Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Matematyczny
specjalność: analiza danych

Mikołaj Słupiński

Hidden Markov Models in ultra-high frequency time series processing

Praca magisterska
napisana pod kierunkiem
dr. hab. Pawła Lorka

Wrocław 2020 r.

Contents

Chapter 1

Introduction

Algorithmic trading is a method of executing orders using automated pre-programmed trading instructions accounting for variables such as time, price, and volume [2]. In recent years the number of transactions on stock markets made by the machines is on the rise. It was estimated that at the London Stack Exchange over 40% of orders in 2006 were executed by algorithms [2]. According to the recent data, over 80% of trading in the FOREX market was performed by trading algorithms rather than humans [3].

In most modern financial markets, trade occurs via a continuous double-auction mechanism called a limit order book (LOB) [8]. During last few years much work was invested into developing methods allowing fast and accurate predictions of changes in limit order books [10, 17, 8, 9]. Techniques used varied from the more classical ones (logistic regression [7], SVMs) to the modern ones (convolutional neural networks, recurrent neural networks, transformers). The main task explored by those models is the prediction of mid-price change.

To the best of our knowledge, despite Hidden Markov Models (HMM) being used in financial modelling, there is no Hidden Markov Model-based approach for mid-price change prediction. In this work the novel approach to mid-price modelling is presented.

Chapter 2

Problem statement

2.1 Basic definitions

The problem of LOB classification gained popularity in recent years [8, 7, 18, 10, 9]. Researchers from different backgrounds try to apply their knowledge of economics, mathematics, computer science and physics in order to develop novel prediction methods.

Nowadays electronic order books are in use in over half of the financial markets [8]. Trade in an LOB occurs via an auction mechanism by which interested parties submit orders.

By an *order* x we understand the triple (p_x, ω_x, t_x) submitted with price p_x and size ω_x at a time t_x. In case $\omega_x > 0$ an offer is a commitment to sell up to $|\omega_x|$ units of the asset at a price no less than p_x. Analogously, for $\omega_x < 0$ an offer is an obligation to buy up to $|\omega_x|$ units of the asset at a price no greater than p_x.

Whenever somebody submits a sell order x, a LOB's algorithm checks whether there exists a possible match to an active buy order y such that $p_y \geq p_x$. If we can find such order the matching happens immediately. When it's not possible x remains *active* until it is matched with an opposing order or is *cancelled*. Analogous mechanism works for buy orders.

Those kind of orders are called *limit orders*. The LOB $\mathcal{L}(t)$ is the set of all active orders for a

given asset on a given platform at a given time t.

The highest stated price among active buy orders is called *bid price*, which we define as:

$$b(t) := \max_{\{x \in \mathcal{L}(t) | \omega_x < 0\}} p_x,$$

similarly we call the lowest stated price among active sell orders, the *ask price* which we denote as

$$a(t) := \min_{\{x \in \mathcal{L}(t) | \omega_x > 0\}} p_x.$$

The bid price and ask price are collectively called the best quotes. *The bid-ask spread* at time t is

$$s(t) := a(t) - b(t).$$

Our main object of interest is the *mid-price* at a time t given by

$$m(t) := \frac{a(t) + b(t)}{2}.$$

We say that a price p is on the sell side of $\mathcal{L}(t)$ if $p \geq a(t)$, on the buy side of $\mathcal{L}(t)$ if $p \leq b(t)$ or inside the bid-ask spread if $b(t) < p < a(t)$. On general, LOBs have two *resolution parameters*: the tick size $\pi \geq 0$, which specifies the smallest permissible price interval between different orders, and the *lot size* $\sigma > 0$, which specifies the smallest amount of the asset that can be traded.

Because the tick size cannot be negative, we treat the price axis of LOB as a one-dimensional lattice, whose points correspond to the natural multiples of π. That way we can regard LOB as a set of queues, each of which consists of active orders.

We can specify the total number of active sell orders at price p and time t as

$$n^a(p, t) := \sum_{\{x \in \mathcal{L}(t) | \omega_x > 0, p_x = p\}} \omega_x$$

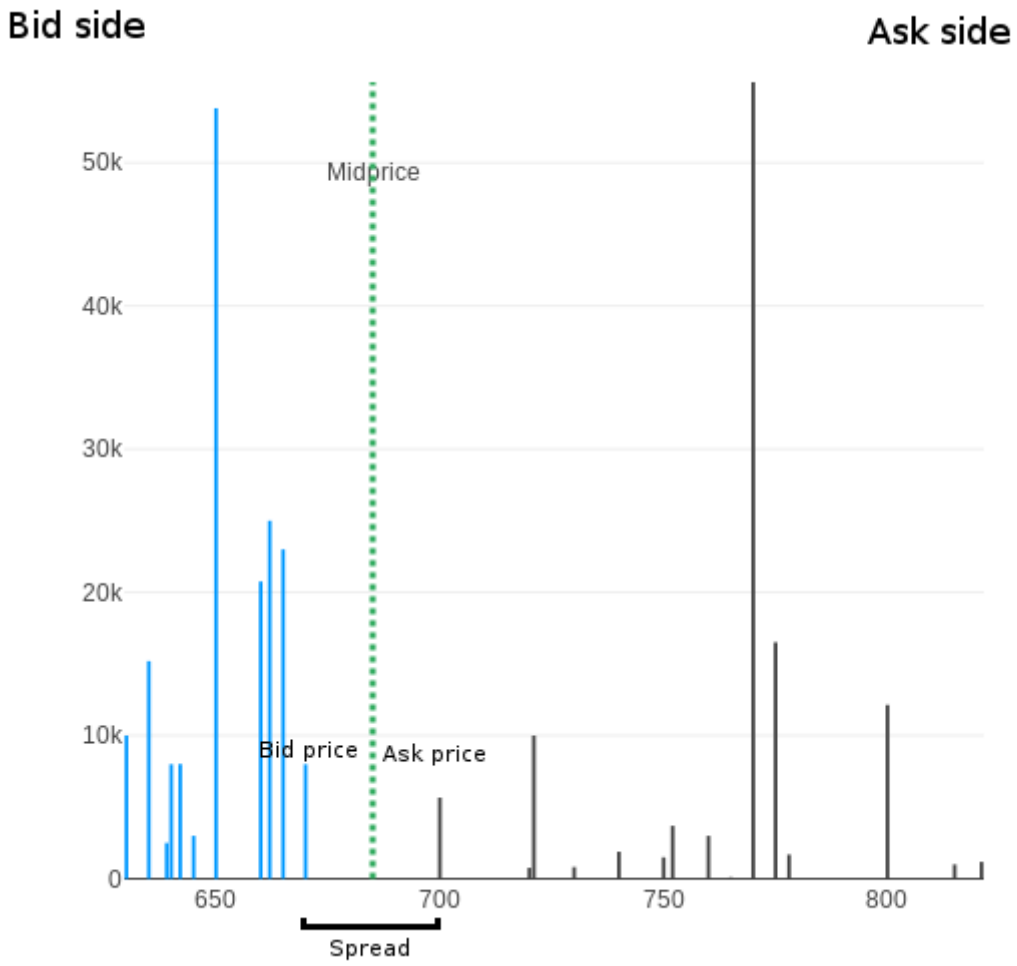

Figure 2.1: Visualisation of LOB.

and the total size of active buy orders

$$n^b(p,t) := \sum_{\{x \in \mathcal{L}(t) | \omega_x < 0, p_x = p\}} |\omega_x| .$$

As one can easily conclude, the value of $a(t)$ drops each time a new sell limit order arrives inside the spread and rises whenever the total volume of sell limit orders are price $a(t)$ drops to 0. The value of $b(t)$ behaves analogously. The value of $m(t)$ changes according to the change of $a(t)$ or $b(t)$.

For more information regarding limit orders please refer to [8], [7].

2.2 Queue Imbalance in a LOB

In their article M. Gould et al. [7] argued that the *queue imbalance* is particularly useful feature in LOB classification. We denote it's value at a specified time t by the formula

$$I(t) := \frac{n^b\left(b(t),t\right) - n^a\left(a(t),t\right)}{n^b\left(b(t),t\right) + n^a\left(a(t),t\right)}.$$

The specified quantity measures the normalized difference between $n^b\left(b(t),t\right)$ and $n^a\left(a(t),t\right)$.

Chapter 3

Existing methods

There were many attempts to predict mid-price movements in the recent years, using wide variety of approaches ranging from logistiuc regression to deep neural networks. There is no point in presenting all of them, but to better understand the problem it is good to review the approach taken by M. Gould et al. [7] based on the queue imbalance as it gave foundations for the newer, more sophisticated, methods.

3.1 Data and sample construction

The data studied by M. Gould et al. in [7] originated from the LOBSTER database, which provided an event-by-event description of the temporal evolution of the LOB for each stock listed on Nasdaq. To obtain their results, they limit the dataset to time-series for a subset of 10 liquid stocks during the entire year of 2014.

The Nasdaq platform operates from 9:30 to 16:00 on each working day. They excluded the first and last 30 minutes of each trading behaviour that can occur shortly after the opening auction or shortly before the closing auction. Concluding, the dataset consisted of 252 trading days and activity from 10:00 to 15:30.

For each stock and each trading day in the sample, there was created an ordered set T of times

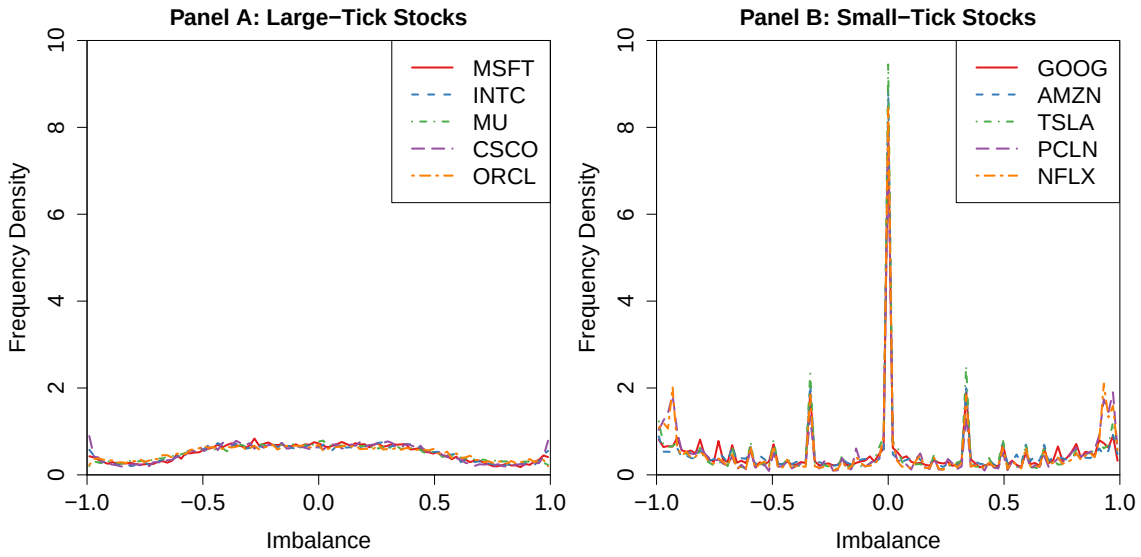

Figure 3.1: Histograms of I for each of the 10 stocks in LOBSTER sample. The left panel presents the results for large-tick stocks and the right panel presents the results for small-tick stocks.

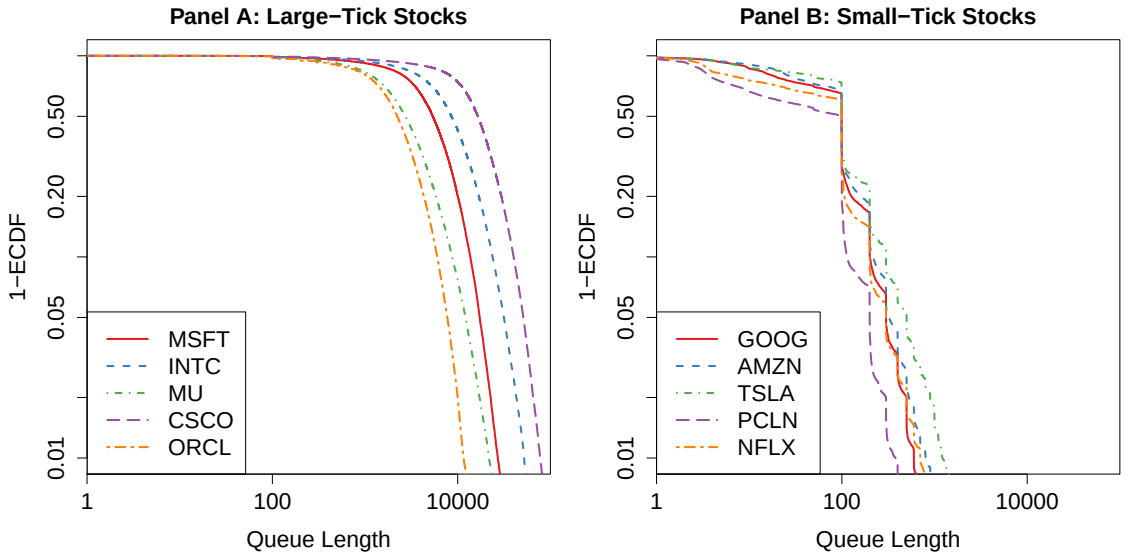

Figure 3.2: Empirical cumulative density functions (ECDFs) of the best-quote queue lengths $n^b(b(t),t)$ and $n^a(a(t),t)$. The plots show the survivor functions (i.e., $1-$ ECDF) in doubly logarithmic coordinates.

at which the mid-price changes,

$$T = \left\{ t | m(t) \neq \lim_{\varepsilon \downarrow 0} m(t - \varepsilon) \right\}.$$

We denote the times in T as the increasing sequence $t_1, t_2, t_3, \ldots, t_N$, additionally t_0 is the time of the first event for the given stock on the given day.

