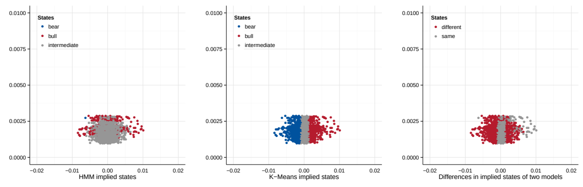
(d) CSI 300 adaptive prediction with confidence band

ILLUSTRATION A–10 CSI 300 10min data simulated prediction results - 2013∼2014

A.3.2 2014 - 2015

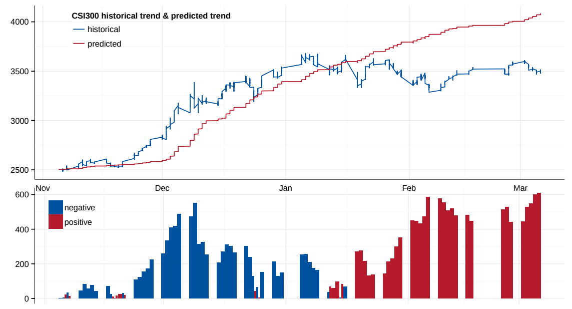


(a) CSI 300 historical hidden states

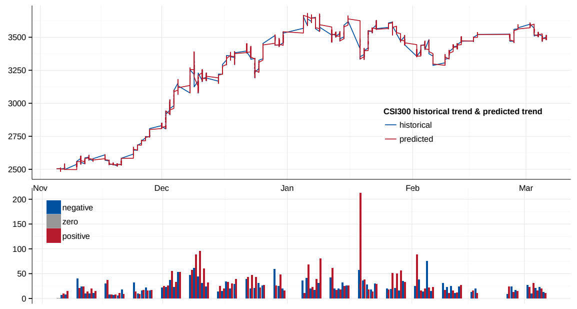


(b) Comparison between K-Means states and HMM hidden states

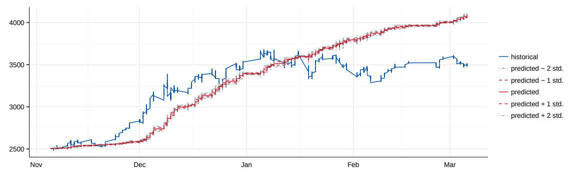
ILLUSTRATION A–11 CSI 300 10min data global decoding results - 2014∼2015



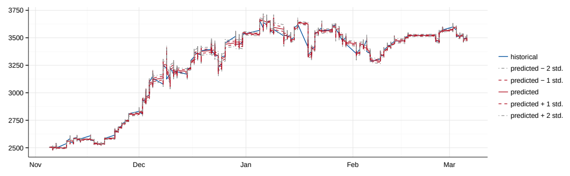
(a) CSI 300 simulated prediction result out-of-sample part



(b) CSI 300 adaptive prediction



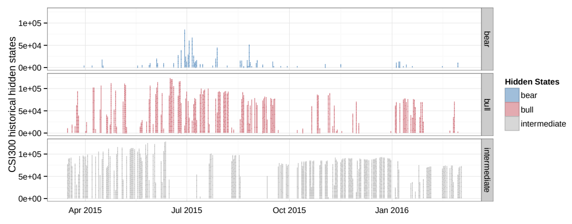
(c) CSI 300 simulated prediction result with confidence band



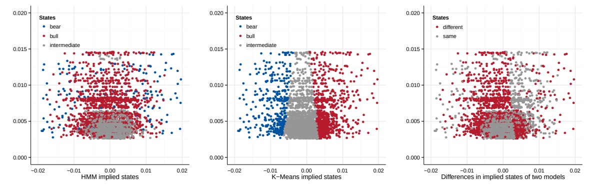
(d) CSI 300 adaptive prediction with confidence band

