Hidden Markov Model for Cryptocurrency Trading

Adam Pešek

Department of Banking and Insurance
Prague University of Economics and Business

This dissertation is submitted for the degree of

December 2022

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Adam Pešek
December 2022

Acknowledgements

And I would like to acknowledge ...

Abstract

Hidden Markov model (HMM) is a statistical signal prediction model, which has been widely used to predict economic regimes and stock prices. In this paper, we introduce the application of HMM in trading stocks (with S&P 500 index being an example) based on the stock price predictions. The procedure starts by using four criteria, including the Akaike information, the Bayesian information, the Hannan Quinn information, and the Bozdogan Consistent Akaike Information, in order to determine an optimal number of states for the HMM. The selected four-state HMM is then used to predict monthly closing prices of the S&P 500 index. For this work, the out-of-sample R^2_{OS}, and some other error estimators are used to test the HMM predictions against the historical average model. Finally, both the HMM and the historical average model are used to trade the S&P 500. The obtained results clearly prove that the HMM outperforms this traditional method in predicting and trading stocks.

Table of contents

List of figures

List of tables

Chapter 1

Markov Processes

1.1 Markov Processes

In order to fully understand and explore the properties and implications of Hidden Markov Models (hereinafter "HMMs") that will be used for cryptocurrency price prediction it is necessary to apriori state the underlying definition of Markov processes imminently followed by Markov Chains and finally Hidden Markov Models.

1.1.1 Definition and Properties

Definition. Given a measurable space (S, B) called state space, where S is a set and B is a sigma-algebra on S. A function P : S × B → R is called a transition probability function if P(x, ·) is a probability measure on (S, B) for all x S and if for every B B, the map s → P(s, B) is B-measurable. Define P 1 (x, B) = P(x, B) and inductively the measures P n+1(x, B) = R S P n(y, B)P(x, dy), where we write R

1.2 Discrete-time Markov Chains

Let $\Omega \neq \emptyset$ and $A \subseteq 2^{\Omega}$ be a σ-algebra on Ω, and P a measure on A with $P(\Omega) = 1$, i.e. P is a *probability measure*. Then the triplet (Ω, A, P) is called a *probability space*. Where Ω denotes a sure event and it holds that $\forall \omega \in \Omega$ is called an elementary event. Furthermore, $\forall A \in A$ is a random event so that $P(A)$ is a probability of such a random event.

Let I be a countable set and Θ a σ-algebra on I. Each $i \in I$ is called a *state* and (I, Θ) a *state-space*. We say that $\lambda = (\lambda_i : i \in I)$ is a measure on I if $0 \leq \lambda_i \leq \infty$. If in addition the *total mass* $\sum_{i \in I} \lambda_i = 1$ then λ is a *distribution* (or probability measure).

Suppose now that we have two measurable spaces (Ω, A) and (I, Θ) and a random variable $X : \Omega \to I$ assuming that X is measurable. Thus we call (I, Θ) a state space and (Ω, A) an underlying space. Therefore we may set:

$$\lambda_X(i) = P(X = i) = P(\{\omega : X(\omega) = i\}) \tag{1.1}$$

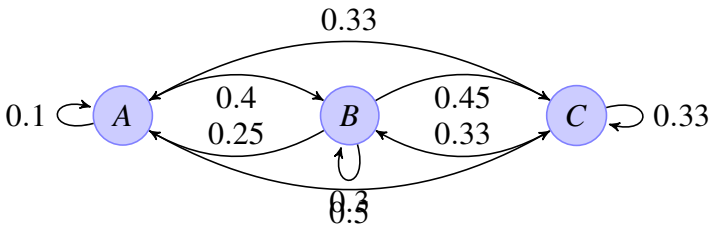
Since we are allowing only for the discrete realisations of the random variable X, given previous assumptions, $\lambda_X(i)$ is a *probability mass function*.

Then measure λ defines a distribution of X. Given such a setting random variable X is assumed to denote random state i with probability λ_i.

To simply illustrate the idea behind discrete Markov Chains let us assume a situation where the future market movements transition between a countable number of states $I =$ {upward movement, no movement, downward movement} and there is a transition matrix (also called *stochastic matrix*) $P = (p_{i,j} : i, j \in I)$ defined as:

$$P = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.25 & 0.3 & 0.45 \\ 0.33 & 0.33 & 0.33 \end{pmatrix}$$

Each row represents full set of transition probabilities between each state which is visible from $\sum_{j \in I} p_{i,j} = 1$, i.e. each row of matrix P represents a full distribution of transitions over I. Such a relationship can be represented as a diagram indexing each state by A, B and C respectively as follows:



This may be easily interpreted for each given state. For example if we assume that the market moved upwards on the last trading day there is a 0.1 chance that the market will move in positive direction today, in other words the conditional probability of observing the state A today given the state A yesterday is 0.1. On the hand if we suppose that today the market actually transitioned to the state B with probability 0.4 there is now a probability of 0.45 to transition to state C since the future transition is only conditioned by its previous state.

Till now we have assumed random variable X that modelled the probability of observing a state $i \in I$ given the initial distribution λ. We ought to consider thus we have to allow for a

system of discrete random variables that are also identically distributed for each time step $\{X_t, t \in T\}$, for $\{t_0, t_1, ..., t_n\} \subset T$ where $n \in \mathbb{N}_0$.

Suppose now that we have observed a given sequence of states for the last week as $\{A, B, C, C, A\}$ and we would like to know its probability of observing given the transition matrix P and distribution λ. Then $P(X_{t_0}, ..., X_{t_n}|P, \lambda)$ can be calculated as:

$$\begin{aligned} P(X_{t_0}, ..., X_{t_n}|P, \lambda) &= P(A, B, C, C, A|P, \lambda) \\ &= P(A) * P(B|A) * P(C|B) * P(C|C) * P(A|C) \\ &= \lambda * p_{1,2} * \ p_{2,3} * \ p_{3,3} * \ p_{3,1} \\ &= \lambda * 0.4 * 0.45 * \ 0.33 * 0.33 \end{aligned}$$

maybe bych tam dal product of Xs

Where the probability of observing state A is determined by our distribution λ evaluated at respective state A since we have no prior knowledge about what exactly happened before t_0.

On the other hand, we might consider a situation in which we have observed such a sequence of events and we need to determine the next state given the sequence. As in the last example, we have our transition matrix P, distribution λ and a sequence of events observed until now $\{A, B, C, C, A\}$ for $t_{k-4}, ..., t_k$. Let us also assume that t_{k+1} is a time of next event for which we are trying to determine its probability.

$$P(X_{t_{k+1}}|X_{t_k}, ..., X_{t_{k-4}}) = P(X_{t_{k+1}}|X_{t_k}) \tag{1.2}$$

We know that last observed state was A which directs us straight to the first row of our transition matrix P since from the properties of Markov Processes (the process is memoryless) we know that the next state will depend solely on the present state so we can abstract from the given sequence of past states and focus only on X_{t_k}. Finally we may conclude that the most likely future state is C with probability of 0.5.