Let

$$y_i := \begin{cases} 1, & \text{if } m(t_i) > m(t_{i-1}) \\ 0, & \text{if } m(t_i) < m(t_{i-1}) \end{cases}$$

be the indicator variable describing whether or not the mid-price movement as $t_i \in T$ was upwards or downwards.

As one can observe the study of price change is restricted simply to the direction of price change, rather than the signed change in mid-price. It is caused by the simple fact that the magnitude of such price changes are affected by more factors than only $n^b(b_t, t)$ and $n^a(a_t, t)$. It is worth noting here that we only say that the price will change, not how big the change will be.

For each time $t_i \in T$, the time $\tilde{t}_i$ was chosen uniformly at random in the open interval (t_{i-1}, t_i). Let's denote imbalance as:

$$I_i = I\left(\tilde{t}_i\right).$$

3.2 Prediction

To perform a binary classification, authors seek to estimate a function $\hat{y}$ that maps queue imbalance onto some subset of $\mathbb{R}$ and threshold value $y* \in \mathbb{R}$ such that:

- if $\hat{y}(I_i) > y*$, the they predict $y_i = 1$,
- if $\hat{y}(I_i) < y*$, the they predict $y_i = 0$,

- if $\hat{y}(I_i) = y*$, the they predict $y_i = 1$ or $y_i = 0$, each with probability $1/2$.

The question can also be considered in terms of a probabilistic classifier, which, for a given queue imbalance I_i, seeks to predict the probability that $y_i = 1$. Let's note here that whether we choose the function y_i for binary classifier such that

$$\hat{y} : (-1, 1) \to [0, 1]$$

and if we interpret $\hat{y}_i$ as

$$\hat{y}_i := \mathbb{P}\left(y_i = 1 | I_i\right),$$

then we can use the same function $\hat{y}$ to perform both binary classification and probabilistic prediction.

Let us consider two queue imbalances $I_{i'}$ and $I_{j'}$ chosen randomly with uniform distribution, $I_{i'}$ among all observations for which $y_i = 1$ and respectively $I_{j'}$ among all observations for which $y_i = 0$.

If I_i provides enough information to perform binary classification, then $\hat{y}$ will satisfy

$$\mathbb{P}\left(\hat{y}_{i'} > \hat{y}_{j'}\right) > 1/2.$$

On contrary if I_i doesn't provide enough predictive power to perform binary classification, $\hat{y}$ will satisfy the equations

$$\mathbb{P}\left(\hat{y}_{i'} > \hat{y}_{j'}\right) = \mathbb{P}\left(\hat{y}_{j'} > \hat{y}_{i'}\right) = 1/2.$$

Analogously, if I_i enables us to do probabilistic classification

$$\mathbb{P}\left(\hat{y}_{i'} > 1/2\right) > 1/2 \text{ and } \mathbb{P}\left(\hat{y}_{j'} > 1/2\right) < 1/2.$$

However, if I_i doesn't supply us with enough predictive power to perform such classification, then $\hat{y}$ satisfies

$$\mathbb{P}\left(\hat{y}_{i'} > 1/2\right) = 1/2 \text{ and } \mathbb{P}\left(\hat{y}_{j'} > 1/2\right) = 1/2.$$

To estimate the function $\hat{y}$, we perform a logistic regression of y_i onto I_i. Namely,

$$\hat{y}(I) = \frac{1}{1 + e^{-(x_0 + Ix_1)}},$$

where x_0 and x_1 are the coefficients we want to estimate.

Authors compared output of their binary classifier to a simple null model in which they assume I doesn't bring any predictive power. That means

$$\hat{y}(I) = 1/2 \text{ for all } I.$$

To asses the predictive power of their fitted logistic regressions for performing probabilistic classification, they use function $\hat{y}$ to make out-of-sample predictions $\hat{y}_i$ for each I_i in the testing set, and compute residuals:

$$r_i := \hat{y}_i - y_i.$$

Then they computed the mean squared residual (MSR) across all observations in the testing set. For the null model $y_i = 0.5$ for all i, so r_i is given by

$$r_i = \begin{cases} -1/2, & \text{if } y_i = 1 \\ 1/2, & \text{if } y_i = 0 \end{cases}$$

thus MSR is exactly 0.25.

3.2.1 Training and testing data

The exact number of mid-price movements that occur in a single trading day varies considerably between different days, also differs between different stocks. To ensure the reduction of that disproportion between data samples, authors draw a random subsample with a fixed size among the N possible choices in the set T. For the results presented below they used a random subsample size of 100. For each stock, they aggregated the subsamples from each of the 252

different trading days to produce an aggregated data set of 25200 data points.They randomly divided each stock's aggregated dataset into two disjoint subsets: a training set, which contained 80% of the data (i.e., 20160 data points), and a testing set containing 20% of the data (i.e., 5040 data points). They perform the fits of logistic regressions and local logistic regressions using the training set (i.e., "in sample"), then evaluate the predictive power of these fits using the testing set (i.e., "out of sample").

3.2.2 Results

Full results are presented in Table 3.1, Table 3.2, Table 3.3 and Table 3.4. We may conclude that Queue Imbalance brings some predictive power.

In each tested case, the out-of-sample ROC curves lie above the grey line on Figure 3.3, which means that the logistic regression fits outperform the out-of-sample predictive power of the fully random null model.

For large-tick stocks, the the ROC AUC ranges from about 0.7 to about 0.8. For small-tick stocks, the results are significantly lower, and range from about 0.6 to about 0.65. Nevertheless, in both cases obtained results indicate that Queue Imbalance provides a substantial improvement in the out-of-sample predictive power of the binary classifier. In order to verify that these results are not caused by over-fitting, the authors also calculated the area under the corresponding ROC curves for the in-sample fits.

For large-tick stocks, the values of MSR range from about 0.18 to about 0.2. For small-tick stocks, the values of MSR range from about 0.235 to about 0.245. For the random null model, the MSR is exactly 0.25. Therefore, when compared to the random model, the logistic regression fits provide a reduction in MSR of about 20% to 30% for large-tick stocks, and about 2% to 6% for small-tick stocks. For all stocks, the in-sample values of the MSR are very similar to the corresponding out-of-sample values, which confirms that the logistic regressions do not suffer from over-fitting.

When it comes to local logistic regression, for large-tick stocks, their regression curves are

	x_0		x_1	
	Estimate	St. Err.	Estimate	St. Err.
MSFT	0.01	(0.02)	2.49	(0.04)
INTC	0.03	(0.02)	2.56	(0.04)
MU	0.03	(0.02)	2.03	(0.04)
CSCO	0.06	(0.02)	2.73	(0.04)
ORCL	0.05	(0.02)	2.25	(0.04)
GOOG	0.03	(0.01)	0.54	(0.02)
AMZN	0.03	(0.01)	0.85	(0.03)
TSLA	−0.01	(0.01)	0.60	(0.03)
PCLN	0.03	(0.01)	0.50	(0.02)
NFLX	0.01	(0.01)	0.65	(0.02)

Table 3.1: Maximum likelihood estimates of the intercept x_0 and coefficient x_1 in the logistic regression fits of $\hat{y}$ versus I. Adopted from [7].

almost monotone increasing functions of I, which suggests that larger values of I_i correspond to larger values of y_i. Once again, the local logistic regressions predict that the probability of an upward price movement ranges from approximately 0.8 to 0.9 when I is close to 1. In contrast to the logistic regression curves, however, the local logistic regression curves suggest that the behaviour of the system exhibits 2 different regimes. For values of I from about -0.25 to about 0.25, the $\hat{y}$ curve is quite steep, meaning that when the bid and ask queues are of similar length, a relatively small difference in the queue imbalance corresponds to a considerable change in the probability that the next price movement will be upwards. Outside of this part, the level of steepness of the $\hat{y}$ curve decreases largely. Therefore, for values of I less than about -0.25 or greater than about 0.25, a further difference in queue imbalance corresponds to a smaller change in the probability that the next price movement will be upwards. For small-tick stocks, the local logistic regression curves predict that the probability of an upward price movement is about 0.6 when I is close to 1. For all small-tick stocks except PCLN, the local logistic regression curves are nonmonotonic in I. This result is rather puzzling, because it suggests that there are cases when a weaker imbalance increases the probability of an upward mid-price movement. This counter-intuitive finding brings into question whether the fitted local logistic regressions $\hat{y}$ really detect a meaningful relationship, or simply over-fit to noise.

For the in-depth analysis of results and its predictive power, please refer to full publication [7].

	In Sample	Out of Sample
MSFT	0.781	0.762
INTC	0.791	0.798
MU	0.747	0.752
CSCO	0.802	0.805
ORCL	0.770	0.770
GOOG	0.592	0.581
AMZN	0.635	0.642
TSLA	0.602	0.602
PCLN	0.592	0.583
NFLX	0.616	0.627

Table 3.2: Area under the ROC curves (see Figure 3.3). Adopted from [7].

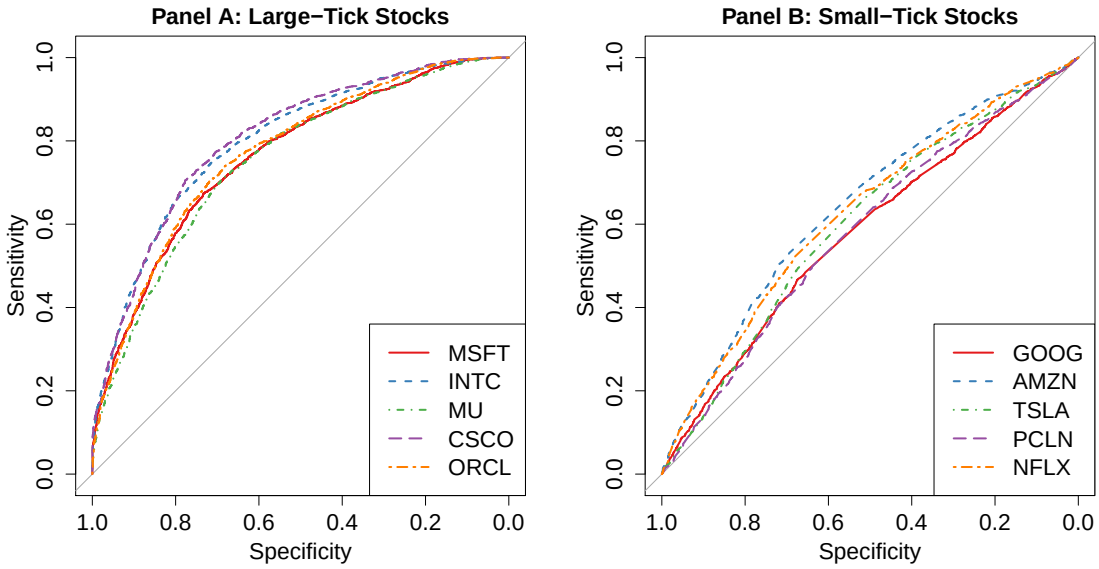

Figure 3.3: Receiver operating characteristic (ROC) curves for the out-of-sample predictive power of a binary classifier, based on the logistic regression fits of $\hat{y}$ versus I. Adopted from [7].

	In Sample	Out of Sample
MSFT	0.191	0.198
INTC	0.186	0.183
MU	0.204	0.202
CSCO	0.181	0.180
ORCL	0.195	0.195
GOOG	0.244	0.246
AMZN	0.237	0.235
TSLA	0.243	0.243
PCLN	0.244	0.245
NFLX	0.240	0.239

Table 3.3: Mean squared residual r_i of the logistic regression fits of $\hat{y}$ versus I. Adopted from [7].