ILLUSTRATION A–12 CSI 300 10min data simulated prediction results - 2014∼2015

A.3.3 2015 - 2016

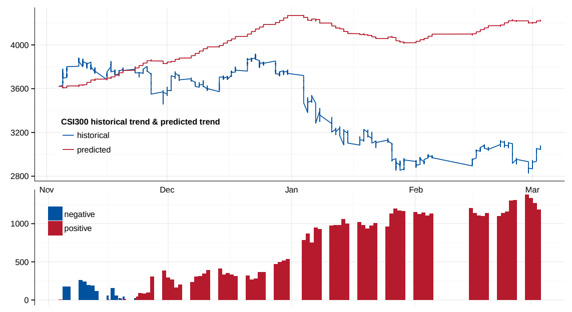


(a) CSI 300 historical hidden states

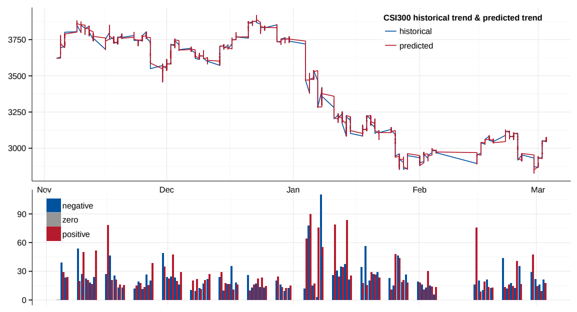


(b) Comparison between K-Means states and HMM hidden states

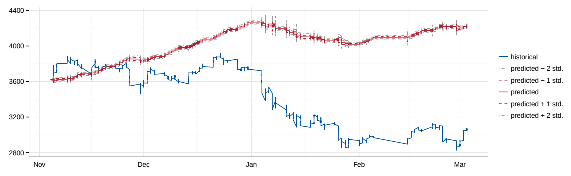
ILLUSTRATION A–13 CSI 300 10min data global decoding results - 2015∼2016



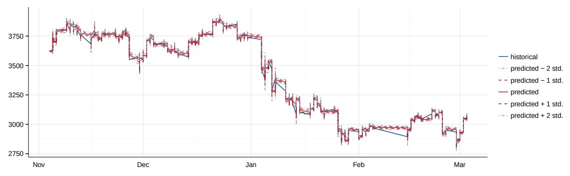
(a) CSI 300 simulated prediction result out-of-sample part



(b) CSI 300 adaptive prediction



(c) CSI 300 simulated prediction result with confidence band



(d) CSI 300 adaptive prediction with confidence band

ILLUSTRATION A–14 CSI 300 10min data simulated prediction results - 2015∼2016

Appendix B Model Realization Python Codes

B.1 User manuscript

All Python source codes for model realization are included in this appendix. In this section we provide a brief instruction to these code files, including the functionalities and how to run the codes, after slight changes on file paths if needed.

Sec. B.2 (`hmmMain.py`) is the main part of the model realization part, which strictly follows the system we describe in Ch. 4. Sec. B.3 (`hmmClass.py`) defines the Python class `HMM`, which is an encapsulated class object for the hidden Markov model. Data members and member functions related to the model estimation are created within (e.g. the observations, transition matrix, conditional distribution parameters, etc.). One can find detailed information about the object in the introduction part of `hmmClass.py`.

Before running the `hmmMain.py` program, make sure that `hmmClass.py` is under the same path of the main program. In order to change the paths for data loading and results saving, the user can modify the paths wherever `.from_csv` and `.to_csv` occur and make the adjustments. After certain changes, one can directly run `hmmMain.py` in an appropriate Python IDE.

Sec. B.4 (`hmmStates.py`) is independent from the main program and also requires (imports) the class definition module. The program is intended to estimate the goodness of fitness of different models with diverse number of hidden states, which generates the results presented in Sec. 5.6.5.

