Financial Applications of Hidden Markov Models

A thesis submitted in partial fulfilment of the requirements for the degree of

Bachelor of Arts in Theoretical Physics
Trinity College Dublin

March 10, 2023

Author:
Harold Hodgins
19335074

Supervisor:
Prof. Dmitri Zaitsev

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Plagiarism Declaration

I hereby declare that this thesis is my own work where appropriate citations have been given and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `https://libguides.tcd.ie/plagiarism/ready-steady-write`.

Signed:Harold Hodgins Date:March 10, 2023 .

Acknowledgments

I would like to acknowledge my parents, for their constant faith and reassurance during my times of self-doubt.

Thank you to my supervisor Dmitri Zaitsev, who introduced me to hidden Markov models and granted me free will to study topics which personally interested me.

Lastly, thank you to all the theoretical physics students who I have had the pleasure of studying alongside for the past four years.

Abstract

Financial markets often change abruptly due to reasons such as government policy, regulatory environment and other macroeconomic effects. The periods in which these changes occur are known as market regimes. It is necessary to be able to detect and categorise these regimes in order to create an optimal trading strategy. One such method for regime detection is a *hidden Markov model (HMM)*.

A HMM is a statistical model which can be trained on observable data to make inferences about underlying, hidden states from which the data was generated. Using the model, the likelihood of observing a sequence of events can be computed which can be used in making predictions and thus improve decision making. This makes the marriage between HMMs and financial markets very natural.

In this study the mathematical foundations of HMMs as well as the core problems associated with them were discussed. This involved discussions of various dynamical programming algorithms.

Financial applications were investigated using the solutions to these problems. A HMM was trained on historical stock data from Medtronic (MDT), from which the daily close prices and log-returns of the stock were predicted. It was also investigated how HMMs can be used for regime identification and specification in financial markets.

Contents

1 Background

To give some context as to why this project studies *hidden* Markov models specifically, we consider the history of hidden Markov models in financial settings.

Renaissance Technologies is arguably one of the most successful hedge funds in history. Founded by American mathematician Jim Simons in 1982, many of its early employees were brought from IBM's speech recognition AI division, a field in which Hidden Markov Models (HMMs) were famously used. They also hired numerous candidates from cryptography and signal processing backgrounds. In fact, Leonard E. Baum, responsible for the Baum-Welch algorithm for HMMs, worked alongside Simons at the fund 'Monemetrics', which was a predecessor of Renaissance Technologies [1]. Sharing information and techniques with others is often looked down upon in trading and it is therefore hard to be certain what exact methods were used but given the above information it is certainly reasonable to assume that HMMs were involved in Renaissance Technologies' trading strategies in their early operations.

The basic theory of HMMs was published in a series of papers by Baum and his colleagues in the late 1960s and early 1970s [2]. Broadly speaking, A HMM is a *Markov process* split into two components; an observable component and a hidden component [3]. We begin our study with some background theory on stochastic processes and Markov models before extending our discussion to HMMs.

2 Markov Processes

2.1 Stochastic Processes

Definition 2.1 (Stochastic Process). A variable whose value changes over time in an uncertain manner is said to follow a *stochastic process*. Mathematically, a stochastic process is best described as a collection of random variables $\{X(t), t \in T\}$ where T is called the index set of the process, meaning for each value t in T, $X(t)$ is an observed value of a random variable [4].

A simple example of a stochastic process is coin tosses. In this case there are two random variables: head or tails. We therefore define the system to have two *states* corresponding to each random variable, where the probability of being in either state is 0.5. We can describe the process visually with the use of a *state diagram*:

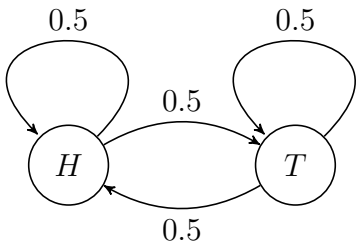

Figure 1: Stochastic process state diagram - Coin toss example.

At evenly spaced discrete times, the system undergoes a change of state, which can include returning to the same state. Generally, to get a full probabilistic description (i.e. to be able to predict future outcomes) of the system, it would be necessary to have information on the current state and *all* previous states. This is where we introduce a specific type of stochastic process known as a Markov process, where only specification of the current and previous state is required.

2.2 Markov Processes

The memorlyess nature of Markov processes with respect to their current state is known as the *Markov property*.

Definition 2.2 (**Markov Property**). Consider a stochastic process which involves a system which can appear in a given state within a state space S at each discrete time instant. Suppose T is a countable set thought of as time, (i.e the positive integers). For each $t \in T$, let X_t be a random variable that gives the state of the process at time t. The *Markov property* is defined as,

$$P(X_t = q_t | X_{t-1} = q_{t-1}, \dots X_1 = q_1, X_0 = q_0) = P(X_t = q_t | X_{t-1} = q_{t-1})$$

for any of the states $q_i \in S$, $i = 0, 1, \dots, t-1, t$ [5].

In simple terms, the state the system is in at the next step of the process depends only on whatever state the system is in at the current step, and is unaffected by any other prior history [5]. The above expression contains conditional probabilities. If we have a probability $P(A|B)$ this is the probability for the event A to occur given that the event B has happened before.

With this we consider the following definition of a Markov process:

Definition 2.3 (**Markov Process**). Suppose a stochastic process involves a system which can appear in a given state at discrete time instants. This process is known as a *Markov process* if it obeys the Markov property. That is, the probability of the process being in a given state depends only on the state the process was in at the previous instant.

This study will only consider discrete-time Markov processes as we have described above. We can think of such a process as running according to a clock; that is, at each tick of the clock the process changes to a given state in the state space. Which state it changes to is determined by a set of conditional probabilities known as *state transition probabilities* [5]- [6].

2.3 State Transition Probabilities

Recall the previous example of a coin toss we used to understand what a stochastic process is. In Figure 1 we observe that the probability of transitioning between any of the possible states (i.e. heads to tails) is always 0.5. In general, there may be a unique probability of transitioning between each state of a system - these probabilities form a set called the *state transition probabilities*.

Definition 2.4 (State transition probabilities). Suppose a stochastic process involves a system which can appear in a given state in a state space S at discrete time instants. In the case that the process obeys the Markov property, and thus is a Markov process, the probability of the process moving between states S_i and S_j is given by:

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$$

Where q_t is the actual state of the system at time t.

These probabilities completely describe the Markov process. If a Markov process describes a system containing N states labelled $S_1, S_2, \ldots S_N$, they have the following properties [6]:

- The process changes state at each time instant i.e. $a_{ij} \geq 0$
- The set of transition probabilities form an $N \times N$ matrix $A = \{a_{ij}\}$ called the *transition matrix* with $1 \leq i, j \leq N$.
- The transition matrix is row stochastic, that is $\sum_{j=1}^{N} a_{ij} = 1$

We note from these properties that the example of the coin toss is in fact a Markov process. Specifically, we can refer to this as an *observable Markov model*. If we are to 'pause' the process at a given time instant we will refer to what state the process is at as the 'output' of the model. In this case, the outputs are a set of physical or *observable* events at each instant of time. The set of these observations is called an *observation sequence*, which we will later explore in further detail.

We have now established the basics of observable Markov models which outline the main features of a Markov model. However, these are quite restrictive and not of any use to more complicated problems like that of analysing financial markets. We therefore direct our focus towards HMMs.

3 Hidden Markov Models

We extend the concept of a Markov model now such that the observation we get from the model is a probabilistic function of the state the model is in. We refer to this as a hidden Markov model [6]. This model is a 'doubly embedded stochastic process'; that is, the observation we obtain from the model arises due to a probability distribution associated with the state the model is in, and the model transitions between these states as was the case with the observable Markov model. However, the latter of these stochastic processes is *hidden* (i.e. the probability distribution which governs how the model switches between states is unknown), and therefore we cannot tell what current state the model is in.

We consider the following example to get a better understanding of the difference between an observable Markov model and a hidden Markov model.

3.1 Example

Consider a Markov process where the possible outputs are weather conditions which can be any of three options - sunny (S), cloudy (C) or rainy (R). The system can change between the states, as well as remain in the same state, governed by state transition probabilities $A = \{a_{ij}\}, i, j \in \{S, C, R\}$. We can visualise this process with the following diagram:

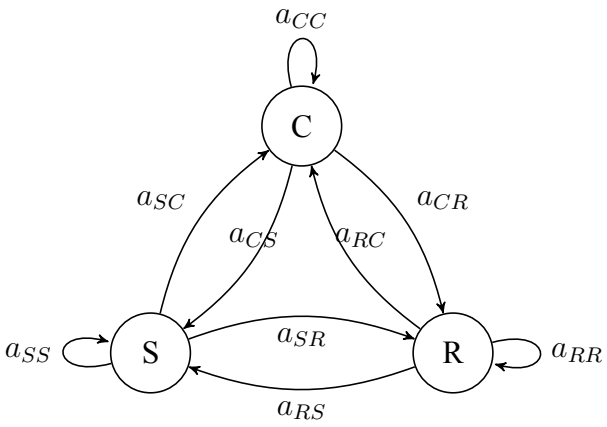

Figure 2: 3-state Markov model with state transition probabilities labelled a_{ij}.

We alter the system now by re-defining what the output of the model is, as well as adding the hidden component. Based on the weather, Oscar's mood will take one of the following states - happy or sad. Assume that on a given day, the only information we have is Oscar's mood, and that we are unaware of the weather. We can see that the weather is the hidden element of the system. If we make a note of Oscar's mood over consecutive days we get an *observation sequence*, which we will denote $\mathcal{O} = \{O_1, O_2, \ldots, O_T\}$. Where each O_i is Oscar's mood, which in general we will call an *observation symbol*.

3.2 Observation Probabilities

The observation symbols occur as a result of whatever hidden state the system is in. We can think of the hidden states of the HMM 'emitting' these symbols, according to a set of probabilities which we will refer to as *observation probabilities* or *emission* probabilities [6].

Definition 3.1 (Observation probabilities). Suppose a hidden Markov process describes a system of N states $S_1, S_2, \ldots S_N$ which it can transition between according to some unknown state transition probabilities. Let these states give rise to an observation sequence of length T, $\mathcal{O} = \{O_1, O_2, \ldots, O_T\}$. Then the probability of the hidden state S_j emitting the k^{th} observation symbol in the sequence O_k is called an *observation probability*, and is defined as [6]:

$$b_j(k) = P(O_k \text{ at time } t | q_t = S_j).$$

With, $1 \leq j \leq N, 1 \leq k \leq T$.