	Area Under ROC Curve		Mean Squared Residual	
	In Sample	Out of Sample	In Sample	Out of Sample
MSFT	0.781	0.762	0.190	0.198
INTC	0.791	0.798	0.185	0.183
MU	0.747	0.752	0.204	0.202
CSCO	0.802	0.805	0.181	0.179
ORCL	0.770	0.770	0.195	0.195
GOOG	0.592	0.581	0.244	0.246
AMZN	0.636	0.642	0.236	0.235
TSLA	0.602	0.603	0.242	0.242
PCLN	0.592	0.583	0.244	0.245
NFLX	0.616	0.627	0.240	0.239

Table 3.4: Statistics describing the predictive power of the local logistic regression fits of $\hat{y}$ versus I. Adopted from [7].

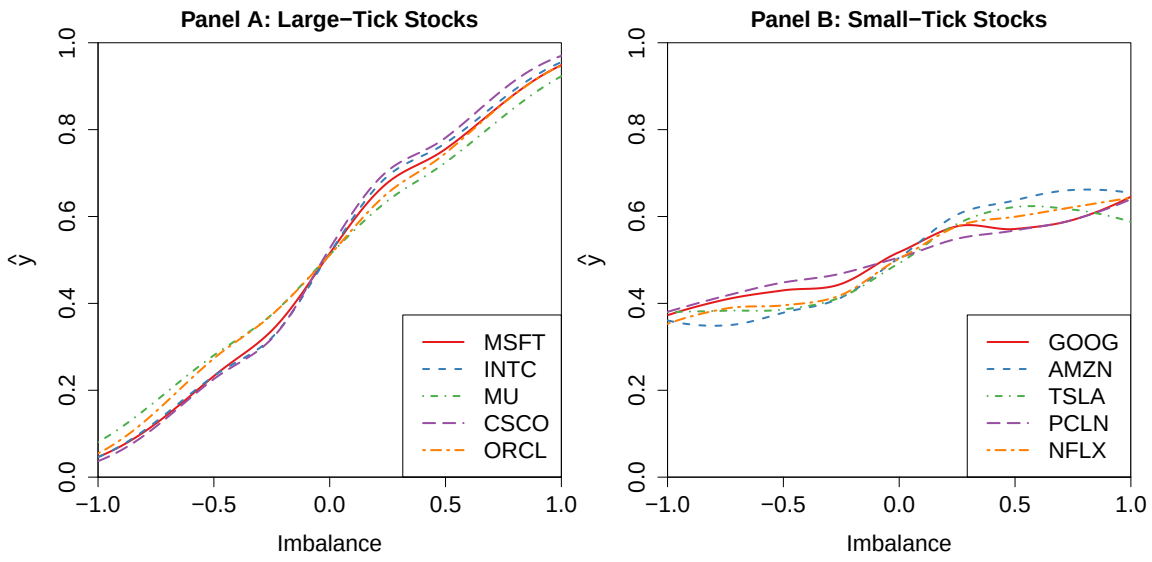

Figure 3.4: Local logistic regression fits of $\hat{y}$ versus I. For each curve, we use a tricube weight function and a nearest-neighbour bandwidth of 0.65. The left panel shows the results for large-tick stocks and the right panel shows the results for small-tick stocks. Adopted from [7]

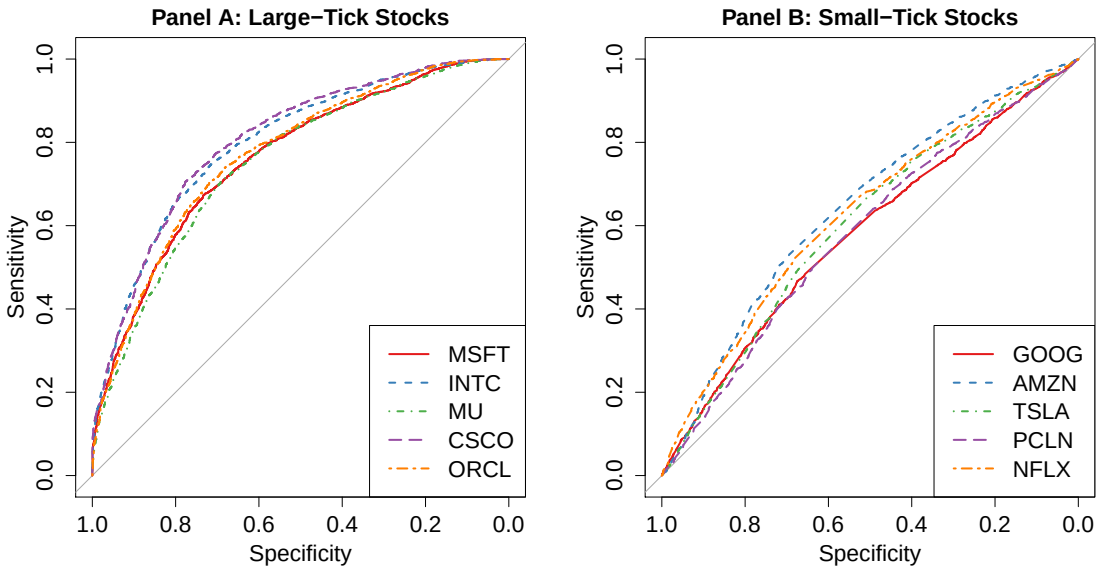

Figure 3.5: Receiver operating characteristic (ROC) curves for the out-of-sample predictive power of the local logistic regression fits of $\hat{y}$ versus I. The left panel shows the results for large-tick stocks and the right panel shows the results for small-tick stocks. The grey line in each plot denotes the expected performance of the null model, which assumes that the probability of an upward price movement is always equal to 1/2, irrespective of the queue imbalance. Adopted from [7].

Chapter 4

Hidden Markov Models

4.1 Markov model

Surprisingly many processes happening in our lives we can model using a stochastic process depending only on the immediately preceding state of the process. One very classical example is "random walk"("birth and death chain") [6]. The state space is $\{0, 1, 2, \ldots, L\}$ The chain goes to the right with probability p and to the left with probability $1 - p$, additionally we assume that going left from 0 or going right from L means staying put.

Turning now to the formal definition, we say that X_n is a discrete time Markov chain with transition matrix $p(i, j)$ if for any $j, i, i_{n-1}, \ldots i_0$

$$P\left(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\right) = p(i, j). \tag{4.1}$$

Above equation explains well what we mean by phrase "given the current state X_n, any other information about the past is irrelevant for predicting X_{n+1}". In formulating 4.1 we have restricted our attention to the temporally homogeneous case in which the transition probability

$$p(i, j) = P\left(X_{n+1} = j | X_n = i\right)$$

does not depend on the time n.

To better illustrate how the Markov process functions, let's look at the simple example.

Example 1. *Let r, s, w be three states, namely rainy, sunny and windy. The weather can transition from one state at time n to another at time $n+1$ with probability $p(x_n, x_{n+1})$.*

We can assume that the model is constructed during the spring and it is sunny most of the time. We can define a state transition matrix:

$$\mathbf{A} = \begin{array}{c|ccc} & \textit{rainy} & \textit{sunny} & \textit{windy} \\ \hline \textit{rainy} & 0.1 & 0.1 & 0.8 \\ \textit{sunny} & 0.2 & 0.7 & 0.1 \\ \textit{windy} & 0.2 & 0.7 & 0.1 \end{array}$$

4.2 Observations

Following [5] we define HMM as follows:

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$$

where S is our state alphabet set, and V is the observation alphabet set:

$$S = (s_1, s_2, \cdots, s_N)$$
$$V = (v_1, v_2, \cdots, v_M)\,.$$

We define Q to be a fixed state sequence of length T, and corresponding observations O:

$$Q = q_1, q_2, \cdots, q_T$$
$$O = o_1, o_2, \cdots, o_T$$

A is a transition matrix, storing the probability of state j following state i. Note here that the state transition probabilities are independent of time:

$$\mathbf{A} = [a_{ij}], a_{ij} = P(q_t = s_j | q_{t-1} = s_i),$$

B is the observation matrix, storing the probability of observation k being produced from the state j, independent of t:

$$\mathbf{B} = [b_i(k)], b_i(k) = P(x_t = v_k | q_t = s_i),$$

π is the initial probability matrix:

$$\pi = [\pi_i], \pi_i = P(q_1 = s_i).$$

Here, two assumptions are made by the model. The first is called the Markov assumption. It states that the current state is dependent only on the previous state. This represents the memory of the model:

$$P(q_t | q_1 \ldots, q_{t-1}) = P(q_t | q_{t-1}).$$

The second one, called the independence assumption. states that the output observation at time t is dependent only on the current state and it is independent of previous observations and states:

$$P(o_t | o_1, \ldots, o_{t-1}, q_1, \ldots q_t) = P(o_t | q_t).$$

Example 2. *Let's get back to our model in Example 1. Let's assume that we want to model one's behavior according to the weather outside. Let's name our hypothetical person Bob. Bob enjoys four activities: walking in park, shopping, cleaning flat and visiting girlfriend.*

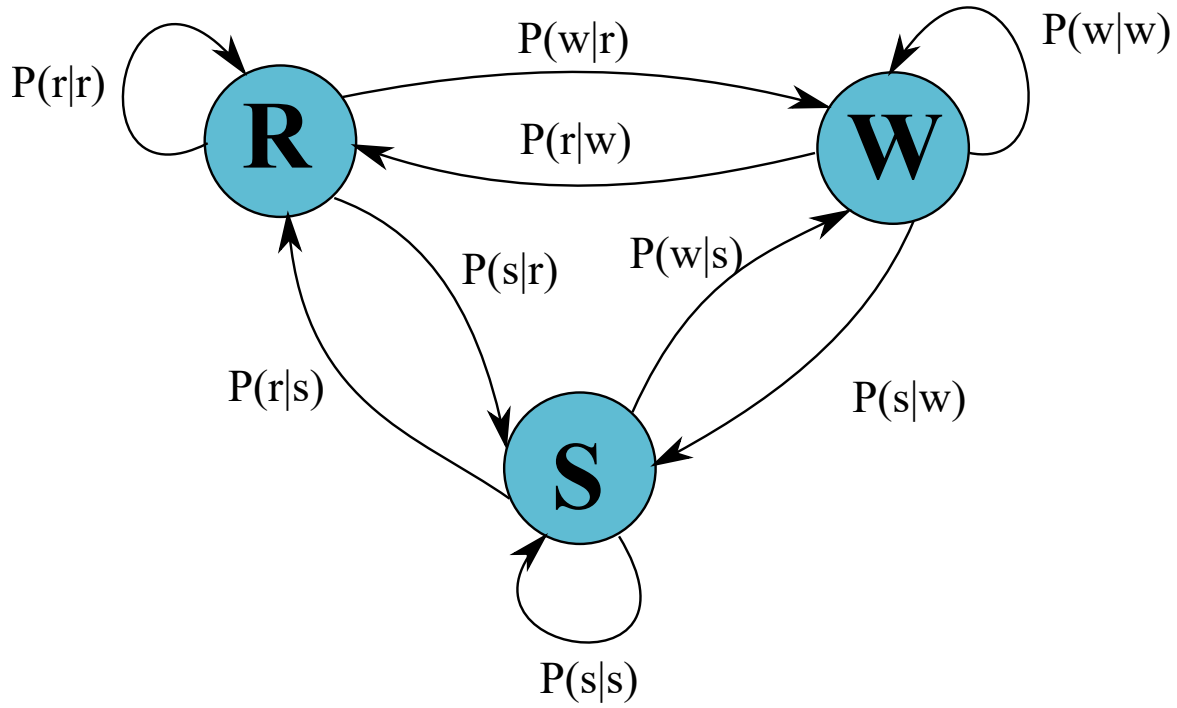

Figure 4.1: Simple weather model with three different states: (R)ainy, (S)unny and (W)indy.