B.2 hmmMain.py

```
# # # Graduation Thesis – Main
#
# The project is intended to construct a complete system for stock returns
# series prediction based on Hidden Markov Model. This program is the main
# part of the project, realizing all aims of the empirical analysis part
# along with other source code files.
#
# Source file lists
#    – hmmMain.py  : main part of the program
#    – hmmClass.py : include construction of the model, in the form of class
#    – hmmStates.py: a separate program to find the optimal number of hidden
#                    states
#
# Data files are import from .csv files and results will be stored in .csv
# format as well. Visualization analysis and presentation of the data will
# be realized in other programs.

# # # # # Program Starts Here # # # # #

## Library Import
import numpy as np
import pandas as pd
from math import sqrt
from sklearn.cluster import KMeans
from hmmClass import HMM

## Data Loading
dataYear = ''
dataFreq = 'daily'
dataFreqMin = 240
#dataFile = '/SP500Data_' + dataFreq + dataYear + '.csv'
dataFile = '/CSI300Data_' + dataFreq + dataYear + '.csv'
dataRaw = pd.DataFrame.from_csv(dataFreq + dataFile,index_col = False)
dataRet = pd.DataFrame({'ret': np.diff(np.log(dataRaw['close'])),
                        'time': dataRaw['time'][1:]})

## In–sample & Out–of–sample Definition
numMin = 240 / dataFreqMin
numDay = len(dataRaw) / numMin
numStdDay = 5
lenInSample = int((numDay – numStdDay)*2/3)
lenOutSample = numDay – lenInSample – numStdDay
dataRetIn = dataRet.ix[0:(lenInSample + numStdDay)*numMin–1].copy()
dataRetIn.ix[:,'std'] = 0.0
for k in range(numStdDay*numMin,(lenInSample + numStdDay)*numMin):
    dataRetIn.ix[k,'std'] = np.std(dataRetIn.ix[(k-numStdDay*numMin):(k-1),
                                                'ret'])
dataRetIn = dataRetIn.tail(lenInSample*numMin)

## K-Means Clustering
# K-Means is used to find the initial probability distribution of the three
# states and corresponding conditional distribution parameters.
# The parameters are used for HMM initialization.
numState = 3
label = ['bear','intermediate','bull']
modelKMeans = KMeans(n_clusters = numState).fit(dataRetIn[['ret','std']])
order = np.argsort(modelKMeans.cluster_centers_.T[0,:])
matParam = modelKMeans.cluster_centers_.T[:,order]
dataRetIn['label'] = ''
for k in range(0,len(dataRetIn)):
    dataRetIn.ix[dataRetIn.index[k],'label'] = label[np.where(order ==
        modelKMeans.labels_[k])[0]]
matInit = np.array((dataRetIn['label'].value_counts())[label]/\
          len(dataRetIn))
matTrans = np.ones([numState,numState])/numState
dataRetIn.to_csv(dataFreq + '/dataReturnInSample' + dataYear + '.csv',
    index = False,encoding='utf-8',columns=['time','ret','std','label'])
dataRetIn.to_csv(dataFreq + '/dataPredict' + dataYear + '.csv',
    index = False,encoding='utf-8',columns=['time','ret','std'])

## Entire Period Estimation
dataRet = dataRet.ix[numStdDay*numMin:,]
model = HMM(dataRet,matTrans,matParam,matInit)
model.EM()
model.viterbi()
dataRet['label'] = ''
for k in range(0,len(dataRet)):
    dataRet.ix[dataRet.index[k],'label'] = label[int(model.matState[k])]
dataRet.to_csv(dataFreq + '/dataHistory' + dataYear + '.csv',
    index = False,encoding = 'utf-8',columns = ['time','ret','label'])

## Initialization
dataPredict = pd.DataFrame({'time':dataRet['time'],'ret':0.0,'std':0.0},
                           index = dataRet.index)
seqTrans = pd.DataFrame({'x11':0.0,'x12':0.0,'x13':0.0,
                         'x21':0.0,'x22':0.0,'x23':0.0,
                         'x31':0.0,'x32':0.0,'x33':0.0},
                         index = dataRet.index)
seqParam = pd.DataFrame({'mu1':0.0,'mu2':0.0,'mu3':0.0,
                         'sigma1':0.0,'sigma2':0.0,'sigma3':0.0},
                         index = dataRet.index)
seqParam.ix[(lenInSample+numStdDay)*numMin-1,] = \
    matParam.reshape(1,2*numState)

## Estimation & Prediction
print 'HMM Loop starts now!'
for k in range(0,lenOutSample*numMin):
    print dataRet.ix[(lenInSample+numStdDay)*numMin+k,'time']
    model = HMM(dataRet.ix[:(lenInSample+numStdDay)*numMin+k-1],
                matTrans,matParam,matInit)
    model.EM()
    seqTrans.ix[(lenInSample+numStdDay)*numMin+k,] = model.matTrans.reshape
        (1,model.numState**2)
    seqParam.ix[(lenInSample+numStdDay)*numMin+k,] = model.matParam.reshape
        (1,2*model.numState)
    dataPredict.ix[(lenInSample+numStdDay)*numMin+k,'ret'] = model.matEnd.
        dot(model.matTrans).dot(model.matParam[0,])
    dataPredict.ix[(lenInSample+numStdDay)*numMin+k,'std'] = sqrt(((model.
        matEnd.dot(model.matTrans))**2).dot(model.matParam[1,]**2))
    pd.DataFrame(dataPredict.ix[(lenInSample+numStdDay)*numMin+k]).T.\
        to_csv(dataFreq + '/dataPredict' + dataYear + '.csv',mode = 'a',
        index = False,encoding = 'utf-8',
        header = False,columns = ['time','ret','std'])

seqTrans['time'] = dataRet['time']
seqParam['time'] = dataRet['time']
seqTrans.to_csv(dataFreq + '/matTrans' + dataYear + '.csv',
                index = False,encoding = 'utf-8',
                columns = ['time','x11','x12','x13',
                                  'x21','x22','x23',
                                  'x31','x32','x33'])
seqParam.to_csv(dataFreq + '/matParam' + dataYear + '.csv',
                index = False,encoding = 'utf-8',
                columns = list(['time']) + list(np.core.defchararray.add(
                    list(np.repeat('mu',3)),map(str,range(1,4)))) \
                + list(np.core.defchararray.add(list(np.repeat('sigma',3)),
                    map(str,range(1,4)))))
```