In simple terms, the observation probabilities are the probability of observing O_k at time t, given the model is in the state $q_t = S_j$ at time t. These probabilities are equipped with similar properties to the state transition probabilities [6]. For a hidden Markov model containing N hidden states $S_1, S_2, \ldots S_N$,

- The set of observation probabilities form an $N \times M$ matrix where M is the number of observation symbols (possible observables) $B = b_j(k)$ with $1 \leq j \leq N, 1 \leq k \leq M$.
- The matrix B is row stochastic, that is $\sum_{j=1}^{N} b_j(k) = 1$

Having described the notion of observation probabilities we now present the following diagram showing the hidden Markov process describing the previous example,

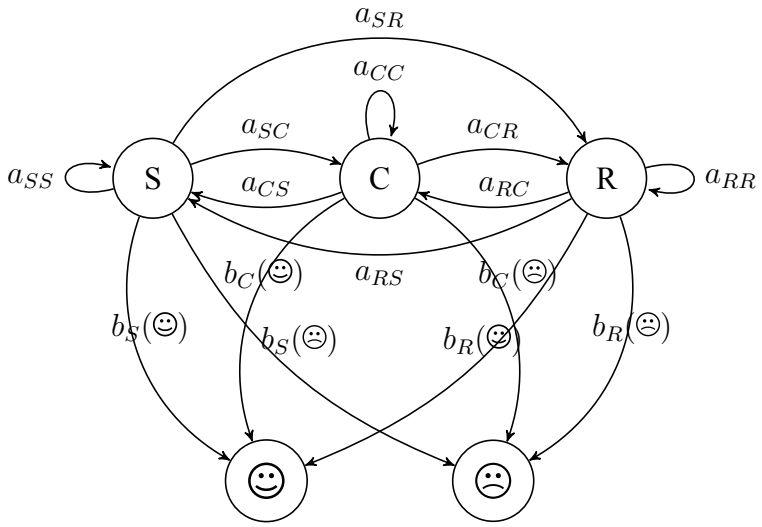

Figure 3: 3-state hidden Markov model with state transition and observation probabilities labelled $a_{ij}, b_j(k)$ respectively.

The challenge we now face is to construct a HMM which we can use to describe the behaviour of a given system. In this study we aim to construct a model which describes the behaviour of a financial market over time. Before doing this we must first consider what goes into building a HMM. In other words: what are the defining characteristics of HMMs?

3.3 Characteristics of HMMs

Hidden Markov models can be characterised by the following elements [6]:

1. N - The number of hidden states which the model can transition between.
2. M - The number of observation symbols which can be emitted from the model.
3. $A = \{a_{ij}\}$ - The state transition probability distribution.
4. $B = \{b_j(k)\}$ - The observation probability distribution.
5. $\pi = \{\pi_j\}$ - The initial state distribution - we define this as the set of the probability of being in each of the hidden states at the first time instant, i.e $t = 1$. Namely:

$$\pi_j = P(q_1 = S_j), \quad 1 \leq j \leq N$$

To summarise, a hidden Markov model - which we will call λ - is specified by the dimensions N and M, the probabilities A and B, and the initial probability distribution π. This is neatly denoted as follows:

$$\lambda = (A, B, \pi)$$

Having now specified what goes into building a HMM, we look to actually create one. It's clear that there are multiple components we must evaluate if we are to accomplish this, one of which being the transition probability distribution. This is, of course, the hidden element of the model, and we will at best only be able to estimate such a distribution so that it describes the observation sequence as closely as possible. There are other questions associated with the model we will wish to answer, the results of which we will use in financial applications later on. We will consider just three problems now, although with these we will be equipped with everything we need to begin building HMMs.

3.4 The Main Problems of HMMs

The following three problems were proposed as the 'basic problems for HMMs' by Lawrence Rabiner in his extremely influential paper on HMMs and their uses in speech recognition [6].

Problem 1.

Given the observation sequence $\mathcal{O} = O_1\, O_2\, \ldots\, O_T$, and the hidden Markov model λ, how do we compute $P(O|\lambda)$, the probability of an observation sequence occurring, given the model?

This is an evaluation problem. In solving this problem we gain the ability to compare how well different observations sequences match the model we are working with. This is useful in applications which involve predicting the most likely observation sequence out of multiple. In addition, this problem allows us to select one model over another, given how well an observation sequence matches the model

Problem 2.

Given the observation sequence $\mathcal{O} = O_1\, O_2\, \ldots\, O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q_1 q_2 \ldots q_N$ which "best" explains the observations?

This problem attempts to uncover the hidden part of the HMM and for this reason it is often referred to as the decoding problem. This problem can be difficult because it requires an optimality criteria to be selected, of which there could be multiple. Solving this problem properly of course yields benefits, as it allows one to gain an insight into which states are causing certain observations to occur. This is particularly useful in regime identification, which we will see further on.

Problem 3.

How do we adjust the model parameters A, B, π to maximise $P(O|\lambda)$? In other words, how do we "train" the model to best fit the observation sequence emitted by the model.

This problem attempts to optimize the model parameters in order to build a model which most accurately describe a given observation sequence. This problem is often called the training problem, as to solve it we use an observation sequence to give a base from which we can adjust the model parameters.

We will now discuss these problems in greater detail and show how we can make use of their solutions in a financial context.

3.5 Solutions to The Main Problems of HMMs

The solutions to these problems are given by Rabiner [6], and further details are given by Mark Stamp [7]. The following solutions use the above sources as well as filling in the gaps between calculations.

3.5.1 Problem 1

Our task is to calculate the probability of the observation sequence $\mathcal{O} = O_1\, O_2\, \ldots\, O_T$ occurring due to the hidden Markov model λ, and thus we must evaluate $P(\mathcal{O}|\lambda)$. Presented here are two methods, the first being a straightforward yet somewhat naive approach.

Naive Approach

This will involve considering every possible state sequence of length T. We recall that at each time instant the model emits an observation symbol. We therefore consider every possible sequence of length T of the hidden states through which the model transitions. Consider one such state sequence,

$$Q = q_1 q_2 \ldots q_T$$

Where each q_t is the hidden state the model is in at time t, making q_1 the initial state of the model. The probability of getting the observation sequence $\mathcal{O}$ for the state sequence Q given the model λ is the product of the individual probabilities of observing each observation symbol O_i in the sequence, as a result of the model being in state q_i. Namely,

$$P(\mathcal{O}|Q, \lambda) = P(O_1|q_1, \lambda)P(O_2|q_2, \lambda) \ldots P(O_T|q_t, \lambda)$$

or simply,

$$P(\mathcal{O}|Q, \lambda) = \prod_{i=1}^{T} P(O_i|q_i, \lambda) \tag{1}$$

We recall that the probability of the observation symbol O_k being emitted due to the state q_t is what we defined to be the *observation probability*, $b_j(k)$. Thus,

$$P(\mathcal{O}|Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \ldots b_{q_T}(O_T) = \prod_{i}^{T} b_{q_i}(O_i) \tag{2}$$

The probability of just the state sequence Q occurring given the model, $P(Q|\lambda)$, is given by multiplying the initial state probability, π_{q_1}, by each of the *state transition probabilities*, $a_{q_i q_j}$, which we interpreted as being the probability of the model transitioning from state q_i to state q_j. This gives,

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} \tag{3}$$

We introduce the probability $P(\mathcal{O}, Q|\lambda)$ which we call a *joint probability*. Such a probability is the product of equations (2) and (3) above. Namely,

$$P(\mathcal{O}, Q|\lambda) = P(\mathcal{O}|Q, \lambda) \cdot P(Q|\lambda) \tag{4}$$

This is derived as follows:

- Using the formula for conditional probability [8], we get
$$P(\mathcal{O}, Q|\lambda) = \frac{P(\mathcal{O} \cap Q \cap \lambda)}{P(\lambda)}$$
- Similarly we note that,
$$P(Q|\lambda) = \frac{P(Q \cap \lambda)}{P(\lambda)}$$
- And,
$$P(\mathcal{O}|Q, \lambda) = \frac{P(\mathcal{O} \cap Q \cap \lambda)}{P(Q \cap \lambda)}$$
- Then,
$$P(\mathcal{O}|Q, \lambda) \cdot P(Q|\lambda) = \frac{P(\mathcal{O} \cap Q \cap \lambda)}{\cancel{P(Q \cap \lambda)}} \cdot \frac{\cancel{P(Q \cap \lambda)}}{P(\lambda)}$$
- And we recover the result of (4),

$$\boxed{P(\mathcal{O}, Q|\lambda) = P(\mathcal{O}|Q, \lambda) \cdot P(Q|\lambda)}$$

Finally, $P(\mathcal{O}|\lambda)$ is found by summing the joint probability over all the possible state sequences. This is because $P(\mathcal{O}|\lambda)$ is the probability of the observation sequence $\mathcal{O}$ occurring given the model, which means it is the probability of $\mathcal{O}$ occurring due to one possible state sequence, *or* another possible sequence *or* another and so on, and in probability theory the probability of one event occurring *or* another is interpreted as the sum of each individual probability. Thus,

$$P(\mathcal{O}|\lambda) = \sum_{\text{possible } Q} P(\mathcal{O}|Q, \lambda) \cdot P(Q|\lambda)$$

$$= \sum_{\text{possible } Q} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{5}$$

This computation requires $2TN^T$ products for a model containing N hidden states, making this an unreasonable solution to the problem [6]. Luckily, there exists an algorithm which yields the same result in a far more efficient manner.

The Forward Algorithm

If we consider equation (5) we notice that there will be many repeated calculations when summing over every possible state sequence. For example, for a two state model and an observation sequence of length three there will be eight summations, four of which will begin with evaluating $\pi_{q_1} b_{q_1}(O_1)$. We aim to use a method where we store these calculations after computing them once, which will exponentially reduce the number of multiplications required [9].

The method we use is called the *forward algorithm*. It works by defining the forward variable, $\alpha_t(i)$ [6]. This is defined as being the probability of having observed the observation sequence $\mathcal{O}$ of length t and now being in the state $q_t = S_i$. Namely:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda) \tag{6}$$

The probability $P(\mathcal{O}|\lambda)$ we wish to calculate is then obtained simply by summing the 'terminal' forward variables, that is for $t = T$, for each state S_i of the model,

$$P(\mathcal{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{7}$$

This is a dynamical programming method as we are dividing the problem of summing over all possible state sequences into sub-cases where we sum over all sequences which end in each of the states of the system. The number of multiplications required by this method is to the order of N^2T [2], which is far less than the previous method.

Clearly, we must figure out how to evaluate the terminal forward variables. To do this, we will derive a recursion relation in which we evaluate the forward variable at each time step. Given the definition of the forward variable these will be interpreted as the probability of a *partial* observation sequence having occurred, and the model being in state S_i when we 'pause' to look at the model at a given time instant.