We can code it in observation probability matrix B defined as

$$\mathbf{B} = \begin{array}{r} \\ rainy \\ sunny \\ windy \end{array} \begin{array}{c} \begin{array}{cccc} walk\ in\ park & shopping & clean\ flat & visit\ girlfriend \end{array} \\ \left\| \begin{array}{cccc} 0.05 & 0.05 & 0.25 & 0.65 \\ 0.7 & 0.15 & 0.05 & 0.1 \\ 0.25 & 0.4 & 0.05 & 0.3 \end{array} \right\| \end{array}$$

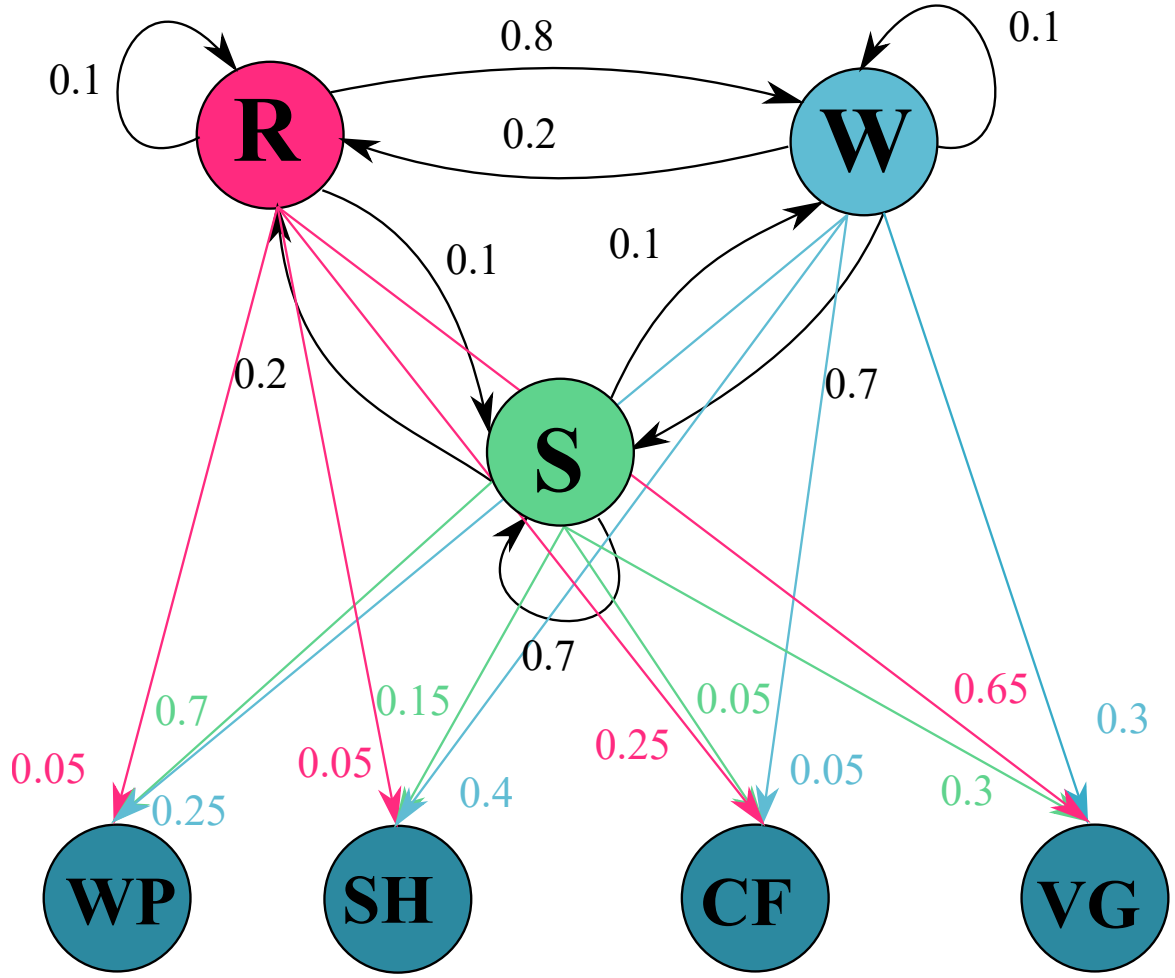

Figure 4.2: Hidden Markov Model with three hidden states: (R)ainy, (S)unny and (W)indy. There are four types of observations: walking in the park (WP), shopping (SH), cleaning flat (CF) and visiting girlfriend (VG).

Chapter 5

Evaluation

Knowing joint distribution of the sequence of observations O and states Q conditioned on λ we can easily compute the probability we are looking for using the law of total probability.

Theorem 1 ([5])**.**

$$P(O|Q,\lambda) = \prod_{t=1}^{T} b_{q_t}(o_t)\,. \tag{5.1}$$

Proof. Using the Markov property we get

$$\begin{aligned} P(O|Q,\lambda) &= P(o_1,\ldots,o_T|q_1,\ldots,q_T,\lambda) = \frac{P(o_1,\ldots,o_T,q_1,\ldots,q_T,\lambda)}{P(q_1,\ldots,q_T,\lambda)} \\ &= \frac{P(o_T|o_1,\ldots,o_{T-1},q_1,\ldots,q_T,\lambda)\,P(o_1,\ldots,o_{T-1},q_1,\ldots,q_T,\lambda)}{P(q_1,\ldots,q_T,\lambda)} \\ &= \frac{P(o_T|q_T,\lambda)\,P(o_{T-1}|o_1,\ldots,o_{T-2},q_1,\ldots,q_T,\lambda)\,P(o_1,\ldots,o_{T-2},q_1,\ldots,q_T,\lambda)}{P(q_1,\ldots,q_T,\lambda)}. \end{aligned}$$

Observation o_{T-1} is independent of the state q_T, so we can get

$$P(o_{T-1}|o_1,\ldots,o_{T-2},q_1,\ldots,q_T,\lambda) = P(o_{T-1}|o_1,\ldots,o_{T-2},q_1,\ldots,q_{T-1},\lambda)\,.$$

Analogously:

$$P(O|Q,\lambda) = \frac{P(o_T|q_t,\lambda)\ldots P(o_1|q_1,\lambda)\,P(q_1,\ldots,q_T,\lambda)}{P(q_1,\ldots,q_T,\lambda)} = \prod_{t=1}^{T} P(o_t|q_t,\lambda)\,.$$

As $P(o_t|q_t, \lambda) = b_{q_t}(o_t)$ we get

$$P(O|Q, \lambda) = \prod_{t=1}^{T} b_{q_t}(o_t).$$

□

Theorem 2 ([5])**.** *The probability of observing O under condition λ is given by the equation:*

$$P(O|\lambda) = \sum_{q_1 \dots q_T \in \mathbb{Q}_\mathbb{T}} \mu_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \dots a_{q_{T-1} q_T} b_{q_T}(o_T).$$

Proof. We can notice that probability $P(q_1, \dots, q_T|\lambda)$ is equivalent to probability that transition between subsequent states occurs according to transition matrix $\mathbf{A}$

$$P(q_1 \dots q_T|\lambda) = P(q_1) P(q_2|q_1) \dots P(q_{T-1}|q_T) = \mu_{q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T}. \tag{5.2}$$

From the formula for total probability we obtain:

$$\begin{aligned} P(O|\lambda) = \frac{P(O, \lambda)}{P(\lambda)} &= \sum_{q_1 \dots q_T \in \mathbb{Q}_\mathbb{T}} \frac{P(O, \lambda|q_1 \dots q_T) P(q_1 \dots q_T)}{P(\lambda)} \\ &= \sum_{q_1 \dots q_T \in \mathbb{Q}_\mathbb{T}} \frac{P(O, \lambda, q_1 \dots q_T) P(q_1 \dots q_T)}{P(q_1 \dots q_T) P(\lambda)} = \sum_{q_1 \dots q_T \in \mathbb{Q}_\mathbb{T}} \frac{P(O, \lambda, q_1 \dots q_T)}{P(\lambda)} \\ &= \sum_{q_1 \dots q_T \in \mathbb{Q}_\mathbb{T}} \frac{P(O|q_1 \dots q_T, \lambda) P(q_1 \dots q_T, \lambda)}{P(\lambda)} = \sum_{q_1 \dots q_T \in \mathbb{Q}_\mathbb{T}} P(O|q_1 \dots q_T, \lambda) P(q_1 \dots q_T|\lambda). \end{aligned} \tag{5.3}$$

Using this result would potentially allow us to evaluate the probability of O, but evaluating it directly has exponential complexity.

A much better approach is to acknowledge that many redundant calculations is made by directly evaluating (5.3) and therefore caching calculations can lead to reduced complexity. We implement the cache as a grid of states at each time step, calculating the cached valued (called α) for each state as a sum over all states at the previous time step. By α we understand the probability of the partial observation sequence $o_1, o_2 \cdots o_t$ and state s_i at time t. Therefore, we define the forward probability variable:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = s_i|\lambda).$$

If we work through the grid filling in the values of α the sum of the final column of the grid will equal the probability of the observation sequence. Mixing (5.1) and (5.2) □

Definition 1 (Forward algorithm)**.** *1. Initialisation:*

$$\alpha_1(i) = \pi_i b_i\left(o_1\right), 1 \leq i \leq N,$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j\left(o_{t+1}\right), 1 \leq t \leq T-1, 1 \leq j \leq N,$$

3. Termination:

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

The induction step is the core of the forward algorithm. For each state $s_j, \alpha_j(t)$ stores the probability of arriving in that state after observing the observation sequence up until time t.

It is easy to notice that by caching α values, the forward algorithm reduces the complexity of calculations involved to roughly N^2T rather than $2TN^T$. We can also define an analogous backward algorithm which is the exact reverse of the forward algorithm with the backward variable:

$$\beta_t(i) = P\left(o_{t+1} o_{t+2} \cdots o_T | q_t = s_i, \lambda\right),$$

as the probability of the partial observation sequence from $t+1$ to T, starting in state s_i.

We can easily extend algorithm to work for continuous observation space.

Chapter 6

Decoding

The aim of decoding is to discover the hidden state sequence that was most likely to have produced a given observation sequence. One solution to this problem is to use the Viterbi algorithm [12] to find the single best state sequence for an observation sequence. We want to compute

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} P(\mathbf{q}|\mathbf{o}).$$

In general, solving this equation would demand a tedious search of all possible observation state sequences $\mathbf{o}$, which would become impossible because of exponential complexity of this problem. Luckily, the structure of hidden Markov models brings us a reduction in the computational complexity.

We know from Bayes' theorem that the posterior probability of observing a vector $\mathbf{q}$ given that we have series of observations $\mathbf{o}$ is expressed in terms of the observation likelihood $P(\mathbf{q}|\mathbf{o})$, the prior distribution $P(\mathbf{o})$ and the marginal distribution $P(\mathbf{o})$ as

$$P(\mathbf{q}|\mathbf{o}) = \frac{P(\mathbf{o}|\mathbf{q})P(\mathbf{q})}{P(\mathbf{o})}.$$

We can write observation likelihood using the independence property of observations:

$$P(\mathbf{o}|\mathbf{q}) = P\left(o_N, o_{N-1}, \cdots, o_2, o_1, o_0 | q_N, q_{N-1}, \cdots, q_2, q_1, q_0\right) =$$
$$P\left(o_N|q_N\right) \cdots P\left(o_2|q_2\right) P\left(o_1|q_1\right) P\left(o_0|q_0\right) = \prod_{n=0}^{N} P\left(o_n|q_n\right).$$

The probability of observing the vector of states $\mathbf{q}$ is given by considering the Markov property, using only the model transition probabilities:

$$P(\mathbf{q}) = P\left(q_N, \cdots, q_2, q_1, q_0\right) =$$
$$P\left(q_N|q_{N-1}, \cdots, q_2, q_1, q_0\right) P\left(q_{N-1}|q_{N-2}, \cdots, q_2, q_1, q_0\right) \cdots P\left(q_1|q_0\right) P\left(q_0\right) =$$
$$P\left(q_N|q_{N-1}\right) P\left(q_{N-1}|q_{N-2}\right) \cdots P\left(q_1|q_0\right) P\left(q_0\right) = \prod_{n=1}^{N} P\left(q_n|q_{n-1}\right) P\left(q_0\right).$$

Using Bayes' rule we can write the a-posteriori probability as:

$$P(\mathbf{q}|\mathbf{o}) = \frac{\prod_{n=1}^{N} P\left(o_n|q_n\right) P\left(q_n|q_{n-1}\right) P\left(q_0\right)}{\sum_{q_N \in S} \cdots \sum_{q_1 \in S} \sum_{q_0 \in S} \prod_{m=1}^{N} P\left(o_n|q_n\right) P\left(q_n|q_{n-1}\right) P\left(q_0\right)},$$

where S is the set of all possible state values. The double product reduces to single product because each observation is conditionally dependent to a single Markov chain state.