B.3 hmmClass.py

```
# # # Graduation Thesis — HMM Class Definition
#
# This program include the construction of a HMM class. The members are
# listed as follows:
#
# Member Data:
#   — obs: the observed sequence/time series
#   — numState: the number of hidden states
#   — matInit: the original probability distribution of hidden states
#   — matTrans: the state transition probability matrix
#   — matParam: the conditional distribution parameters of the observed
#              sequence, corresponding to each hidden state
#   — matDist: the probability distribution matrix, corresponding to each
#             observation at each state
#   — matEnd: the probability distribution of hidden states at the last
#            observation
#
# Member Functions:
#   — __init__: built—in initialization function of the class
#   — forward: calculates the forward probability of observed sequence
#   — backward: calculates the backward probability of observed sequence
#   — expState: calculates the expectation of states
#   — expStateTrans: calculates the expected state transition probability
#   — EM: the major part of HMM parameter (including state transition
#         probabilities, conditional distribution parameter, initial and
#         final state distribution) estimation using Baum—Welch algorithm
#   — viterbi: global decoding to find the most probable sequence of
#             hidden states using Viterbi algorithm

# # # # # Program Starts Here # # # # #
import numpy as np
import pandas as pd
from matplotlib.mlab import *

class HMM:
    def __init__(self,seqObserv,matTrans,matParam,matInit):
        '''HMM is the class defined to hold necessary data and functions
        for a standard hidden Markov model. The initialization of a HMM
        class requires 4 inputs given in the parameter list.

        Parameters
        ——————————
        seqObserv : a data frame holding the stock return series
        matTrans : the initial state transition probability matrix
        matParam : the initial conditional distribution parameters
```

```
                    corresponding to each hidden state
        matInit : the initial probability distribution of hidden states'''
        self.obs = seqObserv['ret'].copy()
        self.matTrans = matTrans.copy()
        self.matInit = matInit.copy()
        self.numState = matTrans.shape[0]
        self.matParam = matParam.copy()
        self.matDist = np.zeros([self.matTrans.shape[0],len(self.obs)])
        matDistTemp = pd.DataFrame(index = self.obs.index)
        for k in range(0,self.matTrans.shape[0]):
            matDistTemp['x' + str(k)] = normpdf(self.obs,self.matParam[0,k
                ],self.matParam[1,k])
        self.matDist = np.array(matDistTemp.T)

    def forward(self):
        T = len(self.obs)
        alpha = np.zeros([self.numState,T])
        scale = np.zeros(T)
        alpha[:,0] = self.matInit[:] * self.matDist[:,0]
        scale[0] = np.sum(alpha[:,0])
        alpha[:,0] /= scale[0]
        for t in range(1,T):
            if np.sum(self.matDist[:,t]) != 0:
                alpha[:,t] = np.dot(alpha[:,t-1].T,self.matTrans).T *\
                             self.matDist[:,t]
                scale[t] = np.sum(alpha[:,t])
                alpha[:,t] /= scale[t]
            else:
                alpha[:,t] = alpha[:,t-1]
                scale[t] = scale[t-1]
        logp = np.sum(np.log(scale[:]))
        return alpha, scale, logp

    def backward(self,scale):
        T = len(self.obs)
        beta = np.zeros([self.numState,T])
        beta[:,T-1] = 1/scale[T-1]
        for t in range(T-1,0,-1):
            beta[:,t-1] = np.dot(self.matTrans,
                                 (self.matDist[:,t]*beta[:,t]))
            beta[:,t-1] /= scale[t-1]
        return beta

    def expState(self,alpha,beta):
        gamma = np.zeros(alpha.shape)
        gamma = alpha[:,:] * beta[:,:]
        gamma = gamma / ((np.sum(gamma,0) == 0) + np.sum(gamma,0))
        return gamma

    def expStateTrans(self,alpha,beta):
        T = len(self.obs)
        xi = np.zeros((self.numState,self.numState,T−1))
        for t in range(T−1):
            denom = np.dot(np.dot(alpha[:,t].T, self.matTrans) * \
                    self.matDist[:,t+1].T,beta[:,t+1])
            for i in range(self.numState):
                numer = alpha[i,t] * self.matTrans[i,:] * \
                        self.matDist[:,t+1].T * beta[:,t+1].T
                xi[i,:,t] = numer / (denom + (denom == 0))
        return xi

    def EM(self):
        T = len(self.obs)
        criterion = 0.001
        alpha, scale, logp = self.forward()
        beta = self.backward(scale)
        gamma = self.expState(alpha,beta)
        xi = self.expStateTrans(alpha,beta)
        matDistTemp = self.matDist
        loop = 0
        while True:
            for i in range(0,self.numState):
                denominator = np.sum(gamma[i,0:T−1])
                for j in range(0,self.numState):
                    numerator = np.sum(xi[i,j,0:T−1])
                    self.matTrans[i,j] = numerator / \
                                    (denominator + (denominator == 0))

            tempMu = np.array([np.sum(gamma*np.array(self.obs.T),1)/np.sum(
                gamma,1)])
            tempSigma = np.sqrt(np.array([np.sum(gamma*(np.array(self.obs.T
                ) − tempMu.T)**2,1) / np.sum(gamma,1)]))
            self.matParam[0,:] = tempMu
            self.matParam[1,:] = tempSigma
            matDistTemp = pd.DataFrame(index = self.obs.index)
            for k in range(0,self.matTrans.shape[0]):
                matDistTemp['x' + str(k)] = normpdf(self.obs,self.matParam
                    [0,k],self.matParam[1,k])
            matDistTemp = np.array(matDistTemp.T)

            self.matDist = matDistTemp
            self.matInit = gamma[:,0]
            self.matEnd = gamma[:,−1]

            alpha, scale, logpNew = self.forward()
            beta = self.backward(scale)
            gamma = self.expState(alpha,beta)
            xi = self.expStateTrans(alpha,beta)

            delta = abs(logpNew - logp)
            logp = logpNew
            loop += 1
            if delta <= criterion:
                break

        self.valAIC = -2*logp + 2*(self.numState**2 + 2*self.numState - 1)
        self.valBIC = -2*logp + \
                      np.log(T)*(self.numState**2 + 2*self.numState - 1)

    def viterbi(self):
        T = len(self.obs)
        psi = np.zeros([self.numState,T])
        delta = np.zeros([self.numState,T])
        matState = np.zeros(T)
        temp = np.zeros([self.numState,self.numState])

        delta[:,0] = np.log(self.matInit) + np.log(self.matDist[:,0])
        for t in range(1,T):
            temp = (delta[:,t-1] + np.log(self.matTrans.T)).T
            ind = np.argmax(temp, axis = 0)
            psi[:,t] = ind
            delta[:,t] = np.log(self.matDist[:,t]) + \
                         temp[ind,range(self.numState)]

        max_ind = np.argmax(delta[:,T-1])
        matState[T-1] = max_ind
        for t in reversed(range(0,T-1)):
            matState[t] = psi[matState[t+1],t+1]

        self.logProbState = delta[:,T-1][max_ind]
        self.matState = matState
```