To begin, we compute the *initialization step* at time t. Namely, we compute the probability of the observation sequence containing a single observation symbol having occurred and the model being in the initial state $q_t = S_i$.

$$\alpha_t(i) = P(O_1\, O_2 \ldots O_t, q_t = S_i|\lambda)$$

Using joint probability rules,

$$\alpha_t(i) = P(O_1\, O_2 \ldots O_t|q_t = S_i) \cdot P(q_t = S_i|\lambda) \tag{8}$$

We recognise the first term as the probability of observing O_t at time t given the model is in the state $q_t = S_i$, which we defined as the *observation probability* $b_i(O_t)$, and the second

term the probability of the model starting in state $q_t = S_i$, which we defined as the *initial probability distribution*, π_i. Thus,

$$\alpha_t(i) = \pi_i b_i(O_t) \tag{9}$$

We now consider the *induction step* and iterate to the next time step and consider the forward variable at time $t+1$, with the final state of the model being $q+t+1 = S_j$.

$$\alpha_{t+1}(j) = P(O_1\, O_2 \ldots O_{t+1}, q_{t+1} = S_j|\lambda) \tag{10}$$

We note that the model could have ended up in state S_j here at time $t+1$ as a result of transitioning from any of the N states in the model at time t. By the Markov property, the probability the model is in state S_j currently depends only on the state it was in at time t. Thus we can rewrite equation (10) as follows:

$$\alpha_{t+1}(i) = \sum_{i=1}^{N} P(O_1\, O_2 \ldots O_{t+1}, q_t = S_i, q_{t+1} = S_j|\lambda) \tag{11}$$

The *chain rule for probability* states the following [10]. For n events $A_1, A_2, \ldots, A_n$,

$$P(A_1, A_2, \ldots, A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2, A_1) \ldots P(A_n|A_{n-1}, \ldots, A_1)$$

Given that multiplications of independent probabilities are commutative, we can rearrange the left hand side of this rule and use it to express (11) as follows:

$$\begin{aligned}\alpha_{t+1}(j) = \sum_{i=1}^{N} P(O_{t+1}|q_{t+1} = S_j, q_t = S_i, O_1 \ldots O_t)P(q_{t+1} = S_j|q_t = S_i, O_1 \ldots O_t)\times \\ \times P(q_t = S_i, O_1 \ldots O_t)\end{aligned} \tag{12}$$

We now use the notion of conditional independence [11]. That is, two events A and B are said to be *conditionally independent* if:

$$P(A|B, C) = P(A|C)$$

That is, B does not contribute to the certainty of A. We note that O_{t+1} is conditionally independent of everything except the hidden state from which it was emitted; that is, $q_{t+1} = S_j$. In addition, by the Markov property, we know that $q_{t+1} = S_j$ only depends on $q_t = S_i$. Therefore (12) becomes;

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} P(O_{t+1}|q_{t+1} = S_j)P(q_{t+1} = S_j|q_t = S_i)P(q_t = S_i, O_1 \dots O_t) \tag{13}$$

We recognise each term to be the following quantities:

1. $P(O_{t+1}|q_{t+1} = S_j)$ is the *observation probability* $b_j(O_{t+1})$.
2. $P(q_{t+1} = S_j|q_t = S_i)$ is the *transition probability* from state S_i to state S_j , a_{ij}.
3. $P(q_t = S_i, O_1 \dots O_t)$ is the *forward variable* $\alpha_t(i)$.

This leaves us with the following expression for the forward variable at time $t + 1$.

$$\alpha_{t+1}(j) = b_j(O_{t+1}) \sum_{i=1}^{N} \alpha_t(i)a_{ij} \tag{14}$$

Finally, we can find $P(\mathcal{O}, \lambda)$ by summing all the terminal forward variables, $\alpha_T(i)$, where T is the length of the observation sequence $\mathcal{O}$. Namely,

$$\boxed{P(\mathcal{O}, \lambda) = \sum_{i=1}^{N} \alpha_T(i)} \tag{15}$$

This is a superior method than the naive approach we initially discussed as it requires far less computation.

3.5.2 Problem 2

When solving Problem 2 we attempt to find the 'optimal' hidden state sequence Q which best describes the observation sequence $\mathcal{O}$ emitted by the model. The most difficult element of this problem is deciding what we define 'optimal' to mean here [6]. Let us discuss the options:

- The first choice is to take inspiration from the solution to Problem 1. This would involve computing the probability of an observation sequence for a given state sequence $P(\mathcal{O}|Q, \lambda)$, and then choosing the state sequence that resulted in the maximum probability. This is not a wise approach as the number of possible state sequences of length T for a model containing N sates is N^T, which scales exponentially.
- Another method is to use a modified version of the forward algorithm which we used in Problem 1 called the *backward algorithm* [6]. The backward algorithm is a time reversed version of the forward algorithm. This method changes our optimality criteria to maximize the number of correct states q_t [7]. We will attempt to calculate the probability that an observation O_t resulted from the hidden state $q_t = S_i$, and choose the most likely state. We now proceed with this method.

The Backward Algorithm

We define the *backward variable* $\beta_t(i)$ to be the probability of the remainder of an observation sequence $O_{t+1}\, O_{t+2} \ldots O_T$, given the model was in state $q_t = S_i$ at time t, where T is the length of the full observation sequence [6]. Thus:

$$\beta_t(i) = P(O_{t+1}, \ldots O_T | q_t = S_i, \lambda) \tag{16}$$

We look to write $\beta_t(i)$ in a form we can use the same way we did for the forward algorithm. We begin with the initialization step at time $t = T$, noting that we are starting at the end of the observation sequence as we are working backwards with this algorithm. The variable is initialised as,

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{17}$$

where we have arbitrarily defined $\beta_T(i)$ to be 1 [6]. The induction step is similar to that of the forward algorithm with the direction of the time step reversed. To derive the induction step we begin by noting that for the model to have been in state S_i at time t we must account for the N possible states $q_{t+1} = S_j$ at time $t + 1$. Thus the backward variable can be written as:

$$\beta_t(i) = \sum_{j=1}^{N} P(q_{t+1} = S_j, O_{t+1}, \ldots, O_T | q_t = S_i, \lambda)$$

Using the fact that the product of independent events is commutative,

$$\beta_t(i) = \sum_{j=1}^{N} O_{t+2}, \ldots, O_T, O_{t+1}, q_{t+1} = S_j | q_t = S_i, \lambda) \tag{18}$$

We now once again use the chain rule for probability [10]. For simplicity let us denote $A = O_{t+2}, \ldots, O_T, B = O_{t+1}, C = q_{t+1} = S_j, D = q_t = S_i$. Then by the formula for conditional probability [8],

$$P(A, B, C|D) = \frac{P(A, B, C, D)}{P(D)} \tag{19}$$

Writing $P(A, B, C, D) = P(D, C, B, A)$ the chain rule gives,

$$P(D, C, B, A) = P(A|B, C, D)P(B|C, D)P(C, D)P(D) \tag{20}$$

Then equating these and subbing back into the formula for conditional probability gives:

$$P(A, B, C|D) = P(A|B, C, D)P(B|C, D)P(C, D) \tag{21}$$

Thus we can rewrite (18) as

$$\beta_t(i) = \sum_{j=1}^{N} P(O_{t+2} \dots O_T | O_{t+1}, q_{t+1} = S_j, q_t = S_i) P(O_{t+1} | q_{t+1} = S_j, q_t = S_i) P(q_{t+1} = S_j | q_t = S_i) \tag{22}$$

As in problem 1, not all the events are conditionally dependent. $O_{t+2}, \dots O_T$ being emitted is only conditionally dependent on the state the model was in at time $t+1$; that is, $q_{t+1} = S_j$. In addition, $Ot+1$ being emitted also depends only on $q_{t+1} = S_j$. Thus:

$$\beta_t(i) = \sum_{j=1}^{N} P(O_{t+2} \dots O_T | q_{t+1} = S_j) P(O_{t+1} | q_{t+1} = S_j) P(q_{t+1} = S_j | q_t = S_i) \tag{23}$$

We recognise the first term to be the backward variable $\beta_{t+1}(j)$, the second as the *observation probability* $b_{t+1}(j)$, and the final term as the *transition probability* a_{ij}. Thus the induction step yields the following result:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{24}$$

From this we can see that the backward variable picks up where the forward variable ends. Having now studied how the backward algorithm works and having obtained a formula for the backward variable of which we can make use, we proceed to solving problem 2.

We begin by considering the probability of the model being in the state $q_t = S_i$ at time t given the observation sequence $\mathcal{O} = O_1, O_2, \dots O_T$ and the model. Using Baye's theorem [8], this is the following probability:

$$P(q_t = S_i | \mathcal{O}, \lambda) = \frac{P(q_t = S_i, \mathcal{O} | \lambda)}{P(\mathcal{O} | \lambda)} \tag{25}$$

We wish to express the numerator of this expression in terms of quantities we have already derived. We do this by writing the observation sequence in the numerator in terms of the two ranges $O_1 \dots O_t$ and $O_{t+1} \dots O_T$ for some arbitrary point t in the sequence. Namely:

$$P(\mathcal{O}, q_t = S_i | \lambda) = P(O_1 \dots O_t, O_{t+1} \dots O_T, q_t = S_i | \lambda) \tag{26}$$

Using the result of equation (21) this gives:

$$P(\mathcal{O}, q_t = S_i | \lambda) = P(O_1 \dots O_t | O_{t+1} \dots O_T, q_t = S_i, \lambda) P(O_{t+1} \dots O_T | q_t = S_i, \lambda) P(q_t = S_i | \lambda) \tag{27}$$

We recognise that the first and last term are simply the *forward variable*, $\alpha_t(i)$ (equation (8)), and the middle term as the *backward variable* (equation (16)), $\beta_t(i)$. Thus,

$$P(\mathcal{O}, q_t = S_i|\lambda) = \alpha_t(i)\beta_t(i) \tag{28}$$

We can now write the probability we wish to evaluate to solve problem 2 (equation (25)) as:

$$P(q_t = S_i|\mathcal{O}, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathcal{O}|\lambda)} = \gamma_t(i) \tag{29}$$

And we recall that $P(\mathcal{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$ is the solution to problem 1.

We have now arrived at a solution for problem 2. The most likely state $q_t = S_i$ at time t which resulted in the observation O_t is the state for which $\gamma_t(i)$ is a maximum. That is:

$$q_t = \text{argmax}(\gamma_t(i)) \quad 1 \leq t \leq T,\ 1 \leq i \leq N \tag{30}$$

This solution is better than the naive approach as we only need to run N calculations at each time t for each hidden state S in the model and choose the most likely one. It is important to note however that this method has limitations. For example, when the model has state transitions that have probability zero ($a_{ij} = 0$), the optimal state sequence found may not be a valid one for the given system described by the HMM [6]. A dynamic programming method using the *Viterbi algorithm* can be used instead, which we will now discuss briefly.

The Viterbi Algorithm

The Viterbi algorithm is another dynamical programming algorithm which works by choosing a different optimality criteria for choosing the best state sequence to describe an observation sequence. This algorithm aims to find the *single* best state sequence $Q = q_1, q_2, \ldots, q_T$ at once, rather than maximizing the number of correct states as was done with the backward algorithm. This is equivalent to maximizing the probability $P(Q, \mathcal{O}|\lambda)$ [6].

The algorithm works by computing the probability that a given state sequence ending in the state $q_t = S_i$ yielded the observation sequence $O_1 \ldots O_t$. We begin by defining $\delta_t(i)$ to be the highest probability along a given state sequence or *path*, ending in the state $q_t = i$, $q_1 \ldots q_t = i$, which yielded the observation sequence $O_1 \ldots O_t$. We will refer to this as a *Viterbi probability*. Namely:

$$\delta_t(i) = \max_{q_1, q_2 \cdots q_{t-1}} [P(q_1 \ldots q_t = i, O_1 \ldots O_t|\lambda)] \tag{31}$$

Using the chain rule for probability this can be written as:

$$\delta_t(i) = \max_{q_1, q_2 \cdots q_{t-1}} P(q_1 \dots q_t | O_1 \dots O_t, \lambda) P(O_1 \dots O_t | \lambda) \tag{32}$$

We now derive the *induction step* of the algorithm, $\delta_{t+1}(i)$. This is interpreted as the most likely path ending in state $q_{t+1} = j$ at time j. The maximum of the N possible states j will have to be considered. Namely:

$$\delta_{t+1}(j) = \max_{q_1, q_2 \cdots q_t} P(q_1 \dots q_t, q_{t+1}, O_1 \dots O_t, O_{t+1} | \lambda) \tag{33}$$

Rewriting the order of the events we obtain the following expression using the chain rule and discarding conditionally independent events as we have done before.

$$\begin{aligned} \delta_{t+1}(j) &= \max_i P(O_{t+1}, q_{t+1} = j, q_1 \dots q_t = i, O_1 \dots O_t | \lambda) \\ &= \max_i P(O_{t+1} | q_{t+1} = j) P(q_{t+1} = j | q_t = i) P(q_1 \dots q_t | O_1 \dots O_t) P(O_1 \dots O_t | \lambda) \end{aligned} \tag{34}$$

Where we recognise the first term to be the observation probability $b_j(O_{t+1})$, the second term to be the transition probability a_{ij}, and the final two terms as the Viterbi probability $\delta_t(i)$. Thus the induction step is:

$$\delta_{t+1}(j) = \max_i \left[\delta_t(i) a_{ij} \right] b_j(O_{t+1}) \tag{35}$$

Which gives the recursion relation ([6]):

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right] b_j(O_t) \quad 2 \le t \le T, \;\; 1 \le j \le N \tag{36}$$

Using this recursion relation we may compute the most likely path the model takes to output the observation sequence, noting that the initialisation step will be $\delta_1(i) = \pi_i b_i(O_1)$, where π_i is the usual probability of starting in state $q_1 = S_i$.

