Machine learning in sentiment reconstruction of the simulated stock market

Mikhail Goykhman, Ali Teimouri

PII: S0378-4371(17)31159-7
DOI: https://doi.org/10.1016/j.physa.2017.11.093
Reference: PHYSA 18861

To appear in: *Physica A*

Received date : 9 August 2017
Revised date : 25 October 2017

Here are the highlights from our paper:

- We studied the framework in which the stock price environment is in one-to-one correspondence with a given driving processes.
- We focused on the systems driven by a sentiment process of the Markov chain type, governing the buy vs. sell decision of agents.
- We demonstrated that the Baum-Welch algorithm of the Hidden Markov Models can be used to recover the transition probabilities matrix of the sentiment process, when the system exhibits sufficiently long-lived sentiment states.
- We discussed application of the Recurrent Neural Network to the problem of reconstruction of the specific sentiment states at each point in time, from the observed stock prices.
- We discussed regimes of applicability of the HMM and RNN methods to the problem of the sentiment reconstruction.

Sincerely,

Mikhail Goykhman, Ali Teimouri

Machine learning in sentiment reconstruction of the simulated stock market

Mikhail Goykhman

Enrico Fermi Institute, University of Chicago,
Chicago, IL 60637, USA

Racah Institute of Physics, Hebrew University of Jerusalem,
Jerusalem, 91904, Israel

E-mail: goykhman@uchicago.edu ‡

Ali Teimouri

Consortium for Fundamental Physics, Lancaster University,
Lancaster, LA1 4YB, United Kingdom

E-mail: a.teimouri@lancaster.ac.uk

Abstract.
In this paper we continue the study of the simulated stock market framework defined by the driving sentiment processes. We focus on the market environment driven by the buy/sell trading sentiment process of the Markov chain type. We apply the methodology of the Hidden Markov Models and the Recurrent Neural Networks to reconstruct the transition probabilities matrix of the Markov sentiment process and recover the underlying sentiment states from the observed stock price behavior. We demonstrate that the Hidden Markov Model can successfully recover the transition probabilities matrix for the hidden sentiment process of the Markov Chain type. We also demonstrate that the Recurrent Neural Network can successfully recover the hidden sentiment states from the observed simulated stock price time series.

1. Introduction

A typical stock market simulation framework considers a discrete-time evolution of a system of agents, each possessing shares of stock and units of cash, who submit orders to the stock exchange, see [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] for some of the original papers. One can assume that at each time step each agent participates in a trade with some probability, which in the simplest models is identical for all of the agents, and is a constant in time. If the agent decides to participate in a trade then it needs to decide on the side of the trade (buy or sell), the limit price (for which it is willing to buy or sell the shares of stock), and the size of the order.

‡ goykhman89@gmail.com

In the simplest models the buy/sell side is determined by a flip of a fair coin, the limit price is normally distributed around the value related to the most recent stock price, and the size of the order is drawn uniformly between a zero and all-in [5]. Under such conditions the stock price time series will exhibit a mean-reverting behavior around the equilibrium, $P_e = M/S$, determined by the total amount of cash M and the total number of shares S in the system. This relation is a simple consequence of the balance of an expected cash flows to and from the stock market capitalization. Similarly, the volatility σ_e around the mean price P_e is determined by the standard deviation σ of the limit orders submitted by the agents, $\sigma_e \simeq k\sigma$, for a certain value of k [5].

The real world stock prices time series are far from being simple mean-reverting processes. In order to obtain interesting stock price dynamics one needs to incorporate a non-trivial behavior of the agents, rather than a random behavior described in the previous paragraph. Various models have been proposed in the literature, incorporating a sophisticated strategies into the behavior of agents, such as trend-following, contrarian, fundamental, utility optimization, etc, see [14] for a review. The common feature of those models is that the agents apply a specified strategy to the observed stock price behavior to make a purposeful decision about their trading actions. Such models have been rather successful in explaining stylized facts of a stock price behavior, such as fat tails of logarithmic returns [15] and volatility clustering.

Another way to model the stock price behavior in a simulated setting has been proposed in [13]. The starting assumption adopted in [13] was to consider a stock market framework in which the strategies of the agents are in one-to-one correspondence with a set of processes, called sentiments. (An interesting review of the sentiment analysis in stock market can be found in [16].) § Examples of such sentiment processes include the perceived volatility sentiment, determining the standard deviation of the submitted limit price, the buy/sell attitude, determining the willingness to buy rather than sell a stock, and the participation sentiment, determining the trading volume. All or large groups of the agents receive sentiments from the same source, probably with some noise around it. For instance, half of the agents might receive an information that a stock has been assigned a positive rating, and therefore the agents in that group will be, *e.g.*, thirty-percent more willing to buy that stock rather than to sell it.

One could ask what would be the reason to model the behavior of the stock market participants using the driving sentiment processes. Indeed, it is likely to expect that the agents participating in the market will readjust their trading decision basing on the observed stock price behavior, rather than persist following the pre-specified sentiment. We answer this question by pointing out that the sentiment processes which we discuss in this paper are emergent rather than imposed, in the sense that the collective behavior of the agents, in the framework considered in this paper, can be described via sentiment states.

§ There are subtle specifics as to how this one-to-one correspondence is to be understood. For instance, we do not distinguish between market frameworks in which a subgroup of agents follows a particular strategy, or each agent in the whole system adopts that strategy with the corresponding probability.

We do not address the question of intelligence of the agents in our stock market simulation framework. Our stock market framework models the situation in which the (large groups of) agents have some trading guidelines which we propose to model by a set of sentiments. This is contrasted with most of the agent-based stock market simulations, ranging in their degree of sophistication in modeling the agents's intelligence: starting from the model [5] of agents with low intelligence (where agents only look at the most recent stock price and stock volatility to decide on the limit orders they submit), to the sophisticated model of [4] with agents shaping their trade decisions using genetic algorithms. The major postulate of our sentiment-driven stock market framework is that all of the intelligent decision-making by the agents has happened elsewhere, and that the outcome of it can be concisely formulated in terms of the driving sentiment processes.

Therefore the sentiment trading conditions discussed in this paper define a market framework as the starting assumption. In a sentiment-driven stock market framework the stock price dynamics is largely determined by the properties of the underlying sentiment processes. Once all the sentiments have been specified we can predict well what will come out of the simulation. We can ask the opposite question: if we observe a stock price behavior and we know that it has originated in a sentiment-driven framework, how do we determine the underlying sentiment processes? This paper is concerned with such a question.

Notice that above we are talking about attempting to explain a stock price behavior using sentiment driving processes when we know for sure that the observed stock price time series has been simulated in a sentiment-driven market simulation. One can ask a question of whether the real-world stock data can be analyzed in a similar way, starting from the assumption that the behavior of the real market participants boils down, within a certain degree of approximation, to a few driving sentiment processes. In this spirit it would be interesting to explore how the sentiment market framework can account for the real market behavior. We will not be addressing this question in this paper, leaving it for future work.