B.4 hmmStates.py

```
# # # Graduation Thesis – Number of States
#
# This program is very similar to the main one, except that is elminates
# the part of prediction and focuses on evaluating the goodness of fit of
# models with different number of hidden states. Notice that the analysis
# is performed only for CSI300 daily data.
#
# Data files are import from .csv files and results will be stored in .csv
# format as well.

# # # # # Program Starts Here # # # # #

## Library Import
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from hmmClass import HMM

## Data Loading
dataPath = 'states'
dataPathMin = 240
dataRaw = pd.DataFrame.from_csv('daily/CSI300Data_daily.csv',
                                index_col = False)
dataRet = pd.DataFrame({'ret': np.diff(np.log(dataRaw['close'])),
                        'time': dataRaw['time'][1:]})

## In–sample & Out–of–sample Definition
numMin = 240 / dataPathMin
numDay = len(dataRaw) / numMin
numStdDay = 5
lenInSample = int((numDay – numStdDay)*2/3)
lenOutSample = numDay – lenInSample – numStdDay
dataRetIn = dataRet.ix[0:(lenInSample + numStdDay)*numMin–1].copy()
dataRetIn.ix[:,'std'] = 0.0
for k in range(numStdDay*numMin,(lenInSample + numStdDay)*numMin):
    dataRetIn.ix[k,'std'] = np.std(dataRetIn.ix[(k–numStdDay*numMin):(k–1),
                                                'ret'])
dataRetIn = dataRetIn.tail(lenInSample*numMin)

## Loop for Different Number of States
seqAIC = pd.DataFrame({'number':range(2,7),'value':0},index = range(2,7))
seqBIC = pd.DataFrame({'number':range(2,7),'value':0},index = range(2,7))

for numState in range(2,7):
    print 'Number of States:', numState
    ## K-Means Clustering
    label = range(0,numState)
    modelKMeans = KMeans(n_clusters = numState).\
                     fit(dataRetIn[['ret','std']])
    order = np.argsort(modelKMeans.cluster_centers_.T[0,:])
    matParam = modelKMeans.cluster_centers_.T[:,order]
    dataRetIn['label'] = ''
    for k in range(0,len(dataRetIn)):
        dataRetIn.ix[dataRetIn.index[k],'label'] = label[np.where(order ==
            modelKMeans.labels_[k])[0]]
    matInit = np.array((dataRetIn['label'].value_counts())[label]/\
            len(dataRetIn))
    matTrans = np.ones([numState,numState])/numState

    ## Entire Period Estimation
    dataRet = dataRet.ix[numStdDay*numMin:,]
    model = HMM(dataRet,matTrans,matParam,matInit)
    model.EM()
    seqAIC.ix[numState,'value'] = model.valAIC
    seqBIC.ix[numState,'value'] = model.valBIC

seqAIC.to_csv(dataPath + '/aic.csv',index = False,encoding = 'utf-8')
seqBIC.to_csv(dataPath + '/bic.csv',index = False,encoding = 'utf-8')
```

