PREDICTION OF FINANCIAL TIME SERIES WITH HIDDEN MARKOV MODELS

by

Yingjian Zhang

B.Eng. Shandong University, China, 2001

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
May 2004

APPROVAL

Name: Yingjian Zhang

Degree: Master of Applied Science

Title of thesis: Prediction of Financial Time Series with Hidden Markov Models

Examining Committee: Dr. Ghassan Hamarneh
Chair

Dr. Anoop Sarkar
Assistant Professor, Computing Science
Senior Supervisor

Dr. Andrey Pavlov
Assistant Professor, Business Administration
Co-Senior Supervisor

Dr. Oliver Schulte
Assistant Professor, Computing Science
Supervisor

Dr. Martin Ester
Associate Professor, Computing Science
SFU Examiner

Date Approved: _____

Abstract

In this thesis, we develop an extension of the Hidden Markov Model (HMM) that addresses two of the most important challenges of financial time series modeling: non-stationary and non-linearity. Specifically, we extend the HMM to include a novel exponentially weighted Expectation-Maximization (EM) algorithm to handle these two challenges. We show that this extension allows the HMM algorithm to model not only sequence data but also dynamic financial time series. We show the update rules for the HMM parameters can be written in a form of exponential moving averages of the model variables so that we can take the advantage of existing technical analysis techniques. We further propose a double weighted EM algorithm that is able to adjust training sensitivity automatically. Convergence results for the proposed algorithms are proved using techniques from the EM Theorem.

Experimental results show that our models consistently beat the S&P 500 Index over five 400-day testing periods from 1994 to 2002, including both bull and bear markets. Our models also consistently outperform the top 5 S&P 500 mutual funds in terms of the Sharpe Ratio.

To my mom

Acknowledgments

I would like to express my deep gratitude to my senior supervisor, Dr. Anoop Sarkar for his great patience, detailed instructions and insightful comments at every stage of this thesis. What I have learned from him goes beyond the knowledge he taught me; his attitude toward research, carefulness and commitment has and will have a profound impact on my future academic activities.

I am also indebted to my co-senior supervisor, Dr. Andrey Pavlov, for trusting me before I realize my results. I still remember the day when I first presented to him the "super five day pattern" program. He is the source of my knowledge in finance that I used in this thesis. Without the countless time he spent with me, this thesis surely would not have been impossible.

I cannot thank enough my supervisor, Dr. Oliver Schulte. Part of this thesis derives from a course project I did for his machine learning course in which he introduced me the world of machine learning and interdisciplinary research approaches.

Many of my fellow graduate students offered great help during my study and research at the computing science school. I thank Wei Luo for helping me with many issues regarding the LaTeXand Unix; Cheng Lu and Chris Demwell for sharing their experiences with Hidden Markov Models; my lab-mates at the Natural Language Lab for the comments at my practice talk and for creating such a nice environment in which to work.

I would also like to thank Mr. Daniel Brimm for the comments from the point of view of a real life business professional.

Finally I would like to thank my mom for all that she has done for me. Words cannot express one millionth of my gratitude to her but time will prove the return of her immense love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis explores a novel model in the interdisciplinary area of computational finance. Similar to traditional financial engineering, we build models to analyze and predict financial time series. What distinguish us from the previous work is rather than building models based on regression equations, we apply a machien learning model–the Hidden Markov Model to financial time series. This chapter describes the basic concepts in financial time series, the difficulties of accurate financial time series prediction, the limitations of regression models, motivation and objective of our machine learning model. The organization of the thesis is also outlined in the later part this chapter.

1.1 Financial Time Series

Financial time series data are a sequence of prices of some financial assets over a specific period of time. In this thesis we will focus our research on the most volatile and challenging financial market – the stock market. More specifically, our experiment data include the daily data of NASDAQ Composite Index, the Dow Jones Industrial Index and the S&P 500 Index.

Over the past 50 years, there has been a large amount of intensive research on the stock market. All the financial analysts carry out many kinds of research in the hope of accomplishing one goal: to beat the market. To beat the market means to have a rate of return that is consistently higher than the average return of the market while keeping the same level of risk as the market. In this thesis, the average rate of return is the Dow Jones Industrial Index , which is a weighted average of 30 significant stocks, and the S&P

500 Index, which is a combination of 500 stocks. Generally there are two main streams in financial market analysis: fundamental analysis and technical analysis.

Fundamental analysis is the examination of the underlying forces that affect the well-being of the economy, industry sectors, and individual companies. For example, to forecast future prices of an individual stock, fundamental analysis combines economic, industry, and company analyses to derive the stock's current fair value and forecast its future value. If fair value is not equal to the current stock price, fundamental analysts believe that the stock is either over or under valued and the market price will ultimately gravitate toward fair value.

There are some inherent weaknesses in fundamental analysis. First, it is very time consuming: by the time fundamental analysts collect information of the company or economy health and give their prediction of the future price, the current price has already changed to reflect the most updated information. Second, there is an inevitable analyst bias and accompanying subjectivity.

Technical analysis is the examination of past price movements to forecast future price movements. In our experiments we base our forecast of the daily stock price on the past daily prices.

The theoretical basis of technical analysis is: price reflects all the relevant information; price movements are not totally random; history repeats itself. And all these are derived from the belief in the non-efficiency of the financial markets. We will discuss the market efficiency hypothesis later in this chapter and the evidence of non-efficiency in the market. We will focus only on methods of technical analysis in this thesis.

In this thesis, we use *forecast* and *prediction* alternatively to represent the process of generating unseen data in a financial time series.

1.2 Return of Financial Time Series

One of the most important indicators of measuring the performance of mutual funds or prediction systems is the return of the system. Usually we have a benchmark series to compare against. In our experiment we use the S&P 500 index as the benchmark.

Let P_t be the price of an asset at time index t. The one-period *simple net return* R_t of a financial time series is calculated with the following formula:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{1.1}$$

For a k-period financial time series, the return at the end of the period is a cumulative return that is composed of all the periods:

$$\begin{aligned} 1 + P_t[k] &= (1 + R_1)(1 + R_2) \dots (1 + R_k) \\ &= \prod_{t=1}^{k} (1 + R_t) \end{aligned} \tag{1.2}$$

In developing a prediction system, our goal is to make as many accurate predictions as possible. This will result in a return curve for the prediction system. The higher the prediction accuracy, the higher the return curve will be above the index curve. Figure 1.1 shows the return based on a strategy using some prediction system. If we invest 1 dollar at the beginning of the period, we will get \$2.1 at the end of the period.

If one get an accuracy 100% for any 400 consecutive days in the ten year period 1987 to 1994, the return at the end of the period would be roughly 25 times the initial input. This is the upper bound for the return. Normally we care more about the return than the accuracy rate of the prediction because the simple net return at each day is different. For example, two days of accurate predictions of small changes may not offset the loss of one day's incorrect prediction of a large change in the market.

Prediction of financial time series is a very extensive topic. In this thesis, we only predict the direction of the time series, generating a signal of either *up* or *down* for each trading day.

1.3 The Standard & Poor 500 data

The real world financial time series we input into the HMM prediction system is the daily return of the Standard & Poor 500 Index (S&P 500). The S&P 500 index is a weighted average of the stock prices of 500 leading companies in leading industry sectors of the US. The index reflects the total market value of all component stocks relative to a particular period of time. The S&P 500 Index is viewed as one of the best indicators of the US economy health. Unlike the Dow Jones Industrial Average, which is an average of the 30 component stock prices, the S&P 500 represents the market value of the 500 component stocks. To

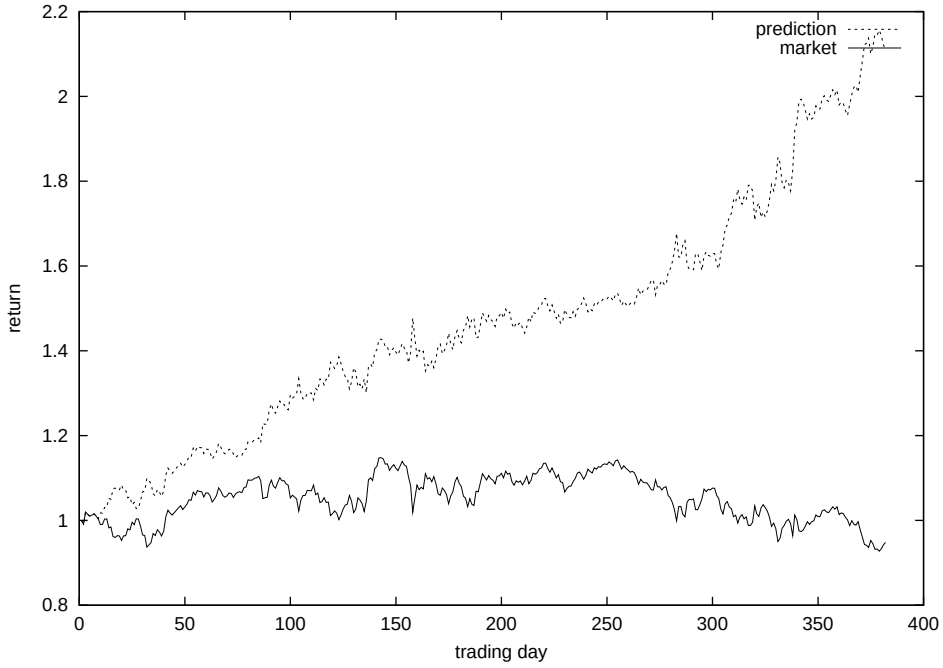

Figure 1.1: The return curve of a prediction system. $1 input at the beginning of the trading period generates a return of $2.1 while the index drops below 1.

calculate the S&P 500 Index, we first add up the total market value of all the 500 stocks, then divide them by the total number of outstanding shares in the market. Because it looks at the total market value and includes stocks from different industries than the Dow, the S&P 500 has some different characteristics from the Dow.

In some scenarios of the experiments we also include the Dow Jones Index that span during the same time period as the S&P 500 Index. One reason to use another time series to facilitate the prediction is that there are sideway movements in the long term trends, also called the *whipsaws*(see 3.1). These are the reverse signals in a long term trend. We need to identify the real trend change signals from the noisy whipsaws. The Dow is a less sensitive index than the S&P 500 so we use it to help smooth the prediction.

The return of the S&P 500 Index for each day is calculated using the techniques introduction in Section 1.2. The price we use for the calculation is the close price for each day. In some experiments we use a second sequence of data: the volume. An example of the data format is shown in Table 1.1:

Date	Close	Return	Volume
Jan-2-2002	1154.67	0.00574	1418169984
Jan-3-2003	1165.27	0.00918	1874700032
Jan-4-2002	1172.51	0.006213	1943559936
Jan-7-2002	1164.89	-0.0065	1682759936

Table 1.1: Data format of the experiment (for the segment from Jan 2, 2002 to Jan 7, 2002)

1.4 Traditional Techniques and Their Limitations

Traditionally, people use various forms of statistical models to predict volatile financial time series [GP86]. Sometimes stocks move without any apparent news, as the Dow Jones Industrial Index did on Monday, October 19, 1987. It fell by 22.6 percent without any obvious reason. We have to identify whether an irrational price movement is *noise* or is part of a pattern. Perhaps the most prominent of all these models is the Black-Scholes model for stock option pricing [BS73] [Hul00]. The Black-Scholes model is a statistical model for pricing the stock option at a future time based on a few parameters. Most of these statistical models require one assumption: the form of population distribution such as Gaussian distributions for stock markets or binomial distribution for bonds and debts markets [MCG04].

Before we go on to the machine learning solution to the problem, let's take a short review of the econometrics method. Classical econometrics has a large literature for the time series forecast problem. The basic idea is as follows:

To make things simple, assume the time series follow a linear pattern. We then find a linear regression predicting the target values from the time, $y_t = a + b \times y_{t-1} + noise$, where a and b are regression weights. Then work out the weights given some form of noise. This method is rather crude but may work for deterministic linear trends. For nonlinear trends, one needs to use a nonlinear regression such as $y_t = a \times y_{t-1} + b \times y_{t-1}^2$. But sometimes this nonlinear formulation is very hard to find.

The econometrics method usually proposes a statistical model based on a set of parameters. As these models are independent of the past prices, some of these parameters must be estimated based on other current situations. The process of the estimation of these parameters will bring in human analyst bias. For example, in the *Black-Scholes* option pricing model,

the no-arbitrage[1] option cost is calculated as:

$$\begin{aligned} C &= C(s,t,K,\sigma,r) \\ &= e^{-rt}E[(se^{W}-K)^{+}] \\ &= E[(se^{W-rt}-Ke^{-rt})^{+}] \\ &= E[(se^{-\sigma^2 t/2+\sigma\sqrt{t}}-Ke^{-rt})^{+}] \end{aligned} \tag{1.3}$$

s is the security's initial price, t is the exercise time of the option, K is the strike price, r is the interest rate. These four parameters are fixed and can be easily identified. W is a normal random variable with mean $(r-\sigma^2/2)$ and variance $\sigma^2 t$. The only way to decide it is to pick a segment of past price data, then assume the logarithm of this data follows the normal random distribution. The choosing of the price segment and the assumption that the logarithm of the prices follows the normal random distribution is rather subjective.

Another inherent problem of the statistical methods is that sometimes there are price movements that cannot be modeled with one single probability density function (*pdf*). In [Ros03], we see an experiment on the crude oil data. The example shows rather than just one function, these are four distributions that determine the difference between the logarithm of tomorrow's price and the logarithm of today's price. These four *pdf*s must work together with each of them providing a contribution at different time points.

Finally, there are times that we cannot find the best approximation function. This can be caused by the rapid change in the data patterns. For different time windows, there are different patterns thus cannot be described with one stationary model. It has been agreed that no single model can make accurate predictions all the time.

1.5 Non-efficiency of Financial Markets

In a classic statement on the *Efficient Market Hypothesis (EMH)*, Fama [Fam65] [Fam70] defined an efficient financial market as one in which security prices always fully reflects the available information. The EMH "rules out the possibility of trading systems based only on currently available information that have expected profits or returns in excess of equilibrium expected profit or return" [Fam70]. In plain English, an average investor cannot hope to

[1] Arbitrage is the simultaneous buying and selling of securities to take advantage of price discrepancies. Arbitrage opportunities usually surface after a takeover offer.

consistently beat the market, and the best prediction for a future price is the current price. This is called a random walk of the market. If equal performance is obtained on the random walk data as compared to the real data then no significant predictability has been found. The EMH was met with harsh challenges both theoretically and empirically shortly after it was proposed.

De Bondt et. [BWT85] compare the performance of two groups of companies: extreme losers and extreme winners. They study the stock price of two groups of companies from 1933 to 1985. They find that the portfolios of the winners chosen in the previous three years have relatively low returns than the loser's portfolio over the following five years. A more prominent example is the so-called *January Effect* [Wac42] [Tha87]. For small companies, there are superior returns around the end of the year and in early January than other times of the year. This phenomenon was observed many years after Wachtel's initial observation and discussion. Interestingly the January effect has disappeared in the last 15 years. But this does not mean the market is fully efficient as the process took a long time and there may have evolved other new patterns that are yet to find out.

Our own program also finds some interesting patterns. There is a *super five day* pattern in the NASDAQ Composite Index from 1984 to 1990. This means the first three days and the last two days of each month generate a higher return in the index than any other day of the month. The shape of the return curve thus looks like a parabola. After 1990, the pattern became more random and less recognizable.

1.6 Objective

Financial time series consists of multidimensional and complex nonlinear data that result in of pitfalls and difficulties for accurate prediction. We believe that a successful financial time series prediction system should be hybrid and custom made, which means it requires a perfect combination of sophistication in the prediction system design and insight in the input data preprocessing.

Both areas have been extensively researched. Data mining people have developed some very useful techniques of dealing with data, such as missing values [HK01], data multiplication [WG93]. Traders and technical analysts also have done lots of work. They derive enhanced features of interest from the raw time series data, such as the variety of indicators including *Moving Average Convergence Divergence (MACD)*, Price by Volume,

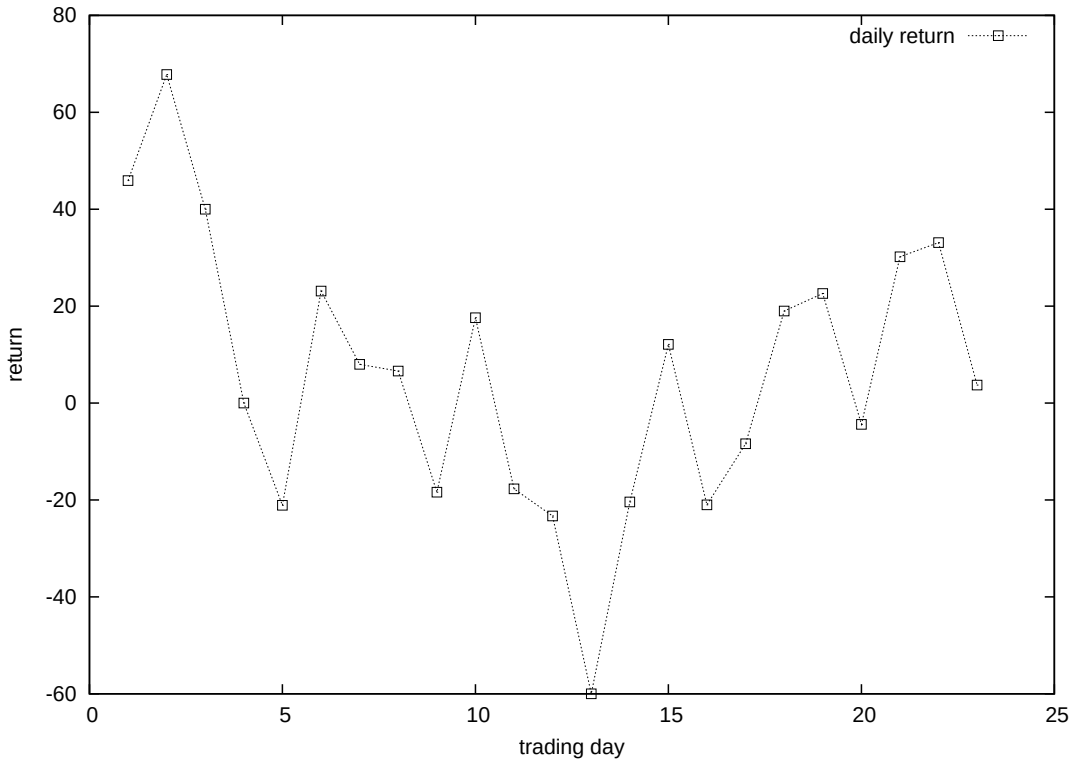

Figure 1.2: A Super Five Day Pattern in the NASDAQ Composite Index from 1984 to 1990. The first three days and the last two days tend to have higher return than the rest of the trading days. However, the pattern diminished in the early 1990s.

Relative Strength Index (RSI) etc. some of which have been confirmed to pertain useful information [STW99].

On the other hand, there have been several straightforward approaches that apply a single algorithm to relatively unprocessed data, such as Back Propagation Network [CWL96], Nearest Neighbor [Mit97], Genetic Algorithm [Deb94], Hidden Markov Models (HMM)[SW97] and so on.

There are three major difficulties about accurate forecast of financial time series. First, the patterns of well-known financial time series are dynamic, i.e. there is no single fixed models that work all the time. Second, it is hard to balance between long-term trend and short-term sideway movements during the training process. In other words, an efficient system must be able to adjusting its sensitivity as time goes by. Third, it is usually hard to determine the usefulness of information. Misleading information must be identified and eliminated.

This thesis aims at solving all these three problems with a Hidden Markov Model(HMM) based forecasting system. The HMM has a Gaussian mixture at each state as the forecast *generator*. Gaussian mixtures have desirable features in that they can model series that does not fall into the single Gaussian distribution. Then the parameters of the HMM are updated each iteration with an *add-drop* Expectation-Maximization (EM) algorithm. At each timing point, the parameters of the Gaussian mixtures, the weight of each Gaussian component in the output, and the transformation matrix are all updated dynamically to cater to the non-stationary financial time series data.

We then develop a novel exponentially weighted EM algorithm that solves the second and third difficulties at one stroke. The revised EM algorithm gives more weight to the most recent data while at the same time adjust the weight according to the change in volatility. The volatility is calculated as the divergence of the S&P Index from the Dow Jones Composite Index. It is included as new information that help make better judgment. Long-term and short-term series balance is then found throughout the combined training-forecast process.

1.7 Contributions of the Thesis

This thesis is an interdisciplinary work that combines computer science and financial engineering. Techniques from the two areas are inter-weaved. The major contributions are:

1. There are few applications of Hidden Markov Model with Gaussian Mixtures in time-dependent sequence data. This is one of the first large scale applications to financial time series.

2. We develop a novel dynamic training scheme. The concept of the dynamically updated *training pool* enables the system to absorb newest information from data.

3. We develop an exponentially weighted EM algorithm. This algorithm realizes that recent data should be given more attention and enables us to recognize trend changes faster than the traditional EM algorithm, especially when training data set is relatively large.

4. Based on the exponentially weighted EM algorithm, we propose a double weighed EM algorithm. The double weighted EM algorithm solves the drawback of the single exponentially weighted EM caused by over-sensitivity.

5. We show the weighted EM algorithms can be written in a form of Exponentially Moving Averages of the model variables of the HMM. This allows us to take the advantage of this existing econometrics technique in generating the EM-based re-estimation equations.

6. Using the techniques from the EM Theorem, we prove that the convergence of the two proposed algorithms is guaranteed.

7. Experimental results show our model consistently outperform the index fund in returns over five 400-day testing periods from 1994 to 2002.

8. The proposed model also consistently outperforms top-listed S&P 500 mutual funds on the Sharpe Ratio.

1.8 Thesis Organization

Chapter 2 of this thesis introduces general form of HMMs. Chapter 3 describes an HMM with Gaussian mixtures for the observation probability density functions and derives the parameter re-estimation formulas based on the classic EM algorithm. Then in chapter 5, some detailed aspects of the HMM are studied, including multiple observation sequence and multivariate Gaussian mixtures as pdfs. We also derive our revised exponentially

weighted EM algorithm and the new parameter updating rules in this chapter. Chapter 6 shows the experimental results and concludes the thesis. Future work that indicate possible improvements of the model are also included in chapter 6.

1.9 Notation

We list the Roman and Greek letters used in this thesis for reference. These symbols are used in the derivations of the re-estimation formulas and proof in chapters 2, 3 and 4.

Symbol	Notation
s_t	The state at time t
π_i	Probability of being at state i at time 0
a_{ij}	Probability of moving from state i to state j
$b_j(o_t)$	Probability of being at state j and observing o_t
w_{jk}	Mixture weight for the k^{th} component at state j
O	The observation sequence
o_t	The t_{th} observation
$\alpha_i(t)$	Forward variable in HMM
$\beta_i(t)$	Backward variable in HMM
$p_t(i,j)$	Prob. of being at state i at time t, and state j at $t+1$
$\gamma_i(t)$	Prob. of being at state i at time t
$\delta_i(t)$	Prob. of the best sequence ending in state j at time t
$\delta_{im}(t)$	Prob. of the best sequence ending at the m^{th} component of the i^{th} state
ψ_t	The node of the incoming arc that lead to the best path
X	Observed data
Y	Hidden part governed by a stochastic process
Z	Complete data set
η	Exponential weight imposed on intermediate variables in EM
ζ	Self-adjustable weight in double weighted EM
ρ	The multiplier in calculating EMA
$\overline{\delta_{im}(T)}$	The EMA of δ_{im} at time T
$\overline{\delta_i(T)}$	The EMA of δ_i at time T
θ_j	Parameters of the j^{th} component in a Gaussian mixture
Θ	Parameters of a Gaussian mixture
L	The likelihood function

Table 1.2: Notation list

Chapter 2

Hidden Markov Models

This chapters describes the general Markov process and Hidden Markov Models (HMMs). We start by introducing the properties of the Markov process, then the characteristics of the HMMs. [RJ86] and [Rab89] provide a thorough introduction to the Markov process, Hidden Markov Models and the main problems involved with Hidden Markov Models. These discussions form the base of our modeling and experiments with financial time series data.