The simplest sentiment-driven stock market environment is defined by a few well-separated sentiment regimes. By calculating the mean stock price in each of those regimes we can derive the corresponding sentiment. The probability to switch between the regimes is then small (of order of an inverse number of steps the market spends in the given sentiment state). We discuss such a situation in section 2, where we consider the market environment in which various groups of agents follow the buy/sell sentiment which changes twice over the time of the simulation. We demonstrate explicitly that the resulting mean stock price in each of the sentiment regimes is consistent with the cash flow balance equation.

A more sophisticated situation is to consider a non-trivial sentiment time series, switching regularly between states with different sentiments. A simple example of such a process, which we will be focusing on for the most part of this paper, is given by a Markov chain, with a certain transition probability matrix. The problem is then to recover the transition probability matrix of the sentiment Markov process from the observed stock price time series. We will address this problem in section 3 using the Baum-Welch algorithm of the Hidden Markov Model (HMM). For an excellent review of the HMM we refer reader to [17].

We notice for completeness that there have been some attempts in the literature to use techniques of the HMM to predict the stock price behavior [18, 19, 20]. A typical goal stated in the literature is to predict the next day's stock price using the HMM trained on the most recent stock price values. We notice that since typically the next day's stock price is correlated with the current day's stock price, any prediction prescription for the next day's price which relies on the current day's price will appear to be successful, and exhibit a high correlation scores with the actual next day's stock price. However we point out that one should be careful about how valuable is in fact such a prediction, and whether one can construct a profitable trading algorithm based on it. In the frameworks aspiring to predict the next day's stock price we suggest to test this goal by calculating the fraction of days on which at least the direction (and at most the return) of the stock price has been guessed correctly. In this paper we are not concerned with an interesting question of predicting the real world stock prices using the HMM.

If we know the transition probabilities matrix for the sentiment Markov process and if we know the current sentiment state which we are in, then we can make an informed trading decision, taking into account the most likely true market value of the stock, and the probability that such a value will change, and by how much. In section 3 we discuss application of the Viterbi algorithm of the HMM to the problem of reconstruction of the underlying hidden sentiment states. We demonstrate that due to limitations of the HMM to describe the sentiment market framework the Viterbi algorithm performs poorly, and predicts the underlying sentiment states with an accuracy being as good as a random guess. We explain in subsection 3.2 that this is essentially due to the fact that the distribution of observed stock prices depends not only on the current sentiment but also on the sequence of the recent sentiments, as contrasted with the local condition of the HMM.

Therefore in order to predict the underlying hidden sentiment states one needs a different method, which would be capable to remember the sequence of states for a fit, rather than a single state. We suggest that for this purpose one can use the technology of the Recurrent Neural Networks (RNN). In section 4 we demonstrate that using the RNN one can improve the prediction of the underlying sentiment states from the observed stock prices, and make it significantly above a random guess.

We discuss our results in section 5. In Appendix A we review the methodology of the Hidden Markov Models relevant for this paper. Appendix B is dedicated to a review of the Recurrent Neural Networks.

2. A simple buy/sell sentiment market

In this paper we study the stock market simulation framework in which all of the non-trivial price dynamics originates from specified processes, which we refer to as sentiments. In this section we discuss a simple example of a sentiment-driven stock market system, and explain our notation on this system. We will be only considering a non-trivial buy/sell sentiment process. It is defined as follows: the agent A is said to follow a buy/sell sentiment $\psi(t)$ if at the time t, provided it decides to participate in a trade, it will submit a buy order with the probability p_b and a sell order with the probability $p_s = 1 - p_b$, determined as

$$\frac{p_b}{p_s} = e^{\psi(t)} \,. \tag{1}$$

Consider the system of $N = 1000$ agents, evolving in a discrete time $t = 1, \dots, T$ over the course of $T = 10^4$ steps. Each agent A_i, $i = 1, \dots, N$, at the beginning of the considered time step, is in a possession of M_i units of cash, and S_i shares of stock. The amount of shares of stock and the amount of cash in the system is constant. At each time step each agent will submit a trade order with probability $\rho = 1/10$ to the stock exchange. A typical way to simulate trading between the agents which can be found in the literature is done by maintaining a limit order book an operating a matching engine, similar to the real stock exchange.

In this paper we clear the unfilled orders from the order book before the next time step, see however [21] for the discussion of time-frame it takes to fill large orders, and how this influences the price change. In this paper we omit discussion of this subtlety, which would be interesting to incorporate in the future work. The model discussed in this paper therefore serves to model a day trading activity, in which the agents submit their orders with the expiration tag of that day's market close. We refer reader to [13] for a recent discussion of the mechanics of the matching engine.

Regardless of the side of the order the agent will submit a limit price for its order by taking a gaussian draw centered around the most recent stock price with the standard deviation $\sigma = 0.01$. The side of the order (buy or sell) in the system discussed in this section is determined as follows. The agents are divided into two groups: the group G_1 has zN agents, $z = 1/4$, and the group G_2 has the remaining $(1 - z)N$ agents. The agents of the group G_1 are following the sentiment process

$$\psi_1(t) = \begin{cases} 0 & \text{if } t \in [0, T/3] \\ \log 2, & \text{if } t \in [T/3, T] \end{cases} \tag{2}$$

and the agents of the group G_2 are following the sentiment process

$$\psi_2(t) = \begin{cases} 0 & \text{if } t \in [0, 3T/4] \\ -\log 2, & \text{if } t \in [3T/4, T] \end{cases} \tag{3}$$

We stress here that the chosen specific sentiment drivers (2), (3) considered in this section serve only the purpose of illustration for the sentiment-driven market

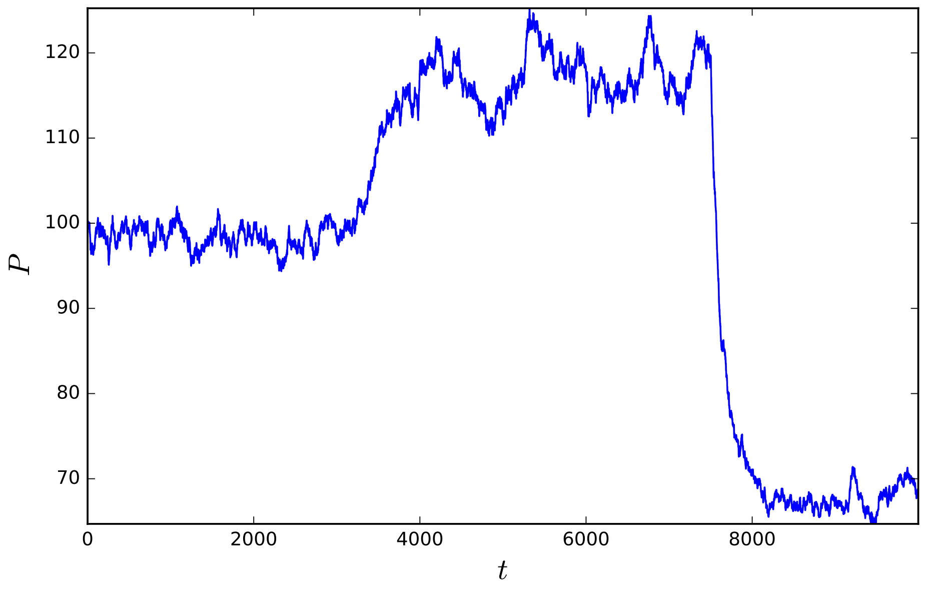

Figure 1. Stock time series for the simulation in section 2. There are three sentiment regimes: $\mathcal{T}_1 \simeq [0, T/3]$, with the mean $P_e^{(1)} = 98.5$ and volatility $\sigma^{(1)} = 1.5$, $\mathcal{T}_2 \simeq [T/3, 3T/4]$, with the mean $P_e^{(1)} = 116.7$ and volatility $\sigma^{(2)} = 3.3$, $\mathcal{T}_3 \simeq [3T/4, T]$, with the mean $P_e^{(1)} = 69.3$ and volatility $\sigma^{(3)} = 4.3$. These values are in agreement with the prediction (7), with $P_1 = 100$.