REFERENCE

[1] WHITTLE P. Hypothesis testing in time series analysis[M], Vol. 4.[S.l.]: Almqvist & Wiksells, 1951.

[2] WIKIPEDIA. Mixture model — Wikipedia, The Free Encyclopedia[R].[S.l.]: [s.n.] , 2016. https://en.wikipedia.org/w/index.php?title=Mixture_model&oldid=718940784.

[3] BEHBOODIAN J. On the Modes of a Mixture of Two Normal Distributions[J]. Technometrics, 1970, 12(1):131–139.

[4] HASSELBLAD V. Estimation of Finite Mixtures of Distributions from the Exponential Family[J]. Journal of the American Statistical Association, 1969, 64(328):1459–1471.

[5] ZUCCHINI W, MACDONALD I L. Hidden Markov models for time series[M], Monographs on Statistics and Applied Probability, vol. 110.[S.l.]: CRC Press, Boca Raton, FL, 2009.

[6] MCLACHLAN G, PEEL D. Finite mixture models[M].[S.l.]: John Wiley & Sons, 2004.

[7] FRÜHWIRTH-SCHNATTER S. Finite mixture and Markov switching models[M].[S.l.]: Springer Science & Business Media, 2006.

[8] WIKIPEDIA. Markov chain — Wikipedia, The Free Encyclopedia[R].[S.l.]: [s.n.] , 2016. https://en.wikipedia.org/w/index.php?title=Markov_chain&oldid=718668115.

[9] KEMENY J G, SNELL J L, et al. Finite markov chains[M], Vol. 356.[S.l.]: van Nostrand Princeton, NJ, 1960.

[10] MEYN S P, TWEEDIE R L. Markov chains and stochastic stability[M].[S.l.]: Springer Science & Business Media, 2012.

[11] MACQUEEN J. Some methods for classification and analysis of multivariate observations[M]//Proc. Fifth Berkeley Sympos. Math. Statist. and Probability (Berkeley, Calif., 1965/66).[S.l.]: Univ. California Press, Berkeley, Calif., 1967:Vol. I: Statistics, pp. 281–297.

[12] FORGY E W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications[J]. Biometrics, 1965, 21:768–769.