In order to obtain the individual states within this path, we must 'backtrack' through the individual Viterbi probabilities $\delta_t(j)$ [6]. We backtrack one step at a time, beginning with the last probability $\delta_T(i)$. We then go back one step and choose the state with the highest probability of transitioning to the current state, and repeat this until each state has been found, thus yielding the full sequence.

3.5.3 Problem 3.

The end goal in solving problem 3 is to have built a HMM in which the model parameters, A, B, π have been adjusted to maximise the probability of the observation sequence $\mathcal{O}$ given the model. This problem is the most difficult of the three as there is no analytical solution, nor is there an optimal way of estimating the parameters [6].

The method we will use to tackle the problem is to attempt to adjust the model parameters such that the probability of the observation sequence given the model is *locally* maximised [6]. There are multiple ways of doing this using iterative schemes. Given the involvement of Baum with Jim Simons, we will choose to use the classic *Baum-Welch* method.

This method involves using the forward and backward algorithms to compute the probability of the model being in state $q_t = S_i$ at time t and state $q_{t+1} = S_j$ at time $t + 1$, given the observation sequence $\mathcal{O}$ and the model λ. With this it is then possible to calculate the expected number of emissions and transitions within the observation sequence, and thus derive new estimated model parameters. The procedure is repeated while the probability of the observation sequence given the model, $P(\mathcal{O}|\lambda)$, is increasing.

To see how this works more formally, we begin by defining $\xi_t(i, j)$ to be the probability of being in state S_i at time t and transitioning to state S_j at time $t + 1$ given the model and the observation sequence [6]. Namely:

$$\xi_t(i, j) = P(S_i = q_t, S_j = q_{t+1}|\mathcal{O}, \lambda) \tag{37}$$

Using the definition of conditional probability ([8]), this can be written as

$$\xi_t(i, j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O_1 \dots O_t, O_{t+1}, O_{t+2} \dots O_T, \lambda)}{P(\mathcal{O}|\lambda)} \tag{38}$$

where we have decomposed the observation sequence in an attempt to hopefully arrive at some familiar quantities. The numerator is the probability $P(q_t = S_i, q_{t+1} = S_j, O_1 \dots O_T|\lambda)$, where we have written it conditioned to the model λ. We can simplify this expression using the chain rule for probability as follows,

$$\begin{aligned} P(q_t = S_i, q_{t+1} = S_j, O_1 \dots O_T|\lambda) &= P(O_1 \dots O_t, O_{t+1}, O_{t+2}, \dots O_T, q_{t+1} = S_j, q_t = S_i, \lambda) \\ &= P(O_1 \dots O_t|O_{t+1} \dots O_T, q_{t+1} = S_j, q_t = S_i, \lambda)\times \\ &\times P(O_{t+1}|O_{t+2} \dots O_T, q_{t+1} = S_j, q_t = S_i, \lambda)\times \\ &\times P(O_{t+2} \dots O_T|q_{t+1} = S_j, q_t = S_i, \lambda)\times \\ &\times P(q_{t+1} = S_j|q_t = S_i, \lambda) \end{aligned} \tag{39}$$

Dropping conditionally independent terms,

$$
\begin{aligned}
P(q_t = S_i, q_{t+1} = S_j, O_1 \dots O_T|\lambda) &= P(O_1 \dots O_t, O_{t+1}, O_{t+2}, \dots O_T, q_{t+1} = S_j, q_t = S_i, \lambda) \\
&= P(O_1 \dots O_t|q_t = S_i, \lambda)\times \\
&\times P(O_{t+1}|q_{t+1} = S_j, \lambda)\times \\
&\times P(O_{t+2} \dots O_T|q_{t+1} = S_j\lambda)\times \\
&\times P(q_{t+1} = S_j|q_t = S_i, \lambda)
\end{aligned}
\tag{40}
$$

we recognise the first term to be the *forward variable* $\alpha_t(i)$, the second to be the *observation probability* $b_j(O_{t+1})$, the third as the *backward variable* $\beta_{t+1}(j)$ and the final term as the *transition probability* a_{ij}. Thus we get the following expression for $\xi_t(i, j)$:

$$
\xi_t(i, j) = \frac{\alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(O_{t+1})}{P(\mathcal{O}|\lambda)} \tag{41}
$$

Recall that the numerator is the probability $P(q_t = S_i, q_{t+1} = S_j, \mathcal{O}|\lambda)$. This is the probability of being in state S_i at time t and state S_j at time $t + 1$, and observing the sequence $\mathcal{O}$. Thus, if we are to sum over all possible state transitions, this will simply be the probability $P(\mathcal{O}|\lambda)$. We can thus write:

$$
\xi_t(i, j) = \frac{\alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(O_{t+1})}{\sum_{i,j=1}^{N} \alpha_t(i)\beta_{t+1}(j)a_{ij}b_j(O_{t+1})} \tag{42}
$$

In this form we can see that we may interpret $\xi_t(i, j)$ as the *expected number of state transitions* from state S_i to S_j [6].

Recall that we defined $\gamma_t(i)$ as being the probability of being in state S_i at time t given the sequence $\mathcal{O}$ (equation (29)). That is,

$$
\gamma_t(i) = P(q_t = S_i|\mathcal{O}, \lambda) = \frac{P(q_t = S_i, \mathcal{O}, \lambda)}{P(\mathcal{O}|\lambda)}
$$

similarly,

$$
\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j|\mathcal{O}, \lambda) = \frac{P(q_{t+1} = S_j)P(q_t = S_i, \mathcal{O}, \lambda)}{P(\mathcal{O}|\lambda)}
$$

Noting that if we are to sum over all of the N possible states S_i, then $\sum_{j=1}^{N} P(q_{t+1} = S_j) = 1$; that is, the model will have to transition to some state at time q_{t+1}, even if it stays transitions to the same state it is already in. Given this, we can see $\gamma_t(i)$ is the sum of $\xi_t(i, j)$ over all states S_j [6]. Namely:

$$
\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)
$$

We now have all the ingredients we need to re-estimate the model parameters. We will go about this by expressing each parameter in terms of the quantities we have derived.

We note if we are to sum $\gamma_t(i)$ over t we will obtain the expected number of times the model will be in state S_i. We can re-phrase this result as the number of times the system transitions *from* state S_i to some other state (not including $t - T$ [6]). In a similar manner, summation of $\xi_t(i, j)$ over t ($t \neq T$) gives the expected value of transitions from state S_i to state S_j.

We are now ready to derive re-estimation formulas for each model parameter [6]:

Initial Probability Distribution π

$\pi^{'}$ = expected number of times model is initially ($t = 1$) in state S_i = $\gamma_1(i)$

State Transition Probability a_{ij}

To update the probability of transitioning from state S_i to S_i, we divide the expected number of transitions from S_i to S_j by the expected total number of times the model is in S_i. This gives us the following re-estimated value for the transition probability $a_{ij}^{'}$.

$$a_{ij}^{'} = \frac{\text{number of transitions from state } S_i \text{ to state } S_j}{\text{number of transitions from state } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Observation Probability $b_j(k)$
Similarly, to update the probability of emitting a particular observation O_k from state S_j, we divide the expected number of times this observation is emitted from S_j by the expected total number of times the model is in state S_j. This gives us the re-estimated value of the observation probability $b_j^{'}(k)$.

$$b_j^{'}(k) = \frac{\text{number of times in state } S_j \text{ observing } O_k}{\text{number of times in state } S_j} = \frac{\sum_{t=1}^{T} \gamma_t(j) \text{ s.t } O_k \text{ observed}}{\sum_{t=1}^{T} \gamma_t(j)}$$

After applying these re-estimations, we get a new model $\lambda^{'}$. We repeat this process while $P(\mathcal{O}|\lambda^{'})$ is increasing at each step, thus improving the accuracy of the model with each iteration.

It is important to note that $P(\mathcal{O}|\lambda^{'})$ is only locally maximized using this scheme and that generally speaking there are multiple local maxima on the optimization surface [6]. This algorithm generally converges quickly, (~ 10 iterations) even for large models and large observation sequences used for training [12].

There are methods we may use to improve the convergence of the re-estimation process. The first is that we should be careful when choosing the initial model parameters. It is essential that π, A, B are *randomized*, since uniform values result in a local maximum from which the model cannot climb [7]. This is because the algorithm used is a *steepest-descent-type* algo-

rithm which climbs the nearest local maximum of the optimization surface [12]. If an educated guess can be made for the initial parameters this should, of course, be done, but in the case that no information is available to guess upon, there are still options as to how to avoid small local maxima. One such method is seen by Krogh [12] where noise is added to the model before each iteration. The amount of noise added decreases linearly to zero over the iterations of the re-estimation procedure. The intuition behind this method is that by adding randomness to the procedure it means the initial iterations will result in worse parameter selection, forcing the algorithm to explore more of the optimization surface, which can stop the algorithm from immediately choosing the closest local maximum it finds and potentially uncovering larger local maxima. This method was further explored and improved upon by Eddy [13].

Having introduced the theory of hidden Markov models and solved the fundamental problems associated with them we now explore their financial applications and show how each of the problems we have solved will play a role.

4 Financial Applications

The first financial application of hidden Markov models we explore is attempting to predict the daily closing price of a stock. In doing this the solutions to problems 1 and 2 which we have just discussed will be used.

4.1 Stock Market Prediction

Stock market prediction is the act of attempting to determine the future value of a company's stock traded on an exchange. This has been studied extensively, as knowing the future value of a stock's price is very useful information to investors looking to make profits. Prediction methods originally comprised of either fundamental analysis; that is, studying the company associated with the stock; or technical analysis, which makes predictions using the previous trends of the stock's price. However, since the invention of the digital computer, machine learning and artificial intelligence have become the go-to approaches for stock market prediction [14].

More recently, hidden Markov models have also been used, which is unsurprising given their successful use in time-series analysis in areas such as speech recognition (which is the purpose of the paper by Rabiner which we used extensively in the previous section of this study [6]). We must acknowledge that in using HMMs to predict stock prices we are assuming that the prices follow a Markov process, which in time we will come to see is quite an oversimplification of an extremely complicated system.