2.1 Markov Models

Markov models are used to train and recognize sequential data, such as speech utterances, temperature variations, biological sequences, and other sequence data. In a Markov model, each observation in the data sequence depends on previous elements in the sequence. Consider a system where there are a set of distinct states, $S = \{1, 2, \ldots, N\}$. At each discrete time slot t, the system takes a move to one of the states according to a set of state transition probabilities P. We denote the state at time t as s_t.

In many cases, the prediction of the next state and its associated observation only depends on the current state, meaning that the state transition probabilities do not depend on the whole history of the past process. This is called a first order Markov process. For example, knowing the number of students admitted this year might be adequate to predict next year's admission. There is no need to look into the admission rate of the previous

years. These are the Markov properties described in [MS99]:

$$P(X_{t+1} = s_k|X_1, \ldots, X_t) = P(X_{t+1} = s_k|X_t) \texttt{ (Limited horizon)} \quad (2.1)$$

$$= P(X_2 = s_k|X_1) \texttt{ (Time invariant)} \quad (2.2)$$

Because of the state transition is independent of time, we can have the following state transition matrix A:

$$a_{ij} = P(X_{t+1} = s_j|X_t = s_i) \quad (2.3)$$

a_{ij} is a probability, hence:

$$a_{ij} \geq 0, \forall i, j \Sigma_{j=1}^{N} a_{ij} = 1. \quad (2.4)$$

Also we need to know the probability to start from a certain state, the initial state distribution:

$$\pi_i = P(X_1 = s_i) \quad (2.5)$$

Here, $\Sigma_{i=1}^{N} \pi_i = 1$.

In a visible Markov model, the states from which the observations are produced and the probabilistic functions are known so we can regard the state sequence as the output.

2.2 A Markov chain of stock market movement

In this section we use a Markov chain to represent stock movement. We present a five state model to model the movement of a certain stock(See Figure 2.1).

In the figure, nodes $1, 2, 3, 4, 5$ represent a possible movement vector {*large rise, small rise, no change, small drop, large drop*}.

From the model in figure we are able to answer some interesting questions about how this stock behaves over time. For example: What is the probability of seeing *"small drop, no change, small rise, no change, no change, large rise"* in six consecutive days?

First we can define the state sequence as $X = \{1, 2, 3, 2, 2, 4\}$. Given the Markov chain, the probability of seeing the above sequence, $P(X|A, \pi)$, can be calculated as:

$$P(X|A, \pi) = P(1, 2, 3, 2, 2, 4|A, \pi) \quad (2.6)$$

$$= P(1)P(2|1)P(3|2)P(2|3)P(2|2)P(4|2) \quad (2.7)$$

$$= \pi * a_{12} * a_{23} * a_{32} * a_{22} * a_{24} \quad (2.8)$$

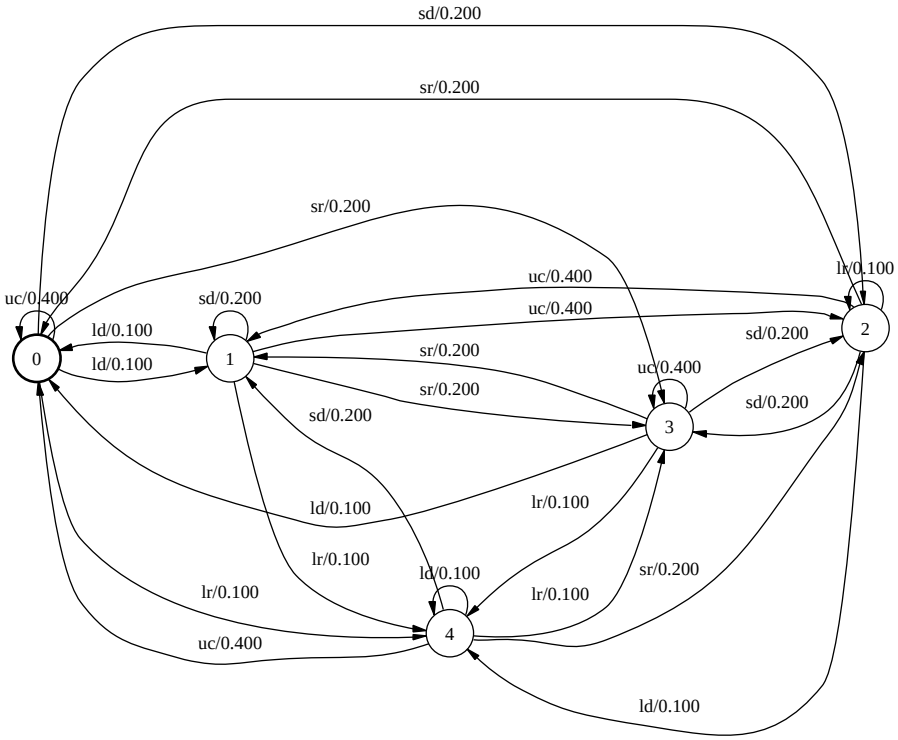

Figure 2.1: Modeling the possible movement directions of a stock as a Markov chain. Each of the nodes in the graph represent a move. As time goes by, the model moves from one state to another, just as the stock price fluctuates everyday.

More generally, the probability of a state sequence $X_1, \ldots, X_T$ can be calculated as the product of the transition probabilities:

$$\begin{aligned} P(X|A,\pi) &= P(X_1)P(X_2|X_1)P(X_3|X_1,X_2)\cdots P(X_T|X_1,\ldots,X_{T-1}) && (2.9)\\ &= P(X_1)P(X_2|X_1)P(X_3|X_2)\cdots P(X_T|X_{T-1}) && (2.10)\\ &= \prod_{t=1}^{T-1} \pi X_t X_{t+1} && (2.11) \end{aligned}$$

2.3 Hidden Markov Models

The Markov model described in the previous section has limited power in many applications. Therefore we extend it to a model with greater representation power, the Hidden Markov Model (HMM). In an HMM, one does not know anything about what generates the observation sequence. The number of states, the transition probabilities, and from which state an observation is generated are all unknown.

Instead of combining each state with a deterministic output(such as *large rise, small drop, etc*), each state of the HMM is associated with a probabilistic function. At time t, an **observation** o_t is generated by a probabilistic function $b_j(o_t)$, which is associated with state j, with the probability:

$$b_j(o_t) = P(o_t|X_t = j) \tag{2.12}$$

Figure 2.2 shows the example of an HMM that models an investment professional's prediction about the market based on a set of investment strategies. Assume the professional has five investment strategies, each of them gives a distinctive prediction about the market movement (large drop, small drop, no change, small rise, and large rise) on a certain day. Also assume he performs market prediction everyday with only one strategy.

Now how does the professional make his prediction? What is the probability of seeing the prediction sequence $\{$*small drop, small rise*$\}$ if the prediction starts by taking strategy 2 on the first day?

To answer the first question, think of HMM as a financial expert. Each of the five strategies is a state in the HMM. The movement set has all the movement symbols and each symbol is associated with one state. Each strategy has a different probability distribution for all the movement symbols. Each state is connected to all the other states with a transition probability distribution, see Figure 2.3 and Table 2.1.

	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5
LR	10%	15%	5%	40%	20%
SR	40%	30%	5%	30%	20%
UC	20%	30%	20%	20%	20%
SD	15%	15%	40%	5%	20%
LD	5%	10%	30%	5%	20%

Figure 2.2: The figure shows the set of strategies that a financial professional can take. The strategies are modeled as states. Each strategy is associated with a probability of generating the possible stock price movement. These probabilities are hidden to the public, only the professional himself knows it.

Unlike the Markov chain of the previous section, the HMM will pick a particular strategy to follow one that was taken based also on the observation probability. The key to the HMM performance is that it can pick the best overall sequence of strategies based on an observation sequence. Introducing density functions at the states of the HMM gives more representation power than fixed strategies associated with the states.

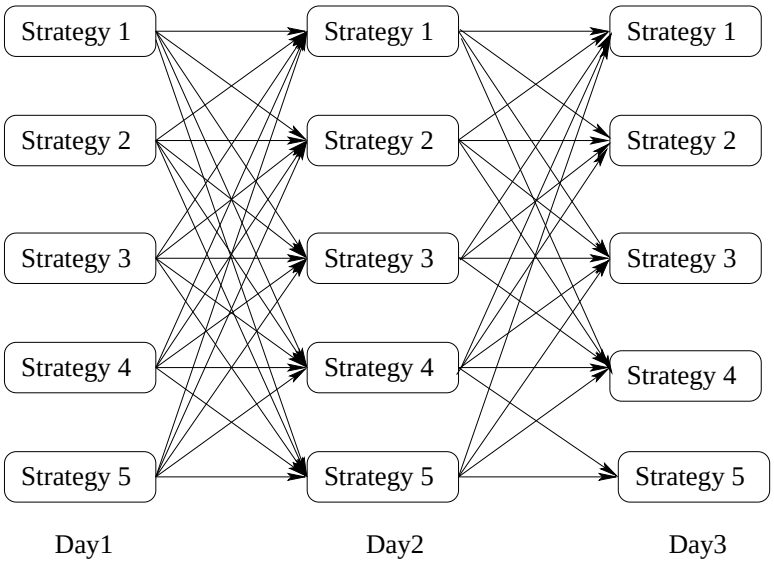

Figure 2.3: The trellis of the HMM. The states are associated with different strategies and are connected to each other. Each state produces an observation picked from the vector [LR, SR, UC, SD, LD].

The steps for generating the prediction sequence:

Probability	Strategy1	Strategy2	Strategy3	Strategy4	Strategy5
Strategy1	10%	10%	20%	30%	30%
Strategy2	10%	20%	30%	20%	20%
Strategy3	20%	30%	10%	20%	20%
Strategy4	30%	20%	20%	15%	15%
Strategy5	30%	20%	20%	15%	15%

Table 2.1: State transition probability matrix of the HMM for stock price forecast. As the market move from one day to another, the professional might change his strategy without letting others know

1. Choose an initial state(i.e. a strategy) $X_1 = i$ according to the initial state distribution Π.

2. Set the time step $t = 1$, since we are at the first day of the sequence.

3. Get the prediction according to the strategy of the current state i, i.e. $b_{ij}(o_t)$. Then we get the prediction for the current day. For example, the probability of seeing a *large rise* prediction on the first day if we take strategy 2 is 15

4. Transit to a new state X_{t+1}, according to the state transition probability distribution.

5. Set t=t+1, go to step 3, if $t \leq T$, otherwise terminate.

To answer the second question, sum up all the probabilities the movement sequence may happen, through different state sequences. We know the prediction starts with strategy 2, which means the initial state is state 2. Then there are five possible states for the second day. So the probability of seeing the movement sequence {*small drop, small rise*} is:

$$0.15 + 0.1 \times 0.4 + 0.2 \times 0.3 + 0.3 \times 0.05 + 0.2 \times 0.3 + 0.2 \times 0.2 = 0.365$$

2.4 Why HMM?

From the previous example, we can see that there is a good match of sequential data analysis and the HMM. Especially in the case of financial time series analysis and prediction, where the data can be thought of being generated by some underlying stochastic process that is not known to the public. Just like in the above example, if the professional's strategies and

the number of his strategies are unknown, we can take the imaginary professional as the underlying power that drives the financial market up and down. As shown in the example we can see the HMM has more representative power than the Markov chain. The transition probabilities as well as the observation generation probability density function are both adjustable.

In fact, what drives the financial market and what pattern financial time series follows have long been the interest that attracts economists, mathematicians and most recently computer scientists. See [ADB98].

Besides the match between the properties of the HMM and the time series data, the Expectation Maximization(EM) algorithm provides an efficient class of training methods [BP66]. Given plenty of data that are generated by some hidden power, we can create an HMM architecture and the EM algorithm allows us to find out the best model parameters that account for the observed training data.

The general HMM approach framework is an unsupervised learning technique which allows new patterns to be found. It can handle variable lengths of input sequences, without having to impose a "template" to learn. We show a revised EM algorithm in later chapters that gives more weight to recent training data yet also trying to balance the conflicting requirements of sensitivity and accuracy.

HMMs have been applied to many areas. They are widely used in speech recognition [Rab89][Cha93], bioinformatics [PBM94], and different time series analysis, such as weather data, semiconductor malfunction [GS00][HG94],etc. Weigend and Shi [SW97] have previously applied HMMs to financial time series analysis. We compare our approach to theirs in Chapter 5.

2.5 General form of an HMM

An HMM is composed of a five-tuple: (S, K, Π, A, B).

1. $S = \{1, \ldots, N\}$ is the set of states. The state at time t is denoted s_t.

2. $K = \{k_1, \ldots, k_M\}$ is the output alphabet. In a discrete observation density case, M is the number of observation choices. In the above example, M equals the number of possible movements.

3. Initial state distribution $\Pi = \{\pi_i\}$, $i \in S$. π_i is defined as

$$\pi_i = P(s_1 = i) \tag{2.13}$$

4. State transition probability distribution $A = \{a_{ij}\}, i, j \in S$.

$$a_{ij} = P(s_{t+1}|s_t), 1 \leq i, j \leq N \tag{2.14}$$

5. Observation symbol probability distribution $B = b_j(o_t)$. The probabilistic function for each state j is:

$$b_j(o_t) = P(o_t|s_t = j) \tag{2.15}$$

After modeling a problem as an HMM, and assuming that some set of data was generated by the HMM, we are able to calculate the probabilities of the observation sequence and the probable underlying state sequences. Also we can train the model parameters based on the observed data and get a more accurate model. Then use the trained model to predict unseen data.

2.6 Discrete and continuous observation probability distribution

2.6.1 Discrete observation probability distribution

There are fixed number of possible observations in the example in section 2.3. The function at each state is called a discrete probability density function. In financial time series analysis, we sometimes need to partition probability density function of observations into discrete set of symbols $v_1, v_2, \ldots, v_M$. For example, we might want to model the direction of the price movement as a five-element vector {*large drop, small drop,no change, small rise, large rise*}. This partitioning process is called *vector quantization*. [Fur01] provides a survey on some of the most commonly used vector quantization algorithms.

After the vector quantization, a *codebook* is created and the following analysis and prediction are with the domain of the *codebook*. In the previous example, the observation symbol probability density function will be in the form:

$$b_j(o_t) = P(o_t = k|s_t = j), 1 \leq k \leq M \tag{2.16}$$

Using discrete probability distribution functions greatly simplifies modeling of the problem. However they have limited representation power. To solve this problem, the continuous probability distribution functions are used during the training phase, enabling us to get as accurate a model as possible. Then in the prediction step, the codebook is used to estimate the probability of each possible symbol and produce the most probable one.

2.6.2 Continuous observation probability distribution

In a continuous observation probability HMM, the function $b_j(o_t)$ is in the form of a continuous probability density function(pdf) or a mixture of continuous *pdf*s:

$$b_j(o_t) = \sum_{k=1}^{M} w_{jk} b_{jk}(o_t), j = 1, \ldots, N \tag{2.17}$$

M is the number of the mixtures and w is the weight for each mixture. The mixture weights have the following constraints:

$$\sum_{k=1}^{M} w_{jk} = 1, j = 1, \ldots, N; w_{jk} \geq 0, j = 1, \ldots, N; k = 1, \ldots, M \tag{2.18}$$

Each $b_{jk}(o_t)$is a D-dimensional log-concave or elliptically symmetric density with mean vector μ_{jk} and covariance matrix Σ_{jk}:

$$b_j k(o_t) = \mathcal{N}(o_t, \mu_{jk}, \Sigma_{jk}) \tag{2.19}$$

Then the task for training is to learn the parameters of these *pdf*s. For details in mixtures of Gaussian distributions, please refer to [Mit97] and [MS99].

This D-dimensional multivariate time series training requires that we pick the most correlated time series as input. For example, in order to predict the price of IBM, it will make more sense to include the prices of Microsoft, Sun and other high tech stocks rather than include Exxon or McDonald's, whose behavior have less correlation with that of IBM. An automatic method for dimension reduction is *Principal Component Analysis(PCA)*. The dimension of the multivariate time series cannot be too large as it increases computation complexity by quadratic factor, we will need to find out the most correlated factors to our prediction. We will describe more about this subject in Chapter 3.

Another problem is the selection of the pdf for each mixture. The most used D-dimensional log-concave or elliptically symmetric density is the Gaussian density, or Normal Distribution.

The single Gaussian density function has the form:

$$b_{jk}(o_t) = \mathcal{N}(o_t, \mu_{jk}, \Sigma_{jk}) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.20}$$

where o_t is a real value.

The multivariate Gaussian density is in the form:

$$b_{jk}(o_t) = \mathcal{N}(o_t, \mu_{jk}, \Sigma_{jk}) = \frac{1}{(2\pi)^{D/2}|\Sigma_{jk}|^{1/2}} exp\left(-\frac{1}{2}(o_t - \mu_{jk})^T \Sigma_{jk}^{-1}(o_t - \mu_{jk})\right) \tag{2.21}$$

where o_t is a vector.

2.7 Three Fundamental Problems for HMMs

There are three fundamental problems about the HMM that are of particular interest to us:

1. Given a model $\mu = (A, B, \Pi)$, and an observation sequence $O = (o_1, \ldots, o_T)$, how do we efficiently compute the probability of the observation sequence given the model? i.e., what is $P(O|\mu)$?

2. Given a model μ and the observation sequence O, what is the underlying state sequence$(1, \ldots, N)$ that best "explains" the observations?

3. Given an observation sequence O, and a space of possible models, how do we adjust the parameters so as to find the model μ that maximizes $P(O|\mu)$?

The first problem can be used to determine which of the trained models is most likely when the training observation sequence is given. In the financial time series case, given the past history of the price or index movements, which model best accounts for the movement history. The second problem is about finding out the hidden path. In our previous example in section 2.3, what is the strategy sequence the financial professional applied on each trading day? The third problem is of the most significant interest because it deals with the training of the model. In the real-life financial times series analysis case, it is about training the model with the past historical data. Only after training can we use the model to perform further predictions.

2.7.1 Problem 1: finding the probability of an observation

Given an observation sequence $O = (o_1, \ldots, o_T)$ and an HMM $\mu = (A, B, \Pi)$, we want to find out the probability of the sequence $P(O|\mu)$. This process is also known as *decoding.* Since the observations are independent of each other the time t, the probability of a state sequence $S = (s_1, \ldots, s_T)$ generating the observation sequence can be calculated as:

$$\begin{aligned} P(O|S,\mu) &= \prod_{t=1}^{T} P(o_t|s_t, s_{t+1}, \mu) && (2.22) \\ &= b_{s_1}(o_1) \cdots b_{s_1 s_2}(o_2) \cdots b_{s_{T-1} s_T}(o_T) && (2.23) \end{aligned}$$

and the state transition probability,

$$P(S|\mu) = \pi_{s_1} \cdot a_{s_1 s_2} \cdot a_{s_2 s_3} \cdots a_{s_{T-1} s_T}. \tag{2.24}$$

The joint probability of O and S:

$$P(O, S|\mu) = P(O|S, \mu)P(S|\mu) \tag{2.25}$$

Therefore,

$$\begin{aligned} P(O|\mu) &= \sum_{S} P(O|S,\mu)P(S|\mu) && (2.26) \\ &= \sum_{s_1 \cdots s_{T+1}} \pi_{s_1} \prod_{t=1}^{T} a_{s_t s_{t+1}} b_{s_t s_{t+1} o_t} && (2.27) \end{aligned}$$

The computation is quite straightforward by summing the observation probabilities for each of the possible state sequence. A major problem with this approach is enumerating each state sequence is inefficiency. As the length of T of the sequence grows, the computation grows exponentially. It requires $(2T - 1) \cdot N^{T+1}$ multiplications and $N^T - 1$ additions.

To solve this problem, an efficient algorithm–the *forward backward algorithm* was developed that cuts the computational complexity to linear in T.

The forward procedure

The forward variable $\alpha_i(t)$ is defined as:

$$\alpha_i(t) = P(o_1 o_2 \cdots o_{t-1}, s_t = i|\mu) \tag{2.28}$$

$\alpha(t)$ stores the total probability of ending up in state s_i at time t, given the observation sequence $o_1 \cdots o_{t-1}$. It is calculated by summing probabilities for all incoming arcs at a trellis node. The forward variable at each time t can be calculated inductively, see Figure 2.4.

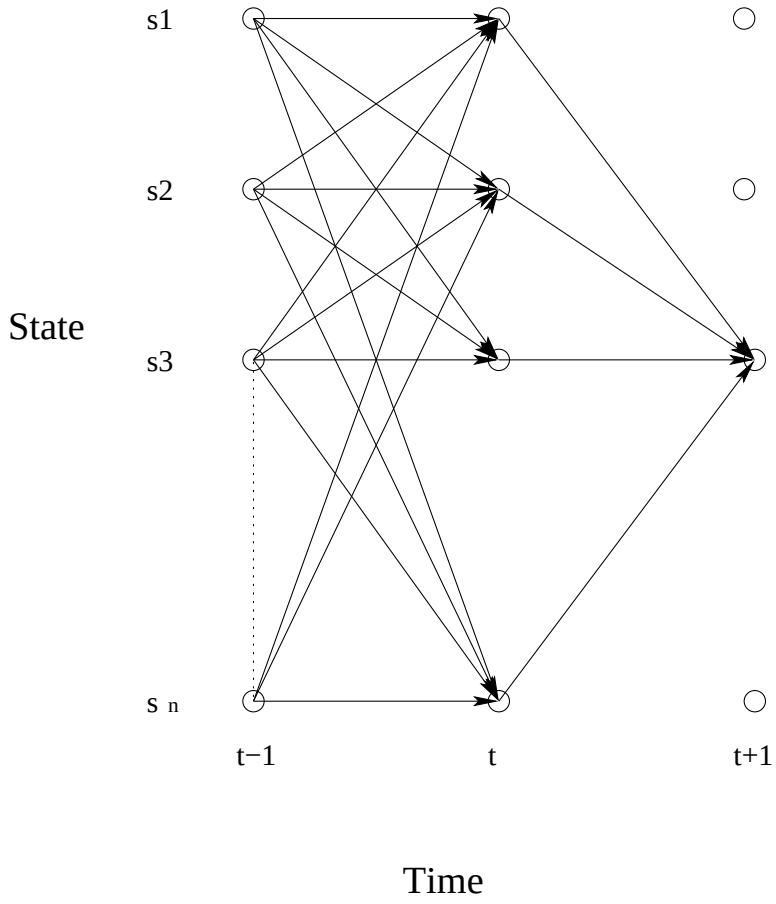

Figure 2.4: The forward procedure

1. Initialization

$$\alpha_i(1) = \pi_i, 1 \leq i \leq N \tag{2.29}$$

2. Induction

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t}, 1 \leq t \leq T, 1 \leq j \leq N \tag{2.30}$$

3. Update time Set $t = t + 1$; Return to step 2 if $t < T$;
   Otherwise, terminate algorithm

4. Termination

$$P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T) \tag{2.31}$$

The forward algorithm requires $N(N+1)(T-1)+N$ multiplications and $N(N-1)(T-1)$ additions. The complexity is $O(TN^2)$.

The backward procedure

The induction computation for the forward procedure can also be performed in the reverse order. The *backward procedure* calculates the probability of the partial observation sequence from $t+1$ to the end, given the model μ and state s_i at time t. The backward variable $\beta_i(t)$ is defined as:

$$\beta_i(t) = P(o_{t+1}o_{t+2}\cdots o_T|s_t = i, \mu) \tag{2.32}$$

The backward algorithm works from right to left on through the same trellis:

1. Initialization

$$\beta_i(T) = 1, 1 \leq i \leq N \tag{2.33}$$

2. Induction

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij}b_{ijo_t}\beta_j(t+1), 1 \leq t \leq T, 1 \leq i \leq N \tag{2.34}$$

3. Update time

   Set $t = t - 1$; Return to step 2 if $t > 0$; Otherwise terminate algorithm.

