A dynamical model for stock forecasting via deep recurrent dictionary learning

Shalini Sharma, Émilie Chouzenoux, Víctor Elvira, Angshul Majumdar

A dynamical model for stock forecasting via deep recurrent dictionary learning

Shalini Sharma, Émilie Chouzenoux, *IEEE Senior Member*, Víctor Elvira, *IEEE Senior Member*, and Angshul Majumdar

***Abstract*—State-space models (SSM) and recurrent neural networks (RNN) are widely used approaches for dynamical system modeling. In the case of SSMs, they include explicit modeling of all components, including the noise characterization, and thus allow for interpretability and uncertainty quantification. However, the underlying dynamical model parameters need to be specified and closed-form inference is possible only in a few simple cases. RNNs, on the other hand, can learn, through supervised training, rather complex nonlinearities from the data but lack the aforementioned advantages of SSMs. In this work, we combine the benefits of both approaches by introducing a Gaussian SSM whose state and evolution operators can be learnt from the data. In order to deal with the ill-posedness of this parameter estimation problem, we propose an innovative factorized form of both the state and observation operators, reminiscent from deep nonnegative matrix factorization models. An expectation-maximization method combined with a block alternating strategy is introduced to estimate each of the involved positive latent factors, while jointly performing the probabilistic state inference. Our resulting formulation and inference tool is called deep recurrent dictionary learning (DRDL). We then specialize DRDL for the problem of stock forecasting, by proposing an online training strategy and a probabilistic assessment of the trading decision. Numerical experiments on a problem of stock market data inference shows its superiority among several state-of-the-art dynamic modeling tools.**

***Index Terms*—Time series analysis; State space models; Deep nonnegative matrix factorization; Kalman filtering; Bayesian smoothing; EM algorithm; Stock forecasting; Stock trading.**

I. Introduction

Modeling dynamical systems has been a topic of interest to signal processing, machine learning and control engineering researchers for more than five decades. Applications range in areas as diverse as financial market analysis to electric demand forecasting. We propose a new dynamical recurrent modeling technique that combines the advantages of state-of-the-art deep learning tools with those of traditional state-space models. The proposed tool is then particularized to the processing of stock market time series. In the following, we review the literature of dynamic modeling around this particular application and we describe the contributions of the paper.

S. Sharma and A. Majumdar are with the Indraprastha Institute of information Technology (India).

E. Chouzenoux is with Université Paris-Saclay, Inria, CentraleSupélec, ´ Centre de Vision Numerique (France)

V. Elvira is with the School of Mathematics at the University of Edinburgh (UK).

Manuscript edited June 1, 2021;

A. State-of-the-art review

Modeling the stock market is a well-known challenging problem [1]. The difficulty lies in the non-stationary and non-linearity of the underlying dynamical process. Moreover, financial markets are not only influenced by consumer behavior but also by a myriad of external factors like natural disasters, administrative policies, political decisions, international relations, etc., to name a few. Therefore developing reliable algorithmic models for stock trading still remains a challenging yet interesting topic from the point of view of both finance and machine learning/signal processing [2], [3].

Auto-regressive moving average (ARMA) models have been used to model stock market [4], [5]. ARMA assumes the stochastic process to be stationary; this turns out to be too simplistic and consequently unrealistic for the stock market. This limitation was partially overcome by autoregressive integrated moving average (ARIMA) [6] models (also referred as Box-Jenkins model). ARIMA has been used in the past for stock forecasting and trading [7], [8]. However, Box-Jenkins/ARIMA methods could not model non-smooth variations in time series [9], [10]. ARIMA with regressors were introduced to overcome the limitations, leading to ARIMAX [11], [12]. Unfortunately, ARIMAX introduced other problems such as over/under-fitting because of the handling of extra predictors and variables.

SSM is another classical approach for modeling and analysing time-series. Many studies used SSMs for stock forecasting, and analysis [13], [14]. The celebrated Kalman filter is a solution to the inference of a linear SSM where the noise is assumed to be Gaussian [15]. The literature illustrates its minimal use in stock forecasting [16] but found its application in other financial analyses, see for example [17]. To overcome the restrictive linearity assumption, extended Kalman filter (EKF) [18], unscented Kalman filter (UKF) [19] were introduced. Particle filters [20], [21] further relaxed the Gaussianity assumption. The advantage of SSM is that it can model uncertainty in the estimate [21], [22], [23]. Uncertainty is crucial for financial markets since it gives a measure of the associated risk [24]. The main drawback of the aforesaid signal processing-based forecasting approaches is that they need the model's specification. Unfortunately, specifying an underlying model for the stock market is difficult, if not impossible. Several works in the literature have thus investigated the learning of model parameters in SSMs. In the case of linear-Gaussian state-space models (LG-SSMs), see for instance the methods in [25], [26], [27], and [28, Chapter 12].

All these works consider the observation and state operators to be unknown and estimated from data using expectation-maximization (EM) methods. However, the aforementioned works can only consider linear models and do not account for any prior knowledge on the involved operators. Inferring model parameters for non-linear SSMs has been explored in more generic algorithms, e.g., particle MCMC methods [29], SMC^2 [30], and nested PFs [31]. In all these cases, the inference is costly, as they use Monte-Carlo sampling methods, and thus do not generally scale well. The problem of scalability in SSM model inference has been midly explored. Let us mention our two recent works [23], [32], both focusing on LG-SSMs. In [23], a sparsity prior in introduced on the linear matrices to infer, providing an interpretable and compressible model. Though this method is promising, it does not allow easily an explicit control of the final dimension of the model, and as such, still requires an increased computational time at inference. In [32], we proposed an online (still EM-based) estimation approach in the context of stock market time series processing. The online processing allows a reduced complexity and memory burden, while being beneficial to the capture of non-linear phenomena in such volatile time series. However, the parametric estimation step lacked of robustness, probably by lack of sufficient imposed structured on the estimated factors.

Neural network (NN) models represent another family of approaches for time series modeling. By construction, these methods excel when model specification is missing, as they learn implicitly the model from the data through the training phase. In particular, the approximation capability of recurrent neural networks (RNNs) for dynamical systems allows to learn the underlying phenomena given enough training data [33], [34]. RNN and its subsequent versions are used in several studies for stock price forecasting [35], [36]. Deeper neural network architectures are known to yield better results than their shallow counterparts [37], [38]. They are engineered to approximate highly non-linear function in high-dimensional spaces and are supposed to be more suitable for challenging problems [39], [40]. 1-D CNN performs better when compared to LSTM and RNN owing to their ease of training. There are studies, such as [41], that use them as financial forecasting. However, 1-D CNNs cannot process streaming data. Hence some works have recently proposed the combination of RNN with 1-D CNN in order to model time-series signals [42]. It must be noted that deep neural networks are computationally intensive [43]. Furthermore, deep neural networks only provide data estimates for each time step and do not provide uncertainty quantification, while such information would be necessary for stock forecasting to assess risks. Therefore, recent works have been dedicated to combine probabilistic forecasting and deep learning techniques, so as to predict the probability distribution of future events in the time series given its past/historical recordings [44], [45]. Deep factor model based on dropout-based heuristic and complex semantics have also been considered in [46]. These techniques provide probability distributions as outputs, thanks to specific learning strategies inherited from the Bayesian NN litterature. However, these works, up to our knowledge, do not mention any explicit strategy to estimate uncertainty/confidence score on future predictions/decisions that would help to assess their reliability. Standard (non deep) machine learning models have also been combined to statistical time series modeling tools. For instance, the work [47] uses ARIMA and support vector machine. Another work uses a combination of ARIMA and random forests for the same task [48]. Finally, several works, such as [49], [50], [51], use information mined from news articles and blogs via natural language processing for stock forecasting. Strictly speaking, this is not artificial intelligence since these are dependent on human cognizance.

B. Contributions compared to existing literature

As a summary, SSMs are valuable tools for probabilistic time series modeling and inference. But there is a crucial need for new strategies to cope with the curse of dimensionality in the learning of SSM model parameters. In this work, we propose to impose a structured prior on the observation/state operators involved in an LG-SSM. We introduce deep nonnegative matrix factorized (deep NMF) models for both operators. Deep NMF [52] is a generalized form of NMF [53] that models latent representations from complex data through a product of a (usually small) number of linear operators (called latent factors) satisfying positivity constraints. Deep NMF has been employed with success on various unsupervised machine learning tasks [54], [55], [56], [57], [58]. When embedded into an NN structure, it leads to the so-called deep ReLu networks [59], [60], [61]. Deep NMF shares connections with the recently introduced deep dictionary learning (deep DL) [62], [63], the main difference being in the priors imposed in the latent factors (positivity, in the case of deep NMF, low-rank/sparsity in the case of deep DL). In our work, deep NMF is neither used for unsupervised representation learning nor in an NN framework. In contrast, it is embedded into a Gaussian SSM to model, allowing to track and predict complex latent phenomena in time series. A novel algorithm is proposed, that learns the positive latent factors jointly with the probabilistic state inferential task induced by our SSM. We call this modelling and inference tool *Deep Recurrent Dictionary Learning* (DRDL). We further specialize this tool, to make it practically efficient in the context of large and volatile time series arising in stock market data. In particular, we perused the online training strategy we previously introduced in [32].

Contributions in a nutshell. In this work, we:

- Introduce an LG-SSM model involving deep positive latent factors;
- Propose a new EM-based inference method to jointly perform the time series prediction task and the deep linear positive factors estimation;
- Devise efficient implementation strategies for practical use of the method in the context of stock market time series analysis;
- Investigate through experiments and benchmark comparisons on real financial data the performance of the novel DRDL approach.

C. Paper organization

The rest of the paper is organized as follows. The proposed DRDL model and its inference is presented in the following Section II. The practical implementation of DRDL in the context of stock market data analysis is discussed in Section III. The experimental results and their analysis are described in Section IV. The conclusions of this work are discussed in section V.