framework proposed in this paper (following [13]). We suggest that the real-world market sentiments are of a much more non-trivial form. In fact, the point of this paper is to discuss an application of the methods of the Hidden Markov Models and the Recurrent Neural Networks to the problem of inferring the real-world market sentiments.

We will assume that regardless of the side each agent A_i submits an order of a uniform size $u_i = \mathcal{U}(0, 1)$, and for a limit price $\mathcal{N}(P_{t-1}, \sigma)$, where P_{t-1} is the most recent stock price. These conditions are designed in the spirit of the sentiment-driven market framework of [13], with a constant volatility sentiment.

The starting stock price is chosen to be $P_1 = 100$. We give each agent $M_i(t = 1) = \mathcal{N}(10^5, 10^3)$ units of cash and $S_i(t = 1) = 10^3$ shares of stock. Non-identical initial wealth allocations in the sentiment-driven market framework have been considered in [13]. Under the conditions of a uniformly distributed trade size and a neutral buy/sell sentiment, the equilibrium price is $P_e = M_i/S_i = 100$, equal to the original stock price P_1. Therefore we expect that until the time $t = T/3$ the stock price will be mean-reverting around the P_1. After that the sentiment of the agents from the group G_1 becomes more bullish, so we expect the price to go up to a new equilibrium value. The length of the transition period depends on the perceived stock volatility σ and the trading intensity of the agents. For the purposes of this section we skip discussion of the transition period and calculate the new equilibrium price of a well-separated new sentiment regime.

In equilibrium the average cash flows to and from the stock market capitalization should balance each other. This condition can be derived by averaging the cash flow balance equation

$$p_b^{\text{eff}}\, M = p_s^{\text{eff}}\, S\, P_e\,, \tag{4}$$

where we have denoted the total cash in the system as $M = \sum_i M_i$, and the total number of shares outstanding as $S = \sum_i S_i$, and introduced the effective buy and sell probabilities. The latter are defined by noticing that having several groups of agents with different sentiments in each group is the same as having one group, where each agent decides at each step to act accordingly with a certain sentiment, with the corresponding probability:

$$p_b^{\text{eff}} = p(\text{buy}|G_1)\, p(G_1) + p(\text{buy}|G_2)\, p(G_2) = \frac{e^{\psi_1}}{e^{\psi_1}+1}\, z + \frac{e^{\psi_2}}{e^{\psi_2}+1}\, (1-z)\,, \tag{5}$$

$$p_s^{\text{eff}} = p(\text{sell}|G_1)\, p(G_1) + p(\text{sell}|G_2)\, p(G_2) = \frac{1}{e^{\psi_1}+1}\, z + \frac{1}{e^{\psi_2}+1}\, (1-z)\,. \tag{6}$$

Specifically for the sentiment processes (2), (3) we obtain the expected equilibrium price time dependence

$$P_e(t) = \begin{cases} P_1 & \text{if } t \in [0, T/3] \\ \dfrac{13}{11}\, P_1 & \text{if } t \in [T/3, 3T/4] \\ \dfrac{5}{7}\, P_1 & \text{if } t \in [3T/4, T] \end{cases} \tag{7}$$

This can be easily confirmed by a simulation, see figure 1.

In the setup considered in this section the market has undergone just two transitions between three sentiment regimes, over a large number of simulation steps (10^4). In such a case different sentiment regimes are well separated from each other, and we can easily calculate the mean stock price and the standard deviation for each sentiment regime, see caption of figure 1, and use the flow balance equation (4) to calculate the corresponding sentiment. We can also infer the order of magnitude of the transition probabilities between the sentiment regimes as being negligibly small, $P_{trans} \simeq \mathcal{O}(10^{-4})$.

3. Sentiment fit using Hidden Markov Models

In section 2 we discussed the simplest example of inferring sentiment properties from the observed stock price time series in a system with a well-separated sentiment regimes. A more sophisticated example of a sentiment-driven market can be constructed by considering a sentiment which is itself a non-trivial time series process. In this section we consider a simple market simulation where trading activity of the agents is influenced by a simple buy/sell sentiment process $\psi(t)$ of the Markov chain type. In subsection 3.1 we describe specifics of the simulated stock market. In subsection 3.2 we discuss how one can reconstruct the properties of the underlying sentiment process using the framework of the HMM. In subsection 3.3 we describe the outcome of multiple simulations and the HMM fit of the sentiments. We refer reader to Appendix A for a review of the HMM relevant for this paper.

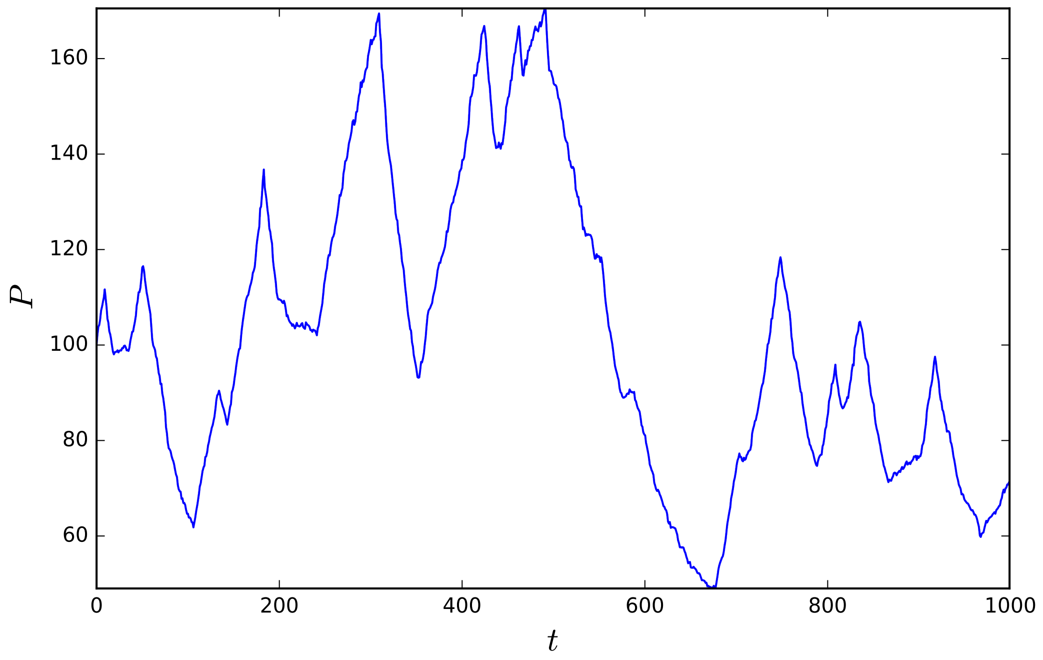

Figure 2. Stock time series for the simulation in section 3.1.