To frame the problem faced to us we recall the key features of a HMM. Hidden Markov models contain hidden states amongst which transitions can occur, and each state emits an observation. In the lens of the stock market, the hidden states determine the price of the stock, and the transitions between these states occur due to features such as policies, current economic climate, and even the public's perception of the company. The set of observations may contain variables such as the open or close price of a given stock - this being the value of the stock at the beginning and end of the trading session (i.e. 9.30am and 4pm for the NYSE, respectively [15]).

There are multiple approaches we can take when using HMMs to forecast stock prices. One method is where the observation emitted by the model is a vector containing historical daily close prices for various stocks [16]. Here, the previously mentioned Viterbi algorithm in our discussion of Problem 2 is used. Recall that this algorithm computes the most likely state sequence given an observation sequence. Having computed the most likely state sequence, the most likely next state s_{t+1} can then be found using the trained model's transition matrix.

Another possible choice for the observation is to use fractional changes of the stock price [17]. We will use this method in this study (and also build upon it to predict log-returns, which we will come to in due course). In this approach, the observation yielded from the model is a vector composed of the following fractional changes,

$$\Delta_{\text{price}} = \frac{\text{close - open}}{\text{open}}$$

$$\Delta_{\text{high}} = \frac{\text{high - open}}{\text{open}}$$

$$\Delta_{\text{low}} = \frac{\text{open - low}}{\text{open}}$$

Where 'open' and 'close' are the daily open and close prices, and 'high' and 'low' are the daily maximum and minimum stock price, respectively. We can use the solutions to the main problems of hidden Markov models we discussed (in conjunction with a maximum a posteriori probability estimate) to predict the most likely fractional change $\tilde{\Delta}_{\text{price}}$, such that the closing price on day t can be predicted by adding the predicted fractional change to the open price on day t. Namely:

$$\text{close}(t) = \text{open}(t) + \tilde{\Delta}_{\text{price}} \tag{43}$$

The process of computing the predicted fractional change begins with building our HMM. The first step is to use the results of the training problem (3.5.3). We train the model on historical values of the three fractional changes described above. In this study all historical financial data was gathered from *Yahoo Finance* [18] and Medtronic (MDT) was the stock chosen to study. The Python library `hmmlearn` [19] was used to build the model. This library makes use of the previously mentioned Baum-Welch algorithm when training the model, which we recall finds the optimal model parameters by using the forward-backward algorithm to compute the probability of observing a certain observation sequence for some set of parameters. It repeats this until the calculated probability stabilises within a specified tolerance.

4.2 Predicting Close Prices

The code (found at [20]) discussed in the following section was inspired and adapted from [21]. I implemented changes such as using random number generation to create possible observation sequences, as well as implementing the ability to predict log-returns instead of close prices.

With the model parameters optimised, we can begin the process of predicting the daily close price of a stock over a selected time period. The initial approach we will take is closely related to Gupta and Dhingra's [17]; that is, we use a Maxium A Posteriori probability estimate to 'score' possible observation sequences to find the most likely one, which will yield us the predicted fractional change, and in turn the close price. In theory, every possible observation sequence should be used, but with computation time being a relevant constraint in this study, a subset of possible observation sequences was selected. Specifically, pseudo-random elements from the set of historical fractional changes are appended to a vector of the form:

$$\mathcal{O}_{\text{possible}} = (\Delta_{\text{price}}, \Delta_{\text{high}}, \Delta_{\text{low}}) \tag{44}$$

This was contained in a `for` loop to create a specified number of possible sequences. In this case an observation sequence containing 5000 possible observations was generated.

The next step is to 'score' these observations using data from a specified number of 'latent' days (50 in this case) to establish the most likely one. The scoring is done by the built in function `model.score()` in the python library. This calculates the log probability of observing a given string of observation sequences, composed of the observations from the previous 50 days along with one of the possible observations generated. This probability is computed using the forward algorithm. This is done individually for each of the generated observations and we compute `np.argmax()` of these probabilities, yielding the observation which had the highest probability of occurring:

$$\mathcal{O}_{\text{predicted}} = (\tilde{\Delta}_{\text{price}}, \tilde{\Delta}_{\text{high}}, \tilde{\Delta}_{\text{low}}) \tag{45}$$

The first element of the observation vector is the predicted fractional change in the open and close price, which is then used to calculate the closing price on a given day using (43).

This process is done for a specified number of days, with the results of such a calculation shown below:

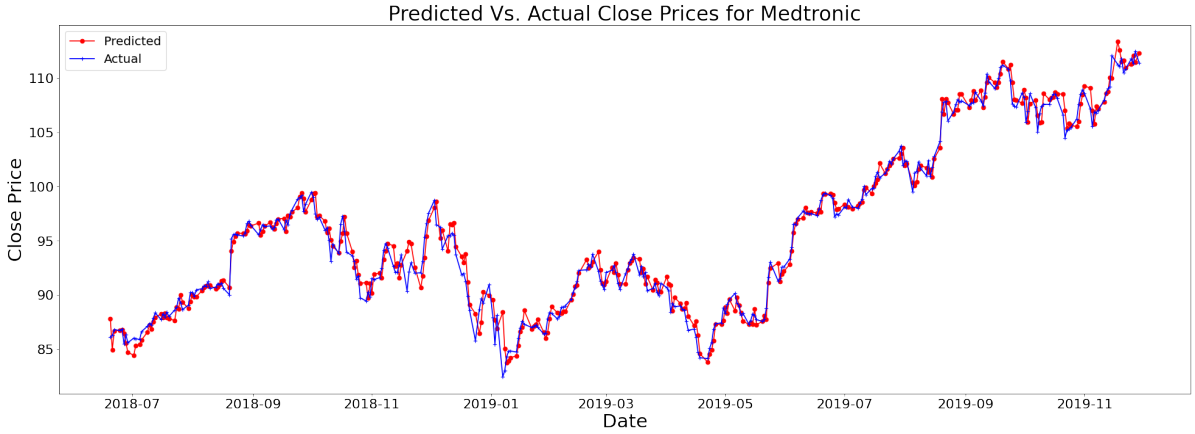

Figure 4: Predicted close price of Medtronic stock for a period of 365 days.

The error associated with these predictions is given below:

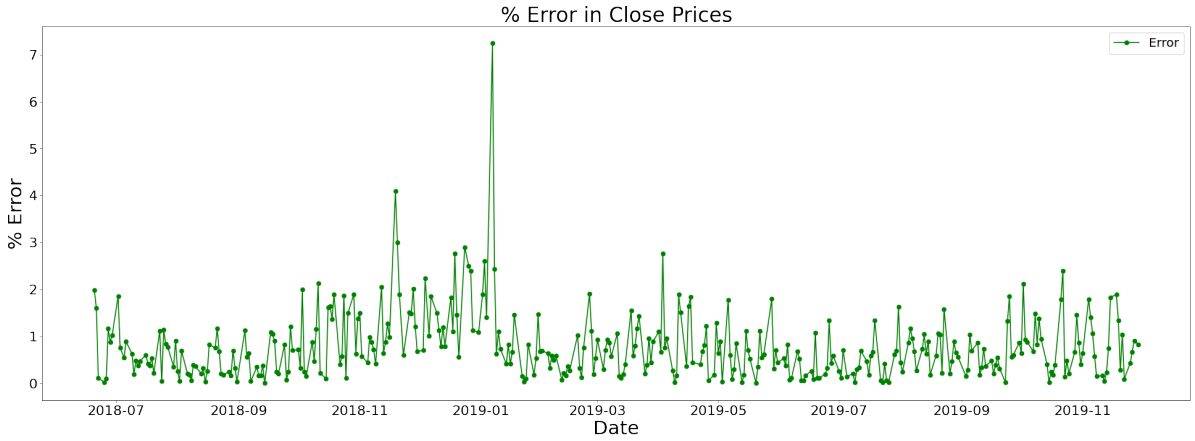

Figure 5: Percentage error in predicted close price of Medtronic stock over a period of 365 days.

Where the percentage error is given by:

$$\delta = \left| \frac{\text{close}_{\text{predicted}} - \text{close}_{\text{actual}}}{\text{close}_{\text{actual}}} \right| \times 100 \tag{46}$$

Yielding an average error of 0.771%.

Comment on Maximum a Posteriori (MAP)

Definition 4.1 (**Maximum a posteriori estimator**). A *maximum a posteriori* probability estimate is an estimate of an unknown quantity. If we want to estimate the parameter θ given data on a variable x, then the MAP estimate of θ is defined to be the mode of the posterior distribution $f(\theta|x)$, which is simply the probability distribution of θ given the data x. Namely: [22]

$$\hat{\theta}_{MAP}(x) = \text{argmax} f(\theta|x)$$

In this case, θ is the observation we are attempting to predict and x is the observations from the previous latent days. The 'argmax' or, arguments of the maxima computes the point(s) of the probability distribution at which it is maximised.

Before we turn out attention towards predicting log-returns, we take a moment to discuss financial returns and why log-returns are a natural and useful quantity when analysing a stock's performance.

4.3 Introduction to Returns

Returns

A financial return, or simply a return, is the amount of money made or lost on an investment over a specified time period [23]. The arithmetic or simple return, R, on an investment is given by [24]:

$$R = \frac{P_1 - P_0}{P_0}$$

where P_0 and P_1 are the initial and final prices of the investment over a certain period (e.g a stock's price). This can be identified as simply the percentage change in price, so the price of the investment after a given period is given by:

$$P_1 = P_0(1 + R)$$

Suppose now we want to compute the interest over a number of sub periods. For n sub periods the final price will be given by:

$$P_1 = P_0(1 + R_1)(1 + R_2)\dots(1 + R_n)$$

Where R_i is the return for each period. The disadvantage of working with arithmetic returns is that we cannot express this quantity neatly or shorten it in any way, since the returns are in general not equal to one another. This is where we direct our focus to logarithmic returns. When discussing logarithmic returns it is the natural logarithm which is used. It will become clear shortly why this is the case.

Consider the following simple example. Suppose we invest $\$1$ in a very generous bank which offers an annual interest rate of 100%. After 1 year our initial investment will have grown by 100%.

$$\$1 \xrightarrow{\text{1 year}} \$1 + \$1 \times 100\% = \$2$$

We now introduce the concept of *compound interest*. Suppose we are instead offered an interest rate of 50%, compounded every 6 months. This will increase our investment as follows:

$$\$1 \xrightarrow{\text{6 months}} \$1 + \$1 \times 50\% = \$1.50 \xrightarrow{\text{6 months}} \$1.50 + \$1.50 \times 50\% = \$2.25$$

Or, more concisely:

$$\$1 \xrightarrow[\text{compounded bi-annual}]{\text{1 year}} \$1\left(1 + \frac{1}{2}\right)^2 = \$2.25$$

We can continue to compound the interest more and more frequently. Specifically, for an investment gaining $r\%$ interest, compounded n times per year, we have:

$$P_1 = P_0\left(1 + \frac{r}{n}\right)^n$$

where we have written $r\%$ simply as r -which is known as the *nominal rate*. Using this expression we note the value of our investment with increasingly shorter compounding periods.

Compounding Frequency	Value of Investment ($)
Yearly (1 times)	2.0
Bi-Annual (2 times)	2.25
Monthly (12 times)	2.613035290224676
Weekly (52 times)	2.692596954437168
Daily (365 times)	2.7145674820219727
Hourly (8760 times)	2.7181266916179077

Table 1: Value of an initial investment of $\$1$ gaining 100% interest compounded n times.