Obviously, the probabilities in previous transition matrix P were imaginary and served only as a mere example of the main properties of Discrete-time Markov Chains. From now on we consider a dataset of BTC-USD daily close prices calculated my CoinMarketCap website from 5th March 2017 to 5th March 2022. FIrst of all we ought to make several assumptions about the data in order to apply the logic and properties of Markov Chains.

Assumption 1: The close daily prices of BTC-USD trading pair are discrete, meaning that we only record prices at exactly 0:00 of each day since there are no closing hours of the major cryptocurrency exchanges. This assumption will be elevated in the next section with Continous-time Markov Chains.

Assumption 2: We will assume that future prices of Bitcoin depend only on their pre

Theorems:

1.3 Continous-time Markov Chains

Viterbi algorithm should, in its most general form, provide a solution to the maximum a posteriori probability (MAP) estimation of the state sequence of a finite-state discrete-time Markov process introduced in the last chapter.

1.4 Hidden Markov Model

Up until now we have considered visible states in a sense that the sequence of states was known, we refer to these models as *visible Markov Models*. In this section we will consider a situation in which we do not observe the states directly but only as a guess given other visible observations that are available to us. These "visible" observations are considered as an emissions from the hidden state sequence. Thus, the observations are assumed to be generated by the hidden states. We may interpret this relation by the following diagram where we have a hidden state sequence $Z = \{z_1, z_2, ...z_T\}$:

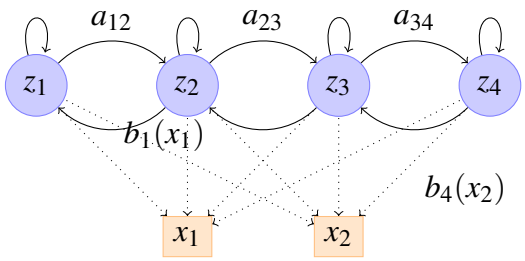


Fig. 1.1 An HMM with 4 hidden states which emits 2 discrete observable states denoted by x_1 or x_2. a_{ij} is the probability to transition from state z_i to state z_j. $b_j(x_k)$ is the probability to emit symbol x_k in state z_j.

There are 3 main assumptions of Hidden Markov Models as a consequence of the properties of Markov processes. For $Z = \{z_t\}_{t=0}^{T}$ being the hidden state sequence:

1) The Markov assumption - this assumption states that the next hidden state z_{t+1} depends only on the current state z_t, so that the transition probabilities are defined as:

$$P(z_{t+1} = j|z_t = i, z_{t-1} = l, ..., z_0 = n) = P(z_{t+1} = j|z_t = i) = p_{ij} \tag{1.3}$$

It it also possible to assume that the states in HMM are dependant beyond the current state therefore giving rise to k-order HMMs as opposed to classical first-order HMM, moreover, such variations are uneasy to analyse.

2) The stationarity assumption - The transition matrix is invariant of the time, thus for any arbitrarily set time t_1 and t_2:

$$P(z_{t_1+1} = j|z_{t_1}) = P(z_{t_2+1} = j|z_{t_2} = i) = p_{ij} \tag{1.4}$$

3) The observation independence assumption - The current observation or output is statistically independent of the previous observations. So if we have a observation sequence $X = \{x_1, x_2, ..., x_T\}$ then:

$$P(X|z_1, z_2, ..., z_T, \lambda) = \prod_{t=1}^{T} P(x_t|z_t, \lambda) \tag{1.5}$$

Given these assumptions we define the joint probability of the hidden states and the observations $P(X, Z)$ as follows:

$$P(X, Z) = \prod_{t=1}^{T} P(z_t|z_{t-1})P(x_t|z_t) \tag{1.6}$$

There are mainly 3 fundamental problems in HMM that need to be resolved as in (Oliver C. Ibe):

1. Evaluation problem - Given a model denoted as $\lambda = (P, \theta, \pi)$ and an observation sequence $X = x_1, x_2, ..., x_T$, how to efficiently compute the probability that the model generated the observation sequence, in other words, what is $P(X|\lambda)$?

2. Decoding problem - Given a model $\lambda = (P, \theta, \pi)$, what is the most likely sequence of hidden states that could have generated a given observation sequence? Thus we would like to find $Z = \arg\max_Z P(Z, X|\lambda)$, where Z is the hidden state sequence.

3. The learning problem - Given a set of observation sequences find the HMM that best explains the observation sequence. Thus, find the values of λ that maximise $P(X|\lambda)$

or in order to estimate the most likely parameters of HMM for a given observation sequence.

The most traditional approaches in solving these 3 fundamental problems differ and one may not suffice in solving all three. The evaluation problem is usually by forward-backward algorithm, the decoding problem by well-known Viterbi algorithm and the last learning problem by Baum-Welsch algorithm which is a special case of Expectation-maximization (EM) algorithm.

1.4.1 Forward and Backward algorithm

While given a sequence of observations, in our case observable states, denoted by $X = x_1, x_2, ..., x_T$ and a model $\lambda = (P, \theta, \pi)$ we wold like to compute the conditional probability of observing the sequence X under the model constraints. Thus:

$$P(X|\lambda) = \sum_{Z} P(X|Z,\lambda) * P(Z|\lambda) \tag{1.7}$$

where $Z = z_1, z_2, ..., z_T$ is a fixed sequence of hidden states. Our goal may be divided into two parts while the first part $P(X|Z,\lambda)$ computes the conditional probability of the observation sequence X given the sequence Z and model λ and the second part $P(Z|\lambda)$ accounts only for the transition probabilities among the hidden states. To summarise:

$$\begin{aligned} P(X|Z,\lambda) &= \prod_{t=1}^{T} P(x_t|z_t,\lambda) = \theta_{z_1}(x_1) \prod_{t=2}^{T} \theta_{z_t}(x_t) \\ P(Z|\lambda) &= \pi_{z_1} * \prod_{t=2}^{T} p_{z_t z_{t+1}} \end{aligned} \tag{1.8}$$

where $\theta_{z_t}(x_t)$ is the emission probability from observation x_t into z_t in our model and $p_{z_t z_{t+1}}$ are the transition probabilities for the given sequence Z. After substitution of terms from Eq. 1.8 into Eq 1.7 we obtain:

$$\begin{aligned} P(X|\lambda) &= \sum_{Z} P(X|Z,\lambda) * P(Z|\lambda) \\ &= \sum_{z_1,...,z_T} \theta_{z_1}(x_1) \pi_{z_1} \prod_{t=2}^{T} p_{z_t z_{t+1}} \theta_{z_t}(x_t) \end{aligned} \tag{1.9}$$

The summation above refers to all possible permutations of the sequence of hidden states Z which implies that we would have N^T possible sequences if we assume that N indicates all possible hidden states at each step. Furthermore, in order to calculate $P(X|\lambda)$ we have $2TN^T$ calculations which is exponential in T and not feasible for real application. As opposed to brute-force procedure as described above we take use of a forward and backward algorithm.