3.1. Simulation setup

Consider the system of $N = 1000$ agents trading over the period of $T = 1000$ steps. At each time step each agent decides to participate in a trade with probability $\rho = 1/10$. If it participates in a trade, it will choose the buy side with the probability p_b, and the sell side with the probability $p_s = 1 - p_b$, such that $p_b/p_s = e^{\psi(t)}$. Regardless of the side of the trade the agent will submit a limit price for its order chosen as a gaussian draw around the most recent stock price with the standard deviation being a stationary volatility sentiment process $\sigma = \mathcal{N}(0.02, 0.005)$. At the beginning we give each agent 1000 shares of stock at the initial price $P_1 = 100$, and $\mathcal{N}(10^5, 10^3)$ units of cash.

The sentiment process $\psi(t)$, governing the buy/sell decision making of the agents, is a Markov chain, with three possible states, and the corresponding equilibrium prices (4) being

- *Buy* with sentiment $\psi_b = 1$, and equilibrium price $P_e^{(b)} = 271$.
- *Neutral* with sentiment $\psi_n = 0$, and equilibrium price $P_e^{(n)} = 100$.
- *Sell* with sentiment $\psi_s = -1$, and equilibrium price $P_e^{(s)} = 36$.

The Markov chain $\psi(t)$ will be initialized randomly at $t = 1$, and the switch between the different sentiment states will occur according to the transition probabilities matrix

$$||a_{ij}|| = \begin{pmatrix} P_{b,b} & P_{b,n} & P_{b,s} \\ P_{n,b} & P_{n,n} & P_{n,s} \\ P_{s,b} & P_{s,n} & P_{s,s} \end{pmatrix}, \tag{8}$$

where $P_{i,j}$ is the probability of moving from i to j in one time step. In order to realize a long-lived hidden states we will be considering the values of a_{ij} drawn uniformly from the interval

$$a_{ii} \in [0.95, 0.98] \,. \tag{9}$$

Then due to (9) the system will spend on average 20-50 time steps in each sentiment state. In such a manner we will run multiple trading simulations, drawing the probability matrix a_{ij} randomly, taking into account the constraint (9), and generate the corresponding stock price time series.

3.2. Single HMM reconstruction

In the previous subsection we discussed the general market environment which will be used in this section to generate the stock price time series governed by a buy/sell sentiment process of the Markov chain kind. In this section we are going to reconstruct the hidden transition matrix, a_{ij}^{fit}, from the observed stock price behavior using the Baum-Welch algorithm of the HMM. We will begin by running a single simulation and explaining our approach, and in the next subsection we proceed to running multiple simulation and gathering fit results.

While we use the methods of the HMM to fit the sentiment transition probabilities matrix a_{ij}, the stock market environment discussed above is not a Hidden Markov system. Indeed, while the hidden sentiment process is constructed to be of the Markov chain type, the observed stock price is derived from an intersection of supply and demand at each time step, rather than obtained using an emission probabilities matrix (as in the HMM) from the hidden states to the observed states.

Specifically, in the HMM for each hidden state the emission probabilities for observed states are always the same. But in the stock market simulation each sentiment does not uniquely determine distribution of possible stock prices. In fact, the price is expected to be essentially influenced by the recent sentiments, for instance, it matters whether the sentiment has recently decreased or increased. Therefore a more appropriate way to fit the sentiment time series would involve a method allowing to keep track of the sequence of states, rather than one state. We will employ such a method in the next section, where we will be considering application of the Recurrent Neural Networks to the problem of inferring the sentiments.

Let us incorporate a buy/sell sentiment driving process of the Markov chain type with the transition probability matrix

$$a_{ij} = \begin{pmatrix} 0.95 & 0.025 & 0.025 \\ 0.035 & 0.93 & 0.035 \\ 0.05 & 0.05 & 0.9 \end{pmatrix} . \tag{10}$$

into the market system described above in this section. Running a simulation over $T = 1000$ steps we have generated the stock price time series plotted in figure 2. We are going to take the stock price time series produced in this simulation as an observable input. Following the Baum-Welch algorithm we were able to reconstruct the transition probabilities matrix as

$$a_{ij}^{\text{fit}} = \begin{pmatrix} 0.93 & 0.07 & 0 \\ 0.06 & 0.82 & 0.12 \\ 0 & 0.09 & 0.91 \end{pmatrix}. \tag{11}$$

Notice that this result is rather close to the actual transition probabilities matrix (10). It is notable that such an accuracy drops sharply if one violates the constrain given in (9). The reason for taking that constraint is to have the market reside in a state with the given sentiment for a long enough time. This way the stock price can spend enough time close to the corresponding equilibrium value $P^{(e)}$, thereby allowing the HMM to train on that value as the one corresponding to each specific sentiment ψ. On the other hand in a system with short-lived sentiment states assumptions of the HMM will be invalid, as discussed above, rendering the Baum-Welch algorithm inapplicable. We will repeat such a simulation and reconstruction below multiple times and confirm this result.

Applying the Viterbi algorithm we obtained the fit score 0.4, that is, only 40% of the Viterbi predictions of the underlying sentiments match the reality. This is almost as good as a random guess of one out of three sentiments, as we will confirm below by running multiple simulations and aggregating scores of the Viterbi predictions. Such a poor performance of the Viterbi algorithm for the market sentiment reconstruction is in fact anticipated, and as discussed above can be explained by the fact that the stock market simulation is not really a Hidden Markov Model.

3.3. Repeated HMM reconstructions

In this subsection we are going to discuss the results of the repeated simulations, similar to those described in subsection 3.2. For each simulation we are going to initialize the transition probabilities matrix of the hidden sentiment Markov process randomly, with the only demand that it satisfies the constraint (9). For each generated sentiment process we simulate the stock market evolution in the setup described in subsection 3.1. We then perform the Baum-Welch fit for the transition probabilities matrix a_{ij}, and the Viterbi fit for the sentiment states $\psi(t)$.

The result of the Baum-Welch fit is a 3×3 plot in figure 3, where each subplot corresponds to a hidden transition probabilities matrix element. Specifically, on each panel we provide a scatterplot of the Baum-Welch fit for the transition probability matrix elements, a_{ij}^{fit}, vs. the actual transition probability matrix elements, a_{ij}, used in that simulation.

For each simulation described above we also use the Viterbi algorithm to reconstruct the sentiment state $\psi(t)$ at each time step t. We plot the results in figure 4. The mean score is 0.33, which is as good as guessing one of three sentiment states at random. This is a reflection of the fact that the sentiment market environment, as discussed above, is not a hidden Markov system.