4. Total

$$P(O|\mu) = \sum_{i=1}^{N} \pi_i\beta_i(1) \tag{2.35}$$

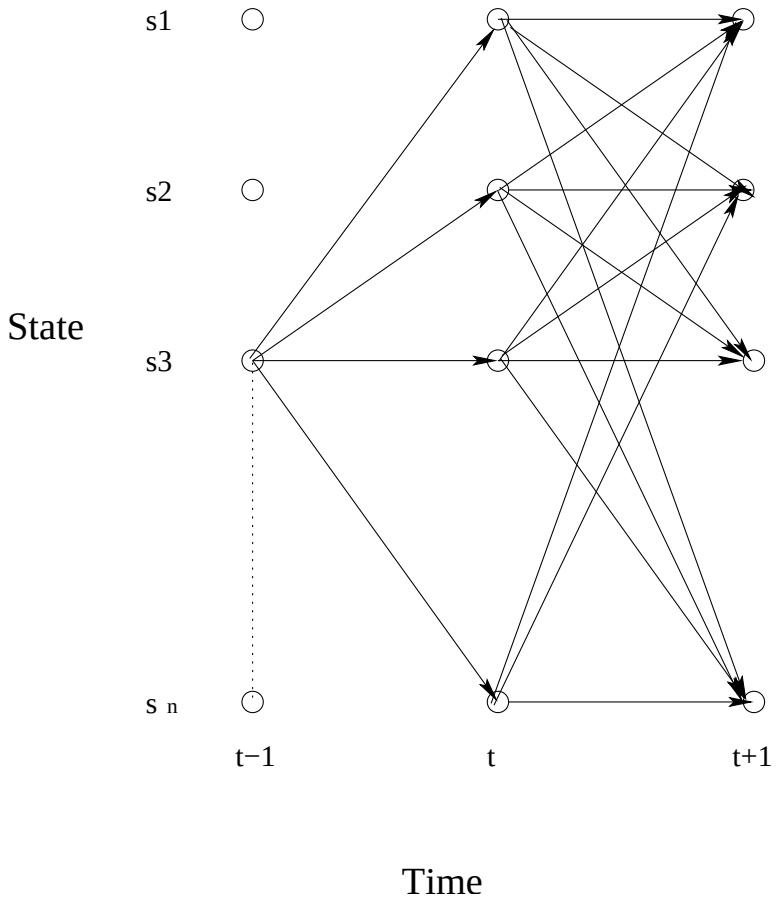

Figure 2.5: The backward procedure

2.7.2 Problem 2: Finding the best state sequence

The second problem is to find the best state sequence given a model and the observation sequence. This is one a problem often encountered in parsing speech and in pattern recognition. There are several possible ways to solve this problem. The difficulty is that there may be several possible optimal criteria. One of them is to choose the states that are individually most likely at each time t. For each time $t, 1 \leq t \leq T+1$, we find the following probability variable:

$$\begin{aligned} \gamma_i(t) &= P(s_t = i|O, \mu) && (2.36) \\ &= \frac{P(s_t = i, O|\mu)}{P(O|\mu)} && (2.37) \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} && (2.38) \end{aligned}$$

The individually most likely state sequence S' can be found as:

$$S' = \underset{1 \leq i \leq N}{\operatorname{argmax}} \gamma_i(t), 1 \leq t \leq T+1, 1 \leq i \leq N \quad (2.39)$$

This quantity maximizes the expected number of correct states. However the approach may generate an unlikely state sequence. This is because it doesn't take the state transition probabilities into consideration. For example, if at some point we have zero transition probability $a_{ij} = 0$, the optimal state sequence found may be invalid.

Therefore, a more efficient algorithm, the *Viterbi* algorithm, based on dynamic programming, is used to find the best state sequence.

The Viterbi algorithm

The Viterbi algorithm is designed to find the most likely state sequence, $S' = (s_1, s_2, \ldots, s_T)$ given the observation sequence $O = (o_1, o_2, \ldots, o_T)$:

$$\underset{S'}{\operatorname{argmax}} P(S'|O, \mu)$$

It is sufficient to maximize for a fixed observation sequence O:

$$\underset{S'}{\operatorname{argmax}} P(S', O|\mu)$$

The following variable:

$$\delta_j(t) = \max_{s_1 \cdots s_{t-1}} P(s_1 \cdots s_{t-1}, o_1 \cdots o_{t-1}, s_t = j|\mu) \quad (2.40)$$

This variable stores the probability of observing $o_1 o_2 \cdots o_t$ using the most likely path that ends in state i at time t, given the model μ. The corresponding variable $\psi_j(t)$ stores the node of the incoming arc that leads to this most probable path. The calculation is done by induction, similar to the *forward-backward algorithm*, except that the forward-backward algorithm uses summing over previous states while the Viterbi algorithm uses maximization. The complete Viterbi algorithm is as follows:

1. Initialization

$$\delta_i(1) = \pi_i b_i(o_1), 1 \leq i \leq N \tag{2.41}$$
$$\psi_i(1) = 0, 1 \leq i \leq N \tag{2.42}$$

2. Induction

$$\delta_j(t) = b_{ijo_t} \max_{1 \leq i \leq N} \delta_i(t-1) a_{ij} \tag{2.43}$$
$$\psi_j(t) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_i(t-1) a_{ij}] \tag{2.44}$$

3. Update time

$$t = t + 1 \tag{2.45}$$

   Return to step 2 if $t \leq T$ else terminate the algorithm

4. Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_i(T)] \tag{2.46}$$
$$s_T^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_i(T)] \tag{2.47}$$

5. Path readout

$$s_t^* = \psi_{t+1}(s_{t+1}^*) \tag{2.48}$$

When there are multiple paths generated by the Viterbi algorithm, we can pick up one randomly or choose the best n paths. Fig 2.6 shows how the Viterbi algorithm finds a single best path.

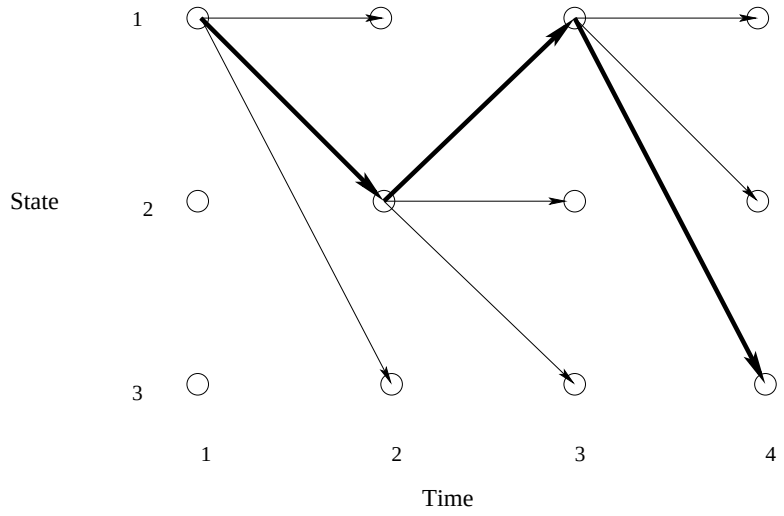

Figure 2.6: Finding best path with the Viterbi algorithm

2.7.3 Problem 3: Parameter estimation

The last and most difficult problem about HMMs is that of the parameter estimation. Given an observation sequence, we want to find the model parameters $\mu = (A, B, \pi)$ that best explains the observation sequence. The problem can be reformulated as find the parameters that maximize the following probability:

$$\underset{\mu}{\operatorname{argmax}} P(O|\mu)$$

There is no known analytic method to choose μ to maximize $P(O|\mu)$ but we can use a local maximization algorithm to find the highest probability. This algorithm is called the *Baum-Welch*. This is a special case of the Expectation Maximization method. It works iteratively to improve the likelihood of $P(O|\mu)$. This iterative process is called the *training* of the model. The *Baum-Welch algorithm* is numerically stable with the likelihood non-decreasing of each iteration. It has linear convergence to a local optima.

To work out the optimal model $\mu = (A, B, \pi)$ iteratively, we will need to define a few intermediate variables. Define $pt(i, j), 1 \leq t \leq T, 1 \leq i, j \leq N$ as follows:

$$
\begin{aligned}
pt(ij) &= P(s_t = i, s_{t+1} = j|O, \mu) && (2.49)\\
&= \frac{P(s_t = i, s_{t+1} = j, O|\mu)}{P(O|\mu)} && (2.50)\\
&= \frac{\alpha_i(t) a_{ij} b_{ijo_t} \beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t)\beta_m(t)} && (2.51)\\
&= \frac{\alpha_i(t) a_{ij} b_{ijo_t} \beta_j(t+1)}{\sum_{m=1}^{N}\sum_{n=1}^{N} \alpha_m(t) a_{mn} b_{mno_t} \beta_n(t+1)} && (2.52)
\end{aligned}
$$

This is the probability of being at state i at time t, and at state j at time $t+1$, given the model μ and the observation O.

Then define $\gamma_i(t)$. This is the probability of being at state i at time t, given the observation O and the model μ:

$$
\begin{aligned}
\gamma_i(t) &= P(s_t = i|O, \mu) && (2.53)\\
&= \sum_{j=1}^{N} P(s_t = i, s_{t+1} = j|O, \mu) && (2.54)\\
&= \sum_{j=1}^{N} p_t(i, j) && (2.55)
\end{aligned}
$$

The above equation holds because $\gamma_i(t)$ is the expected number of transition from state i and $p_t(i, j)$ is the expected number of transitions from state i to j.

Given the above definitions we begin with an initial model μ and run the training data O through the current model to estimate the expectations of each model parameter. Then we can change the model to maximize the values of the paths that are used. By repeating this process we hope to converge on the optimal values for the model parameters.

The re-estimation formulas of the model are:

$$\begin{aligned} \pi'_i &= \text{probability of being at state } i \text{ at time } t = 1 \\ &= \gamma_i(1) \end{aligned} \tag{2.56}$$

$$\begin{aligned} a'_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i} \\ &= \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)} \end{aligned} \tag{2.57}$$

$$\begin{aligned} b'_{ijk} &= \frac{\text{expected number of transitions from } i \text{ to } j \text{ with } k \text{ observed}}{\text{expected number of transitions from } i \text{ to } j} \\ &= \frac{\sum_{t:o_t=k, 1\leq t\leq T} p_t(i,j)}{\sum_{t=1}^{T} p_t(i,j)} \end{aligned} \tag{2.58}$$

In the following calculations, we will replace b_{ijo_t} with b_{jo_t}, i.e. each observation is generated by the probability density function associated with one state at one time step. This is also called a *first order HMM*.

2.8 Chapter Summary

In this chapter, the general form of the HMM is introduced, both theoretically and with examples. The conditions that HMMs must observe, i.e. the Markov properties are also described. Then we studied the three basic problems involved with any HMM. The following model design and experiments are all extensions based on the basic model and the techniques to solve the three problems. In fact, in the prediction task, both Viterbi algorithm and the Baum-Welch algorithm are used in combination to make the system work.

Chapter 3

HMM for Financial Time Series Analysis

In this chapter we construct an HMM based on the discussions in chapter 2. We first give a brief survey of Weigend and Shi's previous work on the application of Hidden Markov Experts to financial time series prediction [SW97], analyze problems with their approach and then propose a new model.

We use an HMM with emission density functions modeled as mixtures of Gaussians in combination with the usual transition probabilities. We introduce Gaussian mixtures which we have as the continuous observation density function associated with each state. The re-estimation formulas are then derived for the Gaussian Mixtures. We apply an *add-drop* approach to the training data.

3.1 Previous Work

In [SW97], Weigend and Shi propose a Hidden Markov Expert Model to forecast the change of the S&P 500 Index at half-hour time step. The "experts" at each state of the Hidden Markov Model applied for financial time series prediction are three neural networks, with characteristics and predicting bias distinct from each other. One of the experts is good at forecasting low volatility data, the second expert is best for high volatility regions, and the

third expert is responsible for the "outliers". [1]

The training of the Hidden Markov Experts mainly involves the re-estimation of the transition probability matrix A. Just like we did in equations (2.58) and (2.59). But there are no re-estimations for each of the three "experts". This means that the experts are fixed and we must trust them during the whole prediction process because there is no way of improving their accuracy of prediction. This is problematic because even if we can divide a long time series into several segments with different volatilities, segments with similar volatilities may not necessarily reflect similar pattern. In the later sections of this chapter we will introduce the Gaussian mixtures where all parameters of each mixture component, the counterpart of the "expert", are adjustable.

Then in the prediction phase, a *training pool* is created. This pool keeps updating by adding the last available data and eliminating the first data in the pool. The data in each pool is then trained together to generate the prediction for the next coming day. In this approach, data at each time step are of equal importance. This can cause big problems during highly volatile period or at the turning point of long term trends. Too much weight is given to the early samples. We will solve this problem with an adjustable weighted EM algorithm in chapter 4.

Another problem with Shi and Weigend's model is the lack of additional information to help prediction decisions. Their model takes the destination time series as the only input time series. By providing more available information, such as similar time series or indicator series derived from the original series, we can make the model more reliable. In section 3.3 we introduce the multivariate Gaussian mixtures that take a vector composed of the Dow Jones Composite Index and the S&P 500 Index as input. Without having to worry about the exactly mathematical relationship between these two series, we let the HMM combine them together to make predictions for the S&P 500 Index. Then in chapter 4, the Dow Jones Index is used as an error controller. When the system prediction of the S&P 500 doesn't coincide with the Dow, we will give less confidence in the HMM. This is because the Dow is a less volatile index than the S&P 500. By comparing the movement prediction of the S&P 500 with the Dow, we can balance between long term trend and short[2] term

[1] Outliers are a few observations that are not well fitted by the "best" available model. Refer to appendix I for more explanation.

[2] To short means to sell something that you don't have. To do this, you have to borrow the asset that you want to short.

whipsaws. [3]

3.2 Baum-Welch Re-estimation for Gaussian mixtures

The re-estimation formulas in the previous chapter have mainly focused on the case where the symbol emission probabilities are discrete, in other words, the emitted symbols are discrete symbols from a finite alphabet list. Although as mentioned before, we can use the *vector quantization* technique to map each continuous value into a discrete value via the *codebook*, there might be serious degradation after the quantization. Therefore it is advantageous to use HMMs with continuous observation densities.

The most generally used form of the continuous probability density function(pdf) is the Gaussian mixture in the form of:

$$b_{im}(o_t) = \sum_{m=1}^{M} w_{im}\mathcal{N}(o_t, \mu_{im}, \Sigma_{im}) \tag{3.1}$$

o_t is the observed vector, w_{im} is the weight of the mth Gaussian mixture at state i. The mixture weights w must satisfy the following constraints:

$$\begin{gathered} \sum_{m=1}^{M} w_{im} = 1, 1 \leq i \leq N \\ w_{im} \geq 0, 1 \leq i \leq N, 1 \leq m \leq M \end{gathered} \tag{3.2}$$

$\mathcal{N}(O, \mu_{im}, \Sigma_{im})$ is a multivariate Gaussian distribution with mean vector μ' and covariance matrix Σ':

$$\mathcal{N}(O, \mu_{im}, \Sigma_{im}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma'|}} exp\left(-\frac{1}{2}(o-\mu')\Sigma'^{-1}(o-\mu')\right) \tag{3.3}$$

The learning of the parameters of the HMMs thus becomes learning of the means and variances of each Gaussian mixture component. The intermediate variable δ is redefined as:

$$\delta_i(t) = \sum_{m=1}^{M} \delta_{im}(t) = \sum_{m=1}^{M} \frac{1}{P}\alpha_j(t-1)a_{ji}w_{im}b_{im}(o_t)\beta_i(t) \tag{3.4}$$

As previously defined M is the total number of Gaussian mixture components at state i and N is the total number of states. The difference with the previous example is that

[3] A whipsaw occurs when a buy or sell signal is reversed in a short time. Volatile markets and sensitive indicators can cause whipsaws

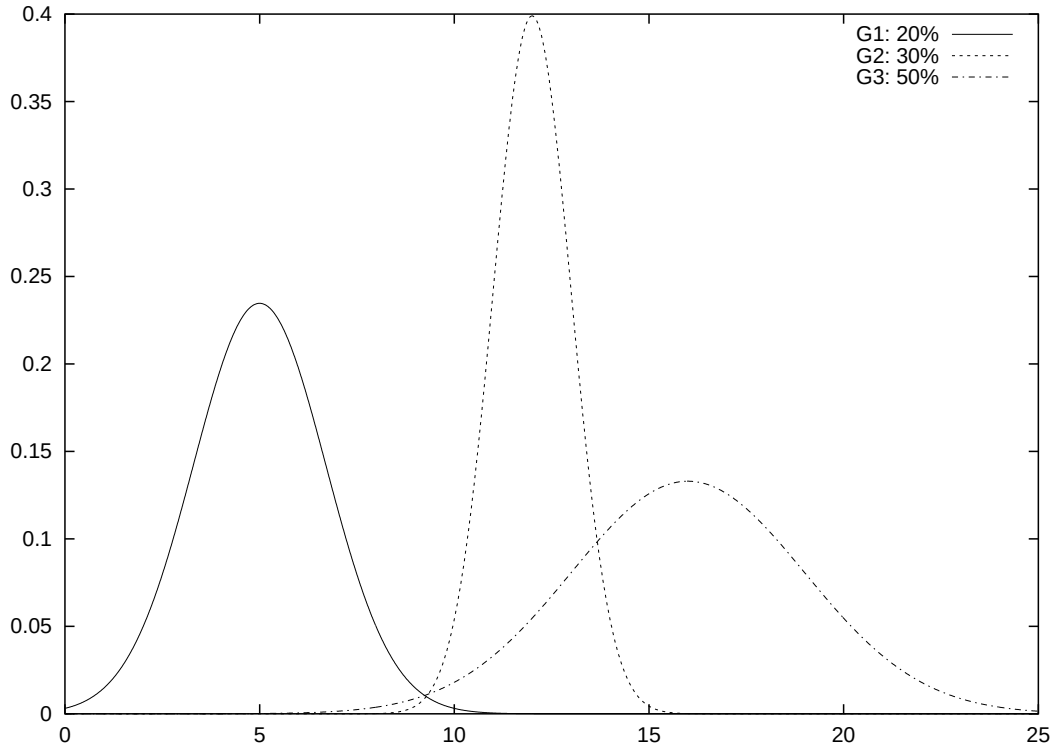

Figure 3.1: Three single Gaussian density functions. Each of them will contributing a percentage to the combined output in the mixture.

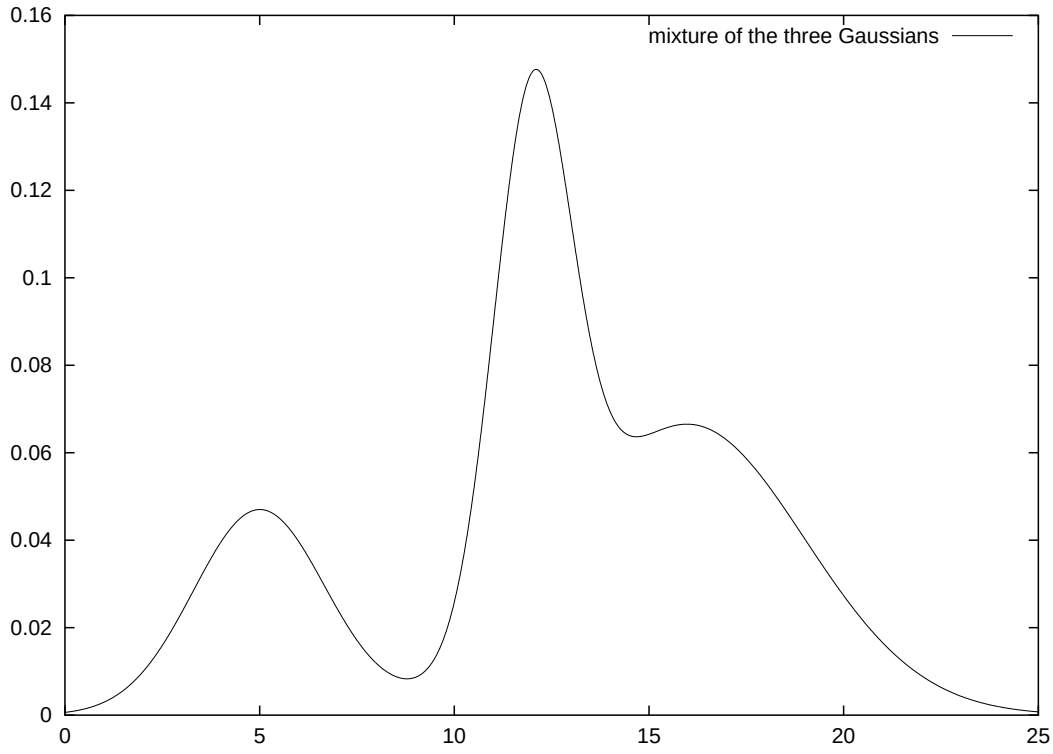

Figure 3.2: A Gaussian mixture derived from the three Gaussian densities above. For each small area δ close to each point x on the x axis, there is a probability of 20% that the random variable x is governed by the first Gaussian density function, 30% of probability that the distribution of x is governed by the second Gaussian, and 50% of probability by the third Gaussian.

rather than associating each state with one probability density function, each observation is generated by a set of Gaussian mixtures, with each mixture component contributing to the overall result by some weight.

Then we also need a variable that denotes the probability of being at state j at time t with the kth mixture component that accounts for the current observation o_t:

$$\gamma_{j,k}(t) = \frac{\alpha_j(t)\beta_j(t)}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)} \cdot \frac{w_{jk}\mathcal{N}(o_t, \mu_{jk}, \Sigma_{jk})}{\sum_{k=1}^{M} w_{jk}\mathcal{N}(o_t, \mu_{jk}, \Sigma_{jk})} \tag{3.5}$$

With the new intermediate variable δ defined, we are able to derive the re-estimation formulas below:

$$\mu'_{im} = \frac{\sum_{t=1}^{T}\delta_{im}(t)o_t}{\sum_{t=1}^{T}\delta_{im}(t)} \tag{3.6}$$

$$\sigma'_{im} = \frac{\sum_{t=1}^{T}\delta_{im}(t)(o_t - \mu'_{im})(o_t - \mu'_{im})'}{\sum_{t=1}^{T}\delta_{im}(t)} \tag{3.7}$$

$$w_{im} = \frac{\sum_{t=1}^{T}\delta_{im}(t)}{\sum_{t=1}^{T}\delta_i(t)} \tag{3.8}$$

$$a'_{ij} = \frac{\sum_{t=1}^{T}\alpha_i(t)a_{ij}b_j(o_{t+1})\beta_j(t+1)}{\sum_{t=1}^{T}\alpha_i(t)\beta_i(t)} \tag{3.9}$$

3.3 Multivariate Gaussian Mixture as Observation Density Function

Sometimes we have observation vectors that have more than one element. In other words the observed data are generated by several distinct underlying causes that have some unknown relationship between them. Each cause contributes independently to the generation process but we can only see the observation sequence as a final mixture, without knowing how much and in what way each cause contributes to the resulted observation sequence.

This phenomenon is of great interest in financial time series analysis. Many time series could be correlated with other time series. For example in stock market, the *volume* by itself is of little interest, but it will contain plenty of information about the trend of the market when combined with *close price*. Canada's stock market is affected by the US economy circle because Canada is the largest trading partner of US. As such the factors that affect

US stock market also have an impact on the Canadian market. Researchers have found that the best Canadian year is the third year of the US president election cycle.

Some day traders have used a technique called "paired-trading". They trade two stocks with similar characteristics at the same time, aiming at profiting from arbitrage opportunities or averse possible risks. If they buy the stocks of *Wal-Mart*, they will short the stocks of *K-mart*. These examples show that there is a need for including relevant information beyond the time series that we are studying.