Forward algorithm

Previous introduction required huge amounts of calculations that were essentially unnecessary, redundant. Each sequence can be decomposed into multiple subsequences which are shared among different sequences and do not need to be recomputed again. These subsequences can be represented by a trellis as shown in Fig 1.2. With a help of such diagram we may imagine recording the probability of each distinct subsequence at each time step. In other words, we wish to compute the joint probability $P(z_t, x_1, x_2, ..., x_t)$ while taking advantage of the conditional independence of x_t which only depends on z_t, moreover, due to Markov assumption, z_t depends only on z_{t-1}. Let us now define the forward probability variable $\alpha_t(i)$ as follows:

$$\alpha_t(i) = P(x_1, x_2, ..., x_{t-1}, x_t, z_t = i|\lambda) \tag{1.10}$$

Which we may also define as the probability of being in state i at time t after having observed the sequence $x_1, x_2, ..., x_t$. The calculation therefore results in summing the incoming arcs at trellis node which is derived from:

$$\begin{aligned}\alpha_t(i) &= P(x_1, x_2, ..., x_{t-1}, x_t, z_t = i|\lambda) \\ &= P(x_t|z_t = I, \lambda)P(x_1, x_2, ..., x_{t-1}, z_t = i|\lambda) \\ &= P(x_t|z_t = i, \lambda)\sum_{j\in I} P(z_t = I|z_{t-1} = j, \lambda)P(x_1, x_2, ..., x_{t-1}, z_{t-1} = j|\lambda) \\ &= \theta_i(x_t)\sum_{j=1}^{N} p_{ji}\alpha_{t-1}(j)\end{aligned} \tag{1.11}$$

Where I is a set of all possible hidden states, i, j denote elements of I and N the size of set $|I|$. We would repeat the iterative procedure until the termination at time T where $P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} P(X, z_T = I|\lambda)$. Moreover, we initialise the trellis with $\alpha_1(i) = \pi_i\theta_i(x_1)$. Resulting algorithm requires time complexity of $O(TN^2)$ which is a significant improvement from the brute force approach with complexity of $O(2TN^T)$.

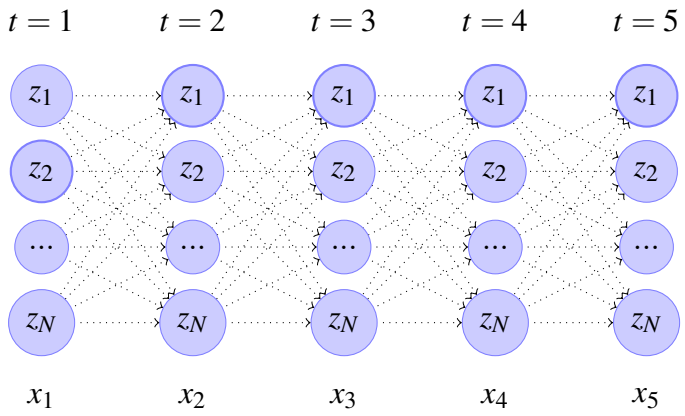


Fig. 1.2 Trellis of the observation sequence x_1,...,x_5 for the above HMM. As an example, the transition between state z_1 at time t=2 and state z_4 at time t=3 has probability $\alpha_2(z_1)p_{14}\theta_{z_4}(x_3)$, where $\alpha_t(i)$ is the probability to be instate i at time t.

Backward algorithm

While computing the backward probability variable denoted as $\beta_t(i)$ we assume the reversed iterative procedure. Let us define the backward probability variable as:

$$\begin{aligned}\beta_t(i) &= P(x_{t+1}, x_{t+2}, ..., x_{t-1}, x_t | z_t = i, \lambda) \\ &= \sum_{i,j \in I} P(x_{t+1}, x_{t+2}, ..., x_{t-1}, x_t, z_{t+1} = j | z_t = i, \lambda) \\ &= \sum_{i,j \in I} P(x_{t+1}|z_{t+1} = j) P(x_{t+2}, ..., x_{t-1}, x_t | z_{t+1} = j) P(z_{t+1} = j | z_t = i, \lambda) \\ &= \sum_{j=1}^{N} \theta_j(x_{t+1}) p_{ij} \beta_{t+1}(j)\end{aligned} \tag{1.12}$$

As we could have in forward algorithm we define effectively 3 steps:

1. Initialization step: with default value of $\beta_T(i) = 1$ for $i \in 1, 2, ..., N$.
2. Induction step:

$$\beta_t(i) = \theta_j(x_{t+1}) p_{ij} \beta_{t+1}(j) \tag{1.13}$$

Afterwards, we update the time t = t-1 and check whether expression above holds which is as long as t > 0.

3. Termination step: once the iterative procedure is exhausted, i.e. when t=0 we have the estimate of $P(X|\lambda)$ as:

$$P(X|\lambda) = \sum_{j=1}^{N} \theta_j(x_1)\pi_i\beta_1(i) = \sum_{j=1}^{N} \alpha_1(i)\beta_1(i) \tag{1.14}$$

Getting everything together we may prove that the forward and backward algorithms coincide in solving the posterior joint and marginal distributions of all hidden states. Consider the case of posterior joint distribution:

$$\begin{aligned} P(X|\lambda) &= \sum_{i=1}^{N} P(X, z_t = i|\lambda) \\ &= \sum_{i=1}^{N} P(x_1, x_2, ..., x_t, x_{t+1}, ..., x_T|z_t = i, \lambda)P(z_t = i|\lambda) \\ &= \sum_{i=1}^{N} P(x_1, x_2, ..., x_t|z_t = i, \lambda)P(x_{t+1}, ..., x_T|z_t = i, \lambda)P(z_t = i|\lambda) \\ &= \sum_{i=1}^{N} P(x_1, x_2, ..., x_t, z_t = i|\lambda)P(x_{t+1}, ..., x_T|z_t = i, \lambda) \\ &= \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \end{aligned} \tag{1.15}$$

See that above we take a use of conditional independence of observations given the hidden state and therefore we may transform the problem using already known variables for forward and backward passes over the hidden states. Thus, the posterior marginal distribution is:

$$P(X, z_t = i|\lambda) = \alpha_t(i)\beta_t(i) \tag{1.16}$$

1.4.2 Viterbi algorithm

Once we solve the evaluation problem, thus finding the optimal procedure in computing the posterior marginal distributions of all hidden states and $P(X|\lambda)$, we may visualise the result in a trellis as shown in Figure 1.2. The second problem, decoding problem, aims to find the sequence of hidden states Z^* that most likely produced observation sequence X. Finding the most likely sequence Z^* maximazes $P(Z|X, \lambda)$. As in the encoding problem, the complexity of the optimisation problem explodes if we decide to compute all possible sequences of

hidden states to N^T calculations. The solution as in the forward and backward algorithm simplifies when we calculate the hidden state with highest probability individually rather than entire sequence up to time t. The idea is that once we find the most likely hidden state given observation and the model at each time step we may discard the rest of the possible hidden states since they obviously could not have most likely produced the observation. The complexity of the optimal sequence of hidden states decreases significantly to NT thus transforming the exponential complexity into linear. As in forward and backward algorithm we define variable $\gamma_t(i)$ as:

$$\gamma_t(i) = P(z_t = i|X,\lambda) = \frac{P(X, z_t = i|\lambda)}{P(X|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \propto \alpha_t(i)\beta_t(i) \tag{1.17}$$