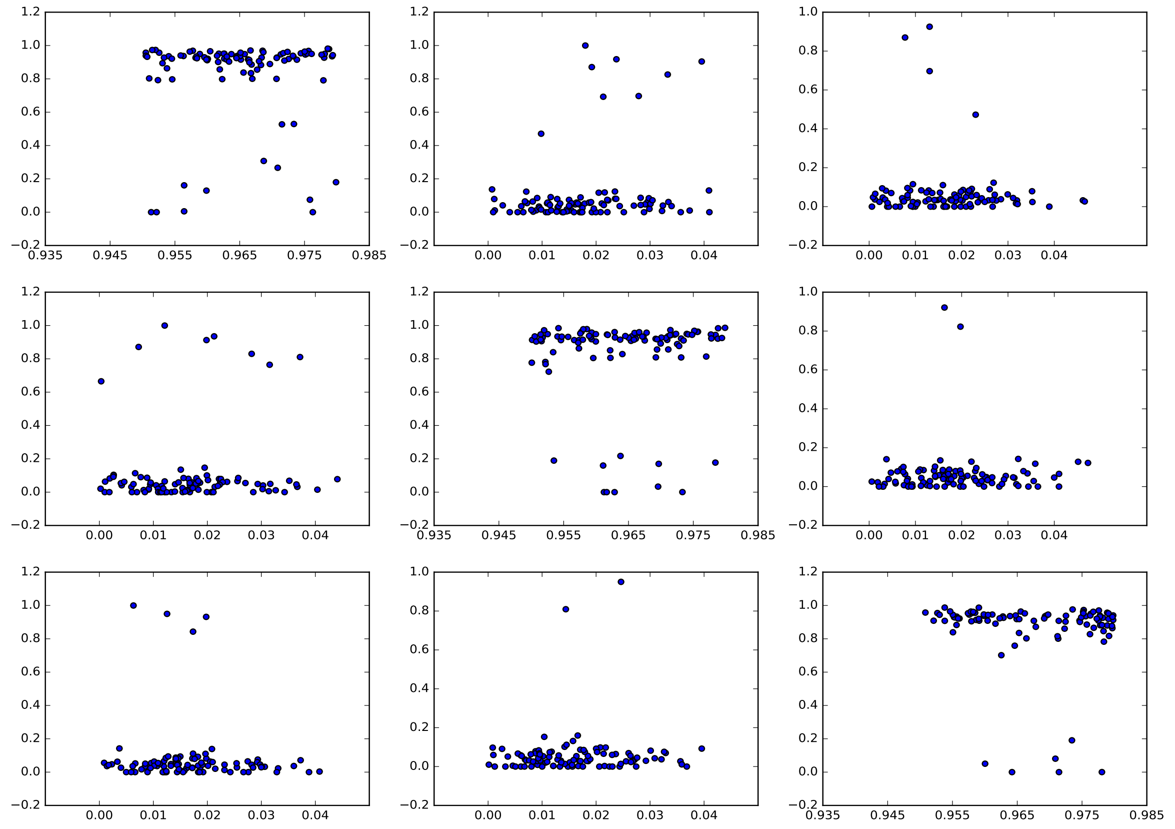

Figure 3. Results for the sentiment transition probabilities matrix fit in subsection 3.3 over 100 simulations. Each pane of the 3×3 plot corresponds to the fit results for one of the nine matrix elements. The horizontal axis represents the actual value of the matrix element a_{ij}, and the vertical axis represents the Baum-Welch fit a_{ij}^{fit}.

4. Sentiment fit using Recurrent Neural Network

In section 3 we discussed an approach to reconstruct properties of the hidden sentiment process from the observed stock price behavior using the methodology of the HMM. We have demonstrated that the Baum-Welch algorithm of the HMM can be used to recover the transition probabilities matrix for the sentiment Markov process with a sufficiently long-lived sentiment states. We have also applied the Viterbi algorithm to estimate the underlying sentiment states from the observed stock prices and demonstrated that the resulting predictions are not satisfactory, and are no better than a random guess. We have argued in subsection 3.2 that such a poor performance of the HMM fit for the sentiment states is due to the fact that the simulated stock market, unlike the HMM, is influenced by the sequence of recent states, rather than a single current state. As an alternative to the Viterbi algorithm in this section we attempt another approach, using the Recurrent Neural Networks, to the problem of recovering sentiment states from the stock price time series. We refer reader to Appendix B for a review of the RNN, relevant for the purposes of this paper.

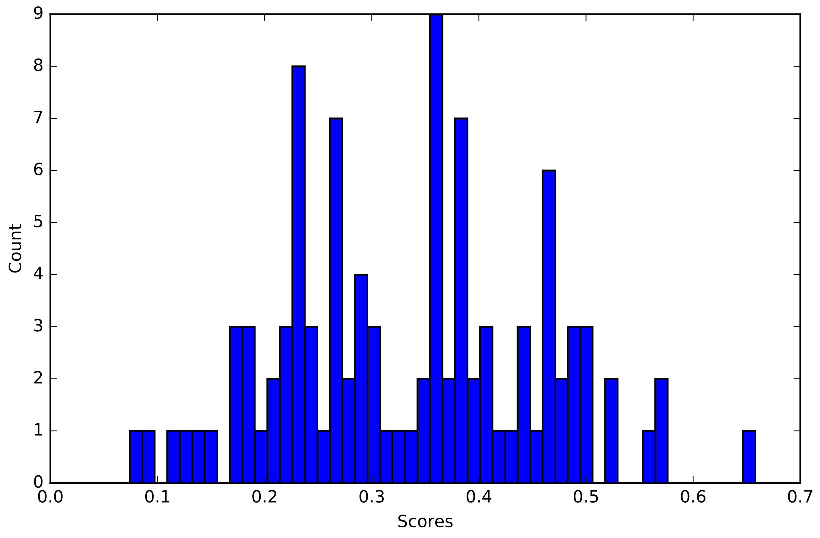

Figure 4. Results for the sentiment states fit in subsection 3.3 over 100 simulations, obtained using the Viterbi algorithm. The score on the x-axis represents the fraction of states reconstructed correctly.