3.3.1 Multivariate Gaussian distributions

The multivariate m-dimensional *Gaussian* has the same parameters as the single normal distribution, except that the mean is in the form of a *vector* $\vec{\mu}$ and the covariance is in the form of a $m \times m$ covariance matrix Σ_j. The probability density function for a Gaussian is given by:

$$n_j(\vec{x}, \vec{m_j}, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^m |\Sigma_j|}} exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu_j})^T \Sigma_j^{-1} (\vec{x} - \vec{\mu_j})\right) \tag{3.10}$$

At each state, we assume the observation is generated by k Gaussians, so the overall output is:

$$\sum_{j=1}^{k} w_j \mathcal{N}(\vec{x}, \vec{\mu_j}, \Sigma_j) \tag{3.11}$$

The initial value of the weights w_j for each mixture component should sum up to 1. The training process then becomes a process of find the parameters that maximizes the probability of this output observation.

3.3.2 EM for multivariate Gaussian mixtures

We now develop the EM algorithm for re-estimating the parameters of the multivariate Gaussian mixture. Let $\theta_j = (\vec{\mu}_j, \Sigma_j, \pi_j)$ be the jth mixture component, then the parameters of the model are defined as $\Theta = (\theta_1, \ldots, \theta_k)^T$. Also define a hidden variable $E[z_{ij}]$ as the probability of generating the ith observation with the jth mixture. The EM algorithm that finds the parameters that maximize the probability of the observation sequence O consists of the following two steps (we use a mixture of two θ_1, θ_2 to explain the algorithm):

Estimation step:
Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, given that we know the current values of $\Theta = (\theta_1, \theta_2)$.

Maximization step:
Calculate a new maximum likelihood value of $\Theta' = (\theta_1', \theta_2')$, assuming the value taken on by each hidden variable z_{ij} is the expected value in the first step.

In the HMM context, the probability of the hidden variable z_{ij} is the intermediate variable $\gamma_{ij}(t)$ we talked about in 3.2. Combining the EM algorithm for multivariate Gaussian mixtures and the EM algorithm for HMMs, we develop the following formulas for each of the parameters of the HMM with multivariate Gaussian mixtures as observation density function:

$$\vec{\mu_{im}}' = \frac{\sum_{t=1}^{T} \delta im(t)\vec{o_t}}{\sum_{t=1}^{T} \delta_{im}(t)} \tag{3.12}$$

$$\Sigma_j' = \frac{\sum_{t=1}^{T} \delta_{im}(t)(\vec{o_i} - \vec{\mu_j'})(\vec{o_i} - \vec{\mu_j'})^T}{\sum_{t=1}^{T} \delta_{im}(t)} \tag{3.13}$$

$$w_{im} = \frac{\sum_{t=1}^{T} \delta_{im}(t)}{\sum_{t=1}^{T} \delta_i(t)} \tag{3.14}$$

$$a_{ij}' = \frac{\sum_{t=1}^{T} \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\sum_{t=1}^{T} \alpha_i(t)\beta_i(t)} \tag{3.15}$$

3.4 Multiple Observation Sequences

In the previous discussions, the EM algorithm is applied to a single observation sequence O. However, the real-life financial time series data can have a very long range. Modeling them as one single observation sequence might give low recognition rate when testing data from some specific period of time. This is because the pattern or trend at some point of time may be quite different to that of the overall pattern. As stated by Graham [Gra86]: "There is no generally known method of chart reading that has been continuously successful for a long period of time. If it were known, it would be speedily adopted by numberless traders. This very following would bring its usefulness to an end."

Rabiner offers a way to solve the similar kind of problem in speech recognition by splitting

the original observation sequence O into a set of k short sequences [Rab89]:

$$O = [O^{(1)}, O^{(2)}, \ldots, O^{(k)}] \tag{3.16}$$

where $O^{(k)} = [O_1^{(k)}, O_2^{(k)}, \ldots, O_{T_k}^{(k)}]$ is the kth observation sequence and T_k is the kth observation sequence length. Assume that each observation sequence is independent of each other, the probability function $P(O|\mu)$ to maximize now becomes:

$$P(O|\mu) = \prod_{k=1}^{K} P(O^{(k)}|\mu) \tag{3.17}$$

The variables $\alpha_i(t), \beta_i(t), p_t(i,j)$ and $\delta_i(t)$ are calculated for each training sequence $O^{(k)}$. Then the re-estimation formulas are modified by adding together all the probabilities for each sequence. In this way the new parameters can be vowed as averages of the optima given by the individual training sequences. The re-estimation formulas for multiple observation sequences are:

$$\mu'_{im} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r} \delta_{im}^{(k)}(t) o_t^{(k)}}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r} \delta_{im}^{(k)}(t)} \tag{3.18}$$

$$\sigma'_{im} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r} \delta_{im}^{(k)}(t)(o_t^{(k)} - \mu_{im})(o_t^{(k)} - \mu_{im})'}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r} \delta_{im}^{(k)}(t)} \tag{3.19}$$

$$w_{im} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r} \delta_{i}^{(k)}(t)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r} \sum_{m=1}^{M} \delta_{im}^{(k)}(t)} \tag{3.20}$$

$$a_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r - 1} \alpha_i^{(k)}(t) a_{ij} b_j(o_{t+1}^{(k)}) \beta_j^{(k)}(t+1)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_r - 1} \alpha_i^{(k)}(t) \beta_i^{(k)}(t)} \tag{3.21}$$

Rabiner's algorithm is used to process a batch of speech sequences that are of equal importance, so he includes all the training sequences in the training process. In our financial time series, often cases are that patterns change from time to time. In order to capture the most recent trend, we create a dynamic training pool. The first data in the pool are popped out and new data are pushed into it when a trading day ends. Each time the training starts from the parameters of last day's trading yet with today's data.

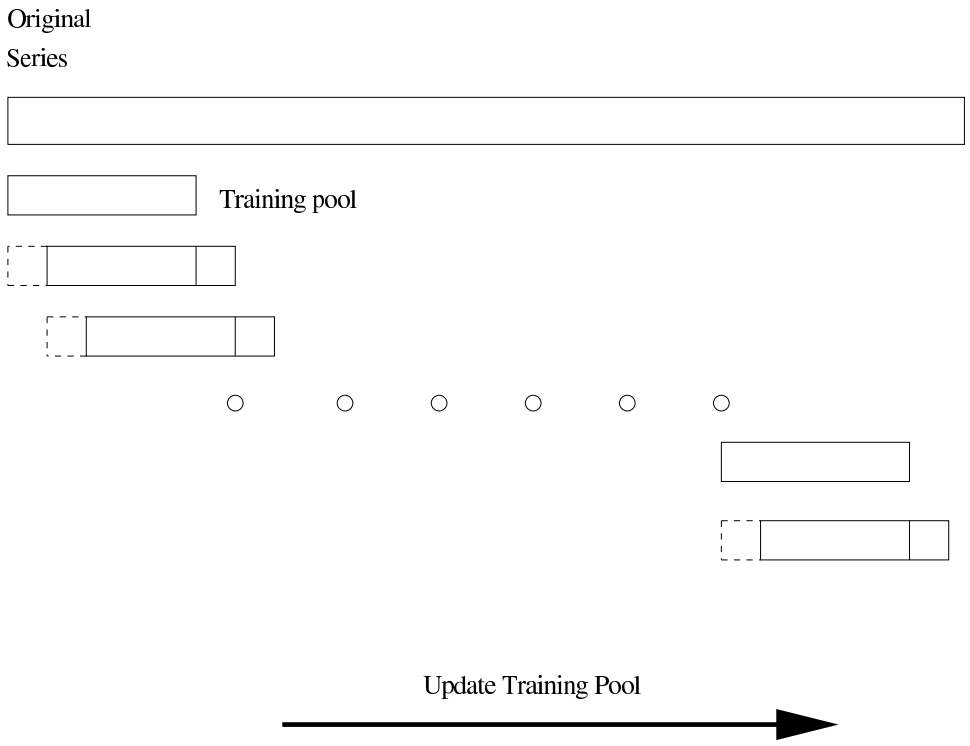

Figure 3.3: The dynamic training pool

3.5 Dynamic Training Pool

We introduce a new concept in the training process called the *training pool*. This is a segment of data that is updated dynamically throughout the training. At the end of a trading day, when the market closes, we can calculate the return for that day and add it to the pool. In the meantime, drop the oldest data in the pool. The training is then performed again based on the parameters from last training and the data from the new pool. This way the system always trains on fresh data. In the training pool each data sample serves as both prediction target and training sample. See figure 3.3 for a graphical illustration of the dynamic training pool.

3.6 Making Predictions with HMM

3.6.1 Setup of HMM Parameters

We construct a HMM with Gaussian Mixtures as observation density function(call it *HMGM*). The HMM has 5 states. Each state is associated with a Gaussian mixture that has 4 Gaussian probability distributions. The HMM is a fully connected trellis, each of the transition probability is assigned a real value at random between 0 and 0.2. The initial state probability distribution probabilities $\Pi = \{\pi_i\}$ are assigned a set of random values between 0 and 0.2. And there are a set of random values ranging between 0 and 0.25 for the initial mixture weights w. The initial values of the *covariance* for each of the Gaussian probability distribution function is assigned 1. And the *mean* of each Gaussian mixture component is assigned a real value between -0.3 and 0.3. This is the normal range of possible returns for each day.

Intuitively, the number of states is the number of the strategies we have to make predictions. This number should be neither too large nor too small. Too many strategies will make little distinction between each other and will increase the cost of computing. On the other hand if the number of states is too small, risk of mis-classification will increase. This can also be explained intuitively using financial theory: the more diversified the strategies, the less investment risk and the less return. The number of the mixture components follow the same discussion. We have launched experiments with different state and mixture numbers. Based on our experience, 5 states and 4 mixture components are reasonable numbers for the model. The architecture of the *HMGM* is shown in Fig 3.4.

This setting is similar to Shi and Weigend's experimental environment where they have one neural net expert associated with each of the three states. See [WMS95] and [SW97]. The optimal number of states can be decided through experiments.

The input into the HMGM is the real number vector introduced in section 1.3. In the following section we will describe how the real number is transformed into long or short signals.

3.6.2 Making predictions

We now come to the core part of the problem: making predictions before each trading day starts. Actually the prediction process is carried out together with the training at the same

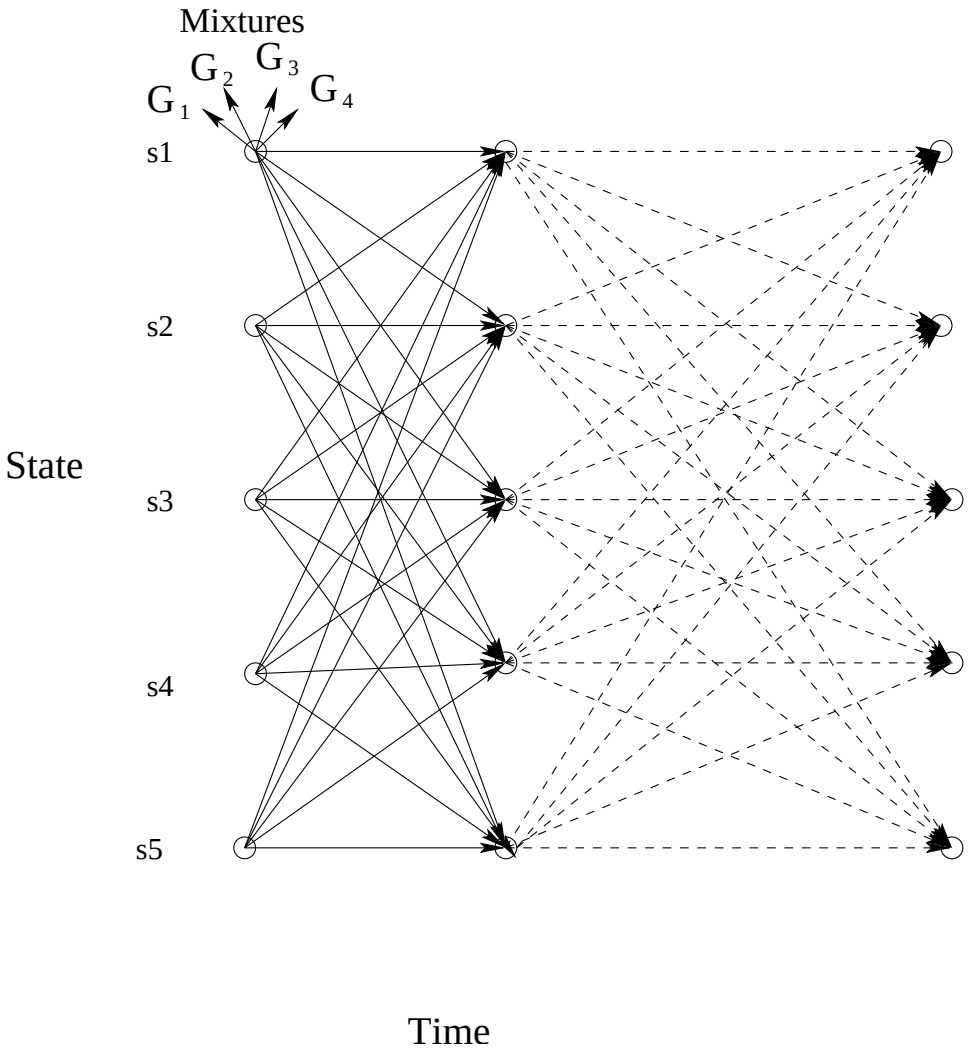

Figure 3.4: Hidden Markov Gaussian Mixtures

time. In other words, at the end of each trading day, when the market return of that day is known to the public, we include it in the training system to absorb the most updated information. This is exactly what human analysts and professional traders do. Many of them work hours after the market closes 4pm Atlantic time. The HMGM is then trained again with the newest time series and the parameters are updated.

When the training is done and parameters of the HMGM are updated, we performed the Viterbi algorithm to find the most "reliable" Gaussian mixtures associated with the most probable state. The weighted average of the mean of the Gaussian mixture is taken as the prediction for tomorrow. The induction step in section 2.7.2 is revised by replacing the function b_{ijo_t} with the Gaussian density function $\mathcal{N}(o_t, \mu_{jm}, \Sigma_{jm})$ for each state j and mixture component m:

$$\delta_j(t) = \max_{1 \le i \le N} \delta_i(t-1) a_{ij} \sum_{m=1}^{M} w_{jm} \mathcal{N}(o_t, \mu_{jm}, \Sigma_{jm}) \tag{3.22}$$

$$\psi_{jk}(t) = \operatorname*{argmax}_{1 \le i \le N} \delta_i(t-1) a_{ij} \sum_{m=1}^{M} w_{jm} \mathcal{N}(o_t, \mu_{jm}, \Sigma_{jm}) \tag{3.23}$$

Next we transform the real values generated by the Gaussian mixtures into binary symbols–positive or negative signals of the market. If the weighted mean of the Gaussian mixtures at the selected state is positive or zero, we output a positive signal, otherwise we output a negative signal. In other words, we have a threshold at zero. In our experiment, a weighted mean of zero will generate a positive signal, that will further result in a long strategy.

Algorithm 3.1 is the revised multi-observation EM algorithm for $HMGM$ written in the pseudo code.

The training set can be of any size, however as most financial analysts believe, a five day or twenty day set would be desirable because five days are the trading days in a week and one month consists of roughly twenty trading days. There is a balance between the size of the training set and the sensitivity of the model. The bigger the training set, the more accurate the model in terms of finding long term trend. However the trained model may not be sensitive to whipsaws.

Algorithm 3.1: Combined training-prediction algorithm for the HMGM

while not End of Series **do**
  initialization
  if tradingDay = first day of series **then**
    assign random values to the HMM parameters
  else
    initialize parameters with the last updated values
  end if
  Training data set update
  1.State drop day 1 in the previous training set
  2.State add current day (the day that has just ended) to the training set
  3.Update all parameters using the EM algorithm
  Prediction
  1.Run the Viterbi algorithm to find the Gaussian mixture with the most probable state
  2.Compare the weighted mean of the Gaussian mixture against the threshold zero
  3.Output the predicted signal
end while

3.7 Chapter Summary

In this chapter, we construct an HMM to predict the financial time series, more specifically the S&P 500 Index. Issues considered in the modeling process include the selection of the emission density function, the EM update rule and characteristics of the selected financial time series. These issues are addressed throughout the explanations in this chapter. Each state of the HMM is associated with a mixture of Gaussian functions with different parameters, which give each state different prediction preference. Some states are more sensitive to low volatile data and others are more accurate with high volatile segments. The revised weighted EM update algorithm puts more emphasis on the recent data. The Dow Jones Index is included in the experiments as a supporting indicator to decide the volatility of the S&P 500 series. We have achieved impressive results that are shown in chapter 5.

Chapter 4

Exponentially Weighted EM Algorithm for HMM

In this chapter we first review the traditional EM algorithm in more depth by taking a close look at the composition of the likelihood function and the Q function. Originally formalized in [DLR77], The EM algorithm for re-estimation is intensively studied under all sorts of problem settings, including Hidden Markov Model [BP66] [RJ86] [Rab89], conditional/joint probability estimation [BF96] [JP98], constrained topology HMM [Row00], coupled HMM [ZG02], hierarchical HMM [FST98], and Gaussian mixtures [RW84] [XJ96] [RY98]. However, all these variances of the EM algorithm do not take the importance attenuation in the time dimension into consideration. They take it for granted that each data sample in the training data set is of equal importance and the final estimation is based on this equal importance assumption, no matter how big the training set is. This approach works well when the training set is small and the data series studied are not time-sensitive or the time dimension is not a major concern. The traditional EM based HMM models have found great success in language and video processing where the order of the observed data is not significantly important.

However financial time series differ from all the above signals. They have some unique characteristics. We need to focus more on recent data while keep in mind of the past patterns, but at a reduced confidence. As we have discussed in chapter 1, many patterns disappeared or diminished after they had been discovered for a while. A most persuasive argument is the change of the Dow Jones Index. It took the Dow Jones Index 57 years to

rise from 200 to 2000, yet it only took 12 years to rise from 2000 to 10,000.

Realizing the change of importance over the time dimension and motivated by the concept of Exponential Moving Average (EMA) of financial time series, we presents the derivation of the exponentially weighted EM algorithm. We show that the final re-estimation of the parameters of the HMM and the Gaussian densities can be expressed in a form of a combination of the exponential moving averages (EMAs) of the intermediate state variables. To find better balance between short-term and long-term trends, we further develop a double weighted EM algorithm that is sensitivity-adjustable by including the volatility information. We show the convergence of the exponentially weighted EM algorithm is guaranteed by a weighted version of the EM Theorem.

4.1 Maximum-likelihood

We have a density function $p(X|\Theta)$ that represents the probability of the observation sequence $X = \{x_1, \ldots, x_N\}$ given the parameter Θ. In the HMGM model Θ includes the set of means and covariances of the Gaussian mixtures associated with each state as well as the transition probability matrix and the probability of each mixture component. Assuming the data are independent and identically distributed (i.i.d.) with distribution p, the probability of observing the whole data set is:

$$p(X|\Theta) = \prod_{i=1}^{N} p(x_i|\theta) = L(\Theta|X) \tag{4.1}$$

The function L is called the *likelihood function* of the parameters given the data. Our goal of the EM algorithm is to find the Θ^* that maximizes L:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\, L(\Theta|X) \tag{4.2}$$

Since the log function is monotonous, we often maximize the logarithm form of the likelihood function because it reduces multiplications to additions thus makes the derivation analytically easier.

4.2 The Traditional EM Algorithm

The EM algorithm takes the target data as composed of two parts, the observed part and the underlying part. The observed data are governed by some distribution with given form

but unknown parameters. The underlying data can be viewed as a random variable itself. The task of the EM algorithm is to find out the parameters for the distribution governing the observed part and the distribution generating the underlying part.

Now assume X as the observed part governed by some distribution with known form but unknown parameters. X is called the *incomplete data*. Then Assume the *complete data* $Z = (X, Y)$. By the Bayes rule, the joint density function of one data in the complete data sample is:

$$\begin{aligned} p(z|\Theta) &= p(x, y|\Theta) \\ &= p(y|x, \Theta)p(x|\Theta) \end{aligned} \tag{4.3}$$

x and y are individual data in the data sample X and Y. Note the a prior probability $p(x|\Theta)$ and the distribution of the hidden data are initially generated by assigning some random values. We now define the *complete data likelihood function* for the overall data set based on the above joint density function of individual data:

$$\begin{aligned} L(\Theta|Z) &= L(\Theta|X, Y) \\ &= p(X, Y|\Theta) \end{aligned} \tag{4.4}$$

When estimating the hidden data Y, X and Θ can be taken as constants and the complete data likelihood becomes a function of Y: $L(\Theta|X, Y) = h_{X,\Theta}(Y)$ where Y is the only random variable.

Next the EM algorithm finds the expected value of the complete data log-likelihood $\log p(X, Y|\Theta)$ with respect to the unknown data Y, given the observed part X and the parameters from last estimation. We define a Q function to denote this expectation:

$$\begin{aligned} Q(\Theta, \Theta^{(i-1)}) &= E\left[\log p(X, Y|\Theta)|X, \Theta^{(i-1)}\right] \\ &= \int_{y\in\Upsilon} \log p(X, y|\Theta) f(y|X, \Theta^{(i-1)})\, dy \end{aligned} \tag{4.5}$$

$\Theta^{(i-1)}$ are the estimates from last run of the EM algorithm. Based on $\Theta^{(i-1)}$ we try to find the current estimates Θ that optimize the Q function. Υ is the space of possible values of y.

As mentioned above, we can think of Y as a random variable governed by some distribution $f_Y(y)$ and $h(\Theta, Y)$ is a function of two variables. Then the expectation can be rewritten in

the form:

$$\begin{aligned} Q(\Theta) &= E_Y\left[h(\Theta, Y)\right] \\ &= \int_y h(\Theta, Y) f_Y(y)\, dy \end{aligned} \tag{4.6}$$

In the M-Step of the EM algorithm, we find the parameter Θ that maximizes the Q function:

$$\Theta^{(i)} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta, \Theta^{(i-1)}) \tag{4.7}$$

The two steps are repeated until the probability of the overall observation does not improve significantly. The EM algorithm is guaranteed to increase the log-likelihood at each iteration and will converge to a local maximum of the likelihood. The convergence is guaranteed by the EM Theorem [Jel97].

4.3 Rewriting the Q Function

An assumption that is not explicitly stated in the derivation of the EM algorithm in all works cited above is that: in calculating the Q function, we take it for granted that each data is of equal importance in the estimation of the new parameters. This is reasonable in problems like speech and visual signal recognition because the order of the data in the observed data sample is not that important, rather the general picture consisting of all data points is of more interest. However as stated in the beginning of this chapter, financial time series have characteristics which make the time dimension very important. As time goes on, old patterns attenuate and new patterns emerge. If the local data set that we use as training set is very large, the new emerged patterns or trends will be inevitably diluted. Therefore we need a new EM algorithm that gives more importance to more recent data.

To achieve this goal, we must first rewrite the Q function. Recall that the Q function is the expectation of the complete data likelihood function L with respect to the underlying data Y, given the observed data X and the last parameter estimates Θ. Now we include a time-dependent weight η in the expectation to reflect information on the time dimension. The revised expectation of the likelihood function $\hat{Q}$:

$$\begin{aligned} \hat{Q}(\Theta, \Theta^{(i-1)}) &= E\left[\log \eta p(X, Y|\Theta) | X, \Theta^{(i-1)}\right] \\ &= \int_{y \in \Upsilon} \log \eta p(X, y|\Theta) f(y|X, \Theta^{(i-1)})\, dy \end{aligned} \tag{4.8}$$

Since the logarithm function is monotonous and $0 < \eta \leq 1$, we can take η out of the log function.

$$\begin{aligned}
&[\log \eta \cdot P(X, y|\Theta) \leq \log \eta \cdot P(X, y|\Theta^g)] \Longleftrightarrow \\
&[\eta \cdot P(X, y|\Theta) \leq \eta \cdot P(X, y|\Theta^g)] \Longleftrightarrow \\
&[\eta \cdot \log P(X, y|\Theta) \leq \eta \cdot \log P(X, y|\Theta^g)]
\end{aligned} \tag{4.9}$$

This move can bring great convenience to the E step. The $\hat{Q}$ function then becomes:

$$\begin{aligned}
\hat{Q}(\Theta, \Theta^{(i-1)}) &= E\left[\eta \log p(X, Y|\Theta)|X, \Theta^{(i-1)}\right] \\
&= \int_{y \in \Upsilon} \eta \log p(X, y|\Theta) f(y|X, \Theta^{(i-1)})\, dy
\end{aligned} \tag{4.10}$$

The weight η is actually a vector of real values between 0 and 1: $\eta = \{\eta_1, \eta_2, \ldots, \eta_N\}$ where N is the number of data. The whole point is that the likelihood of each data is discounted by a *confidence density*. The weighted expectation function can be understood intuitively: Given the last parameter estimates and the observed data, if the expected probability of observing one data is 40%, given its position in the whole data sequence, we might have only 60% confidence in the accuracy of this result–meaning the final expected likelihood is only $0.4 \times 0.6 = 24\%$.