[13] HARTIGAN J A. Clustering algorithms[M].[S.l.]: John Wiley & Sons, 1975.

[14] CONT R. Empirical properties of asset returns: stylized facts and statistical issues[J]. Quantitative Finance, 2001, 1(2):223–236.

[15] Babu M S, Geethanjali N, Satyanarayana B. Clustering Approach to Stock Market Prediction[J]. International Journal of Advanced Networking and Applications, 2012, 3(4):1281–1291.

[16] Gupta A, Sharma S D. Clustering-Classification Based Prediction of Stock Market Future Prediction[J]. IJCSIT) International Journal of Computer Science …, 2014.

[17] Rabiner L, Juang B. An introduction to hidden Markov models[J]. IEEE ASSP Magazine, 1986, 3(1):4–16.

[18] Ephraim Y, Merhav N. Hidden Markov processes[J]. Institute of Electrical and Electronics Engineers. Transactions on Information Theory, 2002, 48(6):1518–1569.

[19] Leroux B G, Puterman M L. Maximum-penalized-likelihood estimation for independent and Markov-dependent mixture models[J]. Biometrics, 1992:545–558.

[20] Haas M, Mittnik S, Paolella M S. A new approach to Markov-switching GARCH models[J]. Journal of Financial …, 2004.

[21] Stratonovich R L. Conditional markov processes[J]. Theory of Probability & Its Applications, 1960, 5(2):156–178.

[22] Wikipedia. Hidden Markov model — Wikipedia, The Free Encyclopedia[R].[S.l.]: [s.n.] , 2016. https://en.wikipedia.org/w/index.php?title=Hidden_Markov_model&oldid=716209275.

[23] Rabiner L R. A tutorial on hidden Markov models and selected applications in speech recognition[J]. Proceedings of the IEEE, 1989, 77(2):257–286.

[24] Ghassempour S, Girosi F, Maeder A. Clustering Multivariate Time Series Using Hidden Markov Models[J]. International Journal of Environmental Research and Public Health, 2014, 11(3):2741–2763.

[25] Baum L E, Petrie T. Statistical inference for probabilistic functions of finite state Markov chains[J]. Annals of Mathematical Statistics, 1966, 37(6):1554–1563.

[26] Baum L E, Eagon J A. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology[J]. Bulletin of the American Mathematical Society, 1967, 73(3):360–363.

[27] Baum L E, Petrie T, Soules G, et al. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains[J]. Annals of Mathematical Statistics, 1970, 41(1):164–171.

[28] Fredkin D R, Rice J A. Bayesian restoration of single-channel patch clamp recordings[J]. Biometrics, 1992:427–448.

[29] VITERBI A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm[J]. Institute of Electrical and Electronics Engineers. Transactions on Information Theory, 1967, 13(2):260–269.

[30] BECKERS J M, RIXEN M. EOF calculations and data filling from incomplete oceanographic datasets[J]. Journal of Atmospheric and Oceanic Technology, 2003, 20(12):1839–1856.

[31] REUTER H I, NELSON A, JARVIS A. An evaluation of void-filling interpolation methods for SRTM data[J]. International Journal of Geographical Information Science, 2007, 21(9):983–1008.

[32] BRAILOVSKIY L, HERMAN M. Prediction of Financial Time Series Using Hidden Markov Models[C]//2014 ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference. .[S.l.]: [s.n.] , 2014.

[33] HASSAN M R, NATH B. Stock market forecasting using hidden Markov model: a new approach[J]. … Design and Applications, 2005.

[34] HASSAN M R, NATH B, KIRLEY M. A fusion model of HMM, ANN and GA for stock market forecasting[J]. Expert Systems with Applications, 2007, 33(1):171–180.

[35] GUPTA A, DHINGRA B. Stock market prediction using Hidden Markov Models[C]//2012 Students Conference on Engineering and Systems (SCES).[S.l.]: IEEE, 2012:1–4.

[36] WASSON T, HARTEMINK A J. An ensemble model of competitive multi-factor binding of the genome[J]. Genome Research, 2009, 19(11):2101–2112.

[37] BENGIO Y, DE MORI R, FLAMMIA G, et al. Global optimization of a neural network-hidden Markov model hybrid[J]. IEEE Transactions on Neural Networks, 1992, 3(2):252–259.

[38] TIPPING M E, BISHOP C M. Mixtures of Probabilistic Principal Component Analyzers[J]. Neural Computation, 1999, 11(2):443–482.