II. Proposed Work

A. Considered model

Let us consider an observed sequence $(\mathbf{x}_k)_{1\leq k\leq K}$ of vectors of size $N_x \geq 1$. We aim to estimate $(\mathbf{z}_k)_{1\leq k\leq K}$, a sequence of unknown hidden/latent vectors of size $N_z \geq 1$. The DRDL approach relies on the following re-parametrized LG-SSM: For every $k \in \{1, \ldots, K\}$,

$$\left\{ \begin{array}{ll} \mathbf{z}_k & = \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_{k-1} + \mathbf{v}_{1,k}, \\ \mathbf{x}_k & = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k + \mathbf{v}_{2,k}. \end{array} \right. \quad (1)$$

where $\mathbf{D}_0 \in [0,+\infty)^{N_z\times N_z}$, $\mathbf{D}_1 \in [0,+\infty)^{N_z\times N_z}$, and $\mathbf{D}_2 \in [0,+\infty)^{N_z\times N_z}$ are three positive-valued linear factors leading to a multi-linear state operator $\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2$. Similarly, $\mathbf{H}_0 \in [0,+\infty)^{N_x\times N_z}$, $\mathbf{H}_1 \in [0,+\infty)^{N_z\times N_z}$, $\mathbf{H}_2 \in [0,+\infty)^{N_z\times N_z}$ are three positive-valued linear factors yielding the multi-linear observation model $\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2$. [1] The process noise $(\mathbf{v}_{1,k})_{1\leq k\leq K}$ is assumed to have a Gaussian distribution with zero-mean and symmetric definite positive covariance matrix $\mathbf{Q} \in \mathbb{R}^{N_z\times N_z}$. The observation noise $(\mathbf{v}_{2,k})_{1\leq k\leq K}$, is also assumed zero-mean Gaussian with symmetric definite positive covariance matrix $\mathbf{R} \in \mathbb{R}^{N_x\times N_x}$. We consider $\mathbf{z}_0 \sim \mathcal{N}(\bar{\mathbf{z}}_0, \mathbf{P}_0)$ as the initial state, with $\bar{\mathbf{z}}_0 \in \mathbb{R}$ and $\mathbf{P}_0 \in \mathbb{R}^{N_z\times N_z}$ defined as definite symmetric positive matrix. The model in Eq. (1) can be interpreted as a multi-linear Gaussian model involving a sequence of K hidden states represented by $(\mathbf{z}_k)_{1\leq k\leq K}$. As discussed earlier, classical inference approaches for SSM require specifying every model parameters. In the model above, this would mean setting the positive latent factor matrices matrices $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ involved in both state and observation models. In practical applications such as stock market analysis, these parameters are unknown and must be learnt from the observed data. The objective is thus to provide a point-wise estimate of the positive latent factor matrices $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ and a probabilistic estimate of sequence $(\mathbf{z}_k)_{1\leq k\leq K}$, given the observed sequence $(\mathbf{x}_k)_{1\leq k\leq K}$. This can be seen as solving jointly (i) two deep NMF problems, (ii) a filtering/smoothing problem.

B. Discussion on the model

We now discuss the main characteristics of the proposed DRDL method. The model is mathematically described in Eq. (1) and displayed in Fig. 1. The top equation describes the hidden state evolution, assuming Markovianity between two consecutive hidden states. The second equation links the hidden and observed states. A first interesting aspect, inherited from the SSM paradigm, is that two Gaussian noise terms are explicitly introduced in DRDL to cope with model uncertainty, which is in contrast with most deep learning models for time series processing (e.g., LSTM). A second novel feature of (1) lies in using deep NMF models instead of generic matrices (in the linear case) or functions (in the non-linear case), as it is usually the case in SSMs [29], [30], [31], taking advantage on the acknowledged representation power of deep NMF [52]. One important benefit of the proposed approach w.r.t. most existing methods in the literature is that we avoid Monte Carlo simulation or complex optimization procedure, which is known to suffer more severely the curse of dimensionality. In our method, each latent factor can be understood as representations in abstract spaces of the phenomena occurring between both pairs of variables. Third, in contrast with the typical usage of deep NMF in machine learning, relying on backpropagation for their model training [57], [58], DRDL model allows the construction of an handcrafted training strategy (see the next section), which benefits from a low computational cost, sound optimality guarantees (in terms of Bayesian estimator), and enables uncertainty quantification.

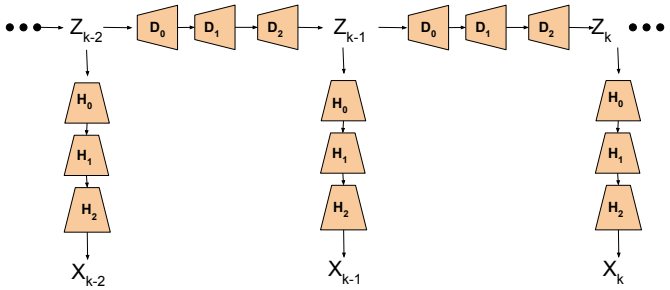


Fig. 1: Schematic Diagram for Deep Recurrent Dictionary Learning

C. DRDL inference algorithm

Using SSM models for time series processing (e.g., for a prediction task) amounts to solving the so-called smoothing/filtering problem, i.e., the probabilistic estimation of the hidden state $(\mathbf{z}_k)_{1\leq k\leq K}$. In our context, as the deep NMF factors $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ involved in the construction of the state transition and the observation transition models are most often unknown, we must also infer them from the observed data, jointly with the hidden states (through the aforementioned filtering/smoothing procedure). To do so, we propose an expectation-maximization (EM) approach (see [28, chap.12] and [26]). The EM method alternates iteratively between the probabilistic inference of the state $(\mathbf{z}_k)_{1\leq k\leq K}$, while the positive factors $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ are fixed (E-step), and the update of these factors, assuming fixed state (M-step). More precisely, the E-step consists in fixing the linear operators obtained in the previous M-step and applying the classical Kalman/RTS recursions, for obtaining the filtered/smooth distributions $p(\mathbf{z}_k|\mathbf{x}_{1:k})$ and

[1] Throughout the paper, we consider three-terms factorizations, for the sake of readibility. The 3-layers modeling and inference methodology has the great advantage of being generic enough to be straightforwardly extended to any number, greater or equals to one, of factors.

$p(\mathbf{z}_k, \mathbf{x}_{1:K})$, respectively. Then, the M-step updates the operators $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ by maximizing an upper bound of:

$$\varphi_K(\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2) = \log p(\mathbf{x}_{1:K}|\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2). \quad (2)$$

We explicit hereafter the construction of the $i+1$-th EM update, given estimates from the previous iteration i.

1) E-step: Kalman/RTS inference: At this step, we considered the factors $\mathbf{D}_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}$ to be fixed (either from the previous M-step or from the initialization at the first iteration), and the goal is the probabilistic estimation of the latent states. As we aforementioned, (1) is a multi-linear Gaussian model whose observation operator $\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2$, evolution/state operator $\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2$, and hidden state $(\mathbf{z}_k)_{1\leq k\leq K}$ must be estimated. For each $k \in \{1, \ldots, K\}$, the probabilistic estimate of the latter conditioned to all available data up to time k, is provided by the Kalman filter through the following Gaussian filtering distribution:

$$p(\mathbf{z}_k|\mathbf{x}_{1:k}) = \mathcal{N}(\mathbf{z}_k; \bar{\mathbf{z}}_k, \mathbf{P}_k). \quad (3)$$

For every k, the mean $\bar{\mathbf{z}}_k$ and the covariance $\mathbf{P}_k$ are given by the Kalman iterations:
For $k = 1, \ldots, K$:
Predict state:

$$\begin{cases} \mathbf{z}_{k|k-1} &= \mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\bar{\mathbf{z}}_{k-1}, \\ \mathbf{P}_{k|k-1} &= \mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\mathbf{P}_{k-1}(\mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]})^\top + \mathbf{Q}. \end{cases} \quad (4)$$

Update state:

$$\begin{cases} \mathbf{y}_k &= \mathbf{x}_k - \mathbf{H}_0^{[i]}\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]}\mathbf{z}_{k|k-1}, \\ \mathbf{S}_k &= \mathbf{H}_0^{[i]}\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]}\mathbf{P}_{k|k-1}(\mathbf{H}_0^{[i]}\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]})^\top + \mathbf{R}, \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}(\mathbf{H}_0^{[i]}\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]})^\top\mathbf{S}_k^{-1}, \\ \bar{\mathbf{z}}_k &= \mathbf{z}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k, \\ \mathbf{P}_k &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top. \end{cases} \quad (5)$$

Hereabove, $\mathbf{y}_k$ represents the measurement pre-fit residual, $\mathbf{S}_k$ represents the pre-fit covariance, $\mathbf{K}_k$ represents Kalman gain, $\bar{\mathbf{z}}_k$ represents the updated (a posteriori) state estimate, $\mathbf{P}_k$ represents the updated (a posteriori) covariance estimate. The backward recursion from the RTS smoother allow to build the smoothing distribution $p(\mathbf{z}_k|\mathbf{x}_{1:K})$.
For $k = K, \ldots, 1$
Backward Recursion:

$$\begin{cases} \mathbf{z}_{k+1}^- &= \mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\bar{\mathbf{z}}_k, \\ \mathbf{P}_{k+1}^- &= \mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\mathbf{P}_k(\mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]})^\top + \mathbf{Q}, \\ \mathbf{G}_k &= \mathbf{P}_k(\mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]})^\top[\mathbf{P}_{k+1}^-]^{-1}, \\ \mathbf{z}_k^s &= \bar{\mathbf{z}}_k + \mathbf{G}_k[\mathbf{z}_{k+1}^s - \mathbf{z}_{k+1}^-], \\ \mathbf{P}_k^s &= \mathbf{P}_k - \mathbf{G}_k[\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-]\mathbf{G}_k^\top. \end{cases} \quad (6)$$

Consequently, for every time step $k \in \{1, \ldots, K\}$, the RTS smoother provides:

$$p(\mathbf{z}_k|\mathbf{x}_{1:K}) = \mathcal{N}(\mathbf{z}_k; \mathbf{z}_k^s, \mathbf{P}_k^s). \quad (7)$$

2) M-step: Evolution operators update: The M-step performs an optimization step to increase the likelihood of the positive latent factors $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$, given the smoothing distribution obtained in the E-step. It proceeds by building the upper-bound:

$$\varphi_k(\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2) \geq \mathcal{Q}(\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2; \mathbf{\Theta}^{[i]}). \quad (8)$$