Where the expressions in the third equality are obtained as a solution to a before mentioned equations. Also, the $P(X|\lambda)$ serves only as a normalising constant and is irrelevant for the optimisation given the respective time step. Moreover, the conditional probability $\gamma_t(i)$ forms a conditional probability mass function since it is non-negative for all possible values of hidden states and observations and the sum over all possible hidden states at each time step is:

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{1.18}$$

This would be one of the possible ways to estimate the most likely hidden states individually and then combine them sequentially to obtain the whole sequence of hidden states as:

$$z_t^* = \arg\max_{i \in I} \{\gamma_t(i)\} \tag{1.19}$$

Method as described by the Eq 1.17 and 1.19 is also called maximum a posteriori probability estimate (hereinafter "MAP estimate") which equals to the mode of the posterior distribution. If the posterior distribution is discrete, the mode is simply the value of a random variable with the highest probability. For continuous distributions the mode is a value of a random variable for which the likelihood/log-likelihood function attains its local maximum/maxima. Assuming now Z denotes the continuous random variable of hidden states and observation sequence X up to time t is present, the MAP estimate of the hidden state $\hat{Z}_{MAP}$ at a given point in time is:

$$\begin{aligned}\hat{z}_{t,MAP} &= \arg\max_{z_t} f(Z_t|X) \\ &= \arg\max_{z_t} \frac{f(X|Z_t)g(Z_t)}{\int_I f(X|Z_t)g(Z_t)\,dZ} \\ &= \arg\max_{z_t} f(X|Z_t)g(Z_t)\end{aligned} \tag{1.20}$$

Where g indicates probability density function of Z_t, so called prior distribution, and I is its domain. The second equality follows from the Bayes' Theorem. The result clearly holds for the discrete case:

$$\begin{aligned}\hat{z}_{t,MAP} &= \arg\max_{z_t} P(Z_t = i|X) \\ &= \arg\max_{z_t} \frac{P(X|Z_t = i)P(Z_t = i)}{\sum_{i=1}^{N} P(X|Z_t = i)P(Z_t = i)} \\ &= \arg\max_{z_t} P(X|Z_t = i)P(Z_t = i) \\ &= z_t^*\end{aligned} \tag{1.21}$$

However, the solution as such might not produce the most likely sequence of states given the sequence of observations. This is due to the fact that the individual estimates do not incorporate the transition probability between most likely states at time t-1 and t. Hence, it might be possible that some states are highly unlikely to transition into other that were evaluated as most likely. Fortunately, the mentioned shortcomings of such approach are easily solved by Viterbi algorithm.

In order to find the most likely sequence of hidden states $Z^* = z_1^*, z_2^*, ..., z_T^*$ given the observation sequence $X = x_1, x_2, ..., x_T$ it is necessary to maximise with respect to the whole sequence rather than individually to avoid less likely transitions as introduced. The algorithm therefore defines a variable as follows:

$$\delta_t(i) = \max_{Z_{t-1}} P(z_1, z_2, ..., z_t = i, x_1, x_2, ..., x_{t-1}, x_t|\lambda) \tag{1.22}$$

That is the most likely state sequence given the observation sequence up to time t. Moreover, for the purpose of the algorithm we also define variable $\psi_t(i)$ that stores the node of the incoming arc that leads to the most probable state path.

$$\psi_t(j) = \arg\max_{i \in I} \delta_{t-1}(i) p_{ij} \tag{1.23}$$

Then we can easily provide the steps to obtain the most likely state sequence:

1. Initialize the algorithm given the initial distribution of hidden states π:

$$\delta_1(i) = \pi_i b_i(o_1) \tag{1.24}$$
$$\psi_1(i) = 0 \tag{1.25}$$

   where we compute the initial values of above variables for each hidden state $i \in I$.

2. Recursive step:

$$\delta_1(i) = \arg\max_{i \in I} \delta_{t-1}(i) p_{ij} b_j(o_t) \tag{1.26}$$
$$\psi_t(j) = \arg\max_{i \in I} \delta_{t-1}(i) p_{ij} \tag{1.27}$$

   At each iterative step we update the time so that $t = t + 1$. The second step continues as long as the $t < T$ if that does not hold the third step terminates the algorithm and we backtrack the most likely state sequence. Although, recursive step is very similar to the induction step in the forward algorithm there exist a main difference between the two that lies in the fact that Viterbi algorithm uses maximization instead of summation over previous states.

3. Termination step for the last observation

$$P^* = \max_{i \in I} \delta_T(i) \tag{1.28}$$
$$q_T^* = \arg\max_{i \in I} \delta_T(i) \tag{1.29}$$

4. Backtrack the optimal state sequence:

$$z_t^* = \psi_{t+1}(z_{t+1}^*) \tag{1.30}$$

For $t = T-1, T-2, ..., 1$.

The resulting most likely state path can be visually viewed as in the form of a path within the trellis introduced in Fig. 1.2:

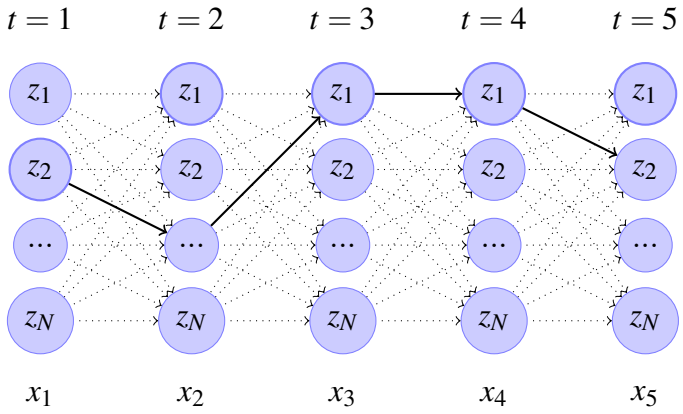


Fig. 1.3 Trellis of the observation sequence x_1,...,x_5 for the HMM. Bold lines illustrate the most likely sequence found by the Viterbi algorithm.

Chapter 2

Parameter Estimation for Hidden Markov Models

2.1 Expectation–Maximization algorithm

Also abbreviated as EM algorithm is an iterative approach for computing the maximum likelihood estimates. It is used in situations where incomplete data are present therefore a part of a complete data set is hidden and we may not be able to apply straightforward analytical procedures for computing maximum likelihood estimates as in a case of complete data.

In other words we want to find the best estimate of the parameters for which the observed sequence is the most likely. This is of great importance and efficiency for Hidden Markov models where we have a sample space of observed variables, let us denote it as X and a hidden discrete sample space Z. We assume that there is a mapping from X into Z, where $\forall x \in X$ is a realisation from sample space X and the mapping from X into Z in many-one, as seen in Figure 1. We also assume that given our model the realisations $z \in Z$ are not observable directly but only as a projection from X into Z.