We note that the value of our investment increases the more times we compound the interest, but by a smaller amount each time. It is approaching a limit which we recognise as the mathematical constant e. In fact, it was from Jacques Bernoulli's work on *continuous* compounding of interest that the limit definition of e was discovered (though it was Leonard Euler, after whom the constant is named, who computed this limit) [25]. That is:

$$\lim_{n\to\infty}\left(1 + \frac{r}{n}\right)^n = e^r$$

This leads to the following formula for the future value of an investment compounded continuously over t years at a nominal rate r [26]:

$$P_1 = P_0 e^{rt} \tag{47}$$

Rearranging this equation we can write the *growth factor* as:

$$\frac{P_1}{P_0} = e^{rt}$$

and taking the natural logarithm of this:

$$\ln\left(\frac{P_1}{P_0}\right) = \ln(e^{rt})$$

and employing basic log identities we find the *logarithmic return* on this investment to be:

$$\ln(P_1) - \ln(P_0) = rt \tag{48}$$

Having now established what log-returns are, we discuss the advantages they hold over arithmetic returns.

Advantages of Log-Returns

Symmetry

Log-returns are symmetric, whereas arithmetic returns are not [24]. To illustrate this let us consider the following example.

Suppose we have an initial investment of $\$100$ that increases by 100% to $\$200$ at the end of the year, and in the following year decreases by 50% to $\$100$.
The following table illustrates the difference in the logarithmic and arithmetic returns in this scenario.

Value of Investment ($)	Arithmetic Return	Log-Return
100	-	
200	$+100\%$	$+69.31\%$
100	-50%	-69.31%

Table 2: Comparison of arithmetic and logarithmic returns on a given investment over 3 years.

From this we can see that an investment which yields an arithmetic return of 100% followed by a return of -50% results in an average return of $+50\%$, which is not at all indicative of the current financial situation which has in fact remained unchanged. In the case of log returns, however, this illustrates that a particular investment such as the example above shows that a log return of $+x\%$ on an investment followed by a log return of $-x\%$ leaves the profit/loss unchanged.

Time-Additive

Log-returns are time additive [24]. This means that the log-return for two successive time periods is simply the sum of the log-return for each successive period. The same is not true for arithmetic returns, as seen in the previous example.

4.4 Predicting Log-Returns

Having introduced the meaning of log-returns and the advantages they offer as a quantity in quantitative finance, we now discuss how they can be predicted using a HMM.

The method used is similar to that used in predicting the close prices as seen in 4.2 (the code used is found at [27]). While it is possible to simply take the log of the predicted close prices and calculate the log-return using these values, this is not natural and a superior method is to train a new model using log-returns. That is, the observation vector the model is to be trained on contains the following elements:

$$\ln(\Delta_{\text{price}}) = \ln(\text{close}) - \ln(\text{open})$$

$$\ln(\Delta_{\text{high}}) = \ln(\text{close}) - \ln(\text{high})$$

$$\ln(\Delta_{\text{low}}) = \ln(\text{close}) - \ln(\text{low})$$

where 'close' and 'open' are still the daily opening and closing prices, and 'high' and 'low' the daily high and low prices.

The model was trained using historical values of these three quantities in the same way as before to optimise the model parameters. With these parameters adjusted, the most likely observation based on a specified number of latent days' worth of data was predicted. The method for calculating the most likely observation sequence was exactly the same as before. Namely, 5000 possible observations were generated using a random number generator to randomly choose an element from the list of historical $\ln(\Delta)$ values. The most likely observation sequence for a given day was then calculated by scoring these possible observations individually. This was done as before by computing the likelihood of an observation sequence occurring given the model comprised of the previous 50 day's observations, along with one of the generated observations. The most likely observation was found and the daily log close price was predicted to be:

$$\ln(\text{close})_t = \widetilde{\ln}(\Delta_{\text{price}}) + \ln(\text{open})_t \tag{49}$$

where $\widetilde{\ln}(\Delta_{\text{price}})$ is the predicted daily value of $\ln(\Delta_{\text{price}})$, or equivalently the first element of the predicted observation vector. Having predicted the daily log close prices for a specified time period, the predicted log return, $\tilde{R}$, on day t, was found to be:

$$\tilde{R} = \ln(\text{close})_t - \ln(\text{close})_{t-1} \tag{50}$$

Using this, the following plot was generated comparing the predicted and actual log returns over a period of one year:

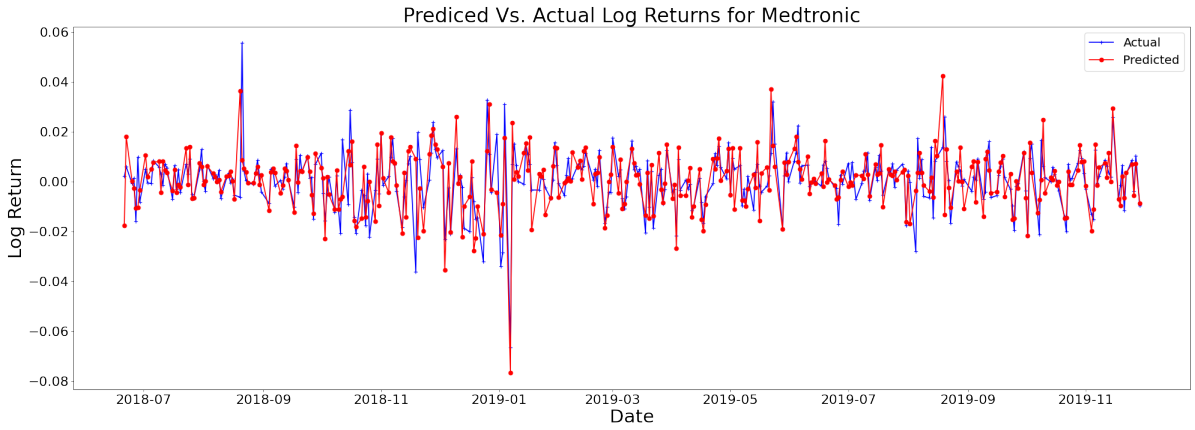

Figure 6: Comparison of predicted vs. actual log-returns on Medtronic stock over a period of 365 days.

And the error associated with these predictions is shown below, with an average error of 0.618%.

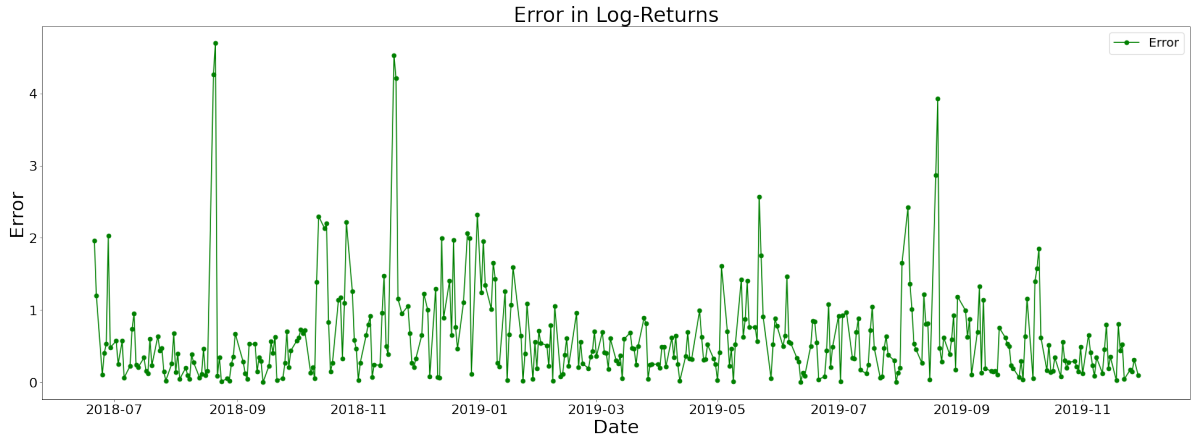

Figure 7: Error in predicted log-returns on Medtronic stock over a period of 365 days.

At first glance, it appears the model made satisfactory predictions of the log-returns, with the calculated values matching up very closely with the actual historical data. Specifically, further inspection of the data shows the model predicts the correct value within a tolerance of 0.01 73.9% of the time. Reducing the error in the predictions can be achieved by generating a larger amount of possible observations or increasing the number of hidden states in the model. The former of these will of course increase the computation time by whatever factor the number of sequences is increased. We will explore these error reduction methods in further sections.

A surprising outcome to note is that training the model using a greater amount of historical data was found to decrease the accuracy of the predictions. This is due to a phenomenon known as 'overfitting', where using a large amount of training data results in the model finding useless patterns in the data, which hinder its ability to make predictions based on new data [28]. This is relevant to the data used to train our HMM as the training set of data is comprised of the initial 80% of the historical data and, if this contains Medtronic's stock data from the 1980s, these stock trends differ significantly when compared to present day.

We will now perform further statistical analysis upon this data in order to determine whether the model has painted the whole picture. We begin by considering a measure known as *kurtosis*.

Kurtosis

Kurtosis is a measurement of the departure from normality of a given distribution [29]. It is also often referred to as the 'tailedness' of a distribution (although it is often wrongly defined as the 'peakedness' of a distribution [29]). Mathematically, it is calculated to be the fourth standardized moment of a probability distribution. To understand this we consider the following definitions [30]:

Definition 4.2 (Expectation value). Let x_i be the numerical outcomes of N repetitions of some experiment/process, and let X_i be random variables with a common mass function f. For each possible x_i, roughly $Nf(x)$ of the X_i variables will take the value x. For large N we describe the *expected value* of a random variable X with mass function f to be:

$$\mathcal{E}(X) = \sum_{x:f(x)>0} xf(x)$$

Definition 4.3 (Central Moment). The k^{th} moment, $\mu_k^{'}$ of a random variable X is defined to be $\mu_k^{'} = \mathcal{E}(X^k)$. The k^{th} moment about the mean, or the k^{th} *central moment* μ_k is defined to be:

$$\mu_k = \mathcal{E}((X - \mu_1^{'}))^k$$

Definition 4.4 (Standardized Moment). The k^{th} standardized moment, $\tilde{\mu_k}$ of a probability distribution is defined to be the ratio of the the k^{th} central moment and the k^{th} power of the standard deviation, σ^k [31]. Namely:

$$\tilde{\mu_k} = \frac{\mu_k}{\sigma^k}$$

Which leads us to the mathematical definition of kurtosis

Definition 4.5 (Kurtosis). The *kurtosis* of a probability distribution is defined to be fourth standardized moment of the distribution. Namely:

$$\text{Kurt} = \frac{\mu_4}{\sigma^4}$$

A perfectly normal distribution has a kurtosis of 3 [32]. Therefore it is common to see Kurtosis defined with 3 subtracted at the end. Karl Pearson coined the terms *leptokurtic, mesokurtic,* and *platykurtic* to indicate cases where kurtosis is greater than, equal to, or less than that of a normal distribution, respectively [33], using the definition which deducts 3.