Hereabove, $\mathbf{\Theta}^{[i]} = \{\mathbf{\Sigma}^{[i]}, \mathbf{\Phi}^{[i]}, \mathbf{B}^{[i]}, \mathbf{C}^{[i]}, \mathbf{\Delta}^{[i]}\}$ gathers five quantities defined from the outputs of the E-step described in Sec. II-C1):

$$\begin{aligned} &\mathcal{Q}(\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2; \mathbf{\Theta}^{[i]}) = \\ &-\frac{K}{2}\mathrm{tr}\Big(\mathbf{Q}^{-1}\mathbf{\Sigma}^{[i]} - \mathbf{C}^{[i]}(\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2)^\top - \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2(\mathbf{C}^{[i]})^\top \\ &\qquad + \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{\Phi}^{[i]}(\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2)^\top\Big) \\ &-\frac{K}{2}\mathrm{tr}\Big(\mathbf{R}^{-1}\mathbf{\Delta}^{[i]} - \mathbf{B}^{[i]}(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top - \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2(\mathbf{B}^{[i]})^\top \\ &\qquad + \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{\Sigma}^{[i]}(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top\Big), \end{aligned} \quad (9)$$

with:

$$\begin{aligned} \mathbf{\Sigma}^{[i]} &= \frac{1}{K}\sum_{k=1}^K \mathbf{P}_k^s + \mathbf{z}_k^s(\mathbf{z}_k^s)^\top, \\ \mathbf{\Phi}^{[i]} &= \frac{1}{K}\sum_{k=1}^K \mathbf{P}_{k-1}^s + \mathbf{z}_{k-1}^s(\mathbf{z}_{k-1}^s)^\top, \\ \mathbf{B}^{[i]} &= \frac{1}{K}\sum_{k=1}^K \mathbf{x}_k(\mathbf{z}_k^s)^\top, \\ \mathbf{C}^{[i]} &= \frac{1}{K}\sum_{k=1}^K \mathbf{P}_k^s\mathbf{G}_{k-1}^\top + \mathbf{z}_k^s(\mathbf{z}_{k-1}^s)^\top, \\ \mathbf{\Delta}^{[i]} &= \frac{1}{K}\sum_{k=1}^K \mathbf{x}_k\mathbf{x}_k^\top. \end{aligned} \quad (10)$$

The updates $\{\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2^{[i+1]}, \mathbf{H}_0^{[i+1]}, \mathbf{H}_1^{[i+1]}, \mathbf{H}_2^{[i+1]}\}$ given the knowledge of $\mathbf{\Theta}^{[i]}$, amounts to maximizing $\mathcal{Q}(\cdot; \mathbf{\Theta}^{[i]})$ under positivity constraints on the factors. In contrast with the linear unconstrained model case studied in [28, Chapter 12], the maximization problem here does not have a closed-form solution. It is highly non-convex due to the multi-linearity of our model. Luckily, it happens to be convex with respect to each of the factors. We thus propose to resort to the following alternating maximization step:

$$\begin{aligned} \mathbf{D}_0^{[i+1]} &= \operatorname*{argmax}_{\mathbf{D}_0\geq 0} \mathcal{Q}(\mathbf{D}_0, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\ \mathbf{D}_1^{[i+1]} &= \operatorname*{argmax}_{\mathbf{D}_1\geq 0} \mathcal{Q}(\mathbf{D}_0^{[i+1]}, \mathbf{D}_1, \mathbf{D}_2^{[i]}, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\ \mathbf{D}_2^{[i+1]} &= \operatorname*{argmax}_{\mathbf{D}_2\geq 0} \mathcal{Q}(\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\ \mathbf{H}_0^{[i+1]} &= \operatorname*{argmax}_{\mathbf{H}_0\geq 0} \mathcal{Q}(\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2^{[i+1]}, \mathbf{H}_0, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\ \mathbf{H}_1^{[i+1]} &= \operatorname*{argmax}_{\mathbf{H}_1\geq 0} \mathcal{Q}(\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2^{[i+1]}, \mathbf{H}_0^{[i+1]}, \mathbf{H}_1, \mathbf{H}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\ \mathbf{H}_2^{[i+1]} &= \operatorname*{argmax}_{\mathbf{H}_2\geq 0} \mathcal{Q}(\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2^{[i+1]}, \mathbf{H}_0^{[i+1]}, \mathbf{H}_1^{[i+1]}, \mathbf{H}_2; \mathbf{\Theta}^{[i]}) \end{aligned}$$

This approach ensures by construction the following inequality:

$$\mathcal{Q}(\mathbf{D}_0^{[i+1]},\mathbf{D}_1^{[i+1]},\mathbf{D}_2^{[i+1]},\mathbf{H}_0^{[i+1]},\mathbf{H}_1^{[i+1]},\mathbf{H}_2^{[i+1]};\boldsymbol{\Theta}^{[i]}) \geq \mathcal{Q}(\mathbf{D}_0^{[i]},\mathbf{D}_1^{[i]},\mathbf{D}_2^{[i]},\mathbf{H}_0^{[i]},\mathbf{H}_1^{[i]},\mathbf{H}_2^{[i]};\boldsymbol{\Theta}^{[i]}), \quad (11)$$

which is key to guarantee the convergence properties for the EM iteration. Indeed, the proposed updates yield an increase of the lower bound of the marginal likelihood, so as a consequence, an increase of the marginal log-likelihood itself. The overall procedure is thus guaranteed to yield a monotonic increase of the marginal log-likelihood function φ_K and classical results about majorization-minimization methods allow to ensure convergence guarantees to a stationary point of φ_K [64]. The six sub-problems are quadratic programming (convex) problems and can be solved through several available solvers. We decided to use the simple and fast alternating least squares approach [53], reminiscent from the literature of deep nonnegative matrix factorization [57], and the deep ReLu neural networks models [60], both showing a satisfactory behavior in preliminary experiments. We start by computing each subproblem solution ignoring the positivity constraints, and then capped the negative entries of the obtained factors. This yields the following analytic updates:

$$\begin{aligned}
\mathbf{D}_0^{[i+1]} &= \text{ReLu}\left(\mathbf{C}^{[i]}(\mathbf{D}_2^{[i]})^\top(\mathbf{D}_1^{[i]})^\top(\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\boldsymbol{\Phi}^{[i]}(\mathbf{D}_2^{[i]})^\top(\mathbf{D}_1^{[i]})^\top)^\dagger\right),\\
\mathbf{D}_1^{[i+1]} &= \text{ReLu}\left(((\mathbf{D}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{D}_0^{[i+1]})^\dagger(\mathbf{D}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{C}^{[i]}(\mathbf{D}_2^{[i]})^\top\right.\\
&\quad\left.\times(\mathbf{D}_2^{[i]}\boldsymbol{\Phi}^{[i]}(\mathbf{D}_2^{[i]})^\top)^\dagger\right),\\
\mathbf{D}_2^{[i+1]} &= \text{ReLu}\left(((\mathbf{D}_1^{[i+1]})^\top(\mathbf{D}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{D}_0^{[i+1]}\mathbf{D}_1^{[i+1]})^\dagger(\mathbf{D}_1^{[i+1]})^\top\right.\\
&\quad\left.\times(\mathbf{D}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{C}^{[i]}(\boldsymbol{\Phi}^{[i]})^{-1}\right),\\
\mathbf{H}_0^{[i+1]} &= \text{ReLu}\left(\mathbf{B}^{[i]}(\mathbf{H}_2^{[i]})^\top(\mathbf{H}_1^{[i]})^\top(\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]}\boldsymbol{\Sigma}^{[i]}(\mathbf{H}_2^{[i]})^\top(\mathbf{H}_1^{[i]})^\top)^\dagger\right),\\
\mathbf{H}_1^{[i+1]} &= \text{ReLu}\left(((\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{H}_0^{[i+1]})^\dagger(\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{B}^{[i]}(\mathbf{H}_2^{[i]})^\top\right.\\
&\quad\left.\times(\mathbf{H}_2^{[i]}\boldsymbol{\Sigma}^{[i]}(\mathbf{H}_2^{[i]})^\top)^\dagger\right),\\
\mathbf{H}_2^{[i+1]} &= \text{ReLu}\left(((\mathbf{H}_1^{[i+1]})^\top(\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{H}_0^{[i+1]}\mathbf{H}_1^{[i+1]})^\dagger\right.\\
&\quad\left.\times(\mathbf{H}_1^{[i+1]})^\top(\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{B}^{[i]}(\boldsymbol{\Sigma}^{[i]})^{-1}\right). \qquad (12)
\end{aligned}$$

Hereabove, $(\cdot)^\dagger$ denotes the pseudo-inverse operator. Moreover, ReLu $(\cdot)$ states for the rectified linear unit function, that projects each entry of its input to the positive orthant.

D. The DRDL algorithm summarized

We summarize in Alg. 1 the DRDL algorithm, for the probabilistic inference of the sequence of hidden state $(\mathbf{z}_k)_{1\leq k\leq K}$, jointly with the point-wise estimation of the latent factors $\{\mathbf{D}_0,\mathbf{D}_1,\mathbf{D}_2,\mathbf{H}_0,\mathbf{H}_1,\mathbf{H}_2\}$, assuming the data follows the DRDL model (1). In practice, DRDL algorithm is ran for a maximum number of iterations $i_{\max}$, set so as to reach stabilisation of the latent factors.

Algorithm* 1.** ***DRDL (3 layers) inference algorithm.

Inputs. *Prior parameters* $(\overline{\mathbf{z}}_0,\mathbf{P}_0)$ *; model noise covariance matrices* $\mathbf{Q}$, $\mathbf{R}$ *; set of observations* $\{\mathbf{x}_k\}_{1\leq k\leq K}$.

Initialization. *Set positive latent factors* $\{\mathbf{D}_0^{(0)},\mathbf{D}_1^{(0)},\mathbf{D}_2^{(0)},\mathbf{H}_0^{(0)},\mathbf{H}_1^{(0)},\mathbf{H}_2^{(0)}\}$.

Recursive step. *For* $i=0,1,\ldots,i_{\max}$*:*

(E step) Run the Kalman filter (4)-(5) *and RTS smoother* (6) *using latent factors* $\{\mathbf{D}_0^{(i)},\mathbf{D}_1^{(i)},\mathbf{D}_2^{(i)},\mathbf{H}_0^{(i)},\mathbf{H}_1^{(i)},\mathbf{H}_2^{(i)}\}$.
Calculate $(\boldsymbol{\Sigma}^{(i)},\boldsymbol{\Phi}^{(i)},\mathbf{B}^{(i)},\mathbf{C}^{(i)},\boldsymbol{\Delta}^{(i)})$ *using* (10).