Let us also denote $\theta \in \Theta$ which is a vector of parameters of the given distribution belonging to sampling distribution Θ. The aim of EM algorithm is essentially to find the best estimate of θ that maximises the likelihood function $L(\theta) = p(X|\theta)$,this is known as Maximum likelihood (ML) estimate of θ.

To visualise and apply the non-iterative approach, let us consider now the daily log returns of BTC/USDT trading pair in past 5 years. If we plot the histogram we may see that the log-returns may asymptotically follow normal distribution. Since the probability density function in such a case is unimodal and has only one global maximum, the logarithmic

transformation converts multiplicative structure into additive with the preservation of global maximum to be optimised by taking the partial derivative w.r.t. each parameter. First of all, we formulate the likelihood function of two-parametric normal distribution.

$$L(\mu,\sigma^2|x_1,...,x_N) = P(\mu,\sigma^2|x_1,...,x_N) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}} \tag{2.1}$$

$$l(\mu,\sigma^2|x_1,...,x_N) = \ln L(\mu,\sigma^2|x_1,...,x_N) \tag{2.2}$$

where $x_1,...,x_N$ is a vector of log-returns of length N and μ and σ^2 are parameters to be estimated.

$$l(\mu,\sigma^2|x_1,...,x_N) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2 \tag{2.3}$$

If we now take the partial derivative w.r.t. the parameter μ and σ^2 and set it to zero we obtain the ML estimate of the parameter as follows:

$$\frac{\partial}{\partial\mu} l(\mu,\sigma^2|x_1,...,x_N) = \frac{1}{\sigma^2}(\sum_{i=1}^{N} x_i - N\mu) \tag{2.4}$$

$$0 = \frac{1}{\sigma^2}(\sum_{i=1}^{N} x_i - N\mu) \tag{2.5}$$

$$\mu_{MLE} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.6}$$

$$\frac{\partial}{\partial\sigma^2} l(\mu,\sigma^2|x_1,...,x_N) = -\frac{N}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i-\mu)^2 \tag{2.7}$$

$$0 = -\frac{N}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i-\mu)^2 \tag{2.8}$$

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N} \tag{2.9}$$

Given the daily returns of BTC/USDT since 2017, ML estimate of the mean denoted by μ is 0.003 and σ^2 is 0.042. Moreover we may compare the ML fit with non-parametric estimate obtained by kernel density estimate that uses a sum of gaussian kernel functions for each sampled data point to estimate the smooth density given only data and prior bandwidth parameter h set to 0.2.

Kernel density estimate is calculated as follows:

$$\rho_K(y) = \sum_{i=1}^{N} K(y - x_i; h) \tag{2.10}$$

$$K(x;h) \propto e^{\left(-\frac{x^2}{2h^2}\right)} \tag{2.11}$$

where $K(x;h)$ denotes the Gaussian kernel function with bandwidth parameter h and $\rho_K(y)$ is a kernel density estimate at point y.

Fig. 2.1 *Candlestick graph of daily spot price of BTC/USD from March 2021 to March 2022 with subplot containing two lines for MACD and Signal line and a bar chart as their difference*

Although the assumption of complete data simplifies the analytical procedure of calculating the ML estimate in closed form, it may only be used by taken in consideration observable data. The task of EM algorithm can be used to iteratively compute the most likely parameter given the data, i.e. the parameter that most likely produced such data, but since the closed form solution is available at this point, it would be redundant to use iterative procedure since both would provide the same result. Nevertheless, the iterative EM algorithm given the fully observed complete data behaves in following way:

In order to estimate the single or vector of parameters θ we use log-likelihood function:

$$l(\theta|X) = \ln p(X|\theta) \tag{2.12}$$

Since the natural logarithm is strictly monotonic increasing function the value of θ maximises the log-likelihood as well as the likelihood function. Afterwards, in a simplistic sense, the EM algorithm iterates over possible values of θ to find the best estimate, i.e. until convergence criterion is satisfied:

$$|l(\theta^i|X) - l(\theta^{i-1}|X)| \leq \varepsilon \tag{2.13}$$

where the current estimate of θ in i^{th} iteration is denoted by θ^i and the convergence threshold ε.

However, certain part of the complete data is hidden in the case of HMM, we speak about observing the incomplete part. The likelihood of the complete data is therefore a joint probability of observed and hidden part of the data. Joint probability of the complete data may also be formulated by summing over all possible values of z.

$$l(\theta|X) = \ln p(X|\theta) = \ln \sum_z p(X, z|, \theta) \tag{2.14}$$

Alternatively, the goal of such an algorithm is to maximise the the loglikelihood of the complete data by estimating the optimal paramater or vector of paramaters denoted by θ. Hence the marginal probability of X given vector of hidden variables Z:

$$\ln p(X|\theta) = \ln \sum_{j=1}^{M} p(x, z = j|\theta) \tag{2.15}$$

Since traditional ML procedure aims to maximise the marginal log-likelihood of complete data, part of which we do not observe, it would be unnecessarily hard to maximise. It is however possible to introduce several assumptions that will eventually suffice in providing direct solution to the maximisation problem. Let us start by constructing the lower bound for the objective function as in Equation 2.15 that is easy to optimise with respect to parameters represented. In order to find the function q, we start by multiplying the marginal likelihood by $\frac{q(z)}{q(z)}$. Such expression will allow for a construction of artificial weights and with the use of Jensen's inequality:

$$\ln \sum_{j=1}^{M} q(z=j) \frac{p(x,z=j|\theta)}{q(z=j)} \geq \sum_{j=1}^{M} q(z=j) \ln \frac{p(x,z=j|\theta)}{q(z=j)} = L(\theta,q) \tag{2.16}$$

$$\ln p(X|\theta) \geq L(\theta,q), for \forall q \in Q \tag{2.17}$$

We have established the Jensen's inequality within the usage of concavity of logarithmic function. The constructed lower bound $l(\theta,q)$ is factorable into the expected value of the loglikelihood of complete data and entropy of probability function $q(x)$. Moreover, the maximisation of such likelihood function is completely determined by the former term since the latter is independent of θ.

$$l(\theta,q) = \sum_{j=1}^{M} q(z=j) \ln p(x,z=j|\theta) + \sum_{j=1}^{M} q(z=j) \ln \frac{1}{q(z=j)} \tag{2.18}$$

The optimization problem transforms into finding the function q for a fixed θ^{k-1}, given the k-th iteration, that maximizes $L(\theta,q)$:

$$q^k = \arg\max_q l(\theta,q) \tag{2.19}$$

In other words, we want to minimise the gap between the complete data log-likelihood function $l(\theta|X)$ and incomplete data $l(\theta,q)$ with respect to function q. Let us now elaborate more on the gap. The final expression that we shall arrive at gives a simplifying interpretation known as Kullback-Leibler divergence (hereafter KL divergence) measures the dissimilarities between two distributions and the measure may be interpreted as a geometrical statistical distance, it also is asymmetric and non-negative. Such measure is commonly used in image or signal processing in calculation of the expected excess surprise from using q as a probability distribution for our model given that the true or actual probability distribution is p. Substantially the measure is a difference of cross-entropy denoted by $H(p,q)$ and entropy by $H(p)$ which is always non-negative as a result of Gibb's inequality and is zero if and only if the two probability measure p and q are equal.