Examples of distributions with high and low kurtosis can be seen below:

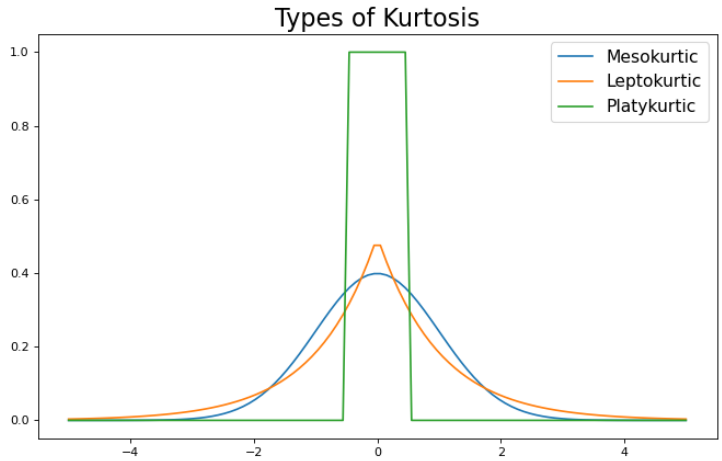

Figure 8: Comparison of laplace (leptokurtic), normal (mesokurtic) and uniform (platykurtic) distributions.

Kurtosis in Finance

In a financial setting, kurtosis is interpreted as a measure of financial risk [34]. This is clear from the definition of kurtosis. High kurtosis means there are more extreme outliers, which is interpreted financially as a higher probability of extremely high or extremely low returns, thus signalling a high level of risk in a given investment. Conversely, low kurtosis indicates that the risk is spread evenly throughout the distribution.

We now consider the kurtosis in the distributions of the actual and predicted log-returns to get a further understanding of the accuracy of the model used.

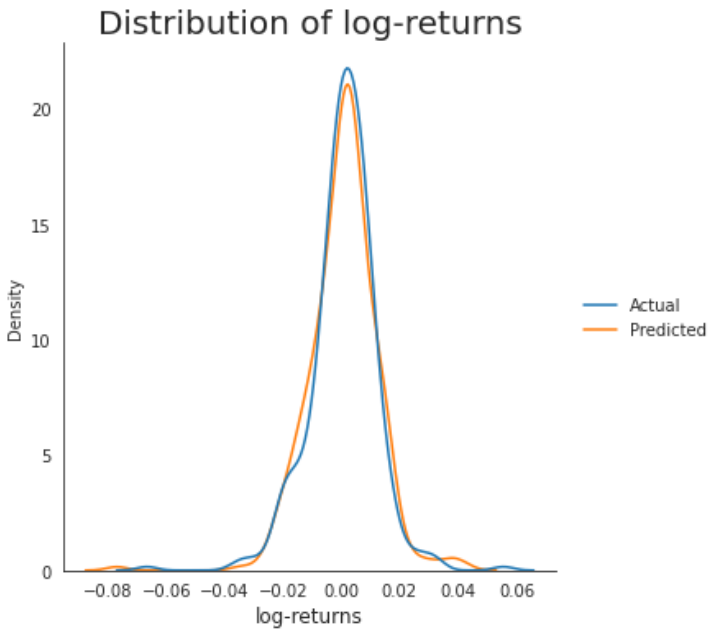

Figure 9: Comparison of the distributions of the actual and predicted log-returns. We note the difference in the tails.

For reference we compare these distributions to that of a random normal distribution:

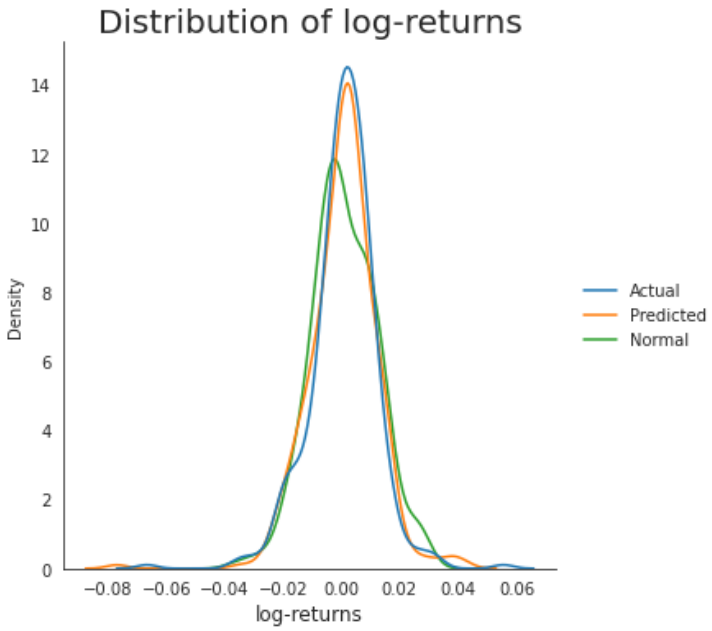

Figure 10: Comparison of the distributions of the actual and predicted log-returns with a normal distribution as a reference.

We note the following statistics from these distributions:

Distribution	Actual	Predicted	Normal
Kurtosis	4.999	5.785	0.205
Standard Deviation	0.01127	0.01161	0.01132

Table 3: Statistics of above distributions.

It is important to note here that we have used the convention that a perfectly normal curve has zero kurtosis. We note that the actual and predicted returns exhibit higher kurtosis than that of the normal curve. Specifically, the predicted log-returns are more kurtotic than the actual values. This highlights the importance of viewing the predictions of a model from multiple angles in order to develop an idea of whether the model is truly an accurate descriptor of the scenario in question. Had we merely considered the relatively low percentage error in the predicted returns (0.618%), the model seems very accurate. However, when we look at other parameters such as kurtosis, we get an idea of how well the model has predicted the less likely, more extreme high and low returns. In this case, the model has predicted a higher probability of extremely high returns than is seen in the historical data.

It is important to note that despite the model predicting higher kurtosis, the standard deviation is almost equal to that of the actual data. Generally speaking, kurtosis on its own is not a very useful quantity in analysing markets. We must also consider the standard deviation. Namely, it is possible that an investment might be highly kurtotic, which is undesirable, but have a low standard deviation which is favorable [34]. Therefore we note that our model has predicted an overall higher level of risk than is realised due to the minor difference in standard deviation.

Generally, it is assumed in financial modelling that returns are normally distributed. However, this is rarely the case when considering actual data. Even when a market appears to be in a

stable state of growth or decline, returns tend to have excess kurtosis [35]. It is impossible to generate a model which captures all the details - something will always be left out. The model used in this study was a Gaussian HMM. That is, the observation probabilities followed a Gaussian distribution (bell-shape). Clearly, it would be better to use a model which uses a leptokurtic distribution as this would be more indicative of the real world data.

We will now briefly explore one final application of HMMs in quantitative finance - regime identification.

4.5 Regime Identification

A market *regime* is a period in which consistent behaviour is observed in the market. For example, a market may have a regime where the market is showing continual growth. Likewise, a neutral regime may exist where the movement of the market is observed to be stagnant. Being able to identity regimes is important for developing an optimal trading strategy as it allows one to build up a description of how the market is evolving and predict trends.

Regime identification can be done using the HMMs, as proposed by James Hamilton [36]. In this study we detect regimes by visualising the returns observed as a result of the hidden states of the model. This is a direct application of Problem 2, where we saw how the Viterbi algorithm can be used to find the most likely hidden state sequence which gave rise to the observables emitted from the model. The hidden states are calculated using the `model.predict()` function in the `hmmlearn` library which was used extensively throughout this study. We will begin by considering a model which has been built to contain three hidden states, and has been trained on the fractional changes of log returns which was discussed in section 4.4 as well as the daily log returns.

The data presented below is a visualisation of the observables emitted by the three hidden states of the model. We recall that the model outputs a given observation according to the observation probability matrix, from one of the hidden states, and it can switch between these states with respect to the transition matrix. We have plotted the log-returns, categorising the values which occurred due to each of the three states and show each collection of points individually.

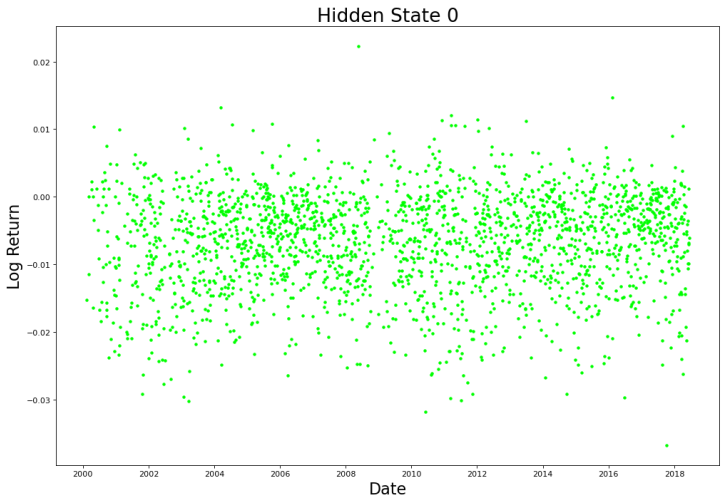

Figure 11: Log-return values observed due to state-0.

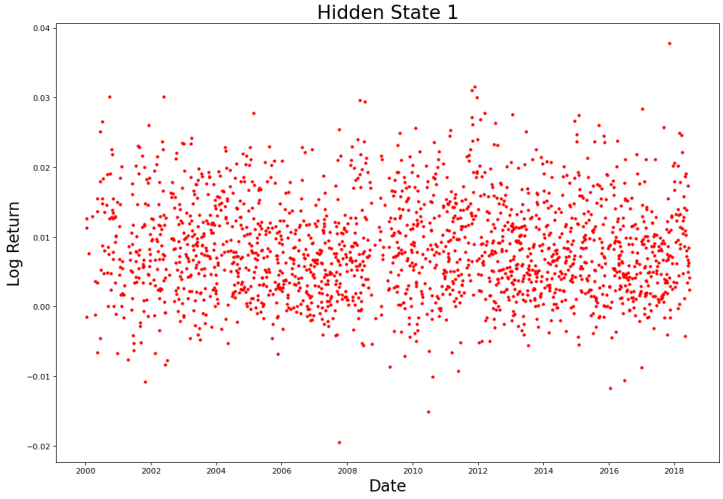

Figure 12: Log-return values observed due to state-1.

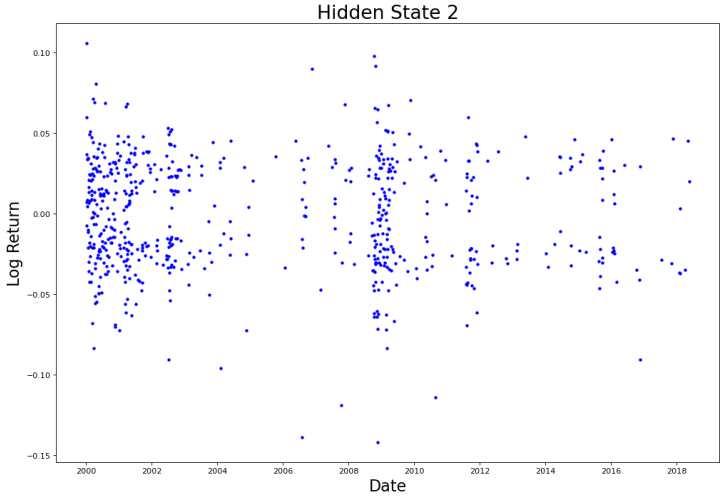

Figure 13: Log-return values observed due to state-2.