(M step) Compute $\{\mathbf{D}_0^{(i+1)},\mathbf{D}_1^{(i+1)},\mathbf{D}_2^{(i+1)},\mathbf{H}_0^{(i+1)},\mathbf{H}_1^{(i+1)},\mathbf{H}_2^{(i+1)}\}$ *using* (12).

Output. *State filtering/smoothing pdfs* (3) *and* (7) *along with point-wise estimates of the latent factor from* (12).

III. Application to Stock Trading

We now particularize the DRDL inference algorithm to the stock trading applications. In particular, we address the forecasting/trading tasks given a set of K daily (i.e., k is a day index) observations of stock market data.

A. Online implementation

First, in order to better cope with high volatility of stock market quantities and allow immediate feedback to the users for on-the-fly trading, we propose here an online implementation of our DRDL approach. We make use of sliding windows of size of $\tau\in\{1,\ldots,K\}$ time steps. The model parameters are inferred for every $k\in\{0,\ldots,K-\tau\}$ using the last τ data points observed in the window, i.e. $\mathcal{X}_k=\{\mathbf{x}_j\}_{j=k+1}^{k+\tau}$, then followed by the EM approach described in detail above. The sliding window approach leverages two advantages. First, it helps in faster processing of the sequence as one can choose a small number τ of data points in the window. Second, it might provide better modeling, since the constant linear factors assumption is likely to better model the time series if τ is small. However, a too small τ might also degrades the inference capabilities. Hence, it is essential to find a tradeoff in finding an optimal τ, as we will show in our experiments. When implementing the online strategy, a warm start approach is employed for the Kalman filter initialization. The observation/state factors are set to their most recent values, and the mean/covariance of the state for processing $\mathcal{X}_{k+1}$ are initialized using the results of the processing of $\mathcal{X}_k$. Let us note that, when we set $\tau=K$, we retrieve the offline version of the algorithm, where the EM inference tool is ran only once.

B. Modeling and post-processing for stock market analysis tasks

Stock market data processing typically amounts to solving two distinct applicative problems, namely daily stock price forecasting and stock trading decision (among 3 options: buy/hold/sell) estimation. We hereafter explain how to post-process DRDL results to tackle both above-stated problems.

1) Stock forecasting: Let us first specify the observation model in stock forecasting. For a given window size $\tau>0$, for each $k\in\{0,\ldots,K-\tau\}$, we observe $(\mathbf{x}_j)_{k+1\leq j\leq k+\tau}\in\mathbb{R}^{15}$,

gathering 14 technical indicators [2] as well as the adjusted close price. Running our DRDL on the considered window yields the following mean estimate of the 15 quantities for the next time step indexed as $k+\tau+1$:

$$\hat{\mathbf{x}}_{k+\tau+1} = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_{k+\tau|k+\tau-1}, \tag{13}$$

with the covariance matrix

$$\mathbf{S}_{k+\tau+1} = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2(\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{P}_{k+\tau}(\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2)^\top + \mathbf{Q}) + \mathbf{R}. \tag{14}$$

Hereabove, $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ are the factors estimated during the M-step of our EM-based inference method, and $(\mathbf{z}_{k+\tau|k+\tau-1}, \mathbf{P}_{k+\tau})$ are byproducts of the Kalman prediction step (4)-(5), computed during the E-step of the EM. The proposed methodology aims at predicting the entire 15-dimensional vector. However, stock forecasting is typically focused on the prediction of a single quantity such as the adjusted close price.

2) Stock trading: In stock trading, a different set of inputs are passed to the model. For each window index $k \in \{0, \dots, K-\tau\}$, the observed data points $(\mathbf{x}_j)_{k+1\leq j\leq k+\tau} \in \mathbb{R}^{17}$, $x_j[i]$, for $i \in \{1, \dots, 14\}$, are the same 14 technical indicators as in stock forecating. Additionally, $[x_j[15], x_j[16], x_j[17]] \in \{0,1\}^3$ gathers the decisions "hold", "buy", or "sell", which are calculated for each stocks for every day so as to maximize the annualized returns. The labels are further turned into soft hot encoded vectors as explained in [32, Sec. 3.3.2]. The mean and covariance of these 17 quantities can be estimated for next day following (13) and (14). We then define our class label for next time step as

$$\ell_{k+\tau+1} = \text{argmax}_{i\in\{1,2,3\}}\, \widehat{x}_{k+\tau+1}[i+14]. \tag{15}$$

C. Probabilistic assessment of stock trading decision

We now describe the procedure to assess the uncertainty quantification associated to the DRDL predictions. Let $k \in \{0, \dots, K-\tau\}$ be the window index on which Algorithm 1 has been run. The probabilistic estimation of the quantities of interest for the next time step (i.e., one-day ahead prediction) $\mathbf{x}_{k+\tau+1}$ conditioned to the data observed in the window $\mathbf{x}_{k:k+\tau}$, reads as a multivariate Gaussian distribution

$$p(\mathbf{x}_{k+\tau+1}|\mathbf{x}_{k:k+\tau}) = \mathcal{N}\left(\mathbf{x}_{k+\tau+1}; \hat{\mathbf{x}}_{k+\tau+1}, \mathbf{S}_{k+\tau+1}\right), \tag{16}$$

with mean and covariance $(\hat{\mathbf{x}}_{k+\tau+1}, \mathbf{S}_{k+\tau+1})$, given by (13) and (14), respectively. Equation (16) assigns a probability score to any decision (e.g., trading) based on the prediction output of the DRDL method. Let us focus on the particular example of assessing the uncertainty of the stock trading decision at time index $k+\tau+1$, given observations at indexes $j \in \{k+1, \dots, k+\tau\}$. The trading decision relies on the discrete maximization step (15). Let us express the probability mass function (pmf) of this decision, from the gaussian predictive probability density function (pdf) of the observed data points in Eq. (16). The pmf can here be summarized as $\mathbf{p}_{k+\tau+1} \in [0,1]^3$ where each $p_{k+\tau+1}[i]$, $i \in \{1,2,3\}$ is a probability, and $\sum_{i=1}^3 p_{k+\tau+1}[i] = 1$. Each $p_{k+\tau+1}[i]$ represents the probability inferred by DRDL that the true value $x_{k+\tau+1}[i+14]$ is greater than $x_{k+\tau+1}[j+14]$, for $j = \{1,2,3\} \setminus i$. According to Eqs. (16) and (15), $\mathbf{p}_{k+\tau+1}$ can be obtained through

$$(\forall i \in \{1,2,3\}) \quad p_{k+\tau+1}[i] = \int_{\mathcal{Y}_i} \mathcal{N}(\mathbf{y}; \hat{\mathbf{x}}_{k+\tau+1}[15:17], \mathbf{S}_{k+\tau+1}[15:17, 15:17])d\mathbf{y}, \tag{17}$$

with

$$\mathcal{Y}_i = \left\{\mathbf{y} \in \mathbb{R}^3 \mid \mathbf{y}[i] \geq \mathbf{y}[j], j = \{1,2,3\} \setminus i\right\}. \tag{18}$$

Due to the intricate form of the constrained set in (18), the integral in (17) is intractable. It can be easily approximated with high precision by direct simulation. In practice, we sampled 10^4 three-dimensional sample from a normal standard distribution. The samples can be re-used for all time steps using coloring and shifting according to the covariance and mean, respectively. Thanks to this procedure, we can infer $\mathbf{p}_{k+\tau+1}$ for every k, and then assess the next day stock trading outcome by using the standard cross-entropy loss:

$$\text{log-loss} = \frac{1}{K-\tau+1}\sum_{k=0}^{K-\tau}\sum_{i=1}^{3} -(L_{k+\tau+1}[i]\log(p_{k+\tau+1}[i])), \tag{19}$$

where the true labels are denoted $\mathbf{L}_{k+\tau+1} \in \{0,1\}^3$ for each time $k+\tau+1$ (hereagain, we use soft hot encoding representation).

D. Summarized pipeline

We provide in Alg. 2 the summary of our proposed pipeline for applying DRDL, in Algorithm 1, in the context of stock forecasting (steps a-b with $N_x = 15$) and trading (steps a-b-c-d with $N_x = 17$).

Algorithm 2. DRDL (3 layers) method for stock forecasting and trading.

Inputs. *Prior parameters $(\overline{\mathbf{z}}_0, \mathbf{P}_0)$; model noise covariance matrices $\mathbf{Q}$, $\mathbf{R}$; set of observations $\{\mathbf{x}_k\}_{1\leq k\leq K}$; windows size τ.*
Initialization. *Set positive latent factors $\{\mathbf{D}_0^{(0)}, \mathbf{D}_1^{(0)}, \mathbf{D}_2^{(0)}, \mathbf{H}_0^{(0)}, \mathbf{H}_1^{(0)}, \mathbf{H}_2^{(0)}\}$.*
Window processing. *For $k = 0, 1, \dots, K-\tau$:*

a. Run DRDL algorithm 1 on sequence $(\mathbf{x}_j)_{k+1\leq k\leq k+\tau}$, initialized with estimates from $k-1$th window (warm start).
b. Calculate one-step ahead predicted mean $\widehat{\mathbf{x}}_{k+\tau+1}$ and its covariance $\mathbf{S}_{k+\tau+1}$ using (13)-(14).
c. Compute one-step ahead predicted label $\ell_{k+\tau+1}$ using (15).
d. Compute $\mathbf{p}_{k+\tau+1}$ using (17)-(18).

Output. *Forecasting/trading predictions and log-loss value* (19).