The nested sum denominator of (6) is computationally expensive. Luckily, we don't have to compute it because it ends up being a constant term. Actually, we can arrive at the same desired solution by maximizing the joint probability:

$$\arg\max_{\mathbf{q}} P(\mathbf{q}|\mathbf{o}) = \arg\max_{\mathbf{q}} P(\mathbf{q}, \mathbf{o}),$$

which can be expanded using the observational conditional dependency and Markov properties to

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} \prod_{n=1}^{N} P\left(o_n|q_n\right) P\left(q_n|q_{n-1}\right) P\left(q_0\right).$$

From a practical viewpoint it is often easier to maximize the log probability, because we avoid numerical errors:

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} \sum_{n=1}^{N} \left(\log P\left(o_n|q_n\right) + \log P\left(q_n|q_{n-1}\right)\right) + \log P\left(q_0\right).$$

We can now reduce the search to exploring a dataset for $\mathbf{q}$ of size $N_S^2 \cdot$ using this algorithm, because now the exploration of all the permutations of $\mathbf{q}$ and $\mathbf{x}$ became redundant. At each time step, we check the paths that lead to each state from the possible old states. For each possible new state, we keep only the path coming from the previous state that has the largest probability, that is:

$$V_q^i = \max\left(V_{q'}^{i-1} + \log P\left(q|q'\right) + \log P\left(o|q^i\right)\right)$$

for each time step. V_q^i at each time step equals to the maximum cumulative log-probability achieved so far in transitioning from state q' to q. By recursively maximizing the joint probability for each possible new state, we maximize the final posterior probability of the entire sequence of states. At any time step k we use the running sum of log-probabilities as the prior probability

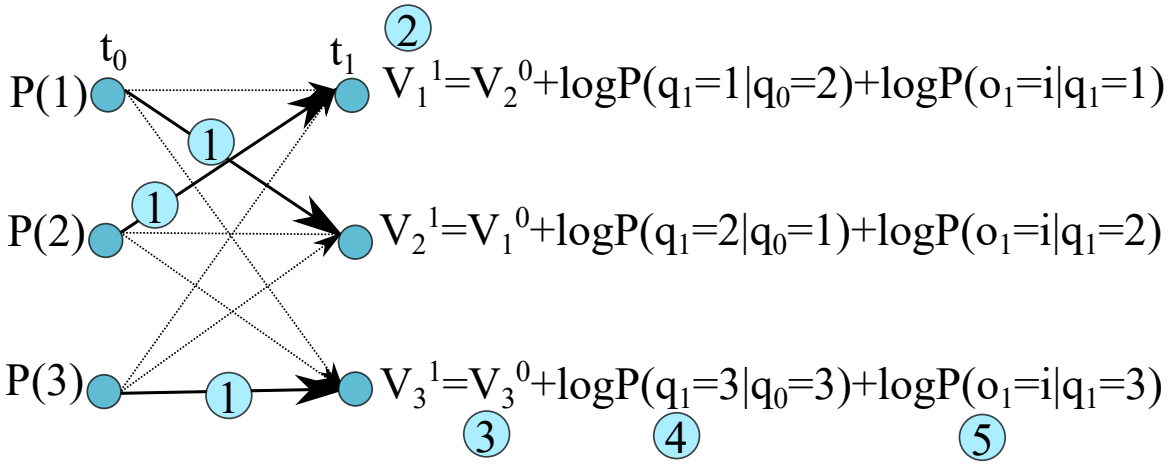

Figure 6.1: A single step in the Viterbi algorithm. By 1 we denote path segments that exhibit largest V_n observing backward from Time 1 to Time 0. By 2 we denote the probability of observing q_n equal to i given transition with maximal likelihood. By 3 we denote prior probability. By 4 we denote likelihood of transition. By 5 we denote observation likelihood for observing output state i

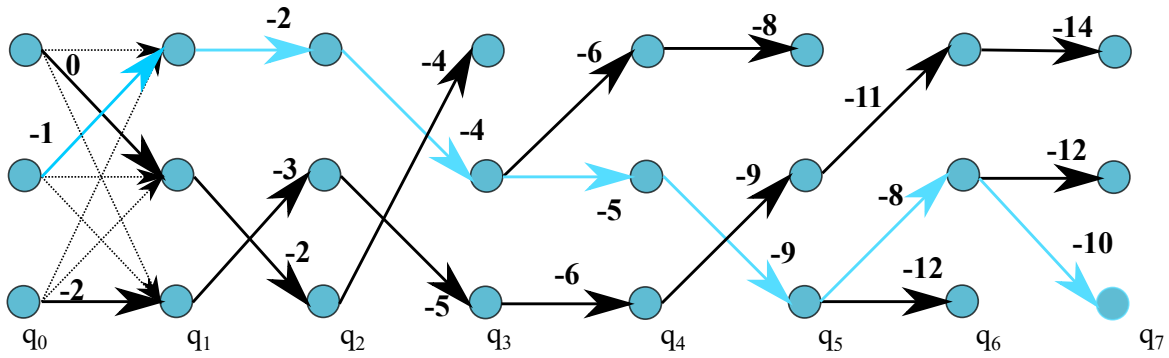

Figure 6.2: The development of the full paths from beginning to end.

$V_{1,2,3}^{k-1}$. Next, we explore all possible transitions leading to the present state, leaving only the transition yielding the highest updated log-probabilities $V_{1,2,3}^{k}$. We keep a list of the most likely transitions and move on to the next step. After putting it all together, we build a series of paths. When we reach the end of the observation sequence, we choose the final sequence that has the highest log-probability and move backwards, following the transitions with highest probability until the start is reached. The path will trace out the most likely sequence of hidden states traversed by the Markov model.

Chapter 7

Baum-Welch algorithm

Sometimes we want to find parameters λ maximizing probability $P(O|\lambda)$. In most cases we can't do it analytically. One of the possible solutions is to find a good approximation using for example Baum-Welch algorithm [5].

Definition 2. *Let $\xi_t(i,j)$ be the probability of being in state s_i at moment t and being in state s_j at moment $t+1$ under the condition, that the model has parameters λ and we observe O:*

$$\xi_t(i,j) = P\left(q_t = s_i, q_{t+1} = s_j | O, \lambda\right), i,j \in \{1,\ldots,N\}, t \in \{1,\ldots,T-1\}.$$

Theorem 3 ([5])**.**

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j\left(o_{t+1}\right) \beta_{t+1}(j)}{\sum_{l=1}^{N}\sum_{k=1}^{N} \alpha_t(l) a_{lk} b_k\left(o_{t+1}\right) \beta_{t+1}(k)},$$

for $i,j \in \{1,\ldots,T-1\}$.

Proof. Using the definition of $\xi_t(i,j)$ we get

$$\begin{aligned}\xi_t(i,j) = P\left(q_t = s_i, q_{t+1} = s_j | O, \lambda\right) &= \frac{P\left(q_t = s_i, q_{t+1} = s_j, O, \lambda\right)}{P(O,\lambda)} \\ = \frac{P\left(q_t = s_i, q_{t+1} = s_j, O|\lambda\right)}{\frac{P(O,\lambda)}{P(\lambda)}} &= \frac{P\left(q_t = s_i, q_{t+1} = s_j, O|\lambda\right)\cdot}{P(O|\lambda)}\end{aligned}$$

Let's just make two observations:

1.

$$P\left(q_t = s_i, q_{t+1} = s_j, O|\lambda\right) = \alpha_t(i) a_{ij} b_j\left(o_{t+1}\right) \beta_{t+1}(j), \tag{7.1}$$

2.

$$P(O|\lambda) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j\left(o_{t+1}\right) \beta_{t+1}(j). \tag{7.2}$$

The probability of obtaining sequence of observations $o_1, o_2, \ldots, o_t$ and being in the state s_i in the moment t is given by forward variable $\alpha_t(i)$. Probability of the transition from state s_i to s_j given observation o_{t+1} is equal to product $a_{ij} b_j\left(o_{t+1}\right)$. The probability of being in state s_j in the moment $t+1$ given sequence of observations $o_{t+2} \ldots o_T$ is equal to backward variable $\beta_{t+1}(j)$.

If we want to obtain the probability of observing the sequence of observations $O = o_1 \ldots o_T$ we have to repeat those steps for every possible value of states q_t and q_{t+1} and sum obtained probabilities.

□

Definition 3. *By $\gamma_t(i)$ we denote the probability of being in state s_i at the moment t, under conditions that model is of the form λ and we observe sequence of observations O:*

$$\gamma_t(i) = P\left(q_t = s_i|O, \lambda\right), i \in \{1, \ldots, N\}, t \in \{1, \ldots, T\}.$$

Observation 1.

$$\gamma_t(i) = P\left(q_t = s_i|O, \lambda\right) = \sum_{j=1}^{N} P\left(q_t = s_i, q_{t+1} = s_j|O, \lambda\right) = \sum_{j=1}^{N} \xi_t(i, j).$$

Let the random variable $C_t(i)$ be equal to 1 at the moment t while process is in state s_i under the condition that model is of the form λ and we observe the sequence O and 0 otherwise,

$$C_t(i) = \mathbb{1}\left(q_t = s_i|O, \lambda\right), i \in \{1, \ldots, N\}, t \in \{1, \ldots, T\}.$$

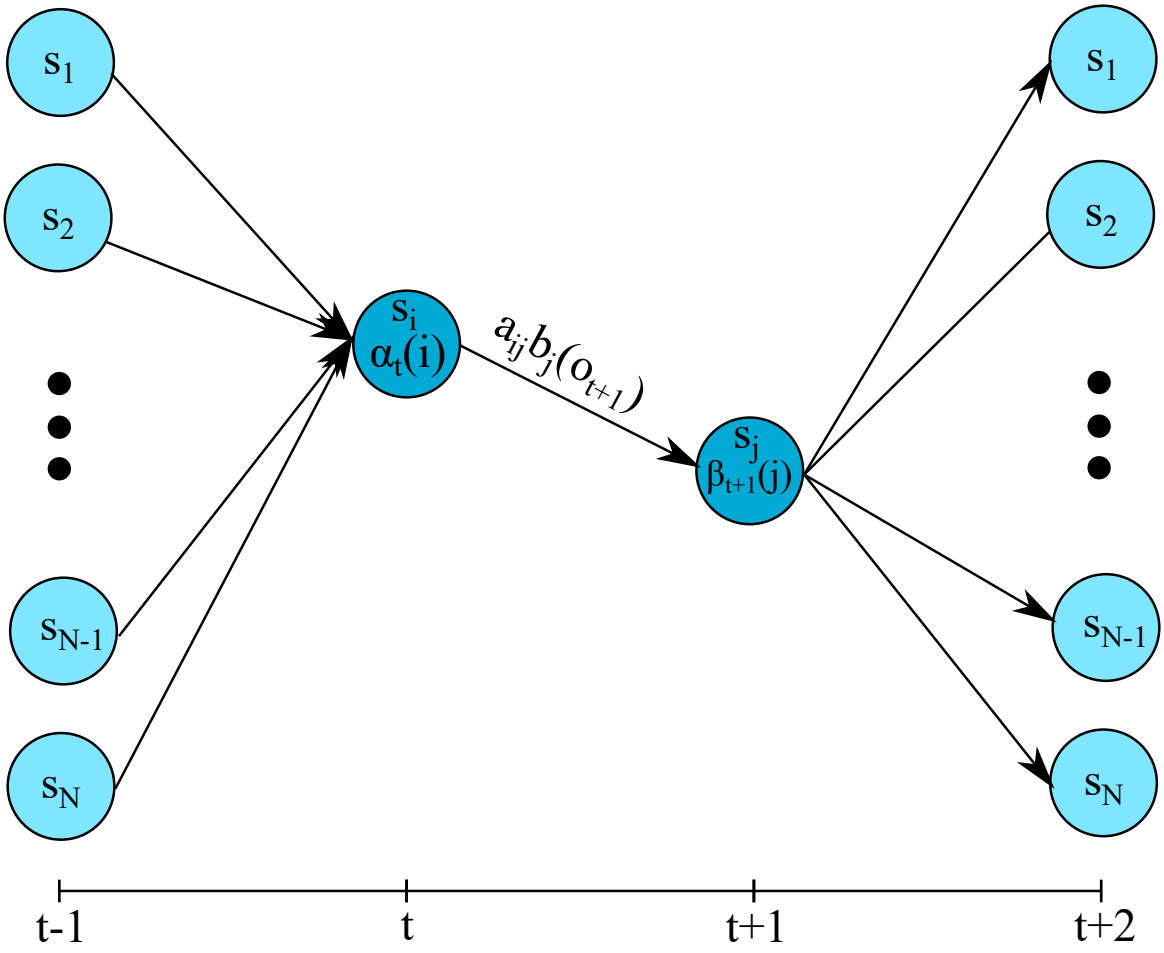

Figure 7.1: Steps required to compute $\xi_t(i, j)$

By $D(i) = \sum_{t=1}^{T} C_t(i)$ we denote random variable counting how many times process achieved state s_i.

Observation 2.

$$\mathbb{E}[D(i)] = \mathbb{E}\left[\sum_{t=1}^{T} C_t(i)\right] = \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{I}\left(q_t = s_i|O,\lambda\right)\right] = \sum_{t=1}^{T} P\left(q_t = s_i|O,\lambda\right) = \sum_{t=1}^{T} \gamma_t(i).$$

By decreasing sum interval to $t \in \{1,\ldots,T-1\}$ we obtain expected number of transitions coming out of s_i. Analogously wan can prove that the value of $\sum_{t=1}^{T-1} \xi_t(i,j)$ is equal to expected value of transitions from s_i to s_j.