In a later section we will show that the $\hat{Q}$ function converges to a local maximum using techniques from the EM Theorem.

4.4 Weighted EM Algorithm for Gaussian Mixtures

We now show how the parameters of the HMGM introduced in chapter 3 can be re-estimated with the weighted EM algorithm. To begin with, we will use a mixture of Gaussians as an example to illustrate the derivation process, the HMGM is only a special case of Gaussian mixtures in nature.

Under this problem setting, first assume the following density for one single data x_j, given the parameter Θ of the Gaussian mixture:

$$p(x_j|\Theta) = \sum_{i=1}^{M} \eta_j \alpha_i p_i(x|\theta_i) \tag{4.11}$$

The parameters now consist of three types of variables $\Theta = (\eta_j, \alpha_1, \ldots, \alpha_M, \theta_1, \ldots, \theta_M)$. η_j is the time-dependent confidence weight for the j^{th} observed data. α_i indicates which mixture

component generates the observed data and it must observe the constraint $\sum_{i=1}^{M} \alpha_i = 1$. p_i is a density function for the i^{th} mixture component with parameter θ_i.

We assume a set of initial parameter values, for example $y_i = k$ indicates the i^{th} data is generated by the k^{th} mixture component. Also assume θ_i as the parameters for the i^{th} Gaussian function of the mixture. (θ_i includes the mean and covariance for the i^{th} Gaussian). Last assume the confidence weight η_n for the n^{th} data. We can obtain the complete data log likelihood function based on these hypothesized parameters:

$$\begin{aligned} \log \hat{L}(\Theta|X,Y) &= \eta \log P(X,Y|\Theta) \\ &= \sum_{i=1}^{N} \eta_i \log P(x_i|y_i)P(y) \\ &= \sum_{i=1}^{N} \eta_i \log \alpha_{y_i} p_{y_i}(x_i|\theta_{y_i}) \end{aligned} \tag{4.12}$$

So we start from a set of guessed parameters and from the assumed parameters we can go on to compute the likelihood function. We guess that $\Theta^g = \left(\eta_1, \ldots, \eta_N; \alpha_1^g, \ldots, \alpha_M^g; \theta_1^g, \ldots, \theta_M^g\right)$ are the best possible parameters for the likelihood $\hat{L}(\Theta^g|X,Y)$. Note the subscripts of η are in the time dimension, there are N data in the training set. The subscripts for α and θ are in the spatial dimension, indicating the parameter of a certain mixture component. η_i and α_{y_i} can be thought of as prior probabilities. And $p_{y_i}(y_i|x_i,\Theta^g)$ in the likelihood function can be computed using the Bayes's rule:

$$\begin{aligned} p(y_i|x_i,\Theta^g) &= \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{p(x_i|\Theta^g)} \\ &= \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(x_i|\theta_k^g)} \end{aligned} \tag{4.13}$$

and same as the basic EM algorithm,

$$p(\mathbf{y}|X,\Theta^g) = \prod_{i=1}^{N} p(y_i|x_i,\Theta^g) \tag{4.14}$$

where $\mathbf{y} = (y_1, \ldots, y_N)$ retain the independent property.

As mentioned above the confidence weight η can be taken as a prior probability. The $p_{y_i}(y_i|x_i,\Theta^g)$ is actually a conditional probability conditional on the data position i. However this conditional probability is often ignored in traditional EM algorithm because it was taken for granted that the prior probability is always equal to 1.

With these initial parameters Θ^g made by guessing, we can try to find out the current parameters Θ that maximizes the $\hat{Q}$ function:

$$\begin{aligned}
\hat{Q}(\Theta, Theta^g) &= \sum_{\mathbf{y}\in\Upsilon} \eta \log(L(\Theta|X,\mathbf{y}))p(\mathbf{y}|X,\Theta^g) \\
&= \sum_{\mathbf{y}\in\Upsilon} \eta \log P(X,\mathbf{y}|\Theta)p(\mathbf{y}|X,\Theta^g) \\
&= \sum_{\mathbf{y}\in\Upsilon} \sum_{i=1}^{N} \eta_i \log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})) \prod_{j=1}^{N} p(y_j|x_j,\hat{\Theta}^g) \\
&= \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_N=1}^{M}\sum_{i=1}^{N} \log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})) \prod_{j=1}^{N} p(y_j|x_j,\hat{\Theta}^g) \\
&= \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_N=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{M} \delta_{l,y_i} \log(\alpha_l p_l(x_i|\theta_l)) \prod_{j=1}^{N} p(y_j|x_j,\hat{\Theta}^g) \\
&= \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l p_l(x_i|\theta_l)) \sum_{y_1=1}^{M}\sum_{y_2=1}^{M}\cdots\sum_{y_N=1}^{M} \delta_{l,y_i} \prod_{j=1}^{N} p(y_j|x_j,\hat{\Theta}^g) \quad (4.15)
\end{aligned}$$

The latter part of the above formula can be simplified as below:

$$\begin{aligned}
&\textstyle\sum_{y_1=1}^{M} \sum_{y_2=1}^{M}\cdots\sum_{y_N=1}^{M} \delta_{l,y_i} \prod_{j=1}^{N} p(y_j|x_j,\hat{\Theta}^g) \\
&= \left(\sum_{y_1=1}^{M}\cdots\sum_{y_{i-1}=1}^{M}\sum_{y_{i+1}=1}^{M}\cdots\sum_{y_N=1}^{M}\prod_{j=1,j\neq i}^{N} p(y_j|x_j,\Theta^g)\right) p(l|x_i,\Theta^g) \\
&= \prod_{j=1,j\neq i}^{N}\left(\sum_{y_j=1}^{M} p(y_j|x_j,\Theta^g)\right) p(l|x_i,\Theta^g) \\
&= p(l|x_i,\Theta^g) \qquad (4.16)
\end{aligned}$$

This simplification can be carried out because of the constraint $\sum_{i=1}^{M} p(i|x_j,\theta^g) = 1$. Although the overall expectation is discounted by the time factor, the constraint on mixture components still holds at each data point over the time axis. Then the $\hat{Q}$ function can be written as:

$$\begin{aligned}
\hat{Q}(\Theta,\Theta^g) &= \sum_{l=1}^{M}\sum_{i=1}^{N} \eta_i \log(\alpha_l p_l(x_i|\theta_l)) p(l|x_i,\Theta^g) \\
&= \sum_{l=1}^{M}\sum_{i=1}^{N} \eta_i \log(\alpha_l) p(l|x_i,\Theta^g) + \sum_{l=1}^{M}\sum_{i=1}^{N} \eta_i \log(p_l(x_i|\Theta_l)) p(l|x_i,\Theta^g) \quad (4.17)
\end{aligned}$$

It becomes clear that to maximize the $\hat{Q}$ function we must maximize both terms in the above formula separately. The first term has α_l which must observe the constraint $\sum_l \alpha_l = 1$. So it is a constrained function optimization problem where the Lagrange multiplier λ can be applied. And the maximum is achieved when the following equation is satisfied:

$$\frac{\partial}{\partial \alpha_l}\left[\sum_{l=1}^{M}\sum_{i=1}^{N}\eta_i \log(\alpha_l)p(l|x_i,\Theta^g) + \lambda\left(\sum_l \alpha_l - 1\right)\right] = 0 \tag{4.18}$$

From above, we have:

$$\sum_{i=1}^{N}\frac{1}{\alpha_l}\eta_i p(l|x_i,\Theta^g) + \lambda = 0 \tag{4.19}$$

Summing both the numerator and denominator of the first term over l, the left side of the equation becomes:

$$\begin{aligned}\frac{\sum_{l=1}^{M}\sum_{i=1}^{N}\eta_i p(l|xi,\Theta^g)}{\sum_{l=1}^{M}\alpha_l} + \lambda &= \frac{\sum_{i=1}^{N}\sum_{l=1}^{M}\eta_i p(l|x_i,\Theta^g)}{\sum_{l=1}^{M}\alpha_l} + \lambda &(4.20)\\ &= \sum_{i=1}^{N}\eta_i + \lambda &(4.21)\end{aligned}$$

Let the above expression equal 0, we can easily get $\lambda = -\sum_{i=1}^{N}\eta_i$. Substituting λ in the above equation we finally get the estimation of α_l:

$$\alpha_l = \frac{\sum_{i=1}^{N}\eta_i p(l|x_i,\Theta^g)}{\sum_{i=1}^{N}\eta_i} \tag{4.22}$$

Next we derive the update rule for the mean and covariance of the Gaussian components. This is done by maximizing the second term in the $\hat{Q}$ function. Given the form of a D-dimensional Gaussian distribution, we know

$$p_l(x|\mu_l,\Sigma_l) = \frac{1}{(2\pi)^d/2|\Sigma_l|^{1/2}} exp\left[-\frac{1}{2}(x-\mu_l)^T\Sigma_l^{-1}(x-\mu_l)\right] \tag{4.23}$$

Substituting this distribution form into the second term of 4.17 and taking away constant terms that will be eliminated after taking derivatives, we get:

$$\begin{aligned}\sum_{l=1}^{M}\sum_{i=1}^{N} &\quad \eta_i \log(p_l(x_i|\Theta_l))p(l|x_i,\Theta^g)\\ = &\quad \sum_{l=1}^{M}\sum_{i=1}^{N}\eta_i\left(-\frac{1}{2}log(|\Sigma_l|) - \frac{1}{2}(x_i-\mu_l)^T\Sigma_l^{-1}(x_i-\mu_l)\right)p(l|x_i,\Theta^g) &(4.24)\end{aligned}$$

The maximization of the above expression is achieved by taking the derivative with respect to μ_l and letting it equal to 0:

$$\sum_{i=1}^{N} \eta_i \Sigma^{-1}(x_i - \mu_l) p(l|x_i, \Theta^g) = 0 \tag{4.25}$$

Then we can work out the current estimation for μ_l:

$$\mu_l = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^g)}{\sum_{i=1}^{N} \eta_i p(l|x_i, \Theta^g)} \tag{4.26}$$

Similarly, take the derivative of (4.45) with respect to the covariance Σ_l and setting it equal to 0, we get:

$$\sum_{i=1}^{N} \eta_i p(l|x_i, \Theta^g) \left(\Sigma_l - (x_i - \mu_l)(x_i - \mu_l)^T\right) = 0 \tag{4.27}$$

So the estimation for Σ is:

$$\Sigma_l = \frac{\sum_{i=1}^{N} \eta_i p(l|x_i, \Theta^g)(x_i - \mu_l)(x_i - \mu_l)^T}{\sum_{i=1}^{N} \eta_i p(l|x_i, \Theta^g)} \tag{4.28}$$

4.5 HMGM Parameter Re-estimation with the Weighted EM Algorithm

After showing the derivation of the exponentially weighted EM algorithm for the Gaussian mixture case, we now return to our HMGM model. As discussed in chapter 2 and 3, we have two intermediate variables describing the whole process of generating the observed data: $\gamma_i(t)$ represents the probability of being in state i at time t and w_{im} is the probability of picking the m^th mixture component at state i. These intermediate variables combine together to generate the probability of the observed data. We can define the probability $\delta_{im}(t)$ that denotes the m^{th} mixture density generating observation o_t as:

$$\delta_{im}(t) = \gamma_i(t) \frac{w_{im} b_{im}(o_t)}{b_i(o_t)} \tag{4.29}$$

We find that the $\delta_{im}(t)$ enjoys the same status as the prior probability $p(l|x_i, \Theta^g)$ in previous discussions. The calculation of the transition probability a_{ij} also involves a time variable

thus can also be weighted. So the exponential weighted update rules for the HMGM are derived in a similar form, taking the time-intensifying weight into consideration:

$$\mu'_{im} = \frac{\sum_{t=1}^{T} \eta_t \delta_{im}(t) o_t}{\sum_{t=1}^{T} \eta_t \delta_{im}(t)} \tag{4.30}$$

$$\sigma'_{im} = \frac{\sum_{t=1}^{T} \eta_t \delta_{im}(t)(o_t - \mu'_{im})(o_t - \mu'_{im})'}{\sum_{t=1}^{T} \eta_t \delta_{im}(t)} \tag{4.31}$$

$$w'_{im} = \frac{\sum_{t=1}^{T} \eta_t \delta_{im}(t)}{\sum_{t=1}^{T} \eta_t \delta_i(t)} \tag{4.32}$$

$$a'_{ij} = \frac{\sum_{t=1}^{T} \eta_t \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\sum_{t=1}^{T} \eta_t \alpha_i(t) \beta_i(t)} \tag{4.33}$$

4.6 Calculation and Properties of the Exponential Moving Average

Before we go on to show how we can rewrite the re-estimation formulas with EMAs of the intermediate variables, let's take a closer look at the calculation and properties of the EMA.

The EMA is invented originally as a technical analysis indicator that keeps track of the trend changes in financial time series. Short-term EMAs response faster than long-term EMAs to changes in the trend. When the curves of two EMA with different length cross each other, this usually means a trend change. Technical analysis people developed other indicators based on the EMA, such as the *Moving Average Convergence Divergence (MACD)*. Figure 4.1 shows the signals indicated by comparing the 30-day EMA and the 100-day EMA for an individual stock *Intel* :

Take the variable $p_t(ij)$ for illustration. We denote $\overline{p_t(ij)}$ as the exponential moving average of the original variable $p_t(ij)$ at time t. As by its name implies, the EMA changes as time moves on. The calculation of a five day exponentially moving average for this new intermediate variable consists of two steps:

First, calculate the arithmetic average of the first five days. Denote this average as $\overline{p_5(ij)}$. Second, starting from the sixth day, the EMA of each day is:

$$\overline{p_t(ij)} = \rho \times p_t(ij) + (1 - \rho) \times \overline{p_{t-1}(ij)} \tag{4.34}$$

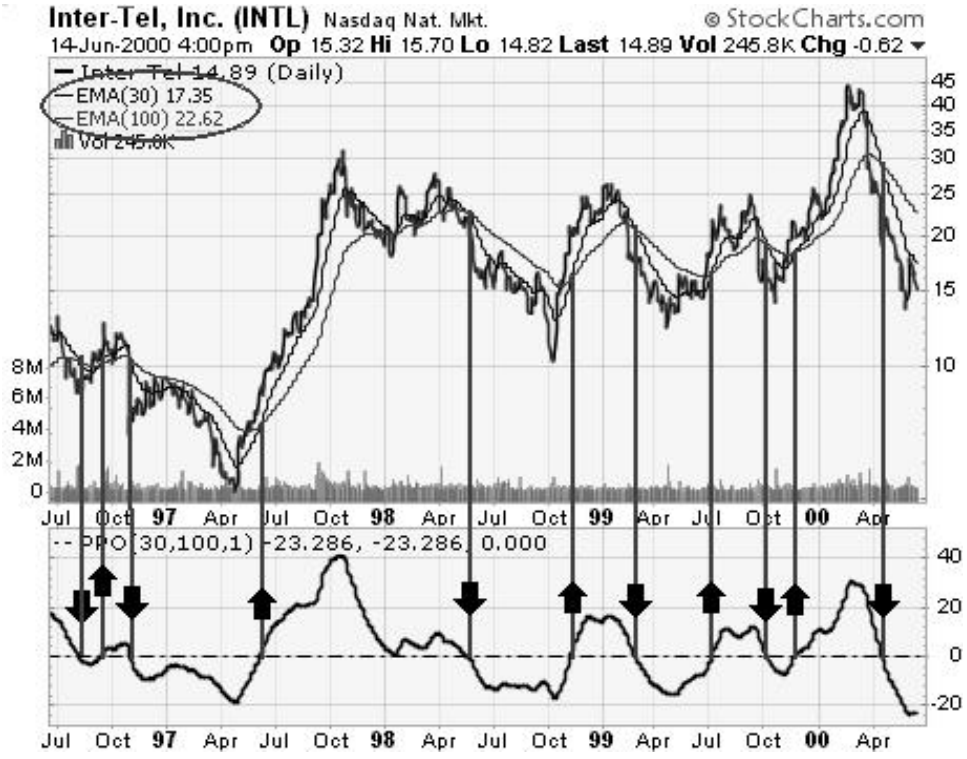

Figure 4.1: When the 30-day moving average moves above the 100-day moving average, the trend is considered bullish. When the 30-day moving average declines below the 100-day moving average, the trend is considered bearish. (picture from *stockcharts.com*)

Note that in the above equation, $p_t(ij)$ is the actual intermediate variable value at time t, while $\overline{p_t(ij)}$ is the exponential moving average of that variable at time t. We keep running the second step until the last day of the training pool.

The EMA of the second intermediate variable $\delta_i(t)$ that stores the probability of being at state i at time t, can be calculated in a similar way:

$$\overline{\delta_i(t)} = \rho \times \delta_i(t) + (1-\rho) \times \overline{\delta_i(t-1)} \tag{4.35}$$

In equations (4.1) and (4.2), ρ is the weight that we have been talking about all the time. The value of ρ is given by the formula: $\rho = \frac{2}{1+k}$, k = the number of days in the moving window

For example, if the size the moving window is 5, which means we are calculating a five day EMA, then the value of ρ would be:

$$\rho = \frac{2}{1+5} = 0.3333... \tag{4.36}$$

So the last day accounts for one third of the final value.

The 5-day EMA at the 20th day of the training pool is calculated by running equation (4.1) until the 20th day:

$$\overline{p_6(ij)} = p_6(ij) \times \rho + \overline{p_5(ij)} \times (1-\rho)$$
$$\overline{p_7(ij)} = p_7(ij) \times \rho + \overline{p_6(ij)} \times (1-\rho)$$
$$.....$$
$$\overline{p_{20}(ij)} = p_{20}(ij) \times \rho + \overline{p_{19}(ij)} \times (1-\rho)$$

As we can see from equation (4.2), the last day in the training window takes up a quite large portion and should notably affect the training procedure. As time goes on, the importance of the early data drops exponentially, as shown in Fig 4.3. This is easily shown when we go into the details of the calculation of equation (4.1):

$$\begin{aligned}
\overline{p_t(ij)} &= \rho \times p_t(ij) + (1-\rho) \times \overline{p_{t-1}(ij)} \\
&= \rho \times p_t(ij) + (1-\rho) \times [\rho \times p_{t-2}(ij) + (1-\rho) \times \overline{p_{t-2}(ij)}] \\
&= \rho \times p_t(ij) + (1-\rho) \times [\rho \times p_{t-2}(ij) + (1-\rho)^2 \times \overline{p_{t-2}(ij)}]
\end{aligned} \tag{4.37}$$

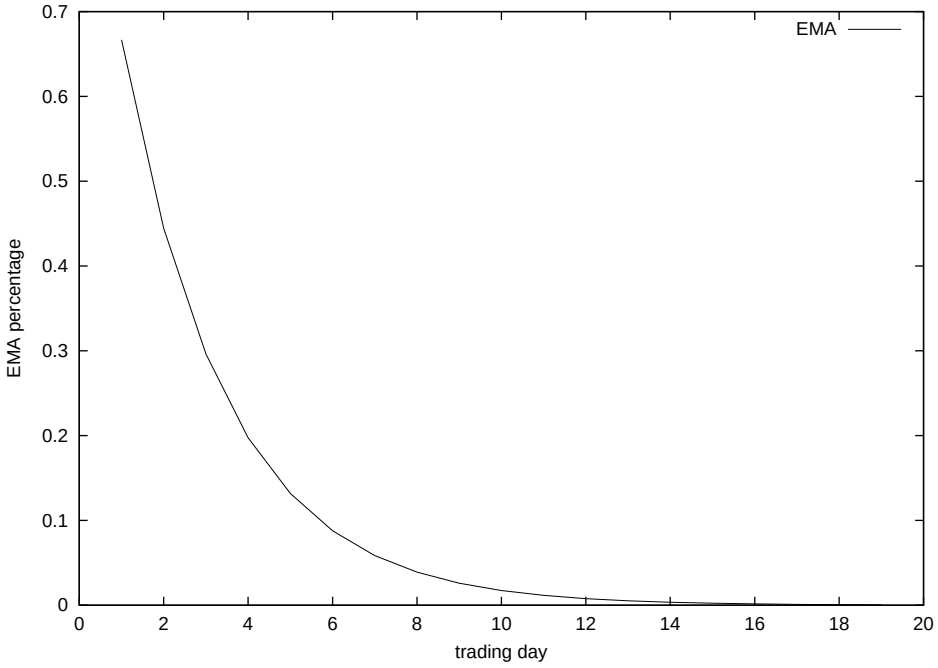

Figure 4.2: The portion of each of the first 19 days' EMA in calculating the EMA for the 20th day. As is shown in the figure, the EMA at the 19th day accounts for 2/3 of the EMA at day 20.

Now we see from (4.5), in calculating the EMA at day t, the weight of the EMA at day $t-1$ takes up a portion of $(1-\rho)$ while the weight of the EMA at day $t-2$ is only $(1-\rho)^2$. We can also know by inference that day $t-3$ will have a portion of $(1-\rho)^3$ in EMA at day t. Fig 4.4 gives a graphical view of the portion each day takes in calculating of the EMA.

4.7 HMGM Parameters Re-estimation in a Form of EMA

In previous sections we derived the exponentially weighted estimation rules for the parameters of the HMGM. We show the formulas with the exponential discount weight η. However, there is still one problem regarding η: how is the time-dependent variable η defined? In fact η can be taken as a density function whose distribution is prior knowledge. Our task now is to find an expression of time for η.

A nice property about the exponentially weighted intermediate parameters is that, when calculating this weighted average, while the large part of weight is given to the recent data, the information from the early data are not lost. The old data are also included in the final

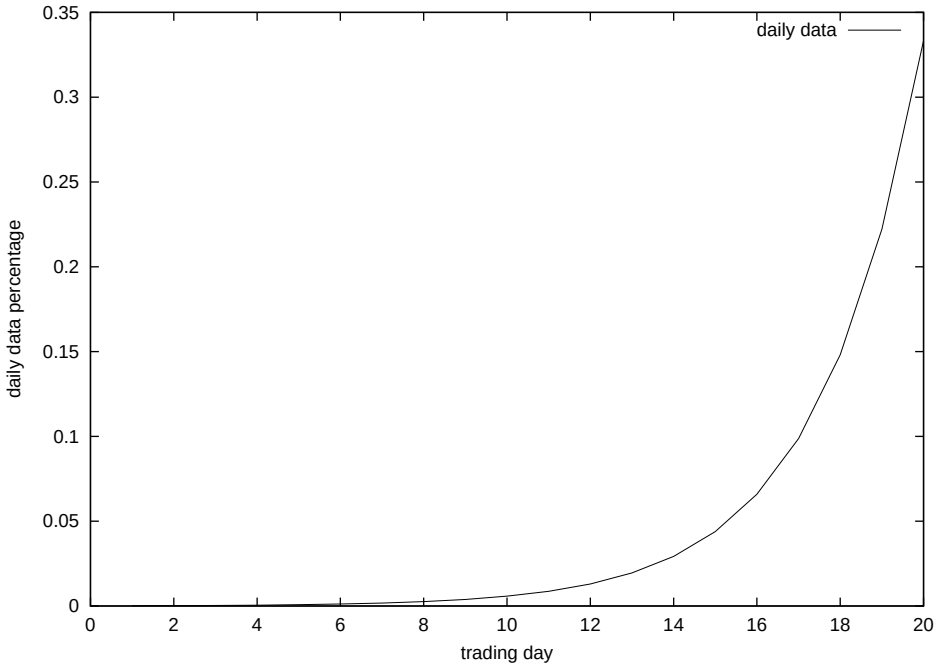

Figure 4.3: The portion of each of the 20 days' *actual* data in calculating the EMA for the 20th day. The last day account for 1/3 of the EMA at day 20. The data earlier than day 10 take up a very small portion but they are still included in the overall calculation

average, by a discounted weight. The older the data, the less important they become, and thus take up less portion in the average. But their impact never disappear.