$$D_{KL}(p||q) = \sum_X p(x) \ln \frac{p(x)}{q(x)} \tag{2.20}$$

$$= \sum_x p(x) \ln \frac{1}{q(x)} - \sum_x p(x) \ln \frac{1}{p(x)} \tag{2.21}$$

$$= H(p,q) - H(p) \tag{2.22}$$

The lower bound introduced in Equation 2.18 and the complete data log-likelihood function $l(\theta|X)$ are obviously not the same following even intuitively from the fact that we are observing only the visible part of the data, the hidden part is still unknown. More rigorously, we have only defined the lower bound for the complete data log-likelihood function using arbitrary probability function $q(z)$, but the deeper examination of the lower bound with the use of KL divergence yields direct choice of the probability function $q(x)$. Let us decompose the lower bound in terms of the original log-likelihood function of the complete data:

$$l(\theta,q) = \sum_{j=1}^{M} q(z=j) \ln \frac{p(x,z=j|\theta)}{q(z=j)} \tag{2.23}$$

$$= \sum_{j=1}^{M} q(z=j) \ln \frac{p(z=j|x,\theta)p(x|\theta)}{q(z=j)} \tag{2.24}$$

$$= \sum_{j=1}^{M} q(z=j) \ln \frac{p(z=j|x,\theta)}{q(z=j)} + \sum_{j=1}^{M} q(z=j) \ln p(x|\theta) \tag{2.25}$$

$$= -D_{KL}(q(z)||p(z|x,\theta)) + \ln p(x|\theta) \tag{2.26}$$

Now we see that the statistical distance between the two likelihood functions is determined solely by the KL divergence. In order to find the minimal distance, ideally distance of zero "length", we simply set the function $q(x)$ equal the the $p(z|X,\theta)$. This might be the posterior distribution Adame ???

$$\ln p(x|\theta) = l(\theta,q) + \sum_{j=1}^{M} D_{KL}(q(z=j)||p(z=j|x,\theta)) \tag{2.27}$$

The last expression is just sum of KL divergences of function $g(z_i = j)$ and the posterior distribution of the latent variable z given the data and current parameter θ.

Recalling Equation 2.18, the optimisation goal lies in finding the probability distribution q that maximises $L(\theta,q)$ given the current estimate of the parameter θ, traditionally called as $E-step$. To extend and present a solution in finding the function q, following interpretation within the knowledge of Equation 2.26 is emphasised. We see that the maximum of the lower bound, i.e. of the log likelihood function, with respect to function $q(z)$ is identical to minimising the KL divergence introduced above because the first term in objective function to minimise is independent of $q(z)$:

$$q^{k+1} = \arg\max_q L(\theta_k, q) = \arg\max_q (\ln p(X|\theta) - \sum_{i=1}^{N} D_{KL}(q(z_i)||p(z_i|x_i,\theta))) \tag{2.28}$$

To visualise E-step, let us consider arbitrary log-likelihood function of complete data $\ln p(X|\theta)$ and $l(\theta^k, q^{k+1})$ for a fixed parameter of θ^k. Notice the visual representation of resulting selection of q^{k+1} as a solution to E-step. Consider also the new parameter estimate of θ^{k+1} which is a part of the next step called Maximisation step, also abbreviated as M-step.

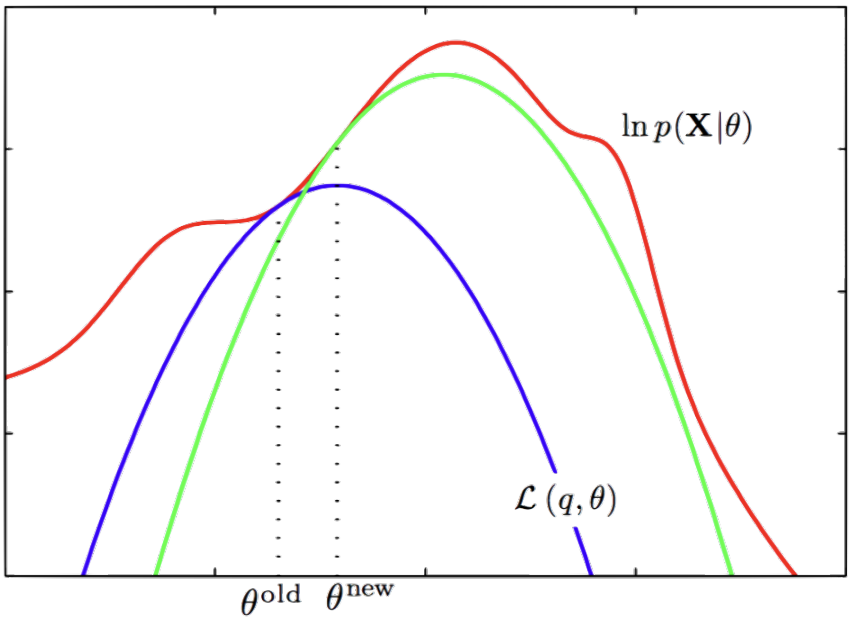


Fig. 2.2 *E-step as a problem of minimisation of KL divergence represented as a gap between two likelihood functions*

Until now we have considered the optimal manner in which we compute the conditional expectation $E_{Z|X,\theta_n}[\ln P(X,z|\theta)]$ by finding the appropriate function q. Next step is, as is typical for coordinate descent, selection of parameter θ_{k+1} by fixing the function q^{k+1} from E-step. The goal of M-step is to:

$$\theta^{k+1} = \arg\max_\theta L(\theta, q^{k+1}) \tag{2.29}$$

Thus, the EM algorithm involves two main steps:

1) Expectation step (E-step) - choose a function q, i.e. probability distribution, that maximises $L(\theta, q)$, which may be also viewed as computing the conditional expectation $E_{Z|X,\theta_n}[\log p(X,z|\theta)]$ based on the current parameter of θ.

2) Maximization step (M-step) - estimating the parameter θ_{k+1} that maximises the conditional expectation $E_{Z|X,\theta_n}[\log p(X,z|\theta)]$.

Both of these steps are repeated until convergence.

2.1.1 Baum-Welsch algorithm

In previous section we defined Expectation-Maximization algorithm used to compute Maximum Likelihood estimates given the incomplete data, i.e. supposing that part of the data is hidden. Two main steps are called E-step and M-step which were discussed generally but we need to establish direct connection to the estimation of the Hidden Markov model parameters. We will show that former step is easily computed given variables α, β and γ established in last chapter, where we defined Viterbi and Forward-Backward algorithm. These computations result in estimating the conditional probability distribution of hidden states given our observations sequence and model parameters $P(z_t = i|X,\theta^k)$.