Definition 4. *By $\overline{\mu_i}$ we denote expected value of frequency of starting the process in state s_i. By $\overline{a_{ij}}$ we understand the ratio of expected number of transitions from the state s_i to the state s_j to the expected number of processes starting in the state s_i. Finally $\overline{b_j}(k)$ is ratio of number of situations where state v_k is observed and the process is in the state s_j to the cumulative expected value of frequency of being in state s_j:*

$$\begin{aligned}
\overline{\mu_i} &= \gamma_1(i), && i \in \{1,\ldots,N\} \\
\overline{a_{ij}} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, && i,j \in \{1,\ldots,N\} \\
\overline{b_j}(k) &= \frac{\sum_{t=1,\{t:o_t=v_k\}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, && j \in \{1,\ldots,N\}, k \in \{1,\ldots,M\}.
\end{aligned}$$

Let:

$$\begin{aligned}
\bar{A} &= [\overline{a_{ij}}], && i,j \in \{1,\ldots,N\} \\
\overline{\mathbb{B}} &= [\overline{b_j}(k)], && j \in \{1,\ldots,N\}, k \in \{1,\ldots,M\}\ , \\
\bar{\mu} &= [\overline{\mu_i}], && i \in \{1,\ldots,N\}
\end{aligned}$$

it can be shown that the model $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\mu})$ doesn't decrease the conditional probability of obtaining given sequence of observations i.e.

$$P(O|\lambda) \leqslant P(O|\bar{\lambda}).$$

Definition 5 (Baum-Welch Algorithm). *1. Setting pleasing value of probability $P(O|\lambda)$.*

2. Defining initial value of parameter λ.

3. Computing value of model λ.

4. replacing old values of λ.

We repeat steps 3 and 4 until we get satisfying values of the probability $P(O|\lambda)$.

Observation 3. *Parameters of model $\overline{\lambda}$ fulfill conditions:*

$$\begin{aligned} \textstyle\sum_{i=1}^{N} \bar{\mu}_i &= 1, \\ \textstyle\sum_{j=1}^{N} \overline{a_{ij}} &= 1, \quad i \in \{1,\dots,N\} \\ \textstyle\sum_{k=1}^{M} b_j(k) &= 1, \quad j \in \{1,\dots,N\} \end{aligned} .$$

Proof.

$$\sum_{i=1}^{N} \bar{\mu}_i = \sum_{i=1}^{N} \gamma_1(i) = \sum_{i=1}^{N} P\left(q_1 = s_i | O, \lambda\right) = 1.$$

For $i \in \{1,\dots,N\}$

$$\begin{aligned} &\textstyle\sum_{j=1}^{N} \overline{a_{ij}} = \sum_{j=1}^{N} \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{j=1}^{N}\sum_{t=1}^{T-1} P(q_t=s_i, q_{t+1}=s_j|O,\lambda)}{\sum_{t=1}^{T-1} P(q_t=s_i|O,\lambda)} \\ &= \frac{\sum_{t=1}^{T-1}\sum_{j=1}^{N} P(q_t=s_i, q_{t+1}=s_j|O,\lambda)}{\sum_{t=1}^{T-1} P(q_t=s_i|O,\lambda)} = \frac{\sum_{t=1}^{T-1} P(q_t=s_i|O,\lambda)}{\sum_{t=1}^{T-1} P(q_t=s_i|O,\lambda)} = 1 \end{aligned} .$$

For $j \in \{1,\dots,N\}$

$$\textstyle\sum_{k=1}^{M} \bar{b}_j(k) = \sum_{k=1}^{M} \frac{\sum_{t=1,\{t:o_t=v_k\}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} = 1$$

□

We assumed that the emission variable has discrete distribution, but the above algorithms work also for a continuous emission space.

We consider a mixture of M multivariate Gaussians for each state where $b_j\left(o_t\right) = \sum_{\ell=1}^{M} c_{j\ell} \mathcal{N}\left(o_t \mid \mu_{j\ell}, \Sigma_{j\ell}\right) = \sum_{\ell=1}^{M} c_{j\ell} b_{j\ell}\left(o_t\right)$.

For Gaussian mixtures, we define the probability that the ℓ^{th} component of the i^{th} mixture

generated observation o_t as

$$\gamma_{i\ell}(t) = \gamma_i(t)\frac{c_{i\ell}b_{i\ell}\left(o_t\right)}{b_i\left(o_t\right)} = p\left(Q_t = i, X_{it} = \ell \mid O, \lambda\right),$$

where X_{it} is a random variable indicating the mixture component at the time t for state i. We might guess that the update equations for this case are:

$$\begin{aligned} c_{i\ell} &= \frac{\sum_{t=1}^{T}\gamma_{i\ell}(t)}{\sum_{t=1}^{T}\gamma_i(t)} \\ \mu_{i\ell} &= \frac{\sum_{t=1}^{T}\gamma_{i\ell}(t)o_t}{\sum_{t=1}^{T}\gamma_{i\ell}(t)} \\ \Sigma_{i\ell} &= \frac{\sum_{t=1}^{T}\gamma_{i\ell}(t)(o_t-\mu_{i\ell})(o_t-\mu_{i\ell})^T}{\sum_{t=1}^{T}\gamma_{i\ell}(t)}. \end{aligned}$$

When there are E observation sequences the e^{th} being of length T_e, the update equations become:

$$\begin{aligned} \pi_i &= \frac{\sum_{e=1}^{E}\gamma_i^e(1)}{E} \\ c_{i\ell} &= \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^e(t)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_i^e(t)} \\ \mu_{i\ell} &= \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^e(t)o_t^e}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^e(t)} \end{aligned} .$$

$$\Sigma_{i\ell} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^e(t)\left(o_t^e-\mu_{i\ell}\right)\left(o_t^e-\mu_{i\ell}\right)^T}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^e(t)}$$

and

$$a_{ij} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\xi_{ij}^e(t)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_i^e(t)}.$$

Full derivation can be found in [4].

Chapter 8

Performance measures

In literature there are many different measures used to determine the quality of classification.

The most straightforward one is *accuracy* which we define as the number of correct labels divided by the sample size. The main drawback of this performance measure is that it's very sensitive to class size imbalance. In order to minimize the impact of disproportion between sample sizes, few other performance measures were introduced.

First, let's assume that we perform a binary classification task. A false positive (fp) is an error in data reporting in which a test result improperly indicates presence of a condition, such as a disease (the result is positive), when in reality it is not present, while a false negative (fn) is an error in which a test result improperly indicates no presence of a condition (the result is negative), when in reality it is present. These are the two kinds of errors in a binary test (and are contrasted with a correct result, either a true positive (tp) or a true negative (tn)).

We define a precision [1] (the fraction of relevant instances among the retrieved instances) as:

$$\text{Precision} := \frac{tp}{tp + fp}.$$

Additionally a recall [1] (the fraction of the total amount of relevant instances that were actually

Table 8.1: Sample classes.

		True		
		rise	fall	same
Predicted	rise	4	6	3
	fall	1	2	0
	same	1	2	6

Table 8.2: Sample classes.

class	precision	recall	F-score
rise	0.308	0.667	0.421
fall	0.667	0.200	0.308
same	0.667	0.667	0.667

retrieved) is defined as:

$$\text{Recall} := \frac{tp}{tp + fn}.$$

The common practice is to use harmonic mean of these two, namely F-measure defined as:

$$F := 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

When it comes to multi-class metrics we can define them in several ways. First, we have to compute per-class measures:

Example 3. *The classification's results are given in Table 8.1.*

The measures of particular scores are presented in Table 8.2.

If we want to measure the quality of multi-class labeling we can do macro-averaging [1], that means simply compute arithmetic measure of score.

Example 4.

$$\textit{Macro-F1} = (0.421 + 0.308 + 0.667)/3 = 0.465,$$

$$\textit{Macro-precision} = (0.31 + 0.67 + 0.67)/3 = 0.547,$$

$$\textit{Macro-recall} = (0.67 + 0.20 + 0.67)/3 = 0.511.$$

We can also used weighted average [1] to compute score.

Example 5. *We have 25 samples in Table 8.2: 6 rises, 10 falls and 9 lacks of changes.*

$$\textit{Weighted-F1} = (6 \cdot 0.421 + 10 \cdot 0.308 + 9 \cdot 0.667)/25 = 0.464,$$

$$\textit{Weighted-precision} = (6 \cdot 0.308 + 10 \cdot 0.667 + 9 \cdot 0.667)/25 = 0.581,$$

$$\textit{Weighted-recall} = (6 \cdot 0.667 + 10 \cdot 0.200 + 9 \cdot 0.667)/25 = 0.480.$$

In the following chapters weighted measures are used.

Chapter 9

Gaussian Hidden Markov Model

9.1 Model description

For the fixed time step t and window size k we are given observation $X(t,k)$ and mid-prices at time steps $t-k+1,\ldots,t$:

$$\mathbf{X}(t,k) := (m(t-k+1), m(t-k+2), \ldots, m(t)).$$

Our goal is to predict if the mid-price in the next step, i.e., $m(t+1)$ will increase or decrease (or changes just slightly), given observation $\mathbf{X}(t,k)$. To be more exact we aim at predicting $l(t) \in \{-1,0,1\}$, where:

$$l(t) := \begin{cases} -1 & \text{if } \frac{m(t+1)-m(t)}{m(t)} < -\varepsilon, \\ 0 & \text{if } |\frac{m(t+1)-m(t)}{m(t)}| \leq \varepsilon, \\ 1 & \text{if } \frac{m(t+1)-m(t)}{m(t)} > \varepsilon, \end{cases} \tag{9.1}$$

for the fixed ε. In other words, we want to classify $\mathbf{X}(t,k)$ into 3 classes, -1,0,+1. Let's denote the resulting classification value by $\hat{l}(t)$.

Construction of $\hat{l}(t+1)$. The precise procedure PREDICT_L is given in Algorithm 3. Roughly speaking, we treat values $\mathbf{X}(t,k) := (m(t-k+1), m(t-k+2), \ldots, m(t))$ as observations of a stochastic process $M_{t-k+1}, M_{t-k+2}, \ldots, M_t$. We assume that the process is a hidden Markov model with three hidden states. We identify the hidden states with $s_1 =$*bull* (upward), $s_2 =$*bear* (downward) and $s_3 =$ stable markets. We also assume that M_w given $Q_w = s_i$ has a normal $\mathcal{N}(\mu_i, \sigma_i^2)$ distribution, $w = t-k+1, \ldots, t$. Using the Baum-Welch Algorithm we train the HMM model obtaining the transition matrix $\mathbf{A}$ (of size 3×3) and parameters $\mu_i, \sigma_i^2, i = 1, 2, 3$. Using the Viterbi algorithm we find the most likely sequence of hidden states $q_{t-k+1}, \ldots, q_t$. The idea is following: we assume that currently (i.e., at step t) the hidden state is indeed q_t. Note that we know the current mid-price $m(t)$ and wish to predict the change in next step. To achieve it we set $a_{-1} = a_0 = a_1$ and we repeat R times the following procedure:

- Simulate $Z \in \{1, 2, 3\}$ according to a distribution $\mathbf{A}(q_t, \cdot)$, i.e.,

$$P(Z = i) = \mathbf{A}(q_t, i).$$

- Assuming that in the next step the system is in a hidden state Z we simulate its emission – a random variable $N \sim (\mu_Z, \sigma_Z^2)$.

- We recored if simulated mid-price at step $t+1$, i.e., N is larger or smaller then the mid-price at step t, i.e., $m(t)$. To be more exact we set (here $\alpha > 0$ is a parameter)

  if $N - m_t < -\alpha$ **then** $a_{-1} = a_{-1} + 1$,

  if $|N - m_t| \leq \alpha$ **then** $a_0 = a_0 + 1$,

  if $N - m_t > \alpha$ **then** $a_1 = a_1 + 1$.