Giving more recent data more weight makes the HMM getting sensitive to trend changes in time series. The HMM will ride on the new trend quickly, without having to make too many mistakes at the beginning of a new trend before it gets fit for the new trend. That is one important motivation of replacing the arithmetic average with EMA in the EM update rule.

Originally, the EMA was invented as a technical analysis tool that deals with financial time series data, such as prices, volume, etc directly. In the EM algorithm for HMM parameters re-estimation, there are a set of middle variables: $\alpha_i(t), \beta_i(t), \delta_{im}(t), \delta_i(t)$. These middle variables all have their actual meanings: they indicate the probabilities of some events at a certain point of time. So these model variables are also related to the time series data. This is another motivation we rewrite the re-estimation formulas in the form of EMA of the model variables. We take the denominator of the μ re-estimation for example. From $\sum_{t=1}^{T} \eta_t \delta_{im}(t)$ we can see each $\delta_{im}(t)$ at a timing point t is associated with one value of η_t. The sum can be extended as:

$$\begin{aligned} \textstyle\sum_{t=1}^{T} & \quad \eta_t \delta_{im}(t) \\ = & \quad \eta_1 \delta_{im}(1) + \eta_2 \delta_{im}(2) + \cdots + \eta_{T-1} \delta_{im}(T-1) + \eta_T \delta_{im}(T) \end{aligned} \tag{4.38}$$

Let $\rho = \frac{2}{1+k}$ where k is for the *multiplier* in a k-EMA. Then set η_t by the following rules:

1. If $1 \leq t \leq k, \eta_t = (1-\rho)^{(T-k)} \cdot \frac{1}{k}$;
2. If $k < t < T, \eta_t = \rho \cdot (1-\rho)^{(T-t)}$;
3. If $t = T, \eta_t = \rho$.

Let $\overline{\delta_{im}(t)}$ represent the k-EMA of δ_{im} at time t. Based on these rules, the expression (4.38) is turned into a form of the EMA of the intermediate variable $\delta_{im}(t)$:

$$\eta_1 \delta_{im}(1) + \eta_2 \delta_{im}(2) + \cdots + \eta_{T-1} \delta_{im}(T-1) + \eta_T \delta_{im}(T)$$

$$
\begin{aligned}
&= (1-\rho)^{(T-k)} \cdot \frac{1}{k} \cdot \delta_{im}(1) + \cdots + (1-\rho)^{(T-k)} \cdot \frac{1}{k} \cdot \delta_{im}(k) \\
&\quad +\rho \cdot (1-\rho) \cdot \delta_{im}(T-1) + \rho \cdot \delta_{im}(T) \\
&= (1-\rho)^{(T-k)} \cdot \overline{\delta_{im}(k)} + \rho \cdot (1-\rho)^{(n-(k+1))} \cdot \delta_{im}(k+1) \\
&\quad +\rho \cdot (1-\rho)^{(n-(k+2))} \cdot \delta_{im}(k+2) + \cdots \\
&\quad +\rho \cdot (1-\rho) \cdot \delta_{im}(T-1) + \rho \cdot \delta_{im}(T) \qquad (4.39)
\end{aligned}
$$

The first k terms work together to generate the EMA for the first k periods, then from $k+1$ to T, apply Formula (4.35) to δ_{im} and induction on Equation (4.39), we can get the following result:

$$
\begin{aligned}
&(1-\rho) \cdot \left[\rho \cdot \delta_{im}(T-1) + (1-\rho) \cdot \overline{\delta_{im}(T-2)}\right] \\
&= (1-\rho) \cdot \overline{\delta_{im}(T-1)} + \rho \cdot \delta_{im}(T) \\
&= \overline{\delta_{im}(T)} \qquad (4.40)
\end{aligned}
$$

Therefore, the re-estimation formulas finally came into a form of EMA of some intermediate variables.

$$\mu'_{im} = \frac{\sum_{t=1}^{T} \eta_t \delta_{im}(t) o_t}{\overline{\delta_{im}(T)}} \qquad (4.41)$$

$$\sigma'_{im} = \frac{\sum_{t=1}^{T} \eta_t \delta_{im}(t)(o_t - \mu'_{im})(o_t - \mu'_{im})'}{\overline{\delta_{im}(T)}} \qquad (4.42)$$

$$w_{im} = \frac{\overline{\delta_{im}(T)}}{\overline{\delta_i(T)}} \qquad (4.43)$$

$$a'_{ij} = \frac{\sum_{t=1}^{T} \eta_t \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\overline{\alpha_i(T)\beta_i(T)}} \qquad (4.44)$$

4.8 The EM Theorem and Convergence Property

This section shows that our exponentially weighted EM algorithm is guaranteed to converge to a local maximum by a revised version of the EM Theorem. For review of the EM Theorem please refer to [Jel97].

The EM Theorem for exponentially weighted EM algorithm: Let X and Y be random variables with distributions $P(\hat{X})$ and $P(\hat{Y})$ under some model $\hat{\Theta}$. And let $P'(\hat{X})$

and $\hat{P'(Y)}$ be the distribution under a model Θ' after running the exponentially weighted EM algorithm. Then,

$$\left[\sum_{Y} \hat{P}(Y|X,\eta) \log \frac{\hat{P}'(Y,X|\eta)}{\hat{P}(Y,X|\eta)} \geq 0\right] \Rightarrow \hat{P}'(X|\eta) \geq \hat{P}(X|\eta) \tag{4.45}$$

This Theorem implies that if for a model $\hat{\Theta}$ and an updated model $\hat{\Theta}'$, if the left hand inequality holds, then the observed data Y will be more probable under $\hat{\Theta}'$ than $\hat{\Theta}$.

Proof: From the derivation of the exponentially weighted update rule we know

$$0 \leq \sum_{Y} \hat{P}(Y|X,\eta) \leq 1$$

And the logarithm function is a monotonous one. So,

$$\begin{aligned}
\log \quad & \hat{P}'(X|\eta) \geq log\hat{P}(X|\eta) \\
\Leftrightarrow \quad & \sum_{Y} \hat{P}(Y|X,\eta) \log \hat{P}'(X|\eta) \geq \sum_{Y} \hat{P}(Y|X,\eta) \log \hat{P}(X|\eta) \\
\Leftrightarrow \quad & \sum_{Y} \hat{P}(Y|X,\eta) \log \hat{P}'(X|\eta) - \sum_{Y} \hat{P}(Y|X,\eta) \log \hat{P}(X|\eta) \geq 0
\end{aligned} \tag{4.46}$$

Then we can perform a series of logarithm transformation operations on expression (4.86):

$$\begin{aligned}
& \textstyle\sum_Y \quad \hat{P}(Y|X,\eta) \log \hat{P}'(X|\eta) - \sum_{Y} \hat{P}(Y|X,\eta) \log \hat{P}(X|\eta) \\
= \quad & \textstyle\sum_Y \quad \hat{P}(Y|X,\eta) \log \frac{\hat{P}'(Y,X|\eta)}{\hat{P}'(Y|X,\eta)} - \sum_{Y} \hat{P}(Y|X,\eta) \log \frac{\hat{P}(Y,X|\eta)}{\hat{P}(Y|X,\eta)} \\
= \quad & \textstyle\sum_Y \quad \hat{P}(Y|X,\eta) \log \frac{\hat{P}'(Y,X|\eta)}{\hat{P}'(Y|X,\eta)} + \sum_{Y} \hat{P}(Y|X,\eta) \log \frac{\hat{P}(Y|X,\eta)}{\hat{P}(Y,X|\eta)} \\
= \quad & \textstyle\sum_Y \quad \hat{P}(Y|X,\eta) \log \frac{\hat{P}'(Y,X|\eta)}{\hat{P}(Y,X|\eta)} + \sum_{Y} \hat{P}(Y|X,\eta) \log \frac{\hat{P}(Y|X,\eta)}{\hat{P}'(Y|X,\eta)} \\
\geq \quad & \textstyle\sum_Y \quad P(Y|X,\eta) \log \frac{\hat{P}'(Y,X)}{\hat{P}(Y,X)} \geq 0
\end{aligned} \tag{4.47}$$

Q. E. D.

The last step follows from the Jensen's inequality. The Jensen's inequality basically states that mis-estimation of the distribution governing a random variable will eventually result in increased uncertainty/information. For a thorough review of the Jensen's inequality please refer to [**?**] [HLP88] and [GI00].

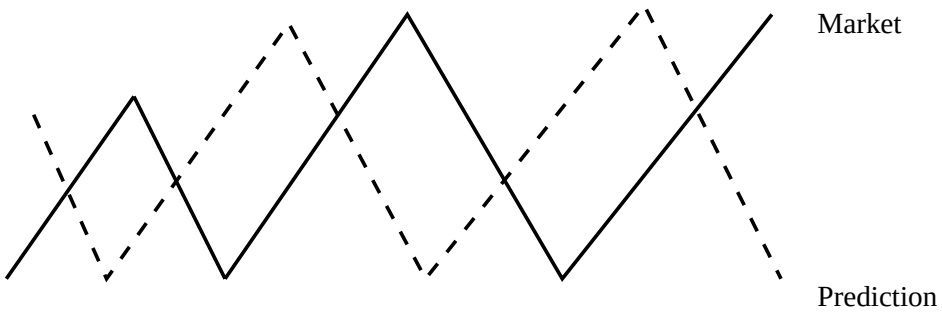

Figure 4.4: The prediction falls behind the market during highly volatile periods of time. This is because the system is too sensitive to recent changes.

4.9 The Double Weighted EM Algorithm

There is still room for improvement in the previous exponentially weighted EM rule. However, just as any other improvements, we have to consider the cost of the improvement of sensitivity and find out a way to overcome any possible problem.

4.9.1 Problem with the Exponentially Weighted Algorithm

The exponentially weighted EM algorithm quickly picks the trend at times when there is a turning of the trend. However, sometimes, especially during the highly volatile periods, the direction of the financial time series reverse at a very high frequency in a very short period of time. In these cases, there is danger of falling right behind the market movement, as shown in Fig 4.5.

Fig 4.5 shows a problem caused by high sensitivity. Not only the turning points of the market are all missed, the prediction of the turning points are not the actual market turning points. This is because the system is too sensitive to recent data that they fall into the pitfall of high volatility. Actually if one keeps making prediction of one same direction, or if he refers to the long term trend during that period of time, he may have a better opportunity of getting a higher gain.

4.9.2 Double Exponentially Weighted Algorithm

To solve this problem, we arm the HMM with a self-confidence adjustment capability. Multiply the weight for the most recent data η with a second weight ζ. ζ is the confidence rate for the weight η. It is calculated by comparing the trend signal, i.e., positive and

negative symbols of the S&P 500 Index with the trend signal of the Dow Jones Index. The Dow Jones is a relatively mild index, comparing to the S&P 500 and NASDAQ. We can of course use other indicators that reserves long term trend well, but for the time being, we will use the Dow for our experiment.

Comparing the S&P 500 with the Dow for the past five days. If they have the same directions for everyday of the past 5 days, the HMM gets full confidence 1. Otherwise, the confidence weight ζ is calculated as the total number of days that Dow coincides with the S&P divided by 5. Then time the two weights together, this is the weight that we will use for further calculation.

The adjustable weighted exponential average for the two intermediate variables are:

$$\begin{aligned} \overline{p_t(ij)} &= \eta \times \zeta \times p_t(ij) + (1 - \eta \times \zeta) \times \overline{p_{t-1}(ij)\delta_i(t)} \\ &= \eta \times \zeta \times \delta_i(t) + (1 - \eta \times \zeta) \times \overline{\delta_i(t-1)} \end{aligned} \tag{4.48}$$

Then apply the formulas (4.13-4.16) to get the updated parameters.

4.10 Chapter Summary

In this chapter we derived a novel exponentially weighted EM algorithm for training HMGMs. The EWEM algorithm takes the time factor into consideration by discounting the likelihood function in traditional EM by an exponential attenuation weight which we call the *confidence weight*. This is very important in financial time series analysis and prediction because financial time series are known to be non-stationary and different periods should be of different interest in generating a prediction for the future. We show that the re-estimation of the parameters come in a form of exponential moving average of the intermediate variables used for calculation of the final parameters. We prove the convergence to a local maximum is guaranteed by an exponential version of the EM Theorem. To find better balance between long-term and short-term trends, we develop a double-weighted EM algorithm.

Chapter 5

Experimental Results

This section has all the experimental results inside. The first section shows the prediction results on a synthetic data set. The second section includes the predictions for the years 1998 to 2000 of the NASDAQ Composite Index with a recurrent neural network. The third section includes prediction results of the HMM with Gaussian mixtures as emission density functions and trained with the simple weighted EM algorithm. We test a set of data with different *training windows*. The last section shows the results from the weighted EM algorithm.

5.1 Experiments with Synthetic Data Set

We first test different models on an artificially generated data set. The data is generated by a function composed of three parts:

$$f(t) = 0.2 * (\tau(t)/100 - 0.005) + 0.7 * f(t-1) + 0.1 * 0.00001 \tag{5.1}$$

$\tau(t)$ is a random number that takes 20% of the overall output. 70% of the data at step t is from that of step $t-1$. This is to try to hold the impact of the recent past. And lastly there is a 10% of up-growing trend at the growth rate of 0.00001. 20% of the data are generated randomly. This may not exactly be the case in real life financial time series, but our model should be able to at least handle it before applied to real data.

We test the recurrent neural network and the HMGM on this artificial data set for 400 days. Both models predict the signal 5 days ahead. Results show that over the test period of 395 days (prediction starts on the 6th day), the HMGM performs better than

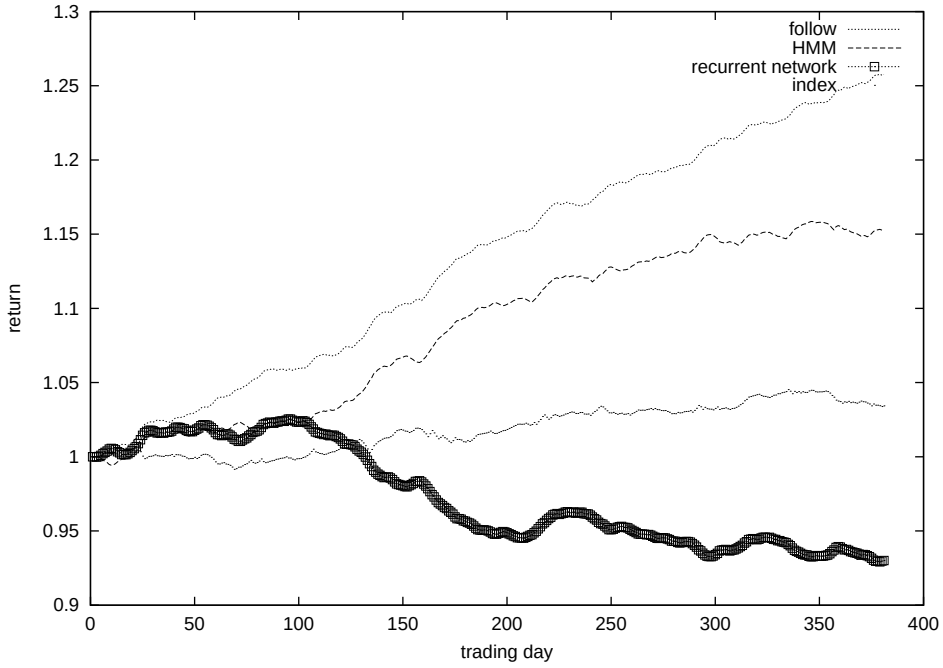

Figure 5.1: Experimental results on computer generated data.

the recurrent neural network by 10.55%, and 22.2% above the cumulative return based on the artificial generated data. The HMGM under-performs a *follow-the-trend* strategy which takes yesterday's signal as today's prediction. This is because the data are composed in a way that a large portion of the data is taken directly from the recent past. However, it is clearly shown that the HMGM is a better model of the two statistical learning models and it works better than the *buy-and-hold* strategy (which is the index). In the following sections we will test the HMGM on real-life data, the S&P 500 Index.

The following table shows the comparisons of accuracy between the recurrent network and the exponentially weighted HMGM. The accuracy rates are based on statistics from ten runs of both models on ten data sets generated by the equation 5.1. As we can see the exponentially weighted HMGM(EW-HMGM) outperforms the recurrent network for positive signals and slightly behind on negative ones, but has a better overall performance. And the EW-HMGM has less volatility in prediction accuracy at different runs of experiment.

	Positive	**Negative**	**Overall**
	Min/Max/Avg	**Min/Max/Avg**	**Min/Max/Avg**
Recurrent Net	0.44/0.72/0.55	0.52/0.79/0.62	0.53/0.65/0.58
EW-HMGM	0.54/0.80/0.64	0.54/0.68/0.57	0.54/0.64/0.61

Table 5.1: Comparison of the recurrent network and exponentially weighted HMGM on the synthetic data testing. The EW-HMGM outperforms the recurrent network in overall result, and has less volatility.

5.2 Recurrent Neural Network Results

The NASDAQ Index data are separated into three years. As can be seen from the data, the recurrent neural network has a positve bias toward steady trends. By that we mean the recurrent network performs better during long term bull market (as in late 1998 and early 1999) or long term bear market (as in the year 2000).

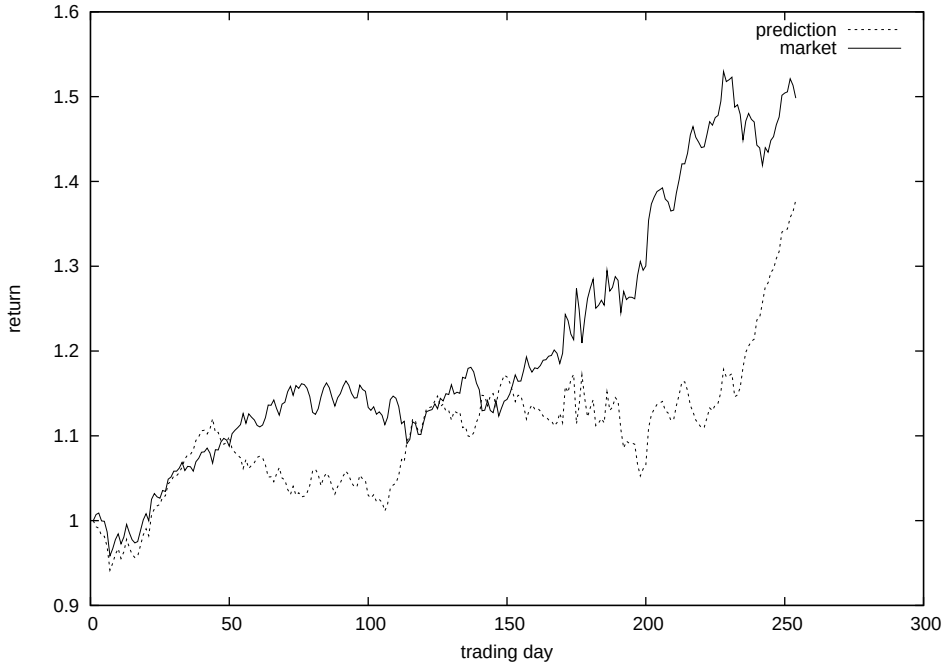

Figure 5.2: Prediction of the NASDAQ Index in 1998 with a recurrent neural network.

As we can see from the above figures, recurrent neural networks work better during low volatile, steady trends, no matter it is a bull market or bear market. This is especially

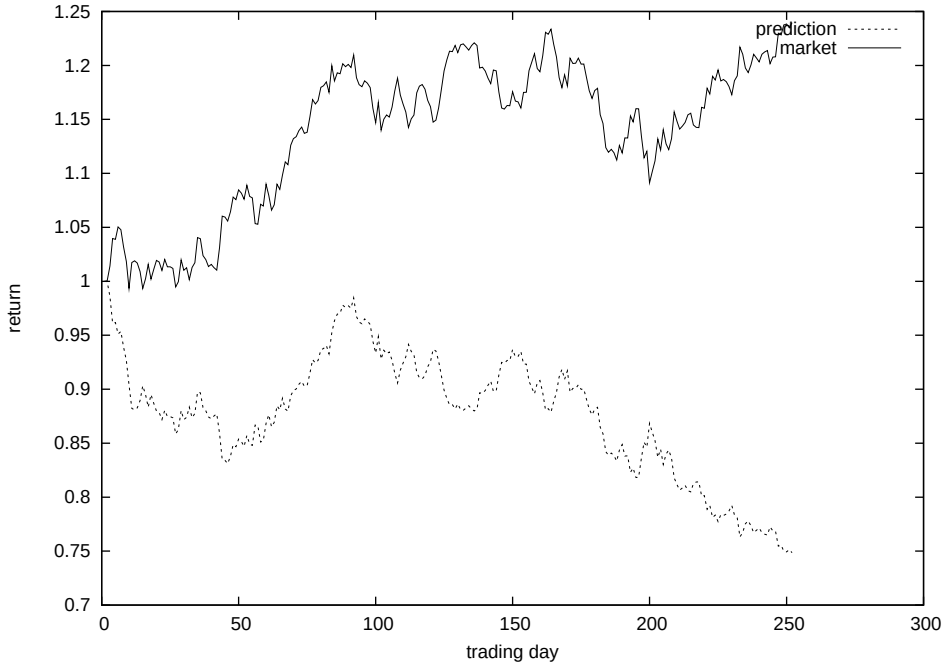

Figure 5.3: Prediction of the NASDAQ Index in 1999 with a recurrent neural network.

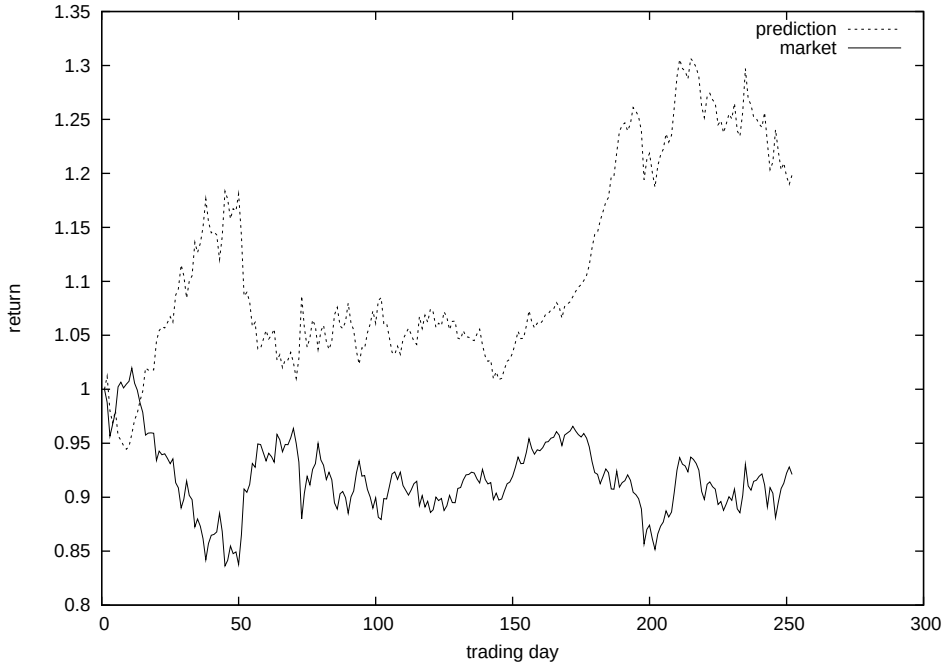

Figure 5.4: Prediction of the NASDAQ Index in 2000 with a recurrent neural network.

apparent in the prediction for the year 2000. Part of the reason is due to the memory brought by the recurrent topology. The past data has too much impact on the current training and prediction.