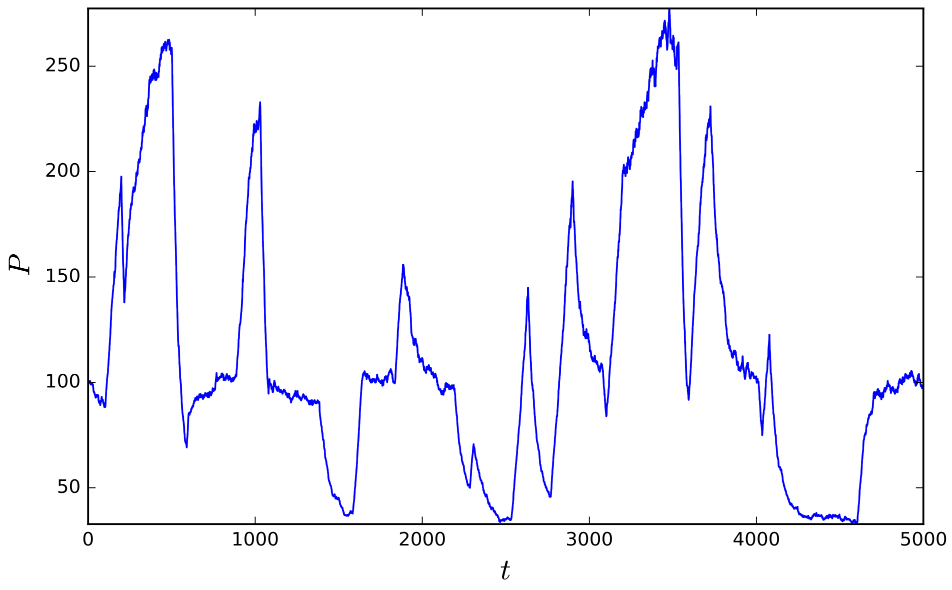

Figure 5. Stock time series for the simulation in section 4.

An RNN is a neural network designed to fit time series, and capable of remembering and training on sequences of states. An input for an RNN is a time series $\{P_t\}$, $t = 1, \dots$ of an arbitrary length, and one can train RNN on sequences of data of length T (unfold the RNN into T time layers). We are interested in reading an output $\hat{\psi}_t$ at each t, which will be the buy/sell sentiment of the market at time t. The hat indicates that this is the fit, rather than the true sentiment ψ_t.

Consider the system of $N = 1000$ agents over $T = 5000$ simulation steps, driven by a buy/sell sentiment process of the Markov chain type with the states $\psi = -1, 0, 1$, and the transition probabilities matrix of the form (8) defined to be equal to

$$a_{ij} = \begin{pmatrix} 0.9948 & 0.0002 & 0.005 \\ 0.0016 & 0.9962 & 0.0022 \\ 0.0044 & 0.0025 & 0.9931 \end{pmatrix}. \tag{12}$$

We also include the volatility sentiment as a stationary process $\sigma(t) \sim \mathcal{N}(0.02, 0.005)$. Running the simulation we obtain the stock price time series in figure 5. Notice that the system spends about 100 steps in each sentiment state. We split the 5000 steps into the first 4500 steps, used as a train set, and the last 500 steps, used as a test set. We train the RNN on the chunks of data with $T = 100$ steps from the train set. The length of the memory layer is taken to be 200. We then ask the RNN to predict sentiments from the stock price observations in the test set. The train score then turns out to be 0.51, and the test score is 0.54, reflecting the fractions of sentiment states inferred correctly by the RNN. This is significantly better than 0.33 score of the random guess.

In order to confirm this result we repeat the calculation over 100 simulations, drawing the diagonal elements of the sentiment transition probability matrix uniformly from the interval $[0.97, 1)$. We noticed that in order to optimize the average RNN performance one should unfold the RNN for the T steps in the range $T \simeq [25, 75]$. In figure 6 we plot the scores obtained for $T = 50$. This makes sense, because we need T to be larger than the length of the transition period between the sentiment states. On the other hand T should be smaller than the life-time of a sentiment state. The train set score and the test set score are equal to 0.56, 0.5 respectively, which is an improvement compared to the 0.33 random guess score. We speculate that not quite perfect performance of an RNN is due to the properties of the simulated stock price time series. It would be interesting to classify systematically what kind of simulated stock price times series in a sentiment-driven stock market framework allows for a more accurate reconstruction of the underlying sentiment states.

5. Conclusions and discussion

In this paper we discussed the problem of inferring the properties of the hidden sentiment process from the observed stock price behavior in a sentiment-driven simulated stock market framework of [13]. We have mostly considered the systems in which the sentiment process is a Markov chain, and approached the problem of retrieving the Markov

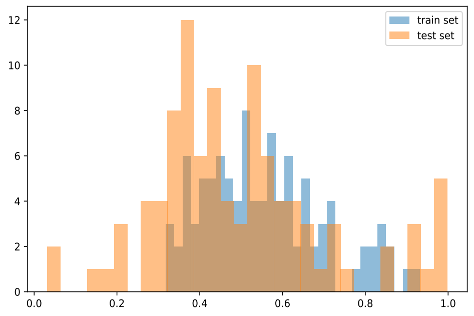

Figure 6. Results for the sentiment states RNN fit in section 4 for 100 simulations. The score on the x-axis represents the fraction of states reconstructed correctly. The train set is the first 90% of the data points, the test set is the last 10%. The mean of the train set is 0.56, the mean of the test set is 0.5.

transition probabilities matrix, and the sentiment states themselves, from the stock price time series. To tackle this problem we proposed to use the methods of the Hidden Markov Model and the Recurrent Neural Network. While it would be interesting to infer the sentiment processes using techniques of the HMM and RNN, in this paper we only considered reconstructing sentiments of the stock time series which have been simulated, rather than taken from the real world market data.

We demonstrated that the Baum-Welch algorithm of the Hidden Markov Model allows to successfully reproduce the sentiment transition probabilities matrix. This is true provided the system exhibits a long-lived sentiment states. We also demonstrated that the prediction performance of the Viterbi algorithm of the HMM, applied to infer the sentiment states at each time step, is as good as a random guess. This is due to the fact that, the Viterbi algorithm is predicting the most probable path between hidden states and observation, assuming that for the given hidden state the distribution of observations is always the same. In other words, in the HMM the observable state, which in our case is the stock price, is determined locally from the hidden state, which in our case is the sentiment, by the fixed emission probability matrix. Therefore the distribution of observables is always the same for the same hidden states. However, in the case of the simulated stock price the distribution of a price at the given time moment depends on the sequence of the recent hidden sentiment state, not only on the single most recent sentiment.

We observed that in order to be able to recreate the underlying sentiment state from the observed stock price time series we need to have a model which will fit the current stock price against the sequence of the recent sentiment states, not just against the current sentiment state. For this purpose we proposed to use the technology of the Recurrent Neural Network, which is known as a framework capable of remembering the sequence of states and fitting time series. We demonstrated that the accuracy of the RNN method is around 50%, being significantly better than the random score of 33%.

We suggest the further tuning of the RNN can improve the fit score even further.

In this paper we have refrained from applying the sentiment-driven market framework to the real-world market, leaving this interesting direction for future work. We note that our results for the Baum-Welch and RNN performance suggest that our method can be adequate for day rather than intra-day stock price time series, so that one would have regimes with potentially long-lived sentiment states. It is worth mentioning that our approach can be used for more complicated models involving option trading. This is because most of options are traded over a long run.

Acknowledgements

M.G. was supported by Oehme Fellowship.

Appendix A. Hidden Markov Models

In this appendix we review the known techniques of the Hidden Markov Models. For an excellent introduction we refer the reader to [17].