Finally, we predict $\hat{l}(t+1)$ according to a situation which happened most frequently:

$$\hat{l}(t+1) = \underset{j \in \{-1,0,1\}}{\arg\max}\, a_j.$$

Algorithm 1 Predicting change in mid-prices: fixed t

procedure PREDICT_L($t, k, \mathbf{X}(t,k), R, \alpha$)
  Input time step t, window size k, number of iterations R, parameter $\alpha > 0$,
    observations $\mathbf{X}(t,k) := (m(t-k+1), m(t-k+2), \ldots, m(t))$
  Output $\hat{l}(t+1)$, prediction of $l(t+1)$
  $\mathcal{M} \leftarrow$ hidden Markov model with 3 states s_1, s_2, s_3, transition matrix $\mathbf{A}$ and emission probabilities $\mathcal{N}(\mu_i, \sigma_i^2), i = 1, 2, 3$.
  Train (use Baum-Welch algorithm) $\mathcal{M}$ on $\mathbf{X}(t,k)$ obtaining $\mathbf{A}$ and $\mu_i, \sigma_i^2, i = 1, 2, 3$.
  Apply the Viterbi algorithm to find the most likely sequence of hidden states $q_{t-k+1}, \ldots, q_t$.
  Set $a_{-1} = a_0 = a_1 = 0$.
  for $r = 1$ **to** R **do**
    Simulate $Z \in \{1, 2, 3\}$ with distribution $P(Z = i) = \mathbf{A}(q_t, i), i = 1, 2, 3$.
    Simulate $N \sim \mathcal{N}(\mu_Z, \sigma_Z^2)$.
    if $N - m_t < -\alpha$ **then** $a_{-1} = a_{-1} + 1$
    if $|N - m_t| \leq \alpha$ **then** $a_0 = a_0 + 1$
    if $N - m_t > \alpha$ **then** $a_1 = a_1 + 1$
  end for
  return $\hat{l}(t+1) = \arg\max_{j \in \{-1,0,1\}} a_j$
end procedure

Classification of changes in all sequence of mid-prices. Given all sequence of mid-prices

$$m(1), \ldots, m(T)$$

and a window size k, we want to predict $l(k+1)$ based on $\mathbf{X}(k,k)$, then $l(k+2)$ based on $\mathbf{X}(k+1,k)$ and so on (till $l(T)$ based on $\mathbf{X}(T-1,k)$). Thus, as in standard machine learning cases, we have a ground truth $l(k+1), \ldots, l(T)$ and predictions $\hat{l}(k+1), \ldots, \hat{l}(T)$ – and may thus evaluate whole model by computing *accuracy, precision, recall* and *F-score* (see Chapter 8). The procedure is given in Algorithm 4.

Algorithm 2 Evaluating the predicting of changes in mid-prices

procedure EVAL($(m(1), \ldots, m(T)), k, R, \alpha$)
  Compute $l(t+1)$ for $t = k, \ldots, T-1$ using (9.1) ▷ compute "ground-truth"
  Set $\hat{l}(t+1) = empty$ $t = k, \ldots, T-1$

  for $i = k$ **to** $T-1$ **do**
    $\hat{l}(i+1) =$PREDICT_L$(i, k, \mathbf{X}(i,k), R, \alpha)$
  end for
  return performance measures for $\{(l(t), \hat{l}(t)), t = k, \ldots, T-1\}$
end procedure

9.2 Data used

This model was tested on one sample company from London Stock Exchange Rebuild Order Book dataset. Data was registered in period 01.09.2013-15.09.2013. The University of Wroclaw is the owner of the dataset.

While creating labels ε was set to 0.0003 to obtain the most balanced distribution of classes.

9.3 Results

In Table 9.1 we can see results of classification for different prediction horizons, window sizes (k), time-steps in minutes (Δt) and values of parameter α.

From observed values we can conclude that the model for short prediction horizons is close to a random prediction. On the other side it gets better results for long-term forecasting (10 minutes).

Table 9.1: Results for univariate Gaussiam emission model.

k	Δt	α	**Accuracy**	**Precision**	**Recall**	**F-Score**
100	1	0.00000001	0.355	0.356	0.341	0.328
100	5	0.00000001	0.474	0.326	0.322	0.324
100	10	0.00000001	0.466	0.324	0.314	0.319
50	1	0.00000001	0.303	0.368	0.345	0.297
50	5	0.00000001	0.305	0.370	0.346	0.299
50	10	0.00000001	0.457	0.319	0.314	0.316
100	1	0.0003	0.335	0.349	0.356	0.314
100	5	0.0003	0.344	0.338	0.350	0.282
100	10	0.0003	0.334	0.311	0.226	0.262
100	1	0.000015	0.357	0.357	0.343	0.329
100	5	0.000015	0.472	0.326	0.320	0.322
100	10	0.000015	0.466	0.346	0.380	0.353
100	1	0.1	0.296	0.099	0.333	0.152
100	1	0.01	0.288	0.096	0.333	0.149
100	1	0.001	0.302	0.395	0.342	0.182
100	1	0.0001	0.351	0.350	0.348	0.348
100	1	0.00001	0.358	0.358	0.345	0.330
100	1	0.000001	0.358	0.356	0.344	0.329
100	1	0.0000001	0.356	0.358	0.342	0.329
100	5	0.1	0.020	0.007	0.333	0.013
100	5	0.01	0.018	0.006	0.333	0.012
100	5	0.001	0.117	0.302	0.344	0.125
100	5	0.0001	0.441	0.337	0.357	0.326
100	5	0.00001	0.472	0.325	0.320	0.323
100	5	0.000001	0.478	0.329	0.326	0.327
100	5	0.0000001	0.479	0.330	0.326	0.328
100	10	0.1	0.019	0.006	0.333	0.012

100	10	0.01	0.016	0.338	0.336	0.013
100	10	0.001	0.166	0.310	0.241	0.166
100	10	0.0001	0.447	0.333	0.302	0.317
100	10	0.00001	0.475	0.331	0.321	0.326
100	10	0.000001	0.487	0.340	0.329	0.334
100	10	0.0000001	0.475	0.331	0.322	0.326
50	1	0.1	0.464	0.155	0.333	0.211
50	1	0.01	0.458	0.153	0.333	0.209
50	1	0.001	0.463	0.355	0.339	0.238
50	1	0.0001	0.345	0.360	0.355	0.343
50	1	0.00001	0.306	0.369	0.348	0.299
50	1	0.000001	0.302	0.364	0.346	0.296
50	1	0.0000001	0.303	0.367	0.347	0.297
50	5	0.1	0.113	0.038	0.333	0.067
50	5	0.01	0.114	0.038	0.333	0.068
50	5	0.001	0.204	0.358	0.355	0.203
50	5	0.0001	0.415	0.345	0.345	0.345
50	5	0.00001	0.435	0.335	0.336	0.327
50	5	0.000001	0.427	0.306	0.325	0.312
50	5	0.0000001	0.450	0.321	0.334	0.325
50	10	0.1	0.024	0.008	0.333	0.016
50	10	0.01	0.022	0.007	0.333	0.014
50	10	0.001	0.186	0.326	0.253	0.185
50	10	0.0001	0.441	0.329	0.302	0.315
50	10	0.00001	0.462	0.322	0.315	0.318
50	10	0.000001	0.465	0.324	0.320	0.322
50	10	0.0000001	0.465	0.324	0.318	0.321
150	1	0.1	0.195	0.065	0.333	0.109

150	1	0.01	0.203	0.068	0.333	0.113
150	1	0.001	0.209	0.350	0.339	0.137
150	1	0.0001	0.345	0.332	0.333	0.330
150	1	0.00001	0.378	0.336	0.334	0.329
150	1	0.000001	0.380	0.341	0.337	0.333
150	1	0.0000001	0.374	0.329	0.329	0.323
150	5	0.1	0.010	0.003	0.333	0.007
150	5	0.01	0.009	0.003	0.333	0.006
150	5	0.001	0.129	0.344	0.320	0.138
150	5	0.0001	0.452	0.344	0.345	0.328
150	5	0.00001	0.493	0.341	0.333	0.337
150	5	0.000001	0.501	0.359	0.392	0.362
150	5	0.0000001	0.493	0.341	0.333	0.337
150	10	0.1	0.000	0.000	0.000	0.000
150	10	0.01	0.000	0.000	0.000	0.000
150	10	0.001	0.159	0.303	0.105	0.154
150	10	0.0001	0.437	0.331	0.295	0.307
150	10	0.00001	0.474	0.332	0.319	0.320
150	10	0.000001	0.463	0.325	0.312	0.312
150	10	0.0000001	0.467	0.327	0.315	0.315

Chapter 10

Multivariate Gaussian Hidden Markov Model

10.1 Model description

Hidden Markov Models allow us to do something more sophisticated than previous one-dimensional model. We can elaborate Gould's Imbalance idea similarly to Xu et al. [16].

Let $b^{(k)}(t)$ be the k-th highest price at the bid side at moment t, namely

$$\begin{aligned} b^{(1)}(t) &:= b(t), \\ b^{(k+1)}(t) &:= \max_{\left\{x \in \mathcal{L}(t) | \omega_x < 0, p_x(t) < b^k(t)\right\}} p_x(t), \\ n_k^b(t) &:= n^b(b^{(k)}(t), t), \end{aligned}$$

analogously let $a^{(k)}(t)$ be the k-th lowest price at the ask side ad moment t, precisely

$$\begin{aligned} a^{(1)}(t) &:= a(t), \\ a^{(k+1)}(t) &:= \min_{\left\{x \in \mathcal{L}(t) | \omega_x > 0, p_x(t) > a^k(t)\right\}} p_x(t), \\ n_k^a(t) &:= n^a(a^{(k)}(t), t). \end{aligned}$$

Then we define following dynamic features:

$$\Delta p^{(i)}(t) := a^{(i)}(t) - b^{(i)}(t),$$
$$\Delta v^{(i)}(t) := n_i^a(t) - n_i^b(t).$$

We assume that vector $\mathbf{x}(t) := (\Delta p^{(1)}(t), \Delta v^{(1)}(t), \Delta p^{(2)}(t), \Delta v^{(2)}(t), \dots \Delta p^{(10)}(t), \Delta v^{(10)}(t)) \in \mathbb{R}^{20}$ is sampled from multivariate Gaussian distribution. Then we define our observations as:

$$\mathbf{X}(t,k) := (\mathbf{x}(t-k+1), \mathbf{x}(t-k+2), \dots, \mathbf{x}(t))$$

We generalize our label definition to:

$$l(t) := \begin{cases} -1 & \text{if } \frac{\frac{1}{h}\sum_{i=1}^{h} m(t+i) - m(t)}{m(t)} < -\varepsilon, \\ 0 & \text{if } |\frac{\frac{1}{h}\sum_{i=1}^{h} m(t+i) - m(t)}{m(t)}| \leq \varepsilon, \\ 1 & \text{if } \frac{\frac{1}{h}\sum_{i=1}^{h} m(t+i) - m(t)}{m(t)} > \varepsilon, \end{cases} \tag{10.1}$$

where h is a parameter called prediction We treat label $l(t)$ as a hidden state at moment t. This allows us to estimate transition matrix $A \in \mathbb{R}^{3\times3}$, mean vector $\mu \in \mathbb{R}^{20}$ and covariance matrix $\Sigma \in \mathbb{R}^{20\times20}$.

By $\mathbf{Y}(t,k)$ we understand:

$$\mathbf{Y}(t,k) := (l(t-k+1), l(t-k+2), \dots, l(t))$$

This solutions differs significantly from the previous one in the fact, that we actually observe past values of hidden states. This allows us to estimate parameters of Hidden Markov Model directly, not using Baum-Welch algorithm.

In the single algorithm iteration we use last k observations to estimate transition matrix $\mathbf{A}_k \in \mathbb{R}^{3\times3}$, mean vector $\mu_k \in \mathbb{R}^{20}$ and covariance matrix $\Sigma_k \in \mathbb{R}^{20\times20}$. Next to predict $l(t+1)$ we assume that $\pi_k(l(t)) = 1$, next using Viterbi algorithm on observations $\{\mathbf{X}(t), \mathbf{X}(t+1)\}$, see

Figure 10.1.