5.3 HMGM Experimental Results

We launch a series of experiments on the *Hidden Markov Gaussian Mixtures (HMGM)*. The *training window* introduced in chapter 3 is adjustable from 10-day to 100-day. The EM algorithm is performed on the training data with different training windows. Results show that the prediction performance changes as the training window size is adjusted. The best result is achieved by the 40-day training day window. And prediction becomes worse as the training window is changed in either direction.

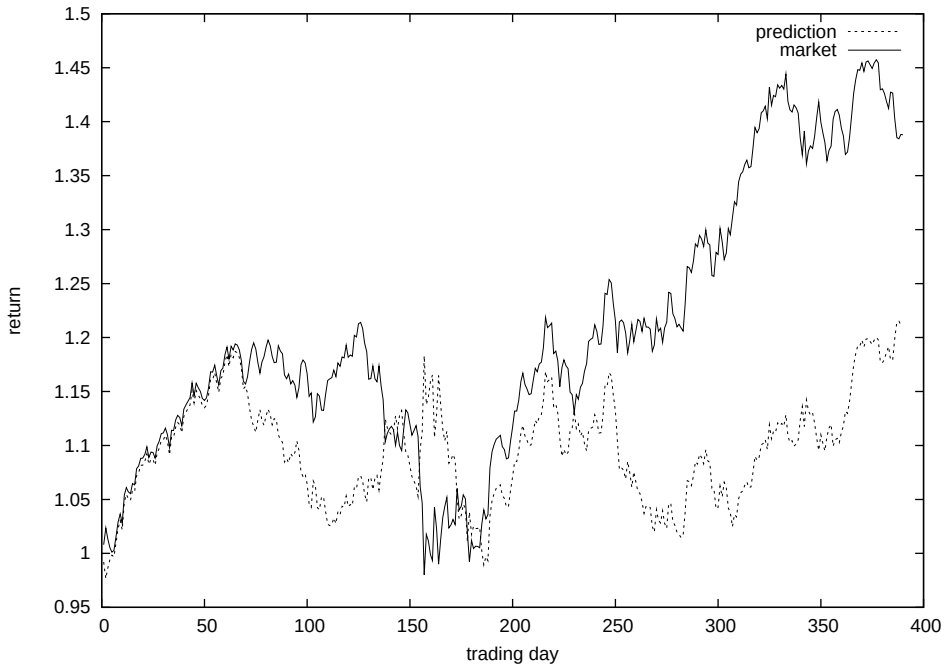

Figure 5.5: Prediction of the 400-day S&P 500 Index starting from Jan 2, 1998. Training window size = 10

We test all the above mentioned approaches in chapter 4 with 400 days of the S&P 500 Index starting from Jan 2, 1998.

First take a look at the HMGM with Rabiner's arithmetic average update rule. From $t213$ to $t228$, the actual market had an apparent reverse in direction, from a decline to an

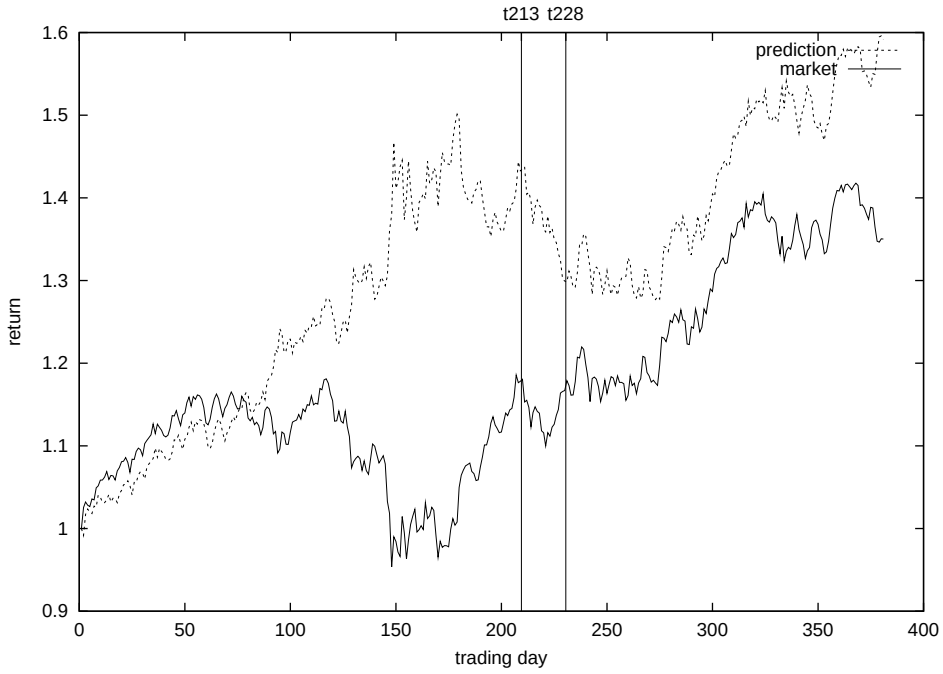

Figure 5.6: Prediction of the 400-day S&P 500 Index starting from Jan 2, 1998. Training window size = 20

increase. The simple HMGM was not able to recognize this in a short time, which costs a loss in profit. But the overall cumulative profit at the end of the testing period is still 20% above the market return.

The multivariate HMGM takes a vector of {S&P daily return, Dow Jones daily return} as input. We do not know the exact relationship between those two financial time series, but the multivariate HMM uses the EM algorithm to find the best parameters that maximizes the probability of seeing both series data. The multivariate HMGM model generates similar cumulative return as the single series HMGM but has less volatility at the later part of the testing data.

The exponentially weighted EM algorithm performs significantly better than the first two models, mainly because it picks up a reverse trend much faster. During the periods of $t164$ to $t178$ and $t213$ to $t228$, the trend reversals were recognized and we gain points here, so that the later cumulative gains can be based on a higher early gain. However, it also loses points during periods where the volatility is higher, such as the period after $t228$. At the end of the whole period, a bull market started and the model responded quickly to this

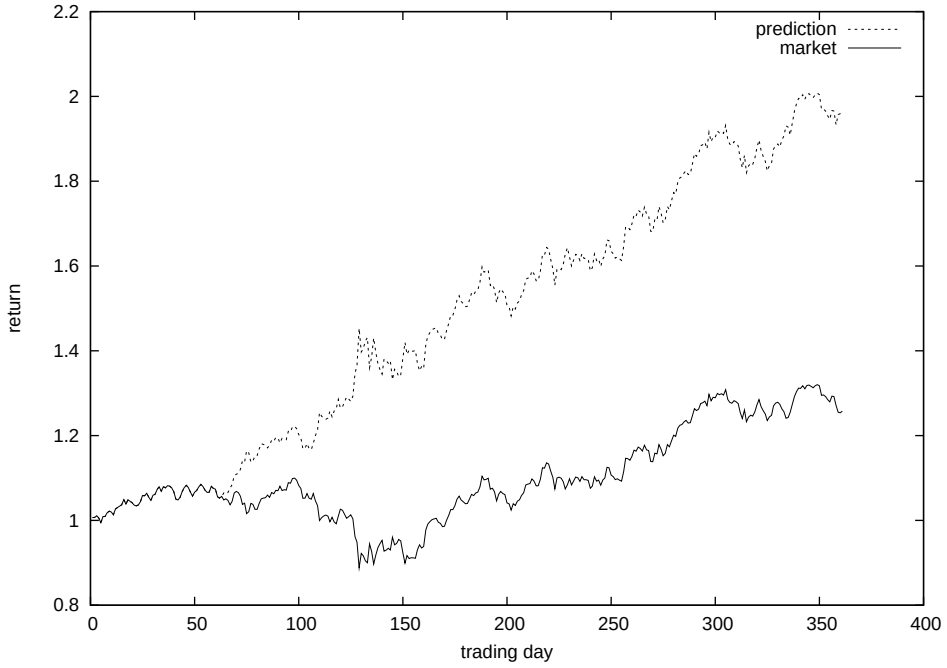

Figure 5.7: Prediction of the 400-day S&P 500 Index starting from Jan 2, 1998. Training window size = 40

signal and made no mistake during the long bull. We made nearly twice as much at the end of the 400 days.

Then we tested the double weighted EM algorithm. There isn't quite much difference from the single exponentially weighted EM algorithm. This is partly because the S&P 500 and the Dow give identical direction signals most of the time during the testing period. But a small difference at the end of the 400 days gives the double weight algorithm a 2% better performance.

Finally we produce an all-in-one picture with all the prediction results from different models for comparison. As seen from the picture, the two weighted HMGM performs convincingly better than the traditional EM algorithm.

5.4 Test on Data Sets from Different Time Periods

To show the results generated by the HMGM are not random and can repeatedly beat the *buy-and-hold* strategy, we test the model on data sets from different time periods. The data tested are four segments of the S&P 500 Index: November-2-1994 to June-3-1996,

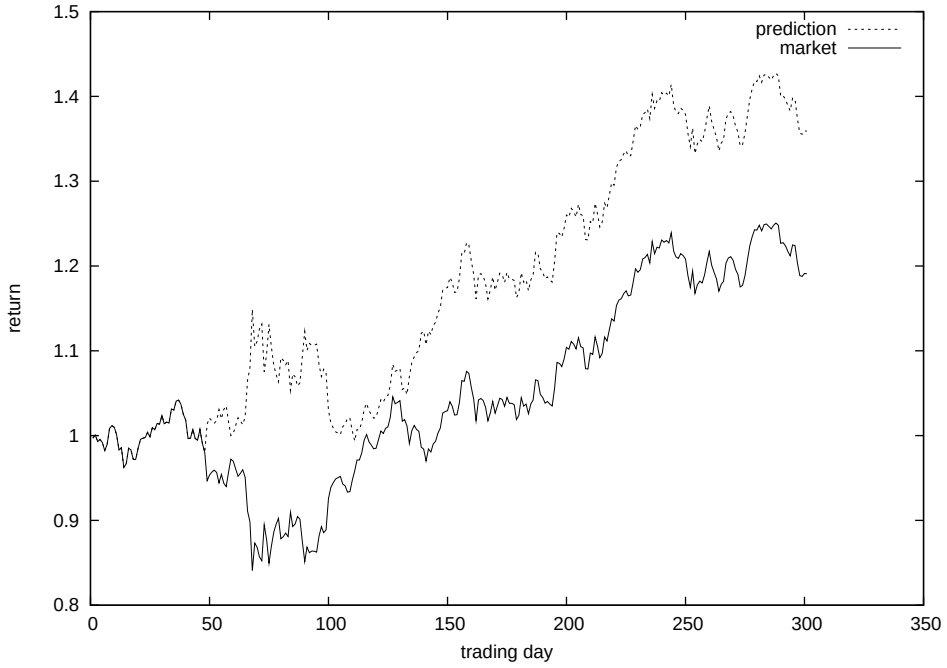

Figure 5.8: Prediction of the 400-day S&P 500 Index starting from Jan 2, 1998. Training window size = 100

June-4-1996 to December-31-1998, August-5-1999 to March-7-2001 and March-8-2001 to October-11-2002 (the period January-2-1998 to August-4-1999 has been tested before), The first two segments are the index in the bull market cycle while the latter two segments are the periods of a bear market. Experiments show that HMGM performs equally well under both situations.

5.5 Comparing the Sharpe Ratio with the Top Five S&P 500 Mutual Funds

The *Sharpe Ratio*, also called the *reward-to-variability ratio*, is a measure of the risk-adjusted return of an investment. It was derived by W. Sharpe [Sha66]. Sharpe argues that the mean and variance by themselves are not sufficient to measure the performance of a mutual fund. Therefore he combined the two variable into one and developed the Sharpe Ratio.

The definition of the Sharpe Ratio is quite straightforward: Let R_{Ft} be the return on the fund in period t, R_{Bt} the return on the benchmark portfolio or security in period t, and

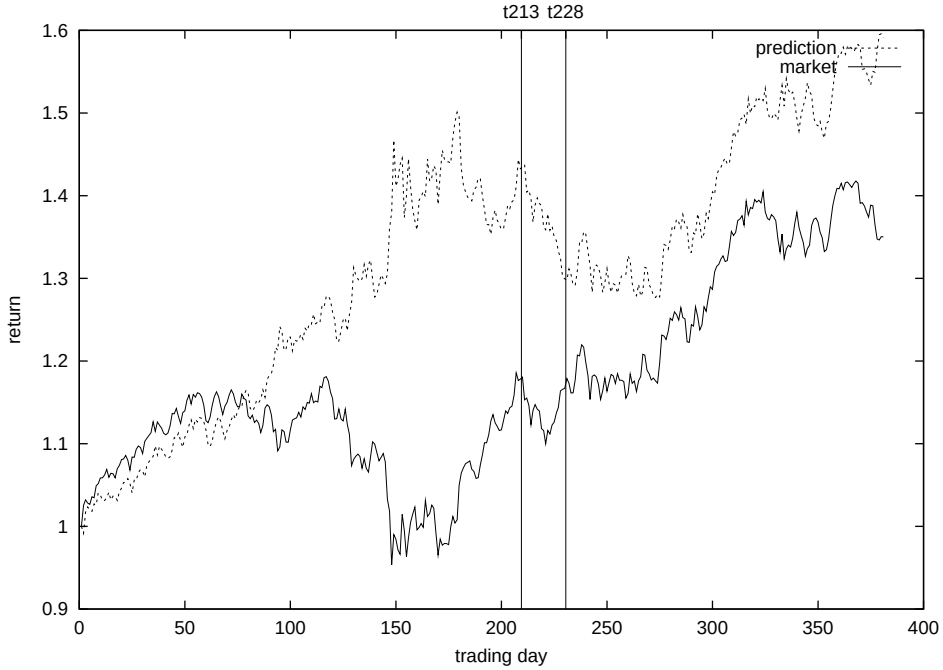

Figure 5.9: Prediction with HMGM for 400 days of the S&P 500 Index. Note from $t213$ to $t228$, the HMM was not able to catch up with the change in trend swiftly.

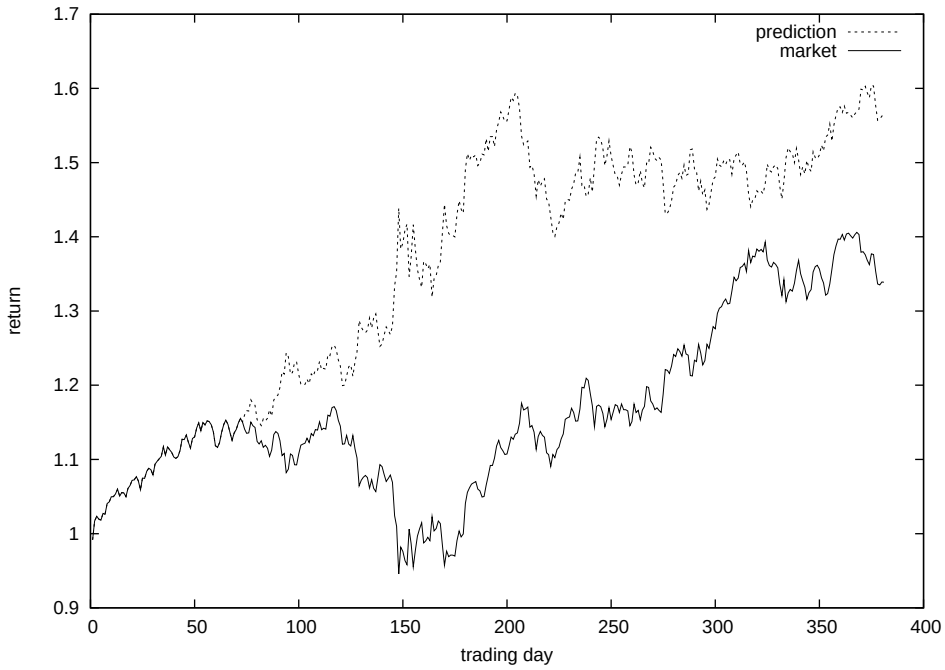

Figure 5.10: Prediction with multivariate HMGM for 400 days of the S&P 500 Index.

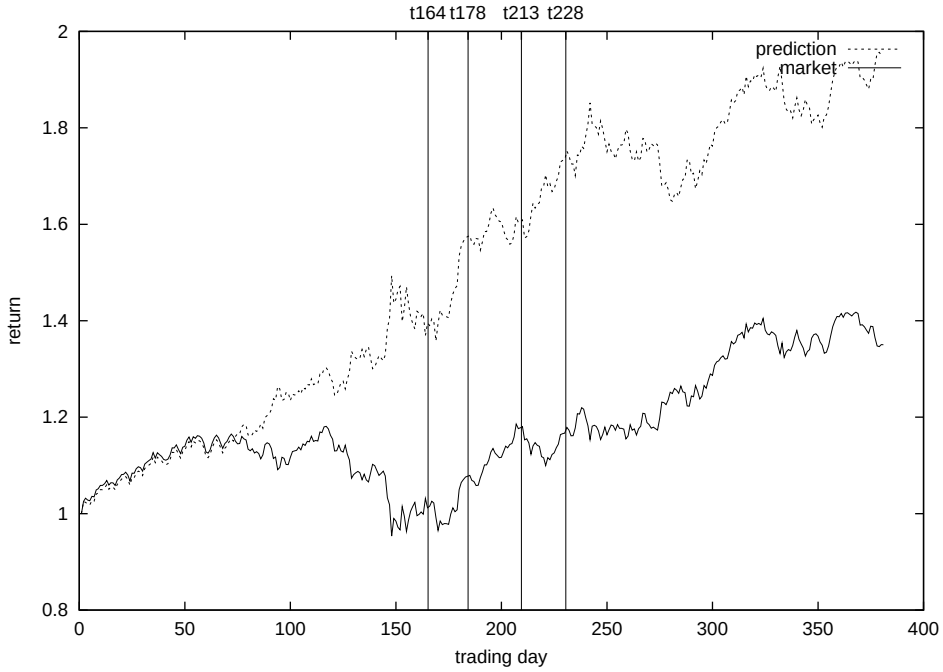

Figure 5.11: Prediction with single exponentially weighted HMGM

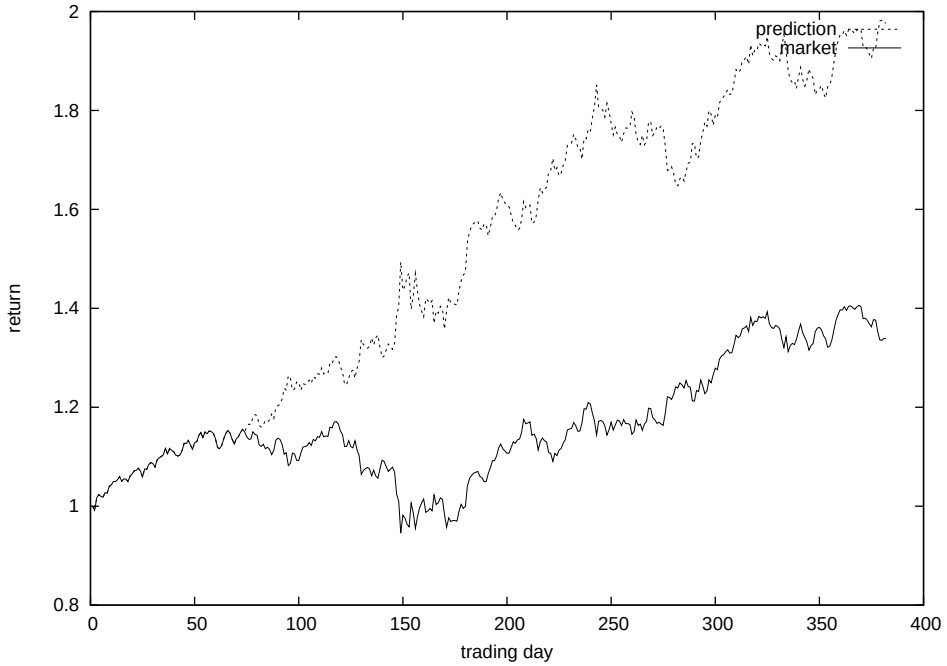

Figure 5.12: Prediction with double weighted HMGM that has a confidence adjustment variable.

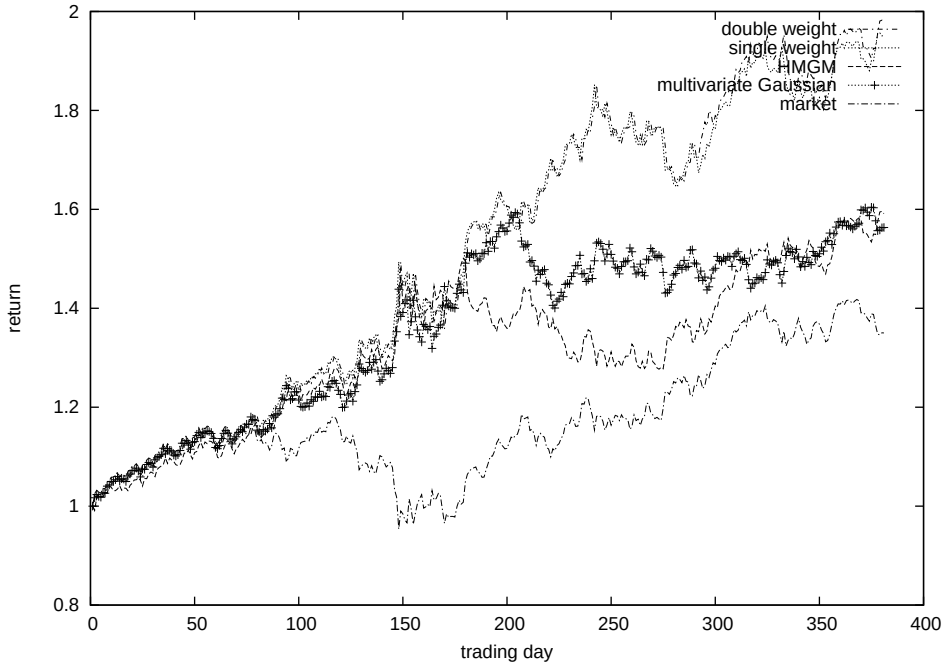

Figure 5.13: Results from all models for comparison.