The framework of HMM is based on the concept of an ordinary Markov chain. Consider a random process $\{x_t\}$ in discrete time, $t = 1, 2, \ldots, T$, where T is the length of the period over which we study the process. At each time step t the random variable x_t can be in one of N states, denoted by $X_1, X_2, \cdots, X_N$. We cannot observe the states $\{x_t\}$ directly. Instead we observe the process $\{y_t\}$, where each y_t can be in one of M states Y_1, Y_2,..., Y_M.

The HMM is characterized by the following parameters:

- Hidden state transition probability distribution $A = \{a_{ij}\}$, where a_{ij} is the $N \times N$ matrix of transition probabilities between hidden states, *i.e.*

$$a_{ij} = P(x_{t+1} = X_j | x_t = X_i), \qquad 1 \leqslant i, j \leqslant N.$$

- Observable state probability distribution $B = \{b_{ik}\}$, where $\{b_{ik}\}$ is the $N \times M$ matrix of emission probabilities between hidden states to observed states, *i.e.*

$$b_{ik} = P(y_t = Y_j | x_t = X_i), \qquad 1 \leqslant i \leqslant N, 1 \leqslant k \leqslant M.$$

- Initial distribution $\pi = \{\pi_i\}$, where π_i is N-tuple of probabilities of the initial hidden states, *i.e.*

$$\pi_i = P(x_1 = X_i), \qquad 1 \leqslant i \leqslant N.$$

In this manner we shall define the discrete HMM by a triplet:

$$\lambda = (A, B, \pi). \tag{A.1}$$

If we observe the sequence $\{y_t\}$, $t = 1, \ldots, T$, and we know the spectrum of observable states $\{Y_t\}$, and the dimension of hidden states, we can use the Baum-Welch algorithm to infer the parameters λ of the HMM. We begin by initializing λ by a guess, and then iterating the following procedure until it converges. First we calculate the forward and backward probabilities

$$\hat{\alpha}_{it} = P(x_t = X_i | y_1 \cdots y_t)\,, \qquad \hat{\beta}_{it} = \frac{1}{c_{t+1} \cdots c_T}\, P(y_{t+1} \cdots y_T | x_T = X_i)\,, \tag{A.2}$$

$$c_t = P(y_t | y_1 \cdots y_{t-1}) \tag{A.3}$$

using the the recursion relations

$$\alpha_{i1} = \pi_i\, b_{ik_1}\,, \quad c_{t+1}\, \hat{\alpha}_{i\,t+1} = \sum_j \hat{\alpha}_{jt}\, a_{ji}\, b_{i\,k_{t+1}}\,, \tag{A.4}$$

$$\hat{\beta}_{iT} = 1\,, \quad c_{t+1}\, \hat{\beta}_{it} = \sum_j \hat{\beta}_{j\,t+1}\, a_{ij}\, b_{j\,k_{t+1}}\,, \tag{A.5}$$

$$\sum_i \hat{\alpha}_{it} = 1\,, \quad \hat{\alpha}_{i1} = \frac{\alpha_{i1}}{c_1}\,, \tag{A.6}$$

where we have denoted $y_t = Y_{k_t}$, Next we calculate the probabilities.

$$\gamma_{it} = P(x_t = X_i | y_1 \cdots y_T) = \hat{\alpha}_{it}\, \hat{\beta}_{it}\,, \tag{A.7}$$

$$\xi_{tij} = P(x_t = X_i, x_{t+1} = X_j | y_1 \cdots y_T) = \frac{1}{c_{t+1}}\, \hat{\alpha}_{it} \hat{\beta}_{j\,t+1}\, a_{ij}\, b_{j\,k_{t+1}}\,. \tag{A.8}$$

Finally we update the λ parameters as

$$\pi_i = \gamma_{i1}\,, \qquad a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{tij}}{\sum_{t=1}^{T-1} \gamma_{it}}\,, \quad b_{ik} = \frac{\sum_{t=1}^{T} \delta(y_t = Y_k)\, \gamma_{it}}{\sum_{t=1}^{T} \gamma_{it}}\,. \tag{A.9}$$

The underlying hidden states $\{x_t\}$ can be inferred using the Viterbi algorithm. One defines two auxiliary matrices R_{it}, Q_{it} of size $N \times T$, where R_{it} is the probability of the most likely sequence $x_1 \cdots x_t$ given the observed sequence $y_1 \cdots y_t$, such that $x_t = X_i$. It can be calculated recursively as

$$R_{i1} = \pi_i\, b_{ik_1}\,, \quad R_{it} = \max_j (R_{j\,t-1}\, a_{ji})\, b_{ik_1}\,. \tag{A.10}$$

The Q_{it} is x_{t-1} of the most likely hidden sequence $x_1 \cdots x_t$ given the observation $y_1 \cdots y_t$. It can be calculated from the known R_{it} as

$$Q_{it} = \arg\max_j (R_{j\,t-1}\, a_{ji})\,. \tag{A.11}$$

The most likely hidden state at $t = T$ is then

$$x_T = \arg\max_i (R_{iT})\,, \tag{A.12}$$

which is then used to initialize iterative calculation

$$x_t = Q_{x_{t+1}\,t+1}\,. \tag{A.13}$$

Appendix B. Recurrent Neural Network

In this appendix we review the simplest kind of a Recurrent Neural Network (RNN). An RNN is the neural network able to fit the data represented as a time series. A typical problem which can be addressed by an RNN is to predict the output times-series $\{y_t\}$ from the input time series $\{x_t\}$, where $t = 1, 2, \ldots$. An RNN is designed to be flexible to the specific length of the time series. We can train it on the chunks of data of the length T. Suppose an input at the given time t is a vector x_t of dimension S and an output y_t is a vector of dimension N. The x_t is received by the input layer, and the y_t is produced by the output layer. Similarly to the usual neural network the RNN has a hidden layer, in between the input and the output layers. The hidden layer m_t is the 'memory' of the RNN, influenced both by the current input x_t, and by all the previous inputs x_τ, $\tau = 1, 2, \ldots, t-1$. We represent the memory layer m_t as a vector of size M.

An RNN is characterized by the hidden bias vector b, output bias vector e, input-to-weight matrix W, memory-to-memory matrix V, and memory-to-output matrix U. We will describe these parameters collectively as

$$\mathcal{R} = (W, V, U, b, e)\,. \tag{B.1}$$

The core functionality of the RNN is described by the equations

$$m_t = f\left(W\,x_t + V\,m_{t-1} + b\right)\,, \tag{B.2}$$

$$y_t = U\,m_t + e\,, \tag{B.3}$$

with some classifier function f, which in this paper we choose to be $f \equiv \tanh$. To summarize, the inner structure of the RNN is characterized by the t-independent parameters $\mathcal{R}$ and by the t-dependent memory vectors m_t. To train an RNN means to fit the parameters $\mathcal{R}$ to produce the desired output $\{y_t\}$ from the given input $\{x_t\}$.