Algorithm 3 Predicting change in mid-prices: fixed t

procedure PREDICT_L($t, k, \mathbf{X}(t,k), \mathbf{Y}(t,k), \mathbf{x}(t+1)$)
  Input time step t, window size k, ,
    observations $\mathbf{X}(t,k), \mathbf{x}(t+1), \mathbf{Y}(t,k)$
  Output $\hat{l}(t+1)$, prediction of $l(t+1)$
  $\mathcal{M} \leftarrow$ hidden Markov model with 3 states s_1, s_2, s_3, transition matrix $\mathbf{A}$ and emission probabilities $\mathcal{N}(\mu_i, \Sigma_i), i = 1, 2, 3$.
  Estimate $\mathcal{M}$ on $Y(t,k)$, computing transitions frequency of subsequent states obtaining $\mathbf{A}$ and compute sample means and covariance matrix $\mu_i, \Sigma_i, i = 1, 2, 3$.
  Set $\pi_{l(t)} = 1$
  Apply the Viterbi algorithm to find the most likely sequence of hidden states $(q(t), q(t+1))$ for observations $(\mathbf{x}(t), \mathbf{x}(t+1))$.
  return $\hat{l}(t+1) = q(t+1)$
end procedure

Algorithm 4 Evaluating the predicting of changes in mid-prices

procedure EVAL($(\mathbf{x}(1), \ldots, \mathbf{x}(T)), k, \mathbf{Y}(t,k), \mathbf{x}(t+1)$)
  Set $\hat{l}(t) = empty\ t = k+1, \ldots, T$

  for $i = k$ **to** $T-1$ **do**
    $\hat{l}(i+1) =$PREDICT_L$(i, k, \mathbf{X}(i,k), \mathbf{Y}(t,k), \mathbf{x}(t+1)\alpha)$
  end for
  return performance measures for $\{(l(t), \hat{l}(t)), t = k+1, \ldots, T\}$
end procedure

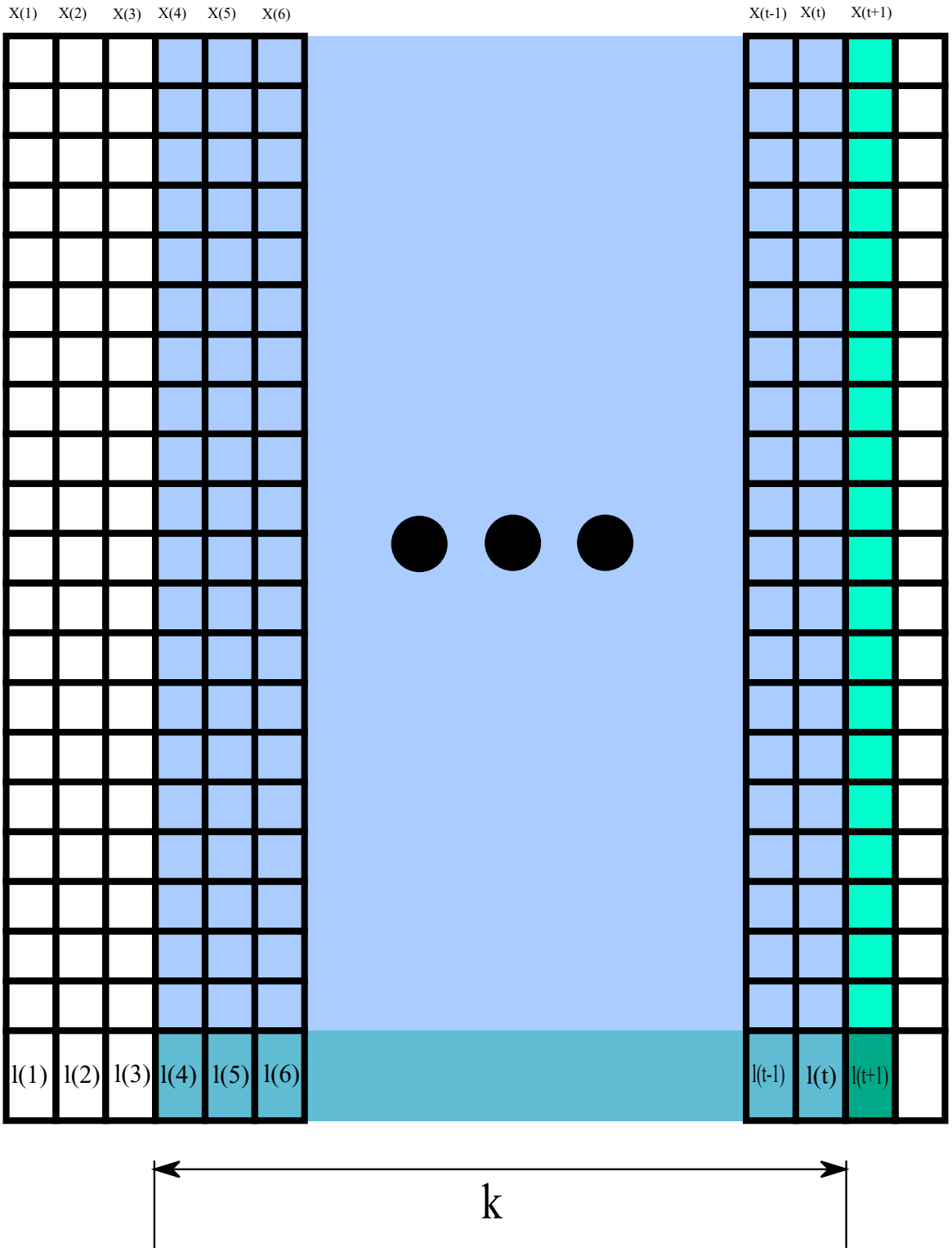

Figure 10.1: Illustration of model construction.

10.2 Data used

In order to evaluate the model and compare it with other state-of-the-art solutions we used FI2010 dataset. Feed data is day-specific and market wide, which means there is one file per day with data over all securities. The information is extracted from five stocks traded on the NASDAQ OMX Nordic at the Helsinki exchange from 1 June 2010 to 14 June 2010 counting ten working days (01.06, 02.06, 03.06, 04.06, 07.06, 08.06, 09.06, 10.06, 11.06, 14.06) Those five stocks are traded exclusively on Helsinki stock exchange.

The Helsinki Stock Exchange, operated by NASDAQ Nordic, is a pure electronic limit order market. The ITCH feed keeps a record of all the events, including those that take place outside active trading hours. At the Helsinki exchange, the trading period goes from 10:00 to 18:25 (local time, UTC/GMT +2 hours).

Labels are created using Equation (10.1). The value of ε is 0.002 percent. In the dataset there are labels for five prediction horizons: $10, 20, 30, 50$ and 100.

Data is normalized using z-score.

Full dataset description can be found in [10].

10.3 Results

Described model was implemented in Python using hmmlearn library. In Table 10.2 and 10.3 we present comparison of our model with current state-of-the-art. As we can see our model outperforms significantly other used solutions. In Table 10.3 we present more results for different choices of model parameters. For the longest prediction horizon HMM obtained results very close to the current most advanced model - DeepLOB which is based on deep neural networks. It is worth noting, that the training time for neural network models is several days and training and evaluation of HMM took approximately 12 hours. We also cannot forget about the main benefit coming from the "moving window" learning scheme. DeepLOB was train on the data

from n days to predict movements on $(n+1)-th$ day, which amounts for a large amount of data, but our model needs only 50 archived states of LOB to make accurate predictions.

Table 10.1: Results for multivariate Gaussian model.

k	**horizon**	**Accuracy**	**Precision**	**Recall**	**F-Score**
50	10	0.573	0.573	0.613	0.589
50	20	0.587	0.587	0.605	0.594
50	30	0.624	0.624	0.637	0.628
50	50	0.683	0.683	0.690	0.685
50	100	0.767	0.767	0.769	0.766
100	10	0.567	0.567	0.615	0.585
100	20	0.585	0.585	0.607	0.594
100	30	0.619	0.619	0.634	0.624
100	50	0.671	0.671	0.678	0.673
100	100	0.747	0.747	0.749	0.747
150	10	0.572	0.572	0.615	0.589
150	20	0.591	0.591	0.611	0.599
150	30	0.621	0.621	0.636	0.626
150	50	0.671	0.671	0.677	0.672
150	100	0.740	0.740	0.741	0.740
200	10	0.576	0.576	0.613	0.591
200	20	0.597	0.597	0.613	0.604
200	30	0.625	0.625	0.638	0.629
200	50	0.672	0.672	0.678	0.673
200	100	0.740	0.740	0.740	0.739

Table 10.2: Current state of the art models. Adopted from [17].

Model	Accuracy	Precision	Recall	F1
Prediction Horizon k = 10				
RR [10]	0.480	0.418	0.435	0.410
SLFN [10]	0.643	0.512	0.366	0.327
LDA [15]	0.638	0.379	0.458	0.363
MDA [15]	0.719	0.442	0.601	0.461
MCSDA [13]	0.834	0.461	0.480	0.467
MTR [15]	0.861	0.517	0.408	0.401
WMTR [15]	0.819	0.463	0.513	0.479
BoF [11]	0.576	0.393	0.514	0.363
N-BoF [11]	0.627	0.423	0.614	0.416
B(TABL) [14]	0.736	0.662	0.688	0.671
C(TABL) [14]	0.780	0.720	0.740	0.728
DeepLOB	0.789	0.785	0.789	0.777
Ours	0.573	0.573	0.613	**0.589**

Table 10.3: Current state of the art models. Adopted from [17].

Model	Accuracy	Precision	Recall	F1
Prediction Horizon k = 50				
RR [10]	0.439	0.436	0.433	0.427
SLFN [10]	0.473	0.468	0.464	0.459
BoF [11]	0.502	0.426	0.496	0.396
N-BoF [11]	0.565	0.472	0.582	0.462
B(TABL) [14]	0.695	0.691	0.688	0.688
C(TABL) [14]	0.748	0.746	0.743	0.743
DeepLOB	0.750	0.751	0.750	0.750
Ours	0.683	0.621	0.623	**0.620**
Prediction Horizon k = 100				
RR [10]	0.429	0.429	0.429	0.416
SLFN [10]	0.477	0.453	0.432	0.410
BoF [11]	0.510	0.425	0.478	0.408
N-BoF [11]	0.564	0.473	0.550	0.469
B(TABL) [14]	0.693	0.690	0.694	0.689
C(TABL) [14]	0.741	0.735	0.738	0.735
DeepLOB	0.767	0.768	0.767	0.766
Ours	0.767	0.767	0.769	**0.766**

Chapter 11

Future work

Obtained results encourage to put more work into researching the applications of HMM in LOB data analysis.

In this work we only used small subset of information contained in the Limit Order Book, but there are more features we can work on. Authors of FI2010 dataset handcrafted 144 features to make LOB predictions. Zihao Zhang et al. [17] used convolutional neural networks for feature extraction. Perhaps, combining neural network based feature extraction could be combined with hmm modelling.

Treating imbalances as sort of the input to model worked well in this work. A well known modification of Hidden Markov Models called Input-Output Hidden Markov Model could benefit this approach.

It is also worth taking note here, that the only aspect explored in this work is mid-price movement prediction. More work can be invested into developping portfolio optimization methods based on HMM classifier.

Chapter 12

Conclusions

In this work two HMM based classifiers for mid-price change were created. One of them got similar results to the other solutions presented in literature, reaching top scores for long prediction horizons, but the main advantage over competing models is fast learning time and low data demand.

Both models showed tendency to give better scores with longer prediction horizons.

This work may be a starting points for research of applications of more advanced probabilistic models with Markov dynamics in LOB prediction. These include Input-Output Hidden Markov Models or switching Linear Dynamical Systems.

Bibliography

[1] Multi-Class Metrics Made Simple, Part II: the F1-score | by Boaz Shmueli | Towards Data Science.

[2] Algorithmic trading, August 2020. Page Version ID: 974661155.

[3] Alessandro Bigiotti and Alfredo Navarra. Optimizing Automated Trading Systems. In Tatiana Antipova and Alvaro Rocha, editors, *Digital Science*, Advances in Intelligent Systems and Computing, pages 254–261, Cham, 2019. Springer International Publishing.

[4] Jeff A Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. page 15.

[5] Phil Blunsom. Hidden Markov Models. September 2004.

[6] Rick Durrett. Essentials of Stochastic Processes. page 226.

[7] Martin D. Gould and Julius Bonart. Queue Imbalance as a One-Tick-Ahead Price Predictor in a Limit Order Book. *arXiv:1512.03492 [q-fin]*, December 2015. arXiv: 1512.03492.

[8] Martin D. Gould, Mason A. Porter, Stacy Williams, Mark McDonald, Daniel J. Fenn, and Sam D. Howison. Limit Order Books. *arXiv:1012.0349 [physics, q-fin]*, December 2010. arXiv: 1012.0349.

[9] Adamantios Ntakaris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Mid-price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators. *arXiv:1907.09452 [cs, q-fin, stat]*, July 2019. arXiv: 1907.09452.

[10] Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Benchmark Dataset for Mid-Price Forecasting of Limit Order Book Data with Machine Learning Methods. *Journal of Forecasting*, 37(8):852–866, December 2018. arXiv: 1705.03233.

[11] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Temporal Bag-of-Features Learning for Predicting Mid Price Movements Using High Frequency Limit Order Book Data. *IEEE Transactions on Emerging Topics in Computational Intelligence*, PP:1–12, October 2018.

[12] George Slade. The Viterbi algorithm demystified. March 2013.

[13] Dat Thanh Tran, Moncef Gabbouj, and Alexandros Iosifidis. Multilinear Class-Specific Discriminant Analysis. *Pattern Recognition Letters*, 100:131–136, December 2017. arXiv: 1710.10695.

[14] Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal Attention augmented Bilinear Network for Financial Time-Series Data Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1407–1418, May 2019. arXiv: 1712.00975.

[15] Dat Thanh Tran, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Tensor Representation in High-Frequency Financial Data for Price Change Prediction. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, November 2017. arXiv: 1709.01268.

[16] Ke Xu, Martin D. Gould, and Sam D. Howison. Multi-Level Order-Flow Imbalance in a Limit Order Book. *arXiv:1907.06230 [q-fin]*, July 2019. arXiv: 1907.06230.

[17] Zihao Zhang, Stefan Zohren, and Stephen Roberts. DeepLOB: Deep Convolutional Neural Networks for Limit Order Books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, June 2019. arXiv: 1808.03668.

[18] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Extending Deep Learning Models for

Limit Order Books to Quantile Regression. *arXiv:1906.04404 [q-fin]*, June 2019. arXiv: 1906.04404.