[2] We retained the relative strength index (RSI), the William percentage range, the absolute price oscillator (APO), the commodity channel index, the Chande momentum oscillator (CMO), the directional movement Indicator (DMI), the ultimator oscillator, the WMA, the exponential moving average (EMA), the Simple Moving Average (SMA), the triple EMA, the moving average convergence (MAC), the percentage price oscillator, the rate of change (ROC). Detailed definitions can be found in https://www.investopedia.com/terms/t/technicalindicator.asp

IV. Experimental Results

A. Dataset

The finance dataset used for experiments is curated from Yahoo finance repository.[3] We curated data for 180 stocks which comprises stocks from USA, UK, India and China. The data is prepared by scrapping daily adjusted close prices, open price, volume, high price, low price for a span of twenty years (i.e., 01/01/1998 to 01/10/2019) using yahoo finance API for Python. Having stocks from different market cap is always advisable by the traders, as it gives them breadth while investing in advanced as well as emerging markets [65]. The diversification also allows to assess the model robustness to various trends [66]. From the knowledge of the close prices, we build two observation sequences associated to the resolution of two specific problems, namely stock forecasting and stock trading, as described in Sec. III-B. The data is scaled by normalising the 14 technical indicator values. For both problems, we will compare DRDL and several state-of-the-art methods arising from signal processing and machine learning literature. In all experiments, each of the 180 observed time series is split into two parts, namely a train phase made of the first recorded 2546 days, and a test phase made of the next 2882 days. The train phase is used to learn the models parameters (for instance, the linear factors involved in DRDL), while the test phase is used to evaluate the performance of the learnt models, their parameters being fixed. More details about DRDL and the retained benchmark methods setting are provided in the next subsection. Figure 2 displays the evolution of 4 of the 14 technical indicators used as input of the inference tools, during the test phase. One can notice the high volatility in the observed data.

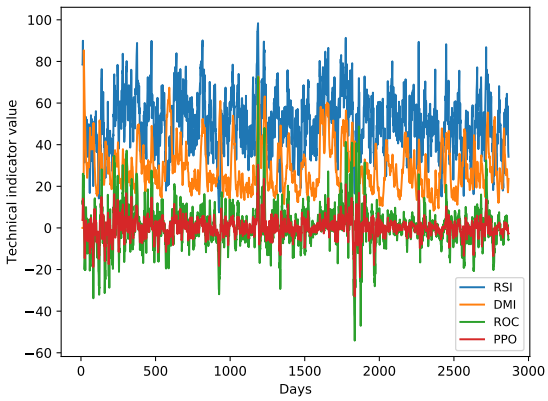


Fig. 2: Evolution of four (among 14) observed technical indicators during the test phase.

B. Practical Settings

1) DRDL settings: As described in Sec. III-B, DRDL can be specified to tackle both stock forecasting problem, in which case $N_x = 15$, and stock trading problem where $N_x = 17$. Three variants of DRDL will be compared, depending on the number of linear factors (i.e., layers) in the multi-linear model. More specifically, we will distinguish in our experiments:

DRDL (1 layer): $\mathbf{D}_0 = \mathbf{D}_1 = \mathbf{D}_2 = \mathbf{Id}$ and $\mathbf{H}_1 = \mathbf{H}_2 = \mathbf{Id}$ fixed and $\{\mathbf{H}_0\}$ is estimated;

DRDL (2 layers): $\mathbf{D}_0 = \mathbf{D}_1 = \mathbf{D}_2 = \mathbf{Id}$ and $\mathbf{H}_2 = \mathbf{Id}$ fixed and $\{\mathbf{H}_0, \mathbf{H}_1\}$ are estimated;

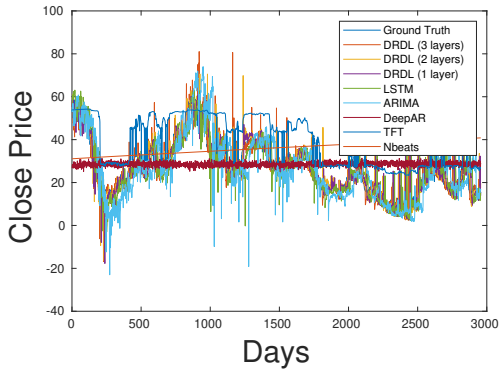
DRDL (3 layers): $\mathbf{D}_0 = \mathbf{D}_1 = \mathbf{D}_2 = \mathbf{Id}$ and $\{\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2\}$ are estimated.

Note that, ignoring the positivity constraint, DRDL (1 layer) would identify with our previously published method RDL [32]. We implement the sliding window approach described in Sec. III-A, for various choices of τ described hereafter. In all experiments, the initial value are set as ; $\bar{\mathbf{z}}_0$ is an all zero vector; $\mathbf{P}_0$=10^{-7}Id, $\mathbf{Q}$=10^{-2}Id and $\mathbf{R}$=10^{-2}Id, where Id states for the identity matrix. Moreover, we set the dimension of the state as $N_z = 14$, which also corresponds to the number of measured technical indicators, we observed better performance of the model. The entries of the linear factors to estimate are initialized at time 0 using independent realizations of a uniform distribution on $[0, 10^{-1}]$. As mentioned in Sec. III-A, to initialize the next processed windows, we used warm start strategy. The estimation of the linear factors (i.e. M-step of the EM method) is only conducted during the training phase. A maximum of 50 iterations of the EM loop are used in Alg. 1, which was observed sufficient to reach stability of the estimated factors. During the test phase, the linear latent factors are fixed, and only the Kalman/RTS inference is ran (i.e., we inhibit M-step in Alg. 1). The scores shown are arithmetic means of 10 random trials, and are computed only during the test phase.

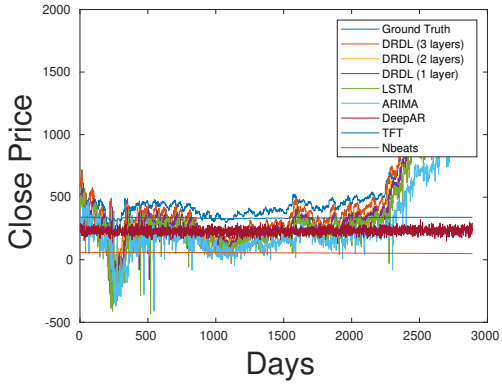
2) Compared methods: The proposed DRDL approach is analyzed by comparing with state-of-the-art-methods deep learning namely Multi-filter neural network [42], Long short term memory [51], 2-D deep CNN (CNN-TA) [41] and ARIMA [6]. We select the state-of-the-art methods for each task for a fair comparison. For stock forecasting, the comparison is done with ARIMA and LSTM. The ARIMA parameter value are set to $(p, d, q) = (5, 1, 5)$. The LSTM is customized from its original version to carry out regression tasks by replacing the softmax output layer with an affine layer. The Adam optimizer is used with learning rate 10^{-4} and 200 epochs. We used a mini-batch strategy where batch-size is fixed to 16 to reduce the objective function's mean square error (MSE). The evaluation of the methods is done using metrics like mean absolute error (MAE), root means square error (RMSE), SMAPE (Symmetric mean absolute percentage error), and Pearson correlation factor.

For the stock trading task, the comparison is made with CNN-TA, Multi-filter neural network, and LSTM implemented with their original architecture and parameter values set to the specified ones in the respective paper [41], [42] and [51]. The evaluation of the methods is done using classification metrics such as the F1-score, recall, precision for each class. We also performed trading simulation experiments in terms of annualized returns. We additionally present the log-loss values provided by DRDL (see Sec. III-C) to illustrate how

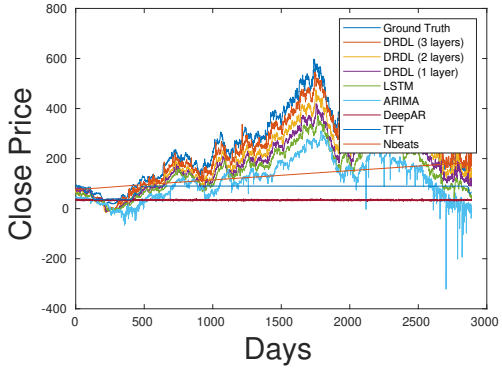
[3]https://yahoo.finance.com



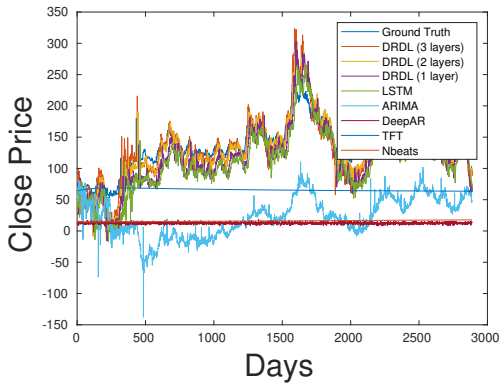
(a) ANF



(b) RELIANCE

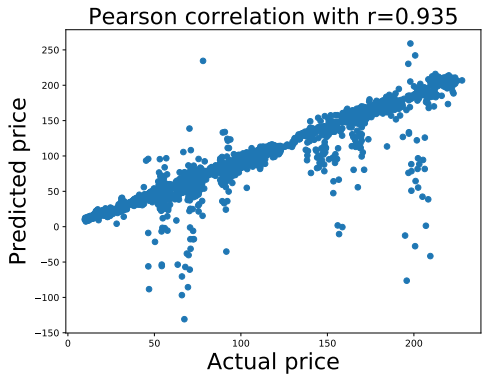


(c) TATAMOTORS

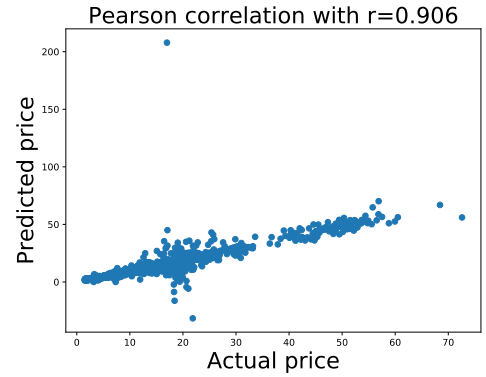


(d) ONGC

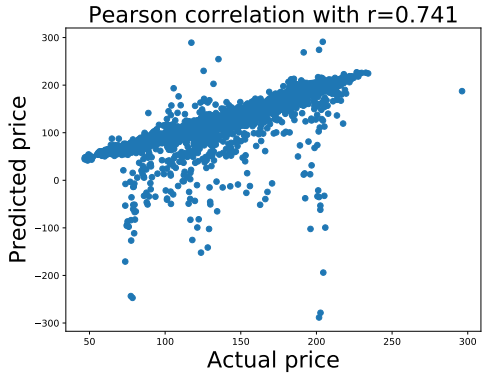
Fig. 3: Ground truth and inferred adjusted close price on the test phase for four different stocks, using DRDL with 1 to 3 layers, LSTM or ARIMA.