In Chapter 1 we denoted the transition matrix as $\{p_{i,j\in I}\}$, i.e. probability of transitioning to state j given that we are at state i. The initial distribution of hidden states π and emission matrix b_j is used as well. First of all we need to specify the equation for complete data log-likelihood function for the Hidden Markov Model, which we can intuitively represent as joint probability distribution of observed and hidden data given our model parameters $\log p(X,z|\theta)$.

$$l(\theta|x) = \log \pi(z_i) + \sum_{t=1}^{n-1} \log p_{ij} + \sum_{t=1}^{n-1} \log b(z_t, x_t) \tag{2.30}$$

2.2 Observable states

There is a huge number of observable variables that one could abstract from cryptocurrency market. A possibility of discrete states within the given state space is plausible and feasible, it would, given our model constraints, provide poor inference since additional information would remain hidden. Imagine a situation where our observable states are defined as a relative change in price or a sudden drop/uprise in the traded volume on the exchange. In order to discreticize our states and construct transition and emission probabilities we are forced to

construct intervals that would well represent the boundaries upon which the model defines structure and predictions.

Assuming that the price increase in the idea predefined Hidden Markov Model will assume

However it is much more efficient to assume continuity in our predefined observable states. It is nowadays empirically proved , as in (citace) that using technical indicators as a predictors for the future spot price yields more accurate machine learning models. As will be demonstrated each of the technical indicators can be classified into several families of indicators, such as momentum, volume, volatility and cycle indicators. For our purposes we will consider mainly momentum indicators that are calculated using Open, High, Low, Close prices (hereinafter "OHLC") and Volume indicators. There is a huge variety of technical indicators to choose from therefore the selection was made according the most used and well known indicators or their transformed versions.

In our case we will consider following observable states that will be defined and elaborated on in the upcoming sections:

1) Moving Average Convergence/Divergence (MACD)

2) Stochastic Oscillator

3) Chaikin Oscillator

4) Relative Strength Index (RSI)

5) Aroon Oscillator

2.3 Moving Average Convergence Divergence

Also knows as MACD is a trend-following momentum indicator that represents the differences between two exponential moving averages (hereinafter "EMA"). The most common and traditional moving averages are 26-period EMA and 12-period EMA.

The indicator is often used with so called ""signal line" that is constructed as a 9-period EMA and is used as a trigger for a buy and sell signal. In practical application a trader decides to buy a stock if the signal line crosses MACD line from above and sell if it crosses from below, assuming simplistic trading strategy using only MACD. EMA also called exponentially weighted moving average is a type of moving average that differs from weighted moving average WMA by the distribution of weights to past observations. While WMA considers the linearly decreasing distribution of weights, the EMA assumes exponential decrease in

weights. Furthermore it is necessary to elaborate over the values of weights because it might not always be unambiguous. WMA distributes weights chronologically and linearly, e.g. 10-period WMA gives weight 1 to the earliest observation and 10 to the most recent observation, the case within EMA is often not that simple. The weights given to each observation are computed as $(1-\lambda)^i$ where $i \in \mathbb{N}_0$ and is bounded from above by the assumed period of interest, e.g. 3, 10, 26-period denoted as T for the sake of . As i increases identically with the time lag the value of weights decreases. The important role that ought to be questioned is the parameter λ that is defined as $\frac{k}{T+1}$ where k represents the so called "smoothing" parameter. Traders and analysts use value 2 for the smoothing parameter but the number may be defined on the interval $(0, T)$. Higher values of k mean bigger weights given to most recent observations.

The Figure 2.1 illustrates MACD line, signal line as well as "MACD histogram" which is displayed as a bar chart indicating the difference of the former ones. Traders use such a distance to identify whether the bullish or bearish momentum is high, i.e. bigger the distances of these two lines higher the price momentum.

MACD has its unfortunate limitations that mainly arise from the non-trending moments. When the price enters sideways movement the MACD histogram signals decreases distances between MACD and signal line, the trend reversal is possible but the price moves sideways which eventually results in false positive signal. Moreover when the price moves sideways for longer periods MACD may signal too many false trend reversals. The most common practice for traders is to combine MACD signals with other indicators such as Relative Strength index (RSI) that measures overbought or oversold market. The RSI uses average price gains and losses usually over 14 periods and yields values between 0 and 100, indicating overbought market for values 70 (80) to 100 and 30 (20) to 0 for oversold market. The idea is that when the distances between MACD line and signal line increase and RSI signals overbought market the trader might consider this as a strong trend reversal signal. The idea is that signals from MACD strategy often produce false signals when price suddenly moves sideways and RSI helps to indicate the false positive signal.

2.4 Stochastic Oscillator

A Stochastic Oscillator is a momentum indicator that compares the most recent closing price of a security with its predeceasing ones. Naturally, the range of preceding closing prices or the range of closing prices is 14 periods but it is regular that such an assumption is often edited to best fit the current needs of a trader. Also slight variation in taking the (weighted) moving average of the oscillator values is often introduced. The indicator is used to generate

Fig. 2.3 *Candlestick graph of daily spot price of BTC/USD from March 2021 to March 2022 with subplot containing two lines for MACD and Signal line and a bar chart as their difference*

trading signals that refer to the current overbought state of the market, which means that the indictor values range from 0 to 100 where the values closer to the number 0 indicate oversold market and inversely values closer to 100 overbought market.

Stochastic Oscillator is computed as follows:

$$SO_t = \frac{C_{t-1} - L_{14}}{H_{14} - L_{14}} \tag{2.31}$$

where C_{t-1} denotes the most recent closing price of a security, H_{14} and L_{14} are the highest and lowest price traded during 14-period interval respectively. SO_t is sometimes referred to as a "fast" stochastic indicator. As said before this interval may be changed arbitrarily. Traders also developed so called "slow" Stochastic Oscillator which is defined as a 3-period moving average of SO_t. Thus when Stochastic Oscillator crosses the smooth "Slow" Stochastic Oscillator a trading signal is generated.

Considering values above 80, the indicator signals overbought market and oversold market when the value drops below 20. Although it remains to hold true that the indicator often produces false indications that may be caused by periods of time where the price remains overbought/oversold for some time and trading with respect to such oscillator may

result in losses. It is rather recommended to observe the values of stochastic oscillator and use it for trend reversal indication.

Fig. 2.4 *Candlestick graph of daily spot price of BTC/USD from March 2021 to March 2022 with subplot with line indicating Stochastic Oscillator*

2.5 Chaikin Oscillator

Chaikin Oscillator is a momentum based indicator of the Accumulation/Distribution Line (hereinafter "A/D line"), which is a cumulative indicator that aims to identify potential divergences between stock price and trading volume. The oscillator is calculated as a difference between 3- day and 10-day Exponential Moving Average of A/D line.

The calculation of the Chaikin Oscillator may be broken down into several steps:

(i) First of all we ought to calculate the Money Flow Multiplier for each time step denoted by "N".

$$N_t = \frac{(Close_t - Low_t) - (High_t - Close_t)}{High_t - Low_t} \tag{2.32}$$

(ii) Now we may multiply N_t by the trading volume in given period of time to get the Money Flow Volume denoted as M_t. With that we recursively construct the A/D line as:

$$ADL_t = M_{t-1} + M_t \tag{2.33}$$

(iii) Given the constructed A/D line, we compute the Chaikin Oscillator values as a difference of 3-day and 10-day exponential moving averages.

$$CO_t = \frac{\sum_{i=0}^{3}(1-\alpha)^i * Close_{t-i}}{\sum_{i=0}^{3}(1-\alpha)^i} - \frac{\sum_{i=0}^{10}(1-\beta)^i * Close_{t-i}}{\sum_{i=0}^{10}(1-\beta)^i} \tag{2.34}$$

where we assume that the weights denoted as α and β are computed as $2/(days+1)$. Numerator as a smoothing factor is often declared as 2. However the indicator may be set to absolutely different number between 0 and 1 according to the needs and assumptions made by the trader/analyst, therefore setting the parameter close to 1 is putting more weight to the most recent price.