[39] DIAS J G, VERMUNT J K, RAMOS S. Clustering financial time series: New insights from an extended hidden Markov model[J]. European Journal of Operational Research, 2015, 243(3):852–864.

[40] CREAL D. A Survey of Sequential Monte Carlo Methods for Economics and Finance[J]. Econometric Reviews, 2012, 31(3):245–296.

[41] KON S J. Models of Stock Returns–A Comparison[J]. The Journal of Finance, 1984, 39(1):147–165.

[42] NYSTRUP P, MADSEN H, LINDSTRÖM E. Stylised facts of financial time series and hidden Markov models in continuous time[J]. Quantitative Finance, 2015.

[43] JACOB P E. Sequential Bayesian inference for implicit hidden Markov models and current limitations[J]. arXiv.org, 2015.

[44] Zhang Y. Prediction of financial time series with Hidden Markov Models[J]. 2004.

[45] Wikipedia. Particle filter — Wikipedia, The Free Encyclopedia[R].[S.l.]: [s.n.] , 2016. https://en.wikipedia.org/w/index.php?title=Particle_filter&oldid=717665870. [Online; accessed 9-May-2016].

[46] Del Moral P. Nonlinear filtering: Interacting particle resolution[J]. Comptes Rendus De L Academie Des Sciences Serie I-Mathematique, 1997, 325(6):653–658.

[47] Liu J S, Chen R. Sequential Monte Carlo methods for dynamic systems[J]. Journal of the American Statistical Association, 1998, 93(443):1032–1044.

[48] Kahn H, Marshall A W. Methods of reducing sample size in Monte Carlo computations[J]. Journal of the Operations Research Society of America, 1953, 1(5):263–278.

[49] Marshall A W. The use of multistage sampling schemes in Monte Carlo computations[J]. 1954.

[50] Richard J F, Zhang W. Efficient high-dimensional importance sampling[J]. Journal of Econometrics, 2007, 141(2):1385–1411.

[51] Rubinstein R Y, Kroese D P. The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning[M].[S.l.]: Springer Science & Business Media, 2013.

[52] Gao T, Li J. A Derivative-Free Trust-Region Method for Reliability-Based Optimizations[J]. arXiv.org, 2016.

[53] Rubin D B. Comment: A noniterative Sampling/Importance Resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The SIR algorithm[J]. Journal of the American Statistical Association, 1987, 82(398):542–543.

[54] Gordon N J, Salmond D J, Smith A F M. Novel approach to nonlinear/non-Gaussian Bayesian state estimation[J]. Radar and Signal Processing, IEE Proceedings F, 1993, 140(2):107–113.

[55] Toni T, Welch D, Strelkowa N, et al. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems[J]. Journal of the Royal Society Interface, 2009, 6(31):187–202.

[56] Dean T A, Singh S S, Jasra A, et al. Parameter estimation for hidden Markov models with intractable likelihoods[J]. Scandinavian Journal of Statistics, 2014, 41(4):970–987.

Acknowledgements

I would like to express my great gratitude to my parents, teachers and friends. Thank you for backing me up all the time. It is never tough to come up with an undergraduation thesis, but it is never easy to keep fighting until the last second of my college life even I am graduating. Life has been especially hard in the last year, and thanks to you all because I know I would be confused without your advice and encouragements.

Thanks to Prof. Jianzhong Lin for being the my supervisor of the thesis, and for spending time reading the thesis and providing suggestions.

Thanks to my parents for never pushing me even slightly. Thank you for telling me to stop and take a break whenever I am working hard. I know I would never stop, yet I still need this to find solace when I am too frustrated to rest.

Special thanks to Prof. Jinglai Li for introducing me to the realm of academic research, and for all the supports during my graduate school applications. Special thanks to Qun Zhao, a true friend and a great mentor, for all kinds of advice and help on career path and academic research and other things that are too numerous to mention. Special thanks to my competitors during all exams, interviews and contests. Thank you for always reminding me that there is still a long long way to go, to become a better person and to do more things.

Finally, special thanks to myself, for never giving up and never having the thought to give up, for all the fun and excitement during the life adventure, and for all the passion that keeps me fighting and looking for new funs.

Life sucks when you are ordinary. So be more than that, beat whatever stands between yourself and your goals, especially your past selves, the vulnerable and weak ones that hinder you from going further. If you should fail, you fail, but never stop trying. If you would live, you live, and always live better, long and prosper.