It is convenient to represent the desired outputs y_t in such a way that at each given time step t we want the output y_t to be equal to the vector R_t with the components

$$R_{it} = \delta_{i\,r_t}\,, \quad i = 1, \ldots, N\,. \tag{B.4}$$

Then we can represent the desired output as the time series of index values $\{r_t\}$ of the non-zero y_t vector component. We can match this desired output vector to the actual output y_t by calculating the probabilities of the r_t values

$$P_{it} = \frac{e^{y_{it}}}{\sum_j e^{y_{jt}}}\,, \quad i = 1, \ldots, N\,, \tag{B.5}$$

and using those probabilities in the loss function, see, *e.g.*, [22]

$$L = -\sum_t \log\left(\sum_i P_{it}\,R_{it}\right)\,. \tag{B.6}$$

The RNN parameters $\mathcal{R}$ can then be tuned over many training iterations using the backpropagation of the loss function L. We want the loss function value to be as small as possible, so we will shift the values of the parameters $\mathcal{R}$ in the direction opposite to the gradients of the loss function L w.r.t. those parameters. We summarized all the key gradients here,

$$\left[\nabla^{(y)} L\right]_{mt} \equiv \frac{\partial L}{\partial y_{mt}} = P_{mt} - \delta_{m\, r_t}\,, \tag{B.7}$$

$$\left[\nabla^{(U)} L\right]_{ij} \equiv \frac{\partial L}{\partial U^{ij}} = \sum_t \left[\nabla^{(y)} L\right]_{it} m_{jt}\,, \tag{B.8}$$

$$\left[\nabla^{(e)} L\right]_{i} \equiv \frac{\partial L}{\partial e^{i}} = \sum_t \left[\nabla^{(y)} L\right]_{it}\,, \tag{B.9}$$

$$\left[\nabla^{(m)} L\right]_{it} \equiv \frac{\partial L}{\partial m_{it}} = \sum_j U^{ji} \left[\nabla^{(y)} L\right]_{jt} + \left[\nabla^{(m_n)} L\right]_{it}\,, \tag{B.10}$$

$$\left[\nabla^{(m_n)} L\right]_{it} \equiv \sum_j \left[\hat{\nabla}^{(m)} L\right]_{j\,t+1} V^{ji}\,, \tag{B.11}$$

$$m_{it} \equiv \tanh \hat{m}_{it}\,, \tag{B.12}$$

$$[\hat{\nabla}^{(m)} L]_{it} \equiv \frac{\partial L}{\partial \hat{m}_{it}} = \left[\nabla^{(m)} L\right]_{it} (1 - m_{it}^2)\,, \tag{B.13}$$

$$\left[\nabla^{(W)} L\right]_{ij} \equiv \frac{\partial L}{\partial W^{ij}} = \sum_t [\hat{\nabla}^{(m)} L]_{it}\, x_{jt}\,, \tag{B.14}$$

$$\left[\nabla^{(V)} L\right]_{ij} \equiv \frac{\partial L}{\partial V^{ij}} = \sum_t [\hat{\nabla}^{(m)} L]_{it}\, m_{j\,t-1}\,, \tag{B.15}$$

$$\left[\nabla^{(b)} L\right]_{ij} \equiv \frac{\partial L}{\partial b^{i}} = \sum_t [\hat{\nabla}^{(m)} L]_{it}\,. \tag{B.16}$$

Once we know the gradients we know by how much we need to shift the parameters $\mathcal{R}$ in order to decrease the loss function,

$$\mathcal{R} \to \mathcal{R} - \frac{r}{\rho}\,[\nabla^{(\mathcal{R})} L]\,, \tag{B.17}$$

where the learning rate is $r \simeq 0.1$, and the adaptive learning rate [23] is

$$\rho = \sqrt{\sum_n [\nabla^{(\mathcal{R})} L]^2}\,, \tag{B.18}$$

and the sum is done over all the iterations we have performed up till now, inclusive.

References

[1] R. G. Palmer, W. B. Arthur, J. H. Holland, B. LeBaron, and P. Tayler, Artificial economic life: a simple model of a stock market, *Physica D*, 75, 264, 1994.

[2] W. B. Arthur, J. H. Holland, B. LeBaron, R. Palmer, and P. Tayler, Asset Pricing Under Endogenous Expectations in an Artificial Stock Market, 1996.

[3] R. G. Palmer, W. B. Arthur, J. H. Holland, and B. LeBaron, An artificial stock market, *Artificial life and robotics*, 3, 1, 27, 1999.
[4] B. LeBaron, W. B. Arthur, and R. Palmer, Time series properties of an artificial stock market, *Journal of Economic Dynamics and Control*, 23, 9, 1487, 1999.
[5] M. Raberto, S. Cincotti, S. M. Focardi, and M. Marchesi, Agent-based simulation of a financial market, *Physica A*, 299, 1, 319, 2001.
[6] E. Bonabeau, Agent-based modeling: methods and techniques for simulating human systems, *Proceedings of the NAS of the USA*, 99, 10, 3, 2002.
[7] J. D. Farmer, D. Foley, The economy needs agent-based modelling, *Nature*, 460, 685, 2009.
[8] S. Pastore, L. Ponta, and S. Cincotti, Heterogeneous information-based artificial stock market, *New Journal of Physics*, 12, 2010.
[9] L. Ponta, M. Raberto, and S. Cincotti, A multi-assets artificial stock market with zero-intelligence traders, *Europhysics Letters*, 93, 2, 2011.
[10] L. Ponta, S. Pastore, S. Cincotti, Information-based multi-assets artificial stock market with heterogeneous agents, *Nonlinear Analysis-Real World Applications*, 12, 2, 1235, 2011.
[11] M. A. Bertella, F. R. Pires, L. Feng , H. E. Stanley, Confidence and the stock market: an agent-based approach, *PLoS ONE*, 9(1): e83488, 2014.
[12] E. Immonen, Simple agent-based dynamical system models for efficient financial markets: Theory and examples, *Journal of Mathematical Economics*, 69, 38, 2017.
[13] M. Goykhman, Wealth dynamics in a sentiment-driven market, *Physica A*, 488, 132, 2017.
[14] E. Samanidou, E. Zschischang, D. Stauffer, and T. Lux, Agent-based models of financial markets, *Reports on Progress in Physics*, 70, 3, 2007.
[15] B. Mandelbrot, The variation of certain speculative prices, *Fractals and scaling in finance*, 371, 1997.
[16] M. Baker and J. Wurgler., Investor sentiment in the stock market, *Journal of Economic Perspectives*, 21(2):129?151, 2007.
[17] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE*, 77, 2, 1989.
[18] M. R. Hassan, B. Nath, Stock Market Forecasting Using Hidden Markov Model: A New Approach, *ISDA*, 2005.
[19] A. Gupta, B. Dhingra, Stock market prediction using Hidden Markov Models, *IEEE*, 2012.
[20] G. Kavitha, A. Udhayakumar, D. Nagarajan, Stock Market Trend Analysis Using Hidden Markov Models, *IJSCNS*, 11(10), 103, 2013.
[21] J. - P. Bouchaud, J. D. Farmer, F. Lillo, How Markets Slowly Digest Changes in Supply and Demand, *Handbooks in Finance*, 57, 2009.
[22] A. Karpathy, CS231n: Convolutional Neural Networks for Visual Recognition, *Stanford CS class.*
[23] J. Duchi, E. Hazan, Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, 12, 2121, 2011.