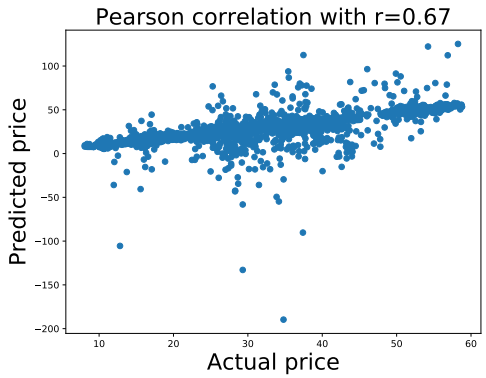
(a) AAPL



(b) ALOKTEXT.BO



(c) BHEL.BO



(d) ANF

Fig. 4: Pearson correlation graph between ground truth adjusted close price and predicted one with DRDL (3 layers), during test phase, for four different stocks.

the probabilistic assessment can be used to let the researchers analyze market sentiments and diversify a balanced portfolio.

3) Hardware and Software descriptions: We curated the data using the python API. The data curation and experimental results for the DRDL model are computed using Python 3.6 code. The implementation is done using the potential python libraries like NumPy, scikit-learn and pandas. In contrast, CNN-TA is implemented in its original version using the keras. MFNN and LSTM are implemented with PyTorch. The technical indicators are evaluated using the libraries Ta-lib[4] and Ta4j[5]. Provided computational times correspond to codes running on Xeon E3-1225V5 clocked at 3.3GHz, with a 4Gb GPU (GeForce GT 730), 16GB RAM, 200GB HDD and Ubuntu OS.

C. Numerical results for stock forecasting problem

1) Influence of window size: The choice of window size is an essential aspect as it can enhance and limit the methodology's potential. To understand the model behavior, we present Table I which provides detailed information on the performance of the model on varying window sizes. The table offers an analysis of various metrics like Pearson correlation (r), RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and SMAPE (Symmetric mean absolute percentage error) for different window sizes τ. The model's performance improves as the window size increases till a stabilization point. We can see that a balanced choice is $\tau = 650$ to reach stabilized performance on this particular task and dataset. We further use this value in upcoming experiments.

Window size τ	r	RMSE ↓	MAE (%) ↓	SMAPE (%) ↓
250	0.45	29.43	0.47	31.8
300	0.49	27.81	0.23	28.6
350	0.53	21.61	0.29	25.5
500	0.69	13.79	0.13	23.4
650	**0.71**	**13.35**	**0.11**	**18.4**
700	0.72	13.62	0.10	18.5

TABLE I: Results of DRDL (3 layers) on stock forecasting problem for different window size. Scores averaged on test phase, on all the 180 stocks.

Model	r	RMSE↓	MAE(%) ↓	SMAPE(%)↓
ARIMA	0.13	78.6	1.23	65.5
LSTM	0.24	297.5	6.12	47
DeepAR	0.40	73.89	0.43	58.13
Nbeats	0.38	95.52	0.57	63.75
TFT	0.52	35.79	0.35	38.25
DRDL (1 layer)	0.65	23.24	0.19	35.3
DRDL (2 layers)	0.69	14.2	0.15	23.2
DRDL (3 layers)	**0.71**	**13.35**	**0.11**	**18.4**

TABLE II: Comparative analysis of DRDL against state-of-the-art methods for stock forecasting problem: Pearson correlation score (r), MAE, RMSE and SMAPE scores on the estimation of next time step adjusted close price, averaged over the data in the test set and the stocks.

.

[4]http://ta-lib.org
[5]http://www.ta4j.org

Method	Train Time cost (h.)	Test Time cost (min.)
DRDL (3 layers)	2.37h	20 min
DRDL (2 layers)	2.12h	18.4 min
DRDL (1 layer)	1.78h	15.8 min
ARIMA	2.01h	36 min
LSTM	8 days	45 min
DeepAR	2.45h	20 min
TFT	2.25h	27 min
Nbeats	3.12h	25 min

TABLE III: Averaged time over 10 random runs for processing the dataset (train(hrs) and test(min)), for DRDL and its competitors.

2) Comparison with benchmark models: To understand better, we present table II which provides comprehensive analysis on performance estimation on the stock forecasting problem using DRDL, LSTM, ARIMA, DeepAR [44], Nbeats [67], and TFT [68]. Table II presents comparison in terms of Pearson correlation factor r, RMSE, MAE and SMAPE. We can see that DRDL (3 layers) architecture outperforms DRDL (2 layers), DRDL (1 layer) as well as the other benchmarks models. We also notice that the average performance of TFT is comparable to DRDL (1 layer) architecture. Fig 4 displays the Pearson correlation analysis between ground truth daily adjusted close price time series and predicted ones along test phase, using DRDL (3 layers) for four representive stock cases.

Table III presents the computational time for forecasting the next day closing price for our dataset. We distinguish the time required to train the methods (on the first ten years) and to test them (on the next ten years) using the walk-forward method described in [32, Section 4.2.1]. We observed the highest computational time with the LSTM approach. The other methods have rather similar computational time, DRDL (1 layer) being the fastest. The computational time of DRDL (3 layers), reaching the best performance metrics, stays reasonable, and is comparable with the one of DeepAR and TFT. Here, we must recall that, in contrast with most of its competitors (except ARIMA), our implementation of DRDL method (for both train/test phases) does not exploit GPU facilities such as PyTorch. Complexity reductions could certainly occur if this was the case.

The stock forecasting results are presented in Fig. 3. The comparison is carried out between the proposed DRDL for different layer number, LSTM, ARIMA, DeepAR, TFT, Nbeats method. We observed that in some cases (c-d), LSTM approach failed to reach satisfying results which might be due to vanishing gradient issues. In cases (a-b), ARIMA performs quite good when compared to its performance in other cases (c-d). In contrast, DRDL (3 layers) reaches stable and satisfactory outcomes. DRDL (2 layers) outperforms DRDL (1 layer) and both benchmark methods but remains lower quality than its 3-layers variant.

D. Numerical results for the stock trading problem

1) Influence of the window size: The challenging task with the DRDL approach is to preserve a balance between the computational time and optimal predictions. We experimented

Method	F1 Score			Precision			Recall			Training time (in hrs)	Testing Time (in min)
	Hold	Sell	Buy	Hold	Sell	Buy	Hold	Sell	Buy		
DRDL (3 layers)	**0.61**	**0.30**	**0.29**	**0.85**	**0.26**	**0.29**	**0.51**	**0.54**	**0.54**	4.12	10.17
DRDL (2 layers)	0.61	0.32	0.34	0.83	0.23	0.26	0.48	0.51	0.53	3.56	12.50
DRDL (1 layer)	0.59	0.19	0.23	0.88	0.15	0.12	0.45	0.52	0.51	2.00	11.56
MFNN	0.58	0.11	0.06	0.79	0.11	0.04	0.47	0.37	0.16	5.34	14.23
LSTM	0.86	0.05	0.05	0.84	0.07	0.06	0.89	0.05	0.05	12.53	13.50
CNN-TA	0.85	0.08	0.09	0.84	0.11	0.09	0.85	0.07	0.10	4.57	14.36

TABLE IV: Comparison of classification scores of different methods on the stock trading problem. All scores are averaged over 180 stocks and over the days of the test phase.

Window size (τ)	F1 Score			Precision			Recall		
	Hold	Sell	Buy	Hold	Sell	Buy	Hold	Sell	Buy
250	0.61	0.23	0.19	0.91	0.15	0.12	0.46	0.51	0.50
300	0.61	0.23	0.19	0.91	0.15	0.12	0.46	0.50	0.51
350	0.61	0.26	0.27	0.89	0.18	0.19	0.47	0.53	0.52
500	0.61	0.26	0.27	0.89	0.18	0.19	0.47	0.52	0.53
650	**0.61**	**0.30**	**0.29**	**0.85**	**0.26**	**0.29**	**0.51**	**0.54**	**0.54**
700	0.61	0.30	0.29	0.87	0.21	0.20	0.48	0.53	0.54

TABLE V: Classification scores of DRDL (3 layers) for varying window size.

Window size (τ)	F1 Score			Precision			Recall		
	Hold	Sell	Buy	Hold	Sell	Buy	Hold	Sell	Buy
250	0.61	0.21	0.19	0.91	0.14	0.12	0.47	0.49	0.48
300	0.61	0.24	0.20	0. 90	0.16	0.13	0.47	0.50	0.49
350	0.62	0.25	0.24	0.89	0.17	0.16	0.48	0.51	0.52
500	0.62	0.25	0.24	0.89	0.17	0.16	0.48	0.52	0.51
650	**0.61**	**0.32**	**0.34**	**0.83**	**0.23**	**0.26**	**0.48**	**0.51**	**0.53**
700	0.59	0.33	0.32	0.84	0.20	0.23	0.46	0.53	0.51

TABLE VI: Classification scores of DRDL (2 layers) with varying window size.

Window size (τ)	F1 Score			Precision			Recall		
	Hold	Sell	Buy	Hold	Sell	Buy	Hold	Sell	Buy
250	0.84	0.10	0.10	0.85	0.10	0.10	0.85	0.12	0.12
300	0.80	0.10	0.12	0.85	0.10	0.10	0.74	0.17	0.14
350	0.68	0.15	0.15	0.82	0.10	0.10	0.62	0.30	0.31
500	0.59	0.18	0.22	0.86	0.15	0.14	0.46	0.51	0.51
650	**0.59**	**0.19**	**0.23**	**0.88**	**0.15**	**0.12**	**0.45**	**0.52**	**0.51**
700	0.59	0.24	0.22	0.90	0.16	0.14	0.46	0.51	0.52

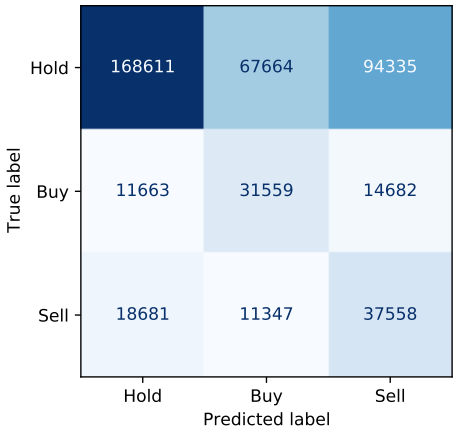
TABLE VII: Classification scores of DRDL (1 layer) with varying window size.