One way to interpret the indicator is to trade with respect to the time when the Chaikin Oscillator crosses zero from below and above which signals buy and sell signals respectively.

Fig. 2.5 *Candlestick graph of daily spot price of BTC/USD from March 2021 to March 2022 with subplot indicating Chaikin Oscillator*

2.6 Relative Strength Index

Given the recent price changes Relative Strength Index measures the its magnitude in order to indicate the overbought or oversold market. In technical analysis such indicator is represented by an oscillator ranging from values 0 to 100. Empirically it was determined that values above 70 and below 30 signal overbought and oversold asset respectively. Therefore we may also use RSI to produce buy and sell signals from former logic, which means that when the RSI crosses 30 from below, buy signal is generated as well as sell signal in case it crosses 70 from above. We also could measure the strength and continuation of the trend for cases in which the RSI crosses value of 50. Such interpretation results from the RSI formula where the value of 50 means that the average gain equals the average loss in the last period.

The formula below explains the procedure within which the values of RSI are calculated. It is obvious that the RSI rises as the number of positive closing prices increases, i.e. the relative change in prices is positive, and falls if otherwise. The standard time interval for the calculation is 14 preceding periods with respect to t, hereby denoted as T.

$$RSI_t = 100 - \frac{100}{1 + r_t} \tag{2.35}$$

where r is a ratio of average gains and losses as follows:

$$r_t = \left| \frac{\sum_{i=1}^{T} \left(\frac{P_{t-i+1}}{P_{t-i}} - 1\right) \mathbb{1}_{[P_{t-i+1} - P_{t-i} > 0]}}{\sum_{i=1}^{T} \left(\frac{P_{t-i+1}}{P_{t-i}} - 1\right) \mathbb{1}_{[P_{t-i+1} - P_{t-i} < 0]}} \right| \tag{2.36}$$

given that P_t denotes the value of an asset at time t.

As stated before there are several drawbacks of using the RSI as a trading indicators only by itself, it happens that the price usually rises and stays overbought for a substantial period of time in times of significant and strong bullish trend. RSI as an oscillator is used as a auxiliary trading tool below the price chart:

Fig. 2.6 *Candlestick graph of daily spot price of BTC/USD from March 2021 to March 2022 with a subplot containing RSI*

2.7 Aroon Indicator

Aroon indicator is used for trend reversal identification and a measure of its strength. Indicator is composed out of two lines aroon up and aroon down that measure the time between new highs or lows respectively. Alternatively, they measure the strength of a bullish or bearish trend. Obviously the main idea of the indicator is based upon the fact that bullish trends are naturely formed by subsequently creating new highs while bearish trends form new lows. Aroon Up and Aroon Down are computed as follows:

$$AroonUp = \frac{25-h}{25} * 100 \tag{2.37}$$

$$AroonDown = \frac{25-l}{25} * 100 \tag{2.38}$$

Where h represents the number of periods from the last 25-period High and l the number of periods from last 25-period Low.

The interpretation of the indicator is very intuitive since the situation in which the Aroon Up line is above Aroon Down line signals bullish trend and when these two lines cross the signal of the trend reversal is generated. That also implies that for higher values of Aroon Up the bigger the strength and for lower values the uptrend is weaker and vice versa. In practice the crossover of these two lines is what generates the buy or sell signals, i.e. if Aroon Up crosses Aroon Down line from below a buy signal is generated and vice versa.

Although, Figure 2.4 graphically illustrates the Aroon Up and Down lines well, it is simpler to transform these two lines into one oscillator that would produce buy or sell signal in the case of zero crossover from above and from below. That is achieved by subtracting Aroon Up and Aroon Down line creating Aroon Oscillator.

Fig. 2.7 *Candlestick graph of daily spot price of BTC/USD from March 2021 to March 2022 with subplots containing two lines for Aroon Up/Down Indicator and Aroon Oscillator*

2.8 Kalman Filters

_

Chapter 3

Gaussian Mixture Models

3.1 Mixture Distributions

As shown in the last chapter the joint probability distribution of the complete data is not strictly concave thus we can not guarantee to converge to the global optima. One thing that we can do is to initialise the parameters multiple times randomly to see if the maxima of the likelihood function increases. Fig 2.2 appropriately displays the shape of the joint probability distribution of the complete data. There are visible multiple local optima of the log-likelihood function caused by the hidden states. Such probability distributions are often called mixture distributions.

They were implicitly introduced in the last chapter while estimating the parameters of the Hidden Markov Model. Although, most of the time we consider unimodal distributions for our data since it may be justifiable empirically or considering the mixture distributions would unnecessarily complicate the computation for an indistinguishable improvement, the statistical inferences about the subpopulations within the population require such tools in cases where the subpopulations significantly differ. However, in these models we pose no requirement of the knowledge about the number of subpopulations within the population or their actual distribution. They are interpreted the same way as for the HMM as hidden variables.

References

[1] Ancey, C., Coussot, P., and Evesque, P. (1996). Examination of the possibility of a fluid-mechanics treatment of dense granular flows. *Mechanics of Cohesive-frictional Materials*, 1(4):385–403.

[2] Read, C. J. (1985). A solution to the invariant subspace problem on the space l_1. *Bull. London Math. Soc.*, 17:305–317.

Appendix A

How to install LaTeX

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from
   https://www.tug.org/texlive/
2. Download WinCDEmu (if you don't have a virtual drive) from
   http://wincdemu.sysprogs.org/download/
3. To install Windows CD Emulator follow the instructions at
   http://wincdemu.sysprogs.org/tutorials/install/
4. Right click the iso and mount it using the WinCDEmu as shown in
   http://wincdemu.sysprogs.org/tutorials/mount/
5. Open your virtual drive and run setup.pl

or

Basic MikTeX - TeX distribution

1. Download Basic-MiKTeX(32bit or 64bit) from
   http://miktex.org/download
2. Run the installer
3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin)
   and choose Package Manager

4. Select or search for packages to install

TexStudio - TeX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Run the installer

Mac OS X

MacTeX - TeX distribution

1. Download the file from
   https://www.tug.org/mactex/

2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - TeX editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Extract and Start

Unix/Linux

TeXLive - TeX distribution

Getting the distribution:

1. TexLive can be downloaded from
   http://www.tug.org/texlive/acquire-netinstall.html.

2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

   ```
   mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
   ```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

   ```
   cd /your/download/directory
   ./install-tl
   ```

4. Enter command 'i' for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64bit OS

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
```

```
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CentOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.