We immediately see a difference in the appearance of the data amongst the states. The data corresponding to states 0 and 1 is much more uniform and clustered together when compared

to state 2. If we compute the covariance matrices of each state and take the diagonal elements we are simply left with the standard deviation σ of each element of the observation vector (see section 4.4). We will also compute the mean of each observation. This yields the following results:

State	$\sigma\,(\ln \Delta_{\text{price}})$	$\sigma\,(\ln \Delta_{\text{high}})$	$\sigma\,(\ln \Delta_{\text{low}})$	σ (log-return)
0	$4.354 \times 10{-}5$	3.845×10^{-5}	1.634×10^{-5}	6.05905×10^{-5}
1	4.415×10^{-5}	1.447×10^{-5}	4.155×10^{-5}	5.899×10^{-5}
2	8.768×10^{-4}	3.167×10^{-4}	4.379×10^{-4}	0.00121

Table 4: Variance in observation elements for each hidden state.

State	$\mu\,(\ln \Delta_{\text{price}})$	$\mu\,(\ln \Delta_{\text{high}})$	$\mu\,(\ln \Delta_{\text{low}})$	μ (log-return)
0	-5.89×10^{-3}	-1.126×10^{-3}	4.74×10^{-3}	-6.41×10^{-3}
1	7.65×10^{-3}	-3.94×10^{-3}	0.01283	8.02×10^{-3}
2	-6.134×10^{-4}	-0.02153	0.02274	-1.6×10^{-3}

Table 5: Mean in observation elements for each hidden state.

We consider the standard deviation and mean in the last columns as this shows the data for the daily log return. From this we see that state 2 has the highest standard deviation and yields a mean negative return. In quantitative finance the standard deviation of returns is called *volatility*. High volatility is unfavorable as it corresponds to high risk of losses [37]. Therefore we can identity the state 2 to represent a *highly volatile regime* in this market.

Similarly, state 1 gives rise to the smallest standard deviation as well as the largest mean return, and thus we identify this as a *lowly volatile regime*. Low volatility is desirable as it generally corresponds to low risk investments.
The final state can be referred to as a *neutral regime*, but in this case we may also identify state 0 as a lowly volatile regime as there is not a large difference in the volatility of each state.

We can also identify these regimes visually by plotting the observed close prices emitted by the hidden states and seeing how the highly volatile regime (blue) dominates periods of loss and how the lowly volatile regimes (red and green) graph the periods of growth.

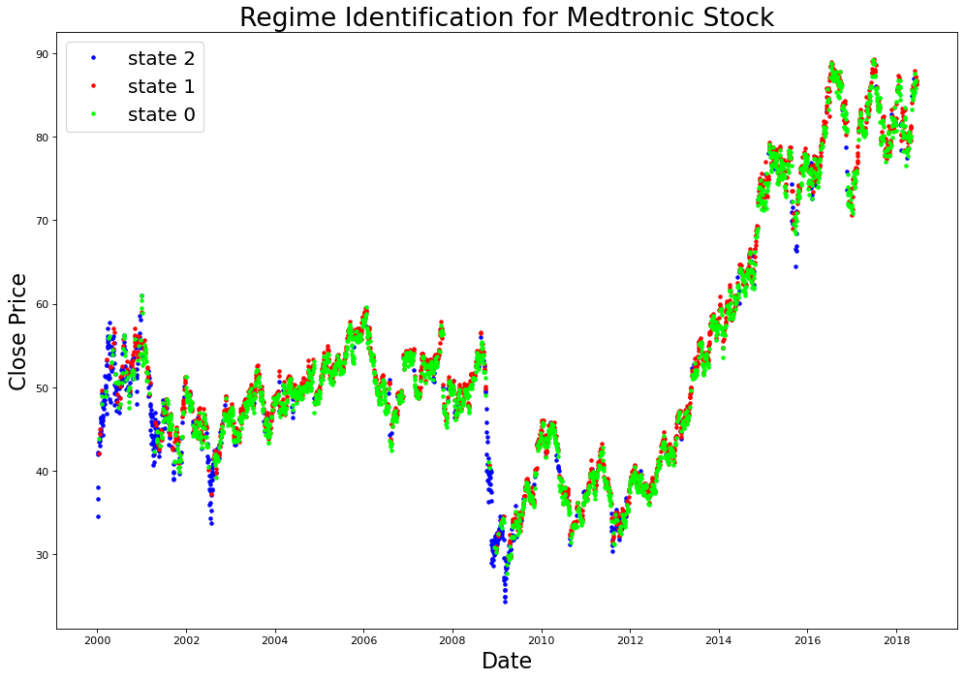

Figure 14: Observe how the highly volatile regime (blue) is observed in periods of losses.

We observe from this application how HMMs can be used to an investor's advantage. Using the basic concepts of HMMs along with a short python script we can easily identify different regimes within the market. The number of states can be increased which may lead to more regimes being uncovered, some of which may not be trivial when simply viewing a graph of closing prices.

This concludes our discussion of financial applications of HMMs. We now briefly discuss potential further study in the areas mentioned in this study.

5 Further Study

One of the major problems associated with building a HMM is choosing the number of hidden states the model is to contain. Intuitively one would assume that having more states would improve the accuracy of the predictions. This was achieved by Bram de Wit [16], where the root mean square error in predicted stock prices was shown to decrease consistently as more hidden states were used. However, in this study, I was unable to reliably conclude if this was true for the data set used. The results showed no correlation between using more states and achieving lower error in predictions. Furthermore, the error associated with each model varied largely each time the calculations were ran.

The most probable reason for these results is that there was a problem with my code, but another explanation is that the specific data set used in this study was not optimal for training a HMM. For example, it is possible that the historical Medtronic stock data used to train the model led to an optimization surface in which the Baum-Welch algorithm regularly got stuck in a local maximum, leading to inferior model parameter estimation. Further study here could involve using a 'noise injection' method as proposed by Krogh and Eddy [12],[13]. As mentioned in problem 3 regarding training a HMM, this method involves adding random noise to the observation sequences in order to force the Baum-Welch algorithm into finding new local maxima in the optimization surface.

6 Conclusion

In this study we have presented a mathematical description of hidden Markov models and solved the fundamental problems associated with them. In doing this we have described multiple approaches to these problems and how optimal solutions can be found using various dynamical programming algorithms such as the *Viterbi Algorithm*, and the *Forward-Backward Algorithm*.

We have also researched two significant financial applications of hidden Markov models : stock market prediction and regime identification. A HMM was trained on historical stock data from Medtronic (MDT). Using the model, daily closing prices as well as log-returns were predicted using a Maximum A Posteriori probability estimate. The results showed that the predicted values closely matched the actual data, differing by less than 1% for the majority of the predictions.

Using the same training data, another model was used to research using HMMs to identify regimes in financial markets. It was found that the HMM reliably identified low and highly volatile regimes.

The findings presented here have demonstrated that hidden Markov models have a wide range of financial applications, and are a valuable tool in financial modelling and decision making.

References

[1] Gregory Zuckerman. *The man who solved the market: How Jim Simons launched the quant revolution*, page 56. Penguin, 2019.

[2] Leonard E. Baum. *An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes*, page 2. 1972.

[3] Ramon van Handel. *Hidden Markov Models*, page 4. 2008.

[4] Emanuel Parzen. *Stochastic processes*, page 7. SIAM, 1999.

[5] James R Kirkwood. *Markov processes*, volume 20. CRC press, 2015.

[6] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):258–265, 1989.

[7] Mark Stamp. *A Revealing Introduction to Hidden Markov Models*, page 3. 2021.

[8] F.M Dekking, C. Kraaikamp, H.P Lopuhaä, and L.E Meester. *A Modern Introduction to Probability and Statistics*, page 26. 2005.

[9] Phil Blunsom. Hidden markov models. *Lecture notes, August*, 15(18-19):3, 2004.

[10] Hossein Pishro-Nik. Introduction to probability, statistics, and random processes. 2016.

[11] A Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(1), 1979.

[12] Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of molecular biology*, 235(5), 1994.

[13] Sean R Eddy et al. Multiple alignment using hidden markov models. In *Ismb*, volume 3, pages 114–120, 1995.

[14] Nusrat Rouf, Majid Bashir Malik, Tasleem Arif, Sparsh Sharma, Saurabh Singh, Satyabrata Aich, and Hee-Cheol Kim. Stock market prediction using machine learning techniques: a decade survey on methodologies, recent developments, and future directions. *Electronics*, 10(21):2717, 2021.

[15] NYSE Trading Hours. `https://www.nyse.com/markets/hours-calendars`, 2022.

[16] Bram de Wit. *Stock Prediction Using a Hidden Markov Model Versus a Long Short-Term Memory*. PhD thesis, 2019.

[17] Aditya Gupta and Bhuwan Dhingra. *Stock market prediction using hidden markov models*, pages 1–4. 2012.

[18] Yahoo! Finance. `https://finance.yahoo.com/`, 2023.

[19] hmmlearn Python library for building HMMs. `https://hmmlearn.readthedocs.io/en/latest/`, 2023.

[20] Google Colab containing code used for close price prediction. https://colab.research.google.com/drive/1xZ0M1ZXtinveJc0hLRenUlSQvt2Ol A2a?usp=sharing, 2021.

[21] Code found on Kaggle. https://www.kaggle.com/code/ehsanamim/stock-market-prediction-using-hmm/notebook, 2023.

[22] Robert Bassett and Julio Deride. Maximum a posteriori estimators as a limit of bayes estimators. *Mathematical Programming*, 174, 2019.

[23] Investopedia.com - What Are Returns in Investing, and How Are They Measured? https://www.investopedia.com/terms/r/return.asp, 2023.

[24] *Why Use Logarithmic Returns In Time Series Modelling*, Lucas Louca. https://lucaslouca.com/Why-Use-Logarithmic-Returns-In-Time-Series-Modelling/, 2021.

[25] Carl B Boyer and Uta C Merzbach. *A history of mathematics*. John Wiley & Sons, 2011.

[26] Ronald J Harshbarger and James J Reynolds. *Mathematical applications for the management, life, and social sciences*. Cengage Learning, 2012.

[27] Google Colab containing code used for log-return prediction and regime identification. https://colab.research.google.com/drive/1H87pL-_iqM9xYg2LExLD15LL_HwZkwCe?usp=sharing, 2023.

[28] Stuart J Russell. *Artificial intelligence : a modern approach*. Prentice Hall, first edition, 1995.

[29] Peter H Westfall. Kurtosis as peakedness, 1905–2014. rip. *The American Statistician*, 68 (3):191–195, 2014.

[30] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2020.

[31] Weisstein, Eric W. "Standardized Moment." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/StandardizedMoment.html, 2023.

[32] Panagiotis Andrikopoulos, Greg N Gregoriou, et al. *Handbook of Frontier Markets: The African, European and Asian Evidence*. Academic Press, 2016.

[33] Karl Pearson. "das fehlergesetz und seine verallgemeiner-ungen durch fechner und pearson." a rejoinder. *Biometrika*, 4(1-2):169–212, 1905.

[34] Informa Investment Solutions, *Informa Stat Facts* - Kurtosis. https://financialintelligence.informa.com/about/statfacts, 2016.

[35] Robert J Frey. Hidden markov models with univariate gaussian outcomes. Technical report, Technical Report, Stony Brook University, 2009.

[36] James D Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the econometric society*, pages 357–384, 1989.

[37] *Volatility From the Investor's Point of View*, Investopedia. https://www.investopedia.com/ask/answers/010915/volatility-good-thing-or-bad-thing-investors-point-view-and-why.asp, 2022.