Stock symbols	DRDL (3 layers)	DRDL (2 layers)	DRDL (1 layer)	CNN-TA	MFNN	LSTM
WIPRO.BO	**-13.89**	-23.26	-29.14	-18.14	-27.81	-47.74
AAPL	**19.12**	11.3	10.14	0	12.92	0
AMZN	-13.23	-11.92	21.23	**30.64**	-20.85	-0.15
IOC.BO	-13.48	-23.28	**-2.68**	-3.03	-26.42	-3.1
TATACHEM.BO	1.23	**3.83**	2.19	-1.54	-8.32	0
SPICEJET.BO	**11.92**	10.17	-8.63	-24.08	-28.21	0
ATML	**-4.13**	-5.78	-10.19	-33.25	-27.07	-33.82
DOM.L	4.56	**9.34**	2.83	0.11	8.22	0.47
INDRAMEDCO.BO	-5.78	-10.34	-3.65	-14.22	**-3.53**	-50.86
Average on all 180 stocks	**3.87**	2.67	2.34	-5.08	-11.45	-13.02

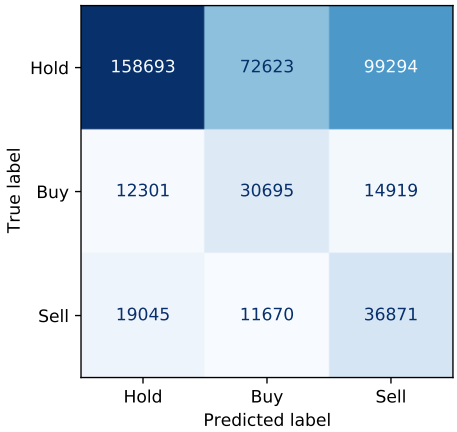
TABLE VIII: Annualized returns resulting from the stock trading decisions of different methods during the test phase.

with various window sizes to analyze and preserve the best parameter for our future experiments. We present Table V which depicts the experimental performance of DRDL (3 layers) architecture, Table VI depicts the experimental performance of DRDL (2 layers) architecture, Table VII depicts the experimental performance of DRDL (1 layer) architecture for different window sizes. The empirical results state that the performance of the approach increases as it feeds more data to the model for better understanding. We can preserve a balance parameter $\tau = 650$, which indicates stabilized performance. We further use this value in upcoming experiments.

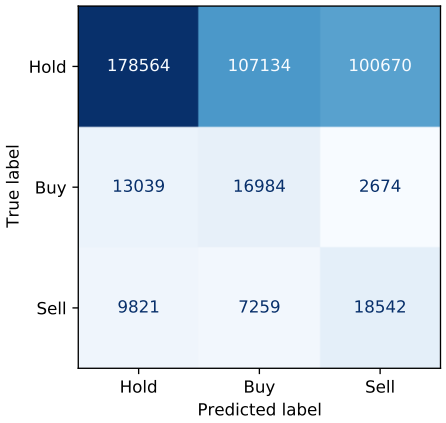
2) Classification metrics: To explain the empirical analyses of the trading process (classification), we present confusion matrices. The trading process involves classifying the signal



(a) DRDL (3 layers)

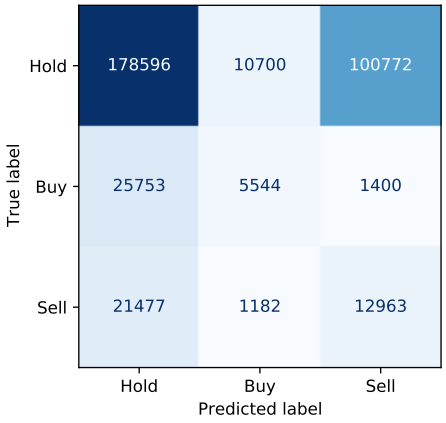


(b) DRDL (2 layers)

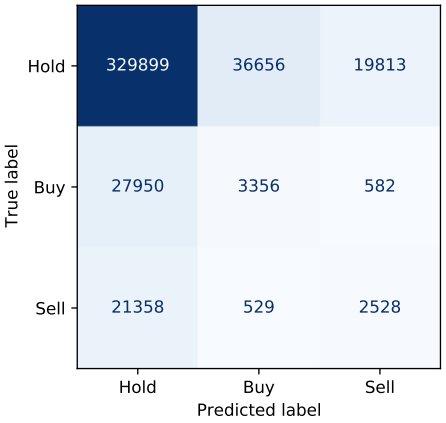


(c) DRDL (1 layer)

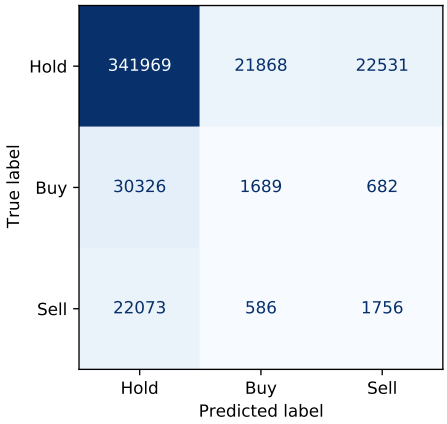
Fig. 5: Confusion matrices on stock trading classification task (averaged over days in test phase and over stocks) for DRDL with 1 to 3 layers, and deep learning techniques MFNN, CNN-TA and LSTM.



(d) MFNN



(e) CNN-TA



(f) LSTM

Fig. 5: Confusion matrices on stock trading classification task (averaged over days in test phase and over stocks) for DRDL with 1 to 3 layers, benchmark teqniques LSTM, MFNN and CNN-TA.

into three classes, namely "Buy," "hold," and "sell" classes. The summarized performance for 180 stocks by DRDL and other state-of-the-art methods is presented in Fig. 5. Among the three classes, we see the prediction of hold class is captured efficiently when compared to the other classes. The LSTM approach predicts the best score over the other state-of-the-art methods when compared to false negatives scores. However, in LSTM and CNN-TA approaches are highlighted many false positives for the "hold" class. It can be noted that these deep learning techniques have labeled most signals as hold class,

jeopardizing the model behavior for the other classes ("buy," "sell"). The nature of the finance market is highly volatile and non-linear; hence we get to see a highly imbalanced dataset. However, we noticed that the DRDL approach handles it by imposing an activation function on the operators. These operators are expected to evolute continuously as we grow deeper with time sequence. The Table. IV and Fig. 5 adds more weight to the analysis. The results state that the DRDL approach managed to predict the highly unbalanced data. The sensitivity score (Recall) is presented well by the DRDL approach compared to the state-of-the-art methods. The diagonals of the confusion matrix of the DRDL approach also takes the maximum values, which a valid classifier should expect. When dealing with highly imbalanced dataset such as finance dataset, it is more important to study classification metrics like F1 Score, Precision and Recall for each class. This helps in analyzing the model behavior for each class. The Table IV presents analysis of these classification metrics for DRDL compared to other deep learning state-of-the-art methods. We conclude that DRDL outperforms the other methods stated.

In Table IV, we also present the computational times (train and test) for conducting trading simulations for our dataset. Hereagain, LSTM is the more demanding method at training. All methods have rather comparable test times, despite DRDL is implemented on CPU only. In particular, besides its probabilistic output, DRDL is not more costly than its competitors. Adding more layers to DRDL slightly increases its train time, but does not affect much the test time.

3) Annualized Returns: Stock market aims to analyze and evaluate the return on investment for a given stock. Every trader is indeed interested in evaluating his investment returns and taking risks accordingly. We simulate market scenarios [41] by evaluating the annualized returns by the predicted stock trading decisions provided by DRDL using 1 to 3 layers as well as the decisions from from the benchmark models. Table VIII presents a detailed study of nine stocks for DRDL methodology and state-of-the-art methods. We display only empirical values for nine stocks and the average results over the 180 stocks. To make it easy for readers we have highlighted best annualized returns in bold. It is clearly evident that the DRDL approach yields higher returns when tested for a duration of 10 years when compared to annualized returns obtained from deep learning state-of-the-art methods predictions.

4) Portfolio diversification: Many researchers and traders believe that it is essential to know the associated sentiments associated with each stock to understand the stock market. Traders support and recommend having a mix of stock sentiments in one's portfolio. The market is very well divided into three types of stock sentiments: small-cap, mid-cap, and large-cap. To read about them in detail please refer to [32][section 4.4.4]. To evaluate this sentiment using the predicted signals from the proposed approach, we calculated probabilistic quantification, as explained in sec. III-C. The practitioner uses this quantification score to have a well-diversified portfolio. The score provides a confidence score that helps the investor decide where to invest in the market to have a balance of market sentiments and maximize returns.

	Stock symbols	DRDL 3 layers	DRDL 2 layers	DRDL 1 layer
Small-cap	ALOKTEXT.BO	**1.04**	1.20	1.09
	ALKYLAMINE.BO	**1.17**	1.19	1.34
	ZEEMEDIA6.BO	0.89	0.99	**0.23**
	PVP.BO	**1.34**	1.98	2.78
Mid-cap	IOC.BO	1.02	1.05	**0.87**
	TATACHEM.BO	0.76	**0.45**	0.94
	SPICEJET.BO	**0.34**	0.65	1.20
	BHEL.BO	**0.20**	0.51	1.15
Large-cap	AAPL	1.13	**0.98**	1.11
	AMZN	**0.11**	0.41	0.43
	HINDZINC.BO	**0.03**	0.65	0.45
	ONGC.BO	0.20	0.13	**0.09**
	SIEMENS.NS	0.12	**0.02**	0.11

TABLE IX: Comparative analysis of uncertainty quantification provided by DRDL using 1 to 3 layers. The quantification is listed for stocks with market capitalization categories. The log-loss is computed over the test phase.

To understand further, we present Table IX which provides a log-loss score. The log-loss score provides the confidence score in terms of its volatility nature, where the smaller value is considered, the better and less volatile. We evaluated the confidence score for the proposed approach for different configuration. The market capitalization of these stocks can be found [6]. The log-loss value provides the probabilistic inference for the predictions. The inference tries to penalize the events for which the method assigns a low probability. We observed that the log-loss value reached a meager value which indicated good prediction accuracy in large-cap stocks, which are expected to be least volatile. In contrast, we achieved a higher log-loss value for predictions associated with small-cap stocks as they are highly unstable and new to the market.