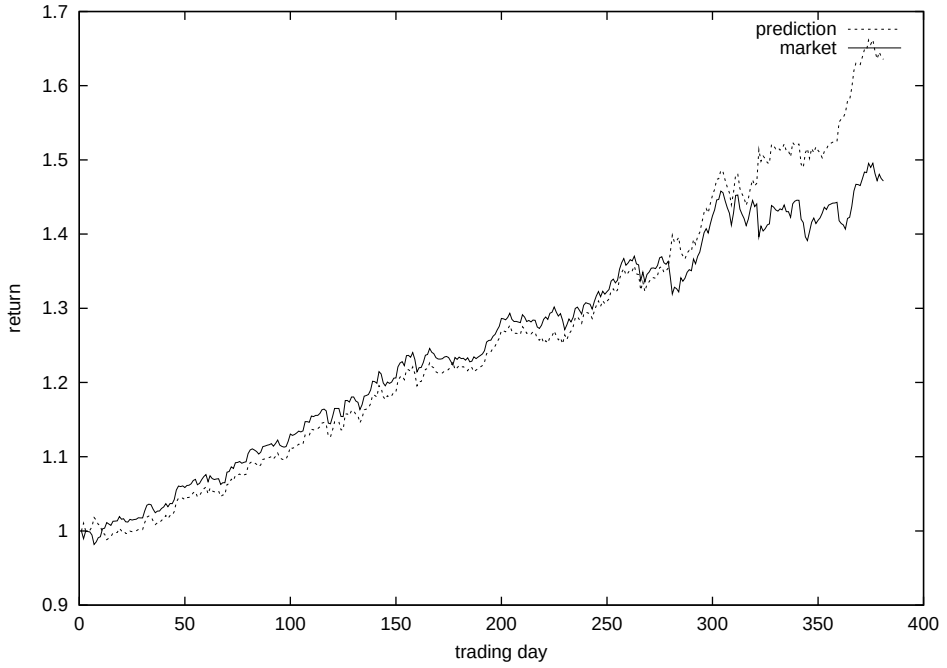

Figure 5.14: Results on the period from November 2, 1994 to June 3, 1996

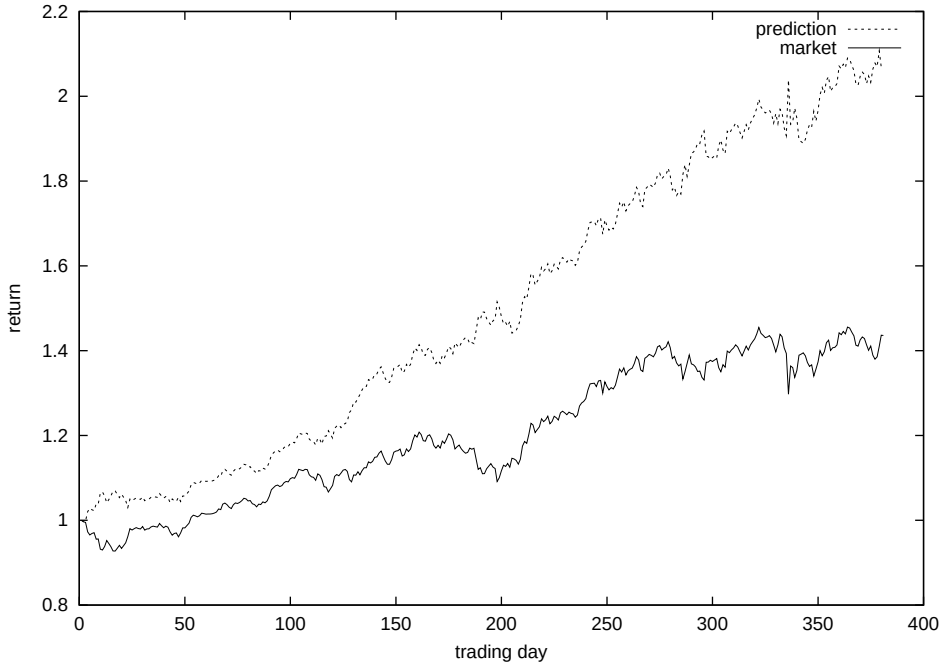

Figure 5.15: Results on the period from June 4, 1996 to December 31, 1998

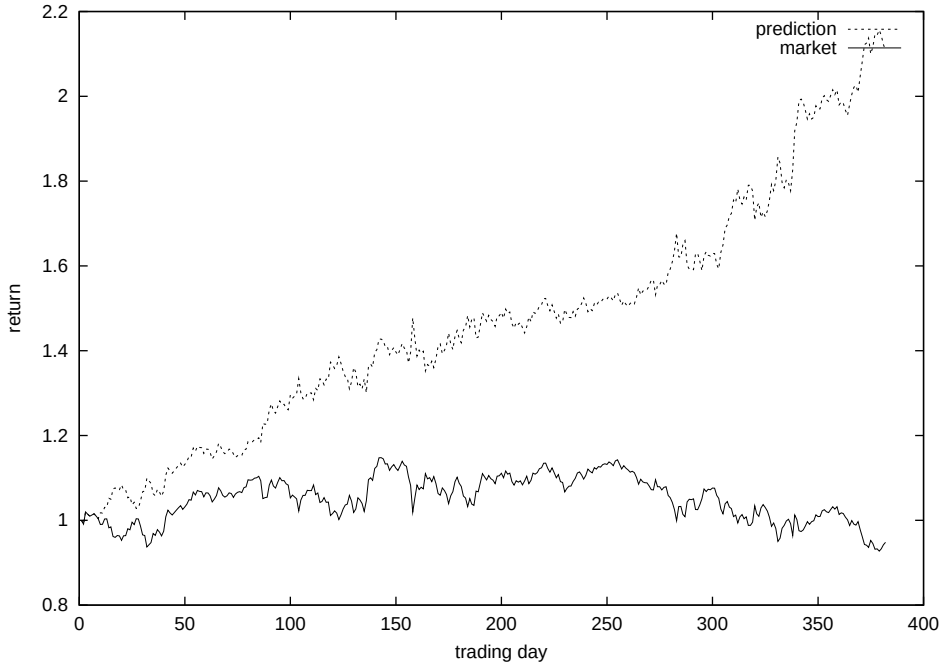

Figure 5.16: Results on the period from August 5, 1999 to March 7, 2001

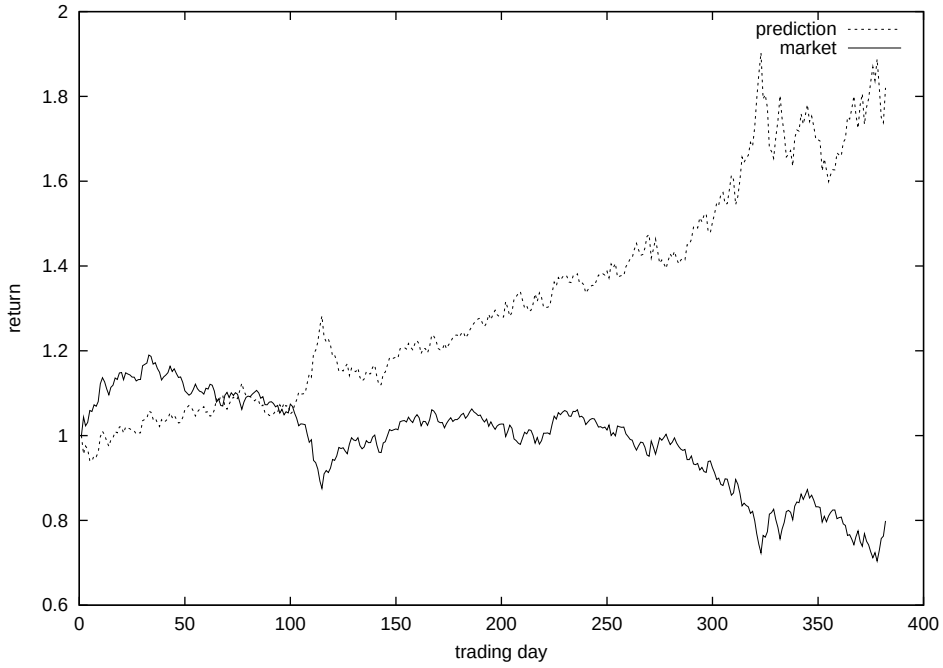

Figure 5.17: Results on the period from March 8, 2001 to October 11, 2002

D_t the differential return in period t:

$$D_t = R_{Ft} - R_{Bt} \tag{5.2}$$

Let $\bar{D}$ be the average value of Dt over the historic period from $t = 1$ through T:

$$\bar{D} = \frac{1}{T} \sum_{t=1}^{T} D_t \tag{5.3}$$

and σ_D be the standard deviation over the period:

$$\sigma_D = \sqrt{\frac{\sum_{t=1}^{T} (D_t - \bar{D})^2}{T}} \tag{5.4}$$

The Sharpe Ratio (S_h) is:

$$S_h = \frac{\bar{D}}{\sigma_D} \tag{5.5}$$

As we can see, the ratio indicates the average differential return per unit of historic variability of the differential return. Using the Sharpe Ratio, we can compare the return of

Symbol	Explanation
BTIIX	-0.02948
FSMKX	0.011475
SPFIX	-0.000502
SWPPX	0.024323
VINIX	-0.00142
HMGM	0.084145

Table 5.2: Comparison of the Sharpe Ratio between the HMM and the top 5 S&P 500 funds during the period of August 5, 1999 to March 7, 2001. Four of the five funds have negative Sharpe Ratios. The absolute value of the positive one is much smaller than the HMGM prediction.

Symbol	Explanation
BTIIX	-0.05465
FSMKX	-0.003319
SPFIX	-0.01856
SWPPX	-0.01726
VINIX	0.001446
HMGM	0.098967

Table 5.3: Comparison of the Sharpe Ratios during the period of March 8, 2001 to October 11, 2002. The Double weighted HMGM consistently outperforms the top five S&P 500 mutual funds.

different strategies at the same risk. This is the meaning of *beat the market* mentioned in chapter 1.

The Sharpe Ratio can be used for significance testing for financial time series analysis and prediction. This is because the *t-statistic* equals the Sharpe Ratio times the square root of the total number of testing units. The Sharpe Ratio is thus proportional to the $t - statistic$ of the means.

We now compare the top 5 S&P 500 mutual funds that performed best over the past three years with the HMGM model we proposed. The HMGM outperforms all of them. 4 of the 5 funds have negative Sharpe Ratios while the positive one is much smaller than the HMGM.

5.6 Comparisons with Previous Work

Hidden Markov Model, as a sequence learning model, has been applied to many fields including natural language processing, bioinformatics, audio and video signals processing (see 2.4). But there have been few applications on the application of HMMs to financial time series analysis and prediction. One reason is it is usually hard to find functions associated to the states that characterizes the behavior of the time series. This is an empirical research. For example, in [MCG04], they use a binomial density function at each state to represent the risk of default.

Second reason lies in the EM algorithm. Financial time series differs from other sequence data in that they are time dependent. In video/audio/language/bio info signals processing, this dependence is not important, or the data must be treated of equal importance. Patterns in financial time series tend to change over time, as we have shown in the *super five day pattern* and the *January Effect* cases.

Bengio [BF96] introduced an Input-Output Hidden Markov Model (IOHMM) and applies it to the Electro-encephalographic signal rhythms [CB04]. Their model enables the HMM to generate the output probability based on the current state as well as the current input data sample from the input sequence. But they still don't realize the importance of time. The input sequence is just another sequence data.

In the two weighted EM algorithms we propose in the thesis, the input sequence is a time dependent value that changes from time to time. The state parameters in the re-estimation formulas all have actual meanings. They represent the probabilities of a variety of situations happening in the HMM. Adding the time-dependent weights incorporates temporal information into the training process of the HMM. Further, the double weighted EM algorithm enables self-adjustment of the weights. These techniques enrich the representation power of the HMM in dealing with financial time series data.

We now compare our experimental results with Shi and Weigend's work [SW97]. As described before, they used three neural networks (called *experts*) as the emission functions at each state. So there are three states in all. Each expert has its own expertise: the first one is responsible for low volatility regions, the second one is good at high volatility regions, and the third expert acts as a collector for the outliers.

They test on 8 years of daily S&P 500 data from 1987 to 1994. Their return over the 8 year period is 120%. In our experiments, each of the five 400-day testing periods generates

a return close to 100%.

However the main problem of their prediction is not the rate of return. As we can see from the profit curve (Figure 5.6), the profit curve has a similar shape as the index curve. This means that there are too many positive signals generated by the prediction system. In other words, many times they are taking a long position. The profit of the prediction thus follows the index behavior to a large extent.

Figure 5.18: Too many positive signals result in a curve similar to the index.

In all our five 400-day periods, the profit curves based on the prediction have a different shape than the index curve. This reflects a more sophisticated strategy that generates dynamically changing signals.

Chapter 6

Conclusion and Future Work

This chapter concludes the thesis. We show our findings through the experiments and indicate possible improvements to the tested model.

6.1 Conclusion

Our experiments can be sorted into three categories: testing of HMGM with different training window sizes; testing of the weighted EM algorithm in comparison with the classic EM algorithm; the use of supporting time series to help predict the targeting series.

Comparing with neural network experts, the parameters of the HMGM as emission density functions can be clearly derived from the EM algorithm. By using a mixture of Gaussian density functions at each state, the emission functions provide a broad range of predictions.

Results show that there is no incorrect bias as shown in the neural network experiments. The HMGM performs equally good in both bull market and bear market. Although at times where the financial time series is volatile, it will make more mistakes, HMGM can eventually catch up with the trend. In a 20-day experiment the cumulative rate of return is 20% above the actual market return over the 400 days of training and testing period.

We introduce an important concept in the model design, the *training window*. In fact, the training window is used in both training and testing as these two processes are combined together throughout the experiment. It is an empirical study what the size of the training window should be. Experimental results show that a 40-day training window generates the best prediction. The accuracy of prediction decreases as the size of the training window

changes in either direction. This means to get the best prediction, we have to balance between short-term sensitivity and long-term memory.

Noticing that recent data should be given more emphasis than older data from the past, we revise the general EM algorithm to a *exponentially weighted EM algorithm*. This algorithm is motivated by the *exponential moving average* technique used widely in financial engineering. The weighted EM algorithm shows an increased sensitivity to recent data. However during highly sensitive periods, the model might fall in the pitfall of frequently changing *whipsaws*, i.e., buy and short signal reverses in very a very short time. To overcome this we add a second weight to the exponential weight. The second weight which is calculated by the divergence of the S&P 500 and the Dow Jones Index, represents the market volatility. In a 20-day training window testing within the same training data, the revised model generates more stable predictions and a higher rate of return. The end of the 400-day training and testing period generates a return of 195%.

6.2 Future Work

Rather than the predefined 20-day training window, the size of the training pool and training window could be decided by a hill climbing or dynamic programming approach. Identifying useful side information for prediction, such as developing of new indicators will help making better predictions. Then we can extend the prediction to a form of density function rather than simply values. This way can we output richer information about the market movements.

Functions and how functions are mixed at each state are not studied in this thesis. This can be a valuable work to do. Gaussian distribution is the most popular distribution when modeling random or unknown processes. But it has been observed that many time series may not follow the Gaussian distribution.

In this thesis, we discussed the application of the Hidden Markov Model to one single financial time series, more specifically, we studied the S&P 500 Composite Index. We didn't take the portfolio selection problem into consideration. Since different financial time series have different characteristics, a portfolio composed of different stocks will have different rate of returns and risk levels. In other words, they will have different patterns. It will be interesting to apply the HMM technique to a portfolio of financial assets and study the time series generated by these groups of assets. The Viterbi algorithm could play an important

role in this application. If we put functions that are estimated to best characterize different assets at different states, the path of the Viterbi algorithm will be able to tell what assets should be included in the portfolio, observing the price movements. [Cov91] offers a nice introduction to universal portfolio selection problem.

The double weights in the HMGM model can also be trained. These weights decide the sensitivity of the model, so by setting up a function that studies the relationship between sensitivity and accuracy, we can expect to find better weights.

The outputs of the HMGM models we present in the thesis are positive and negative symbols. A more delicate model will be able to output densities rather than symbols, i.e., by how much probability the market will go up or down. Furthermore, the density can be designed to describe the probability of falling into a range of values.

Appendix A

Terminology List

outlier Outliers are a few observations that are not well fitted by the *best* available model. Usually if there is no doubt about the accuracy or veracity of the observation, then it should be removed, and the model should be refitted.

whipsaw A whipsaw occurs when a buy or sell signal is reversed in a short time. Volatile markets and sensitive indicators can cause whipsaws.

trend Refers to the direction of prices. Rising peaks and troughs constitute an uptrend; falling peaks and troughs constitute a downtrend. A trading range is characterized by horizontal peaks and troughs.

volatility Volatility is a measurement of change in price over a given period. It is usually expressed as a percentage and computed as the annualized standard deviation of the percentage change in daily price.

risk Same meaning as volatility. The more volatile a stock or market, the more risky the stock is, and the more money an investor can gain (or lose) in a short time.

indicator An indicator is a value, usually derived from a stock's price or volume, that an investor can use to try to anticipate future price movements.

Sharpe Ratio Also called the *reward-to-risk ratio*, equals the potential reward divided by the potential risk of a position.

Exponential Moving Average A moving average (MA) is an average of data for a certain number of time periods. It *moves* because for each calculation, we use the latest x

number of time periods' data. By definition, a moving average lags the market. An exponentially smoothed moving average (EMA) gives greater weight to the more recent data, in an attempt to reduce the lag.

profit Profit is the cumulative rate of return of a financial time series. To calculate today's cumulative return, multiply today's actual gain with yesterday's cumulative return.

short To short means to sell something that you don't have. To do this, you have to borrow the asset that you want to short. The reason to sell short is that you believe the price of that asset will go down.

Bibliography

[ADB98] A. S. Weigend A. D. Back. What Drives Stock Returns?-An Independent Component Analysis. In *Proceedings of the IEEE/IAFE/INFORMS 1998 Conference on Computational Intelligence for Financial Engineering*, pages 141–156. IEEE, New York, 1998.

[APM02] M. Bicego A. Panuccio and V. Murino. A Hidden Markov Model-Based Approach to Sequential Data Clustering . In *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops SSPR 2002 and SPR 2002, Windsor, Ontario, Canada, August 6-9, 2002, Proceedings*, volume 2396 of *Lecture Notes in Computer Science*, pages 734–742. Springer, 2002.

[BF96] Y. Bengio and P. Frasconi. Input-Output HMM's for Sequence Processing. *IEEE Transaction on Neural Networks*, 7(5):1231–1249, 1996.

[Bil97] J. A. Bilmes. *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*. TR-97-021. International Computer Science Institute, UC Berkeley, 1997.

[Boy83] R. A. Boyles. On the Convergence Properties of the EM Algorithm. *Journal of the Royal Statistical Society*, Ser B 44:47–50, 1983.

[BP66] L. E. Baum and T. Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Annals of Mathematical Statistics*, 37:1559–1563, 1966.

[BS73] F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81:637, 1973.

[BWT85] De Bondt, F. M. Werner., and R. H. Thaler. Does the market overreact? *Journal of Finance*, 40:793–805, 1985.

[CB04] S. Chiappa and S. Bengio. HMM and IOHMM modeling of EEG rhythms for asynchronous BCI systems. *European Symposium on Artificial Neural Networks, ESANN*, 2004.

[Cha93] E. Charniak. *Statistical Language Learning.* MIT Press, Cambridge, Massachusetts, 1993.

[Chi95] G. Chierchia. *Dynamics of Meaning: Anaphora, Presupposition and the Theory of Grammar.* The University of Chicago Press: Chicago, London, 1995.

[Cov91] T. Cover. Universal Portfolios. *Mathematical Finance*, 1(1-29), 1991.

[CWL96] W. Cheng, W. Wagner, and C.-H Lin. Forecasting the 30-year U.S. treasury bond with a system of neural networks. *NeuroVe$t Journal*, 1996.

[Deb94] G. Deboeck. *Trading on the edge : neural, genetic, and fuzzy systems for chaotic financial markets.* Wiley, New York, 1994.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Ser B 39:1–38, 1977.

[Dor96] G. Dorffner. Neural Networks for Time Series Processing. *Neural Network World*, pages 447–468, 1996.

[Fam65] E. F. Fama. The Behaviour of Stock Market Prices. *Journal of Business*, 38:34–105, 1965.

[Fam70] E.F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25:383–417, 1970.

[FST98] S. Fine, Y. Singer, and N. Tishby. The hierarchical Hidden Markov Model: analysis and applications. *Machine Learning*, 32, 1998.

[Fur01] S. Furui. *Digital speech processing, synthesis, and recognition.* Marcel Dekker, New York, 2001.

[GI00] I. S. Gradshteyn and I.M.Ryzhik. *Table of Integrals, Series, and Products.* Academic Press, San Diego, CA, 2000.

[GLT01] C. Lee Giles, S. Lawrence, and A.C. Tsoi. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. *Machine Learning*, 44:161–183, 2001.

[GP86] C. W. J. Granger and P.Newbold. *Forecasting Economic Time Series.* Academic Press, San Diego, 1986.

[Gra86] B. Graham. *The Intelligent Investor: The Classic Bestseller on Value Investing.* HarperBusiness, 1986.

[GS00] X. Ge and P. Smyth. Deformable Markov model templates for time-series pattern matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* , pages 81–90. ACM Press New York, NY, USA, 2000.

[HG94] J. P. Hughes and P. Guttorp. Incorporating spatial dependence and atmospheric data in a model of precipitation. In *Journal of Applied Meteorology*, volume 33(12), pages 1503–1515. IEEE, 1994.

[HK01] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, San Francisco, 2001.

[HLP88] G. H. Hardy, J. E. Littlewood, and G. Plya. *Inequalities*. Cambridge University Press, Cambridge, England, 1988.

[Hul00] J. Hull. *Futures, Options and Other Derivatives*. Prentice-Hall, Upper Saddle River, NJ, 2000.

[Jel97] F. Jelinek. *Statistical methods for speech recognition*. MIT Press, Cambridge, Mass.; London, 1997.

[JP98] T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the cem algorithm. *Advances in Neural Information Processing Systems*, 10, 1998.

[MCG04] M. Davis M. Crowder and G. Giampieri. A Hidden Markov Model of Default Interaction. In *Second International Conference on Credit Risk*, 2004.

[Mit97] T. Mitchell. *Machine Learning*. McGraw Hill, New York ; London, 1997.

[MS99] C. D. Manning and H. Schütze. Chapter 9: Markov Models. In *Foundations of Statistical Natural Language Processing*, Papers in Textlinguistics, pages 317–379. The MIT Press, 1999.

[Nic99] A. Nicholas. Hidden Markov Mixtures of Experts for Prediction of Switching Dynamics. In E. Wilson Y.-H. Hu, J. Larsen and S. Douglas, editors, *Proceedings of Neural Networks for Signal Processing IX: NNSP'99*, pages 195–204. IEEE Publishing, 1999.

[PBM94] T. Hunkapiller P. Baldi, Y. Chauvin and M.A. McClure. Hidden Markov Models of Biological Primary Sequence Infromation. In *Proceedings of National Academy of Science USA* , volume 91(3), pages 1059–1063. IEEE, 1994.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of IEEE*, volume 77, pages 257–286, 1989.

[RIADC02] B. C. Lovell R. I. A. Davis and T. Caelli. Improved Estimation of Hidden Markov Model Parameters from Multiple Observation Sequences. In *16th International Conference on Pattern Recognition (ICPR'02) Quebec City, QC, Canada* , volume 2, page 20168, 2002.

[RJ86] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Mag.*, June:4–16, 1986.

[Ros03] S. M. Ross. *An Introduction to Mathematical Finance.* Cambridge University Press, Cambridge, New York, 2003.

[Row00] S. Roweis. Constrained Hidden Markov Models. *Advances in Neural Information Processing Systems*, 12, 2000.

[RW84] R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2), 1984.

[RY98] E. Ristad and P. Yianilos. Towards EM-style Algorithms for a posteriori Optimization of Normal Mixtures. *IEEE International Symposium on Information Theory*, 1998.

[Sha49] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[Sha66] W. F. Sharpe. Mutual Fund Performance. *Journal of Business*, 39:119–138, 1966.

[STW99] R. Sullivan, A. Timmermann, and H. White. Data-snooping, technical trading rule performance and the bootstrap. *Journal of Finance*, 54:1647–1691, 1999.

[SW97] S. Shi and A. S. Weigend. Taking Time Seriously: Hidden Markov Experts Applied to Financial Engineering. In *Proceedings of the IEEE/IAFE 1997 Conference on Computational Intelligence for Financial Engineering* , pages 244–252. IEEE, 1997.

[Tha87] R. Thaler. Anomalier: The January Effect. *Journal of Economics Perspectives*, 1:197–201, 1987.

[Wac42] S. Wachtel. Certain Observations on Seasonal Movements in Stock Prices. *Journal of Business*, 15:184–193, 1942.

[WG93] A. S. Weigend and N.A. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the past.* Addison-Wesley, 1993.

[WMS95] A. S. Weigend, M. Mangeas, and A. N. Srivastava. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. In *International Journal of Neural Systems*, volume 6, pages 373–399, 1995.

[Wu83] C. F. J. Wu. On the Convergence Properties of the EM Algorithm. *The Annals of Statistics*, 11:95–103, 1983.

[XJ96] L. Xu and M. Jordan. On Convergence Properties of the EM Algorithm for Gaussian Mixtures. *Neural Computation*, 8(1):129–151, 1996.

[XLP00] M. Parizeau X. Li and R. Plamondon. Training Hidden Markov Models with Multiple Observations-A Combinatorial Method. In *IEEE Transaction on Pattern Analysis and Machine Intelligence*, volume 22(4), pages 371–377. IEEE pressing, 2000.

[ZG02] S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification. *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2:1254–1159, 2002.