ACKNOWLEDGMENT

This work was supported by the CNRS-CEFIPRA project under grant NextGenBP PRC2017. E.C. acknowledges support from the European Research Council Starting Grant MAJORIS ERC-2019-STG-850925.

V. CONCLUSION

In our approach, time-series sequences are modeled with a flexible Gaussian SSM. The transition matrices (state and observation models) are unknown, and are estimated thanks to an expectation-minimization strategy, assuming a particular deep NMF structure. The DRDL approach inherits advantages from sophisticated modeling techniques while quantifying the uncertainty in the predictions. We have then adapted the DRDL approach to deal with a challenging large scale financial time series problem, to target stock forecasting and trading tasks. In particular, the method is able to successfully operate in an online processing manner, allowing to capture piece-wise linear characteristics in the data. The results show that the proposed method outperforms the state-of-the-art techniques. Given these promising results, we plan as future work to delve deeper into the area of financial forecasting, including the application of our technique in forecasting derivatives.

[6]https://finance.yahoo.com/screener

REFERENCES

[1] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology & Decision Making*, vol. 5, no. 04, pp. 597–604, 2006.

[2] E. F. Fama, "Efficient capital markets a review of theory and empirical work," *The Fama Portfolio*, pp. 76–121, 2021.

[3] D. Shah, H. Isah, and F. Zulkernine, "Stock market analysis: A review and taxonomy of prediction techniques," *International Journal of Financial Studies*, vol. 7, no. 2, p. 26, 2019.

[4] W. Zhao-yang, "Forecasting stock indexes based on a revised grey model and the arma model," *CAAI Transactions on Intelligent Systems*, vol. 3, 2010.

[5] G. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques-part i: Conventional methods," *Journal of Computational Optimization in Economics and Finance*, vol. 2, no. 1, pp. 45–92, 2010.

[6] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. IEEE, 2014, pp. 106–112.

[7] B. U. Devi, D. Sundar, and P. Alli, "An effective time series analysis for stock trend prediction using arima model for nifty midcap-50," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 1, p. 65, 2013.

[8] A.-C. Petrica, S. Stancu, and A. Tindeche, "Limitation of arima models in financial and monetary economics." *Theoretical & Applied Economics*, vol. 23, no. 4, 2016.

[9] S. Makridakis and M. Hibon, "Arma models and the box–jenkins methodology," *Journal of Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[10] T. M. O'Donovan, "Short term forecasting: An introduction to the box-jenkins approach." *John Wiley & Sons, INC., 605 Third AVE., New York, NY 10158, USA, 1983, 256*, 1983.

[11] S. Chadsuthi, C. Modchang, Y. Lenbury, S. Iamsirithaworn, and W. Triampo, "Modeling seasonal leptospirosis transmission and its association with rainfall and temperature in thailand using time–series and arimax analyses," *Asian Pacific Journal of Tropical Medicine*, vol. 5, no. 7, pp. 539–546, 2012.

[12] K. A. Ababio, "Comparative study of stock price forecasting using arima and arimax models," Ph.D. dissertation, 2012.

[13] R. Östermark, "Vector forecasting and dynamic portfolio selection: Empirical efficiency of recursive multiperiod strategies," *European Journal of Operational Research*, vol. 55, no. 1, pp. 46–56, 1991.

[14] N. Saini, A. K. Mittal *et al.*, "Forecasting volatility in indian stock market using state space models," *Journal of Statistical and Econometric Methods*, vol. 3, no. 1, pp. 115–136, 2014.

[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME, Journal Basic Eng.*, vol. 82, no. 1, pp. 35–45 (11 pages), 1960.

[16] T. Choudhry and H. Wu, "Forecasting the weekly time-varying beta of uk firms: Garch models vs. kalman filter method," *The European Journal of Finance*, vol. 15, no. 4, pp. 437–444, 2009.

[17] C. Wells, *The Kalman filter in finance*. Springer Science & Business Media, 2013, vol. 32.

[18] L. Ljung, "Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 36–50, 1979.

[19] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, 2000, pp. 153–158.

[20] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, "Particle filtering," *IEEE signal processing magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[21] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[22] F. R. Bach and M. I. Jordan, "Learning graphical models for stationary time series," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2189–2199, 2004.

[23] E. Chouzenoux and V. Elvira, "Graphem: Em algorithm for blind kalman filtering under graphical sparsity constraints," in *ICASSP 4 May 2020-8 May 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona*. IEEE, 2020, pp. 5840–5844.

[24] L. Rigotti and C. Shannon, "Uncertainty and risk in financial markets," *Econometrica*, vol. 73, no. 1, pp. 203–243, 2005.

[25] S. Sharma, A. Majumdar, V. Elvira, and E. Chouzenoux, "Blind kalman filtering for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4916–4919, 2020.

[26] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the em algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[27] M. E. Khan and D. N. Dutt, "An expectation-maximization algorithm based Kalman smoother approach for event-related desynchronization (ERD) estimation from EEG," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 7, pp. 1191–1198, 2007.

[28] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, no. 3.

[29] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[30] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos, "SMC2: an efficient algorithm for sequential analysis of state space models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 397–426, 2013.

[31] D. Crisan and J. Miguez, "Nested particle filters for online parameter estimation in discrete-time state-space markov models," *Bernoulli*, vol. 24, no. 4A, pp. 3039–3086, 2018.

[32] S. Sharma, V. Elvira, E. Chouzenoux, and A. Majumdar, "Recurrent dictionary learning for state-space models with an application in stock forecasting," *Neurocomputing*, vol. 450, pp. 1–13, 2021.

[33] M. A. Gonzalez-Olvera and Y. Tang, "Black-box identification of a class of nonlinear systems by a recurrent neurofuzzy network," *IEEE Transactions on Neural Networks*, vol. 21, no. 4, pp. 672–679, 2010.

[34] S. H. Won, I. Song, S. Y. Lee, and C. H. Park, "Identification of finite state automata with a class of recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 9, pp. 1408–1421, 2010.

[35] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1456–1470, 1998.

[36] P. Tino, C. Schittenkopf, and G. Dorffner, "Financial volatility trading using recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 865–874, 2001.

[37] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[38] L. Shao, D. Wu, and X. Li, "Learning deep and wide: A spectral method for learning deep networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2303–2308, 2014.

[39] P. Cheridito, A. Jentzen, and F. Rossmannek, "Efficient approximation of high-dimensional functions with neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[40] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 8, pp. 1553–1565, 2014.

[41] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.

[42] W. Long, Z. Lu, and L. Cui, "Deep learning-based feature engineering for stock price movement prediction," *Knowledge-Based Systems*, vol. 164, pp. 163–173, 2019.

[43] E. Abbe and C. Sandon, "Provable limitations of deep learning," *arXiv preprint arXiv:1812.06369*, 2018.

[44] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[45] W. Jiang, "Applications of deep learning in stock market prediction: Recent progress," *Expert Systems with Applications*, vol. 184, p. 115537, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421009441

[46] L. Chauhan, J. Alberg, and Z. Lipton, "Uncertainty-aware lookahead factor models for quantitative investing," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1489–1499.

[47] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, pp. 497–505, 2005.

[48] M. Kumar and M. Thenmozhi, "Forecasting stock index returns using arima-svm, arima-ann, and arima-random forest hybrid models," *Inter-*

national Journal of Banking, Accounting and Finance, vol. 5, no. 3, pp. 284–308, 2014.
[49] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Twenty-fourth International Joint Conference On Artificial Intelligence 10-15th September 2015 Buenos Aires*.
[50] E. Chong, C. Han, and F. C. Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," *Expert Systems with Applications*, vol. 83, pp. 187–205, 2017.
[51] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.
[52] P. De Handschutter, N. Gillis, and X. Siebert, "A survey on deep matrix factorizations," *Computer Science Review*, vol. 42, p. 100423, 2021.
[53] A. Cichocki, R. Zdunek, A. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley-Blackwell, 2009.
[54] L. Yu, C. Liu, and Z.-K. Zhang, "Multi-linear interactive matrix factorization," *Knowledge-Based Systems*, vol. 85, pp. 307–315, 2015.
[55] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep matrix factorization method for learning attribute representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 417–429, 2016.
[56] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, vol. 17. Melbourne, Australia, 2017, pp. 3203–3209.
[57] Z. Chen, S. Jin, R. Liu, and J. Zhang, "A deep non-negative matrix factorization model for big data representation learning," *Frontiers in Neurorobotics*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2021.701194
[58] J. Flenner and B. Hunter, "A deep non-negative matrix factorization neural network," Tech. Rep., 2017, https://www1.cmc.edu/pages/faculty/BHunter/papers/deep-negative-matrix.pdf.
[59] B. Liu and Y. Liang, "Optimal function approximation with relu neural networks," *Neurocomputing*, vol. 435, pp. 216–227, 2021.
[60] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova, "Nonlinear approximation and (deep) relu networks," *Constructive Approximation*, vol. 55, no. 1, pp. 127–172, 2022.
[61] M. Chen, H. Jiang, W. Liao, and T. Zhao, "Efficient approximation of deep relu networks for functions on low dimensional manifolds," *Advances in neural information processing systems*, vol. 32, 2019.
[62] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, "Deep dictionary learning," *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.
[63] S. Mahdizadehaghdam, A. Panahi, H. Krim, and L. Dai, "Deep dictionary learning: A parametric network approach," *IEEE Transactions on Image Processing*, vol. 28, no. 10, pp. 4790–4802, 2019.
[64] M. W. Jacobson and J. A. Fessler, "An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms," *IEEE Transactions on Image Processing*, vol. 16, no. 10, pp. 2411–2422, 2007.
[65] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.
[66] V. Kumar and D. Shah, "Expanding the role of marketing: from customer equity to market capitalization," *Journal of Marketing*, vol. 73, no. 6, pp. 119–136, 2009.
[67] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.
[68] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *CoRR*, vol. abs/1912.09363, 2019. [Online]. Available: http://arxiv.org/abs